A Sampling-Based Partial Motion Planning Framework for System-Compliant Navigation along a Reference Path

Author(s):
Schwesinger, Ulrich; Rufli, Martin; Furgale, Paul; Siegwart, Roland

Publication date:
2013

Permanent link:
https://doi.org/10.3929/ethz-a-010022855

Originally published in:
https://doi.org/10.1109/IVS.2013.6629500

A Sampling-Based Partial Motion Planning Framework for System-Compliant Navigation along a Reference Path

Ulrich Schwesinger[1], Martin Rufli[1], Paul Furgale[1] and Roland Siegwart[1]

***Abstract*— In this paper a generic framework for sampling-based partial motion planning along a reference path is presented. The sampling mechanism builds on the specification of a vehicle model and a control law, both of which are freely selectable. Via a closed-loop forward simulation, the vehicle model is regulated onto a carefully chosen set of terminal states aligned with the reference path, generating system-compliant sample trajectories in accordance with the specified system and environmental constraints. The consideration of arbitrary state and input limits make this framework appealing to nonholonomic systems. The rich trajectory set is evaluated in an online sampling-based planning framework, targeting real-time motion planning in dynamic environments.**

In an example application, a Volkswagen Golf is modeled via a kinodynamic single-track system that is further constrained by steering angle/rate and velocity/acceleration limits. Control is implemented via state-feedback onto piecewise $\mathcal{C}^0$-continuous reference paths. Experiments demonstrate the planner's applicability to online operation, its ability to cope with discontinuous reference paths as well as its capability to navigate in a realistic traffic scenario.

I. Introduction

While autonomous mobile robots have been developed since the 1960s, research on autonomous cars only intensified in the 1990s—particularly at Carnegie Mellon University (Navlab project [1], 4500 km of autonomous driving) and the University of Parma (ARGO Project [2], 2000 km of autonomous driving). At that time, operation was restricted to lane following and platooning, however. Later, the various DARPA challenges, culminating in the 2007 Urban Challenge, bundled efforts that led to systems approaching fully autonomous operation in urban-like scenarios, including interaction with other traffic participants and adherence to traffic regulations.

One of the central building blocks required for autonomous operation—the navigation framework—remains inherently difficult to approach in a unified algorithm. As a consequence, hierarchical frameworks consisting of a global planner (commonly, graph search) and a system-compliant local planner have found general acceptance. To ensure safety in dynamic environments, the local planner must ensure that returned paths are actually dynamically feasible—that they are within the dynamic capabilities of the robotic platform. Hence, the boundary between planning and traditional control becomes somewhat blurred. In this paper, we exploit this connection to develop a partial motion planning framework that utilizes the system model and controller to ensure that the resulting plans are dynamically feasible by construction.

*The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013, Challenge 2, Cognitive Systems, Interaction, Robotics, under grant agreement No 269916, V-Charge (`http://www.v-charge.eu`).

[1]Autonomous Systems Lab, ETH Zurich, Tannenstrasse 3, 8092 Zurich Corresponding author:`sculrich@mavt.ethz.ch`

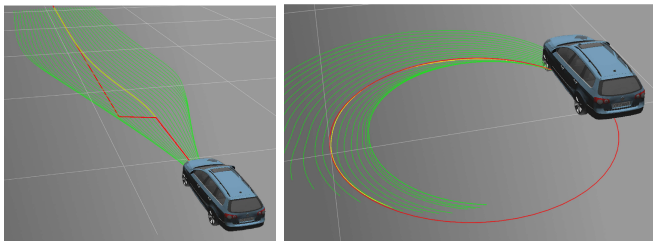

Fig. 1: *Left*: Trajectories with a fixed longitudinal speed profile (green) generated with respect to a $\mathcal{C}^0$-discontinuous reference path (red) and the minimum cost trajectory (yellow). *Right*: Trajectories generated for a reference path with curvature close to the vehicle's maximum steering angle.

A. Related Work

Related work on *local* motion planning & motion control is commonly separated into approaches that compute solution curves directly, and such that require a reference path (or trajectory) for guidance. In the former case, classic collision avoidance methods [3], [4] have gradually been replaced by state lattice graphs [5], [6] and Rapidly-exploring Random Tree (RRT) [7] methods. Nonetheless, these methods remain restricted to approximately four dimensional state spaces if online operation is desired—thereby limiting the fidelity of the agent models they can faithfully reproduce. Consequently, they typically find application in situations where no external structure can be extracted for guidance, such as off-road or on large-scale parking lots.

If a reference path (or trajectory) is available for guidance, higher-dimensional state-space representations become approachable. Road and lane information in particular enact narrow constraints on heading and curvature values. A common approach is thus to align the endpoints of local trajectory samples with the reference path [8]. This technique simultaneously reduces the search complexity and overcomes the danger of entering unsafe end states. Most of these approaches are based on geometric primitives instead of actual vehicle models, separating the computation of velocity profiles from the geometric construction of the path [9], [10], [11], [12]. This separation may result in conflicts with nonholonomic platform constraints—especially at low speeds. Consequently, trajectories need to be validated in a post-

processing step—which may in turn lead to the pruning of a considerable amount of candidate motions [8]. In contrast, the local on-road planner described in Urmson et al. [13] generates motion samples by solving two-point boundary value problems subject to the dynamics of a detailed vehicle model directly [14]. Unfortunately, this method is prone to convergence issues related to the non-invertibility of non-holonomic vehicle models.

We thus observe that existing approaches face difficulties in capturing system constraints characteristic for autonomous cars (including steering angle, steering rate, and acceleration limits, as well as reaction delays) explicitly. Furthermore, non-holonomic constraints cannot be integrated—nor inverted—in closed form.

B. Contributions

Our approach presented in this paper fully addresses these important real-world platform and environmental constraints without having to revert to model inversion techniques. It proposes a *generic* sampling-based framework requiring only (i) a reference path verifying piece-wise $\mathcal{C}^{\geq 0}$ continuity, (ii) a system description suitable for numerical forward propagation, and (iii) obstacle information expressed in vehicle frame. Its output consists of a system-compliant command sequence, which may be employed for feed-forward control of the platform directly.

The remainder of this paper is organized as follows: in Section II we introduce the overall structure of the proposed motion planning framework. Section III then describes a sample implementation for a Volkswagen Golf. Section IV reports on experiments conducted in simulation as well as during physical runs on a full-scale test vehicle. Finally, Section V concludes this paper and sketches further work.

II. Framework

This section introduces the main contribution of this paper, consisting of a novel framework for partial motion planning along reference paths. The proposed planning framework consists of a graph search routine, a system model and a control law. Graph search operates on a tree that is constructed online. Nodes, n, represent particular vehicle state / time pairs, $(\mathbf{x}, t)$, while edges, e, are constructed by system-compliant trajectory segments. An outline of a single planning cycle is portrayed in Algorithm 1.

The search routine operates online at a fixed rate in accordance with environmental requirements. During initialization, any remaining nodes in N are removed. Then, the most recent system state, $\mathbf{x}_0$, obtained from the localization module is inserted at the root of the tree at time t_0. To account for the planning time, T_{cycle}, during which the planner is busy, the root vertex is expanded according to the initial segment of the solution found during the previous search iteration (if available). Actual search thus begins from $(\mathbf{x}_1, t_0 + T_{\text{cycle}})$ using a user-supplied graph search method which needs to provide *push* and *pop* operations to insert and retrieve items from the search queue, Q, respectively.

Algorithm 1 Planning cycle

```
1: N ← ∅,   E ← ∅
2: n ← (x_0, t_0),   N ← N ∪ {n}
3: (n*, e*) ← simulate(n, f, g, T_cycle, u_0)
4: N ← N ∪ {n*},   E ← E ∪ {e*}
5: Q.push(n*)
6: for l = 1 → desired tree depth do
7:     for all nodes at tree level l do
8:         n ← Q.pop()
9:         for j = 1 → |M| do
10:            (d_ref, v_ref) ← drawSample(M)
11:            (n*, e*) ← expand(n, f, g, d_ref, v_ref, T_sim)
12:            N ← N ∪ {n*},   E ← E ∪ {e*}
13:            Q.push(n*)
14:        end for
15:    end for
16: end for
17: computeCosts(E, N)
18: n̂ ← minimum cost node at last tree level
19: ê ← edge in tree level 1 leading to n̂
20: return ê
```

Within a loop, pop operations are then triggered one at a time leading to node expansion. Successor edges are constructed by (numerical) forward propagation of a general user supplied vehicle model, **f**, with

$$\mathbf{x}\left(t + \Delta t\right) = \mathbf{f}\left(\left[\mathbf{x}\left(t\right), \dot{\mathbf{x}}\left(t\right), \ldots\right], \mathbf{u}\left(t\right), t, \Delta t\right). \qquad (1)$$

Uniquely, this allows to incorporate arbitrary system descriptions—even non-linear and non-invertible ones that are hard to work with in competing approaches. In order to contain the size of the state space when using high-dimensional system representations, however, a control-based approach building on terminal manifolds is enforced. The notion of the terminal manifold [8] refers to a reduced subspace of the state-time space towards which node expansion is directed. In our framework it takes the form of heading and curvature alignment (θ and c) with respect to a user-supplied reference path,

$$\mathbf{z}\left(s\right) = \begin{bmatrix} \theta\left(s\right) - \theta_{\text{ref}}\left(s\right) \\ c\left(s\right) - c_{\text{ref}}\left(s\right) \end{bmatrix} = \mathbf{0} \;\; \forall s > s_0, \qquad (2)$$

leaving the desired lateral offset to the reference path, d_{ref}, as well as the desired velocity, v_{ref}, at the curvilinear abscissa, s, as free variables. The alignment is enforced for all states ahead of the current curvilinear abscissa, s_0. Along the reference path, the terminal manifold, **z**, is then sampled into a set of terminal states, $M = D \times V$, built from a discrete set of lateral offsets, D, and longitudinal vehicle velocities, V. During node expansion, samples $m = (d_{\text{ref}}, v_{\text{ref}})$, are iteratively drawn from M and used as a target states for closed-loop system control. The function `expand` uses a user-supplied controller, **g**, with $\mathbf{u}\left(t\right) = \mathbf{g}\left(\mathbf{x}\left(t\right), m\right)$ to regulate the forward-simulated system model **f** towards the sample m over a fixed time period, T_{sim}. The newly retrieved state and its incoming trajectory segment are pushed back onto Q as a node / edge-pair, (n^*, e^*).

Once the tree has been expanded to a desired search depth (or maximum planning time has elapsed), edge and accumulated node costs are computed. Then, the lowest

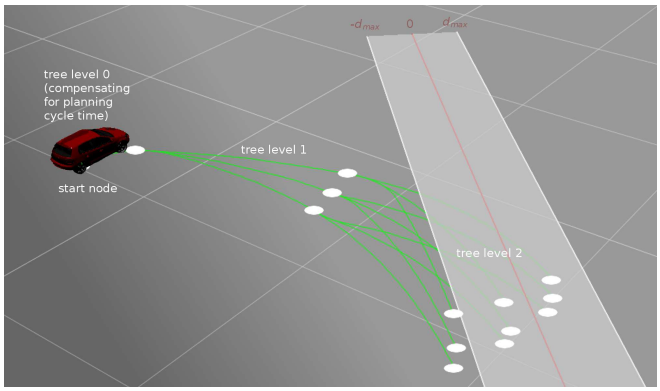

Fig. 2: A tree composed of system-compliant trajectory segments. The edge in tree level zero corresponds to the initial part of the solution found during the previous search iteration. For subsequent tree levels, the lateral terminal manifold is (for illustrative purposes) discretized into a coarse set of three samples $\{-d_{\max}, 0, d_{\max}\}$.

cost node at maximal search depth is identified and back-propagated, which results in the construction of the solution trajectory. Depending on the fidelity of the system model employed during edge construction, inputs corresponding to the solution trajectory may be directly executed on the platform as feed-forward commands.

Figure 2 illustrates a search tree created using a coarse sampling of the terminal manifold. For illustrative purposes, only one velocity profile and three lateral offsets are explored in a tree of depth three.

III. Application to an Autonomous Car

In this section, we illustrate the proposed framework of Section II by applying it to the problem of on-road autonomous driving using lane centers as reference. Our experimental platform—both in simulation and real-world experiments—is a VW Golf prepared for the European V-Charge project.

A. System Modeling

The described operating conditions require a modeling approach that is particularly suitable for low-velocity maneuvers, where non-holonomic constraints dominate. Nonetheless, it also needs to maintain accuracy at higher velocities, provided steering and acceleration commands remain within comfortable ranges and operation is limited to surfaces with sufficient traction (i.e., to paved roads).

The kinodynamic single-track model fulfills all of these conditions. At the same time it remains suitably efficient to implement and forward simulate. It combines 2D position, $(x\,/\,y)$, heading, θ, steering angle, ϕ, and longitudinal speed, v into its state vector, $\mathbf{x}$. The dynamics are then captured by

$$\underbrace{\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{\phi}\\ \dot{v}\end{bmatrix}}_{\dot{\mathbf{x}}} = \begin{bmatrix}\cos(\theta)\cdot v\\ \sin(\theta)\cdot v\\ \frac{v}{L}\tan(\phi)\\ 0\\ 0\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & 0\\ 1 & 0\\ 0 & 1\end{bmatrix} \underbrace{\begin{bmatrix}u_1\\ u_2\end{bmatrix}}_{\mathbf{u}} \tag{3}$$

This translates into a system model as specified in equation 1 via $\mathbf{f} = \mathbf{x}(t_0) + \int_{t_0}^{t+\Delta t} \dot{\mathbf{x}}(t)\,\mathrm{d}t$. The inputs take the form of steering speed, u_1, and longitudinal acceleration, u_2, stacked into the control vector, $\mathbf{u}$. L represents the inter-axle distance of the vehicle. In addition, state and actuator constraints are considered during numerical integration. We limit steering angle, $\phi \leq \phi_{\max}$, steering speed, $u_1 \leq \dot{\phi}_{\max}$, longitudinal speed, $v \leq v_{\max}$, longitudinal acceleration, $u_2 \leq \dot{v}_{\max}$, and longitudinal deceleration, $u_2 \geq \dot{v}_{\min}$. Numerical values used in our experiments derived from the manufacturer's specifications are given in Table I. Retrieving a more involved system model via an identification step is left as future work.

Parameter	Value
maximum steering angle, $\phi_{\max}$	$0.64\,\mathrm{rad}$
maximum steering speed, $\dot{\phi}_{\max}$	$0.57\,\mathrm{rad/s}$
longitudinal acceleration, $\dot{v}_{\max}$	$1.0\,\mathrm{m/s^2}$
longitudinal deceleration, $\dot{v}_{\min}$	$-1.5\,\mathrm{m/s^2}$
inter-axle distance, L	$2.578\,\mathrm{m}$

TABLE I: Vehicle parameters for the VW Golf platform. Acceleration limits selected to ensure comfort.

B. Controller Design

Control of the single-track vehicle model given in (3) onto a set of reference trajectories following the road (and thus the generation of a set of locally expressive trajectory segments) is achieved via the nonlinear state-feedback controller described in [15]. The controller is designed to follow a moving reference cart, where center of the rear vehicle axle serves as reference position. The error states for the lateral control are given by the heading difference, $\Delta\theta$, and the lateral distance, Δd, to the reference cart in the ego-vehicle frame, both depicted in Figure 3. The reference cart is placed with a speed-dependant look-ahead along the reference path. The lateral control output for the turning rate ω then is

$$\omega = \omega_f + k_1 v_{ref} \frac{\sin(\Delta\theta)}{\Delta\theta}\Delta d - k_2 \Delta\theta \tag{4}$$

with an optional feed-forward term for the turning rate, $\omega_f = v_{\mathrm{ref}} c_{\mathrm{ref}}$, based on the curvature of the reference path.

While this control law guarantees the regulation of the error states to zero eventually for an unconstrained system, the introduction of actuator constraints may negate these guarantees. In practice we have been successful in scheduling the lateral control gains k_1 and k_2 for specific longitudinal

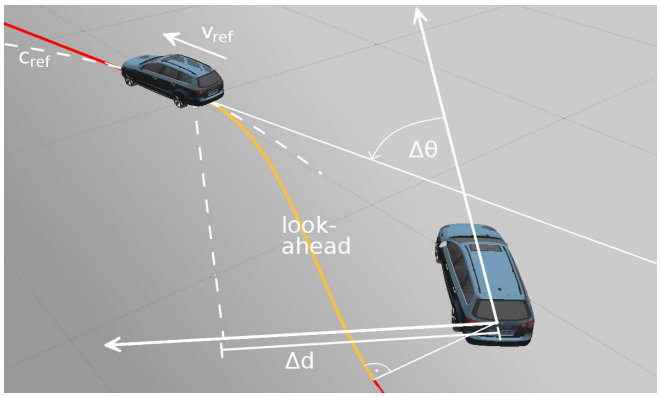

Fig. 3: Control variables for lateral control, the nonlinear state feedback regulates the error states $\Delta\theta$ and Δd to zero.

speed ranges. Note that a proof of stability is in this case not required, as the controller only serves as a generator of a locally expressive trajectory set, which is then assessed for suitability by the optimization routine, i.e., the cost function.

C. Cost Function

The design of the cost function should thus both penalize divergence from the reference path as well as rewarding progress along it. In general, these two criteria are conflicting in situations where the vehicle is not well aligned with the reference path. Due to the nonholonomic constraints acting on the platform, the vehicle then needs to deviate from the reference path in order to make progress along it. To generate a behavior that attempts to fulfill both close and quick path following, we linearly combine a lateral cost J_d with a longitudinal cost J_s according to Equation (7).

$$J_d = \frac{1}{d_{max} c_f} \int_0^{c_f} d\,(c)\, dc \tag{5}$$

$$J_s = \frac{c_f}{v_{max} T_{sim}} \tag{6}$$

$$J = k J_d + (1-k)\, J_s \tag{7}$$

J_d integrates the lateral distance to the reference path over the curvilinear abscissa c, where c_f represents the curvilinear abscissa of the reference path associated with the final state of the simulated trajectory. The integral penalizes a divergence of the platform from the reference path along the simulated trajectory, which counteracts shortcutting in sharp turns. The longitudinal cost term, on the other hand, is comprised of the final curvilinear abscissa c_f, normalized by the largest feasible travel distance $v_{max} T_{sim}$. Hence, both cost terms are normalized to lie within the range $[0, 1]$, and an intuitive combination into a single scalar cost term J becomes feasible. Via the factor k, preference between close versus rapid path following can be expressed.

Trajectories in collision with an obstacle at any single time during simulation are assigned infinite cost.

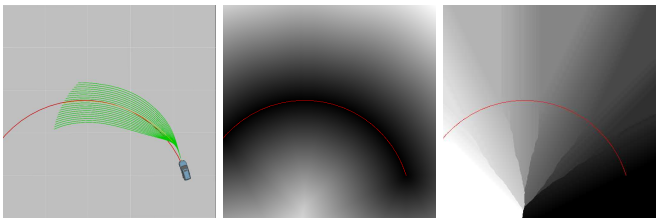

Fig. 4: *Left:* Reference path (red) and forward simulated trajectories (green) that have been obtained by controlling the vehicle model onto the terminal manifold. *Center:* Path distance transform of the reference path. Lighter colors refer to larger distances. *Right:* Voronoi labeling. Colors refer to the curvilinear abscissa of the associated reference path item.

D. Implementation Details

This section presents implementation details for aspects of our sample application.

a) Collision Checking: The static map is represented as an occupancy grid. To perform fast collision checks, we compute a one-time dilatation on the grid at the start of the planning cycle with the distance transform described in [16]. The rectangular shape of the car is approximated by a set of discs. The dilatation map is thresholded by the disk radius which allows for a single look-up collision check per disk. Collision checking with dynamic obstacles is achieved in a similar manner. Predicted positions and timestamps of obstacles are stored in an occupancy grid. Using a one-time dilatation we obtain a map that, at a given position, allows for fast look-ups of timestamps associated with that position's spatially closest dynamic obstacles. Note that this method is limited to situations where predicted obstacle paths do not overlap. For spatially colliding objects the time-gap is inspected.

b) Path Distance Transform: Our proposed controller requires the orthogonal projection of the current position onto the reference path for all visited cells during the forward simulation of the system. For a fast (approximate) computation of said projection, we calculate in a precomputation step the Voronoi decomposition of the discretized reference path with the method described in [17]. Each cell in the map is associated with the curvilinear abscissa of the nearest reference path item. The path distances themselves are needed to calculate the lateral cost term J_d. A sample of the output of this step is displayed in figure 4.

c) Parallelization: The map dilatation of the occupancy grid for fast collision checking and path distance transform including Voronoi decomposition are operations that have to be performed in a pre-processing step, but may run on individual cores in parallel. We found the implementation in *OpenCV* to operate sufficiently fast. In *OpenCV* $\geq$ 2.4.3, the function *cv::distanceTransform* provides both distance transform and Voronoi decomposition used for orthogonal projection of any given position on the map onto the reference path. Computation times are given in table II.

The remainder of the total planning time is subsequently allocated to the processes of trajectory generation and col-

lision checking. Both of these operations can be fully parallelized among CPU and GPU cores, thereby offering the potential for substantial gains in search qaulity if adequate hardware is available. Computation times provided in table II are in terms of a single CPU.

IV. Experimental Evaluation

Below we present experimental results based on the system and controller implementations described in Section III. We first analyze the effect of discontinuous and strongly curved reference paths on the closed-loop system response. Then, we qualitatively demonstrate the capabilities of our framework in simulated inner-city traffic scenario. Finally, we provide runtime results.

A. Sensitivity to Reference Path Shape

The reference path represents a key aspect of our framework in that it defines the target manifold of the sampling strategy. It is thus important that its shape does not negatively affect planning results. In this regard, figure 1 indicates that motion segments are not even unduly affected by path discontinuities and curvatures close to vehicle limits. In fact, it is the use of a closed-loop control strategy, which allows to obtain system-compliant trajectories despite reference path discontinuities. This property serves well in practice where global localization jumps may result in similar effects.

B. Navigation in Simulated Traffic Scenario

In a simulated inner-city traffic scenario, we demonstrate our planning framework's capacity for complex decision-making—including lane and intersection handling, static and dynamic obstacle avoidance, as well as platooning[1]. Importantly, the handling of these situations *emerges* from the interplay between the framework (in particular state-time space collision checking) and the environment and thus reduces the need for a separate situational awareness module.

Figure 5 shows an orthographic image of Zurich, Hegibachplatz together with an artificially created reference path for the ego vehicle and three dynamic obstacle tracks. Each dynamic obstacle track was populated with several circular dynamic objects of radius $1\,\mathrm{m}$ with a constant time gap of $4\,\mathrm{s}$ between them. Obstacle predictions for the planner were taken from the known obstacle tracks. The reference travel speeds for the ego vehicle and the dynamic obstacles on tracks 1 and 2 were set to $8.33\,\mathrm{m/s} = 30\,\mathrm{km/h}$, while those on track 3 were set to $2\,\mathrm{m/s}$ to cause a congestion.
We applied a binary mask to the static environment to mask impassable areas outside of drivable roads. In this regard note, that the tram displayed in the orthographic image was not considered in the scenario. The reference path for the ego vehicle was created based on a hand-drawn reference path in the image. Heading and curvature information was extracted by pixel-based finite differences, post-processed via a moving average filter for smoothing purposes. Despite the smoothing step, heading as well as steering angle information

[1]A video of the run can be found at `http://www.asl.ethz.ch/people/sculrich/hegibachplatz_iv13.avi`.

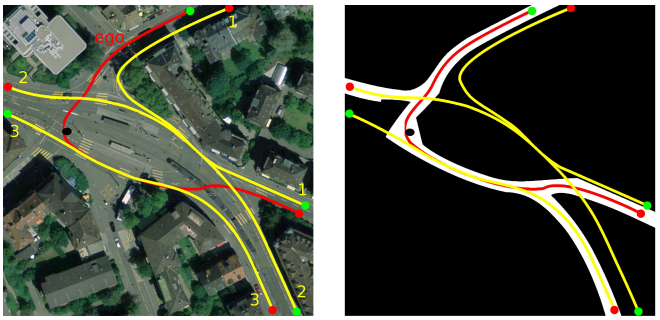

Fig. 5: Simulated run at Zurich, Hegibachplatz. *Left*: Orthographic image of Zurich, Hegibachplatz. Reference paths for the ego vehicle (red) and three dynamic obstacle tracks (yellow) are displayed, with their start (green) / end points (red) indicated. *Right*: Binary occupied(black) / free(white) space mask applied to the image, representing road information.

is rather noisy, which enables the demonstration of discontinuous reference information handling within our framework. After every planning cycle, the vehicle in simulation executes the initial parts of the solution $\mathbf{u} = \left[\dot{\phi}, \dot{v}\right]$ as feed-forward commands. In figure 6 four snapshots of the simulated run are displayed, capturing four distinct situations that challenge the planner's capabilities: *Intersection handling*, *static obstacle avoidance*, *platooning* and pure *lane following*. In the initial part of the run ($t = [0\,\mathrm{s}, 10\,\mathrm{s}]$), the ego vehicle faces oncoming vehicles on track 1. The knowledge that their future movement is constrained to the oncoming lane allows the ego vehicle to avoid improper evasive maneuvers. After $10\,\mathrm{s}$, while approaching the first intersection, the vehicle needs to yield to dynamic obstacles on track 2. The ego vehicle slows down to let the obstacles pass, as time does not permit pulling out in front of them. At $t = 18\,\mathrm{s}$, the vehicle encounters a static obstacle (black ellipse) in the middle of the intersection and steers to the right to pass by. This illustrates that the reference path is readily departed if circumstances call for it. Thereafter, the vehicle faces a convoy of four slow traveling obstacles with $2\,\mathrm{m/s}$. It decelerates and follows the convoy with a constant time gap of $3\,\mathrm{s}$ as space does not permit an overtaking maneuver. As soon as the road is clear, the ego vehicle accelerates again. In order to come to a standstill at the end of the reference path the ego vehicle brakes at the end of the run. All these maneuvers emerged from the same cost function without switching between different behavior or parameter sets.

C. Framework Runtime Results

A key aspect of every online planning algorithm is its cycle runtime. Table II displays the execution times of the main computation steps of our algorithm on a single core of an Intel Core i7@2.67 GHz, 4 MiB cache. Using this setup, a set of approximately 3500 trajectories can be evaluated (based on 100 samples per trajectory) during a planning cycle of $200\,\mathrm{ms}$. This set is large enough to retain adequate motion diversity as well as sampling density up to a tree depth of two. Parallelization and GPU implementations would further

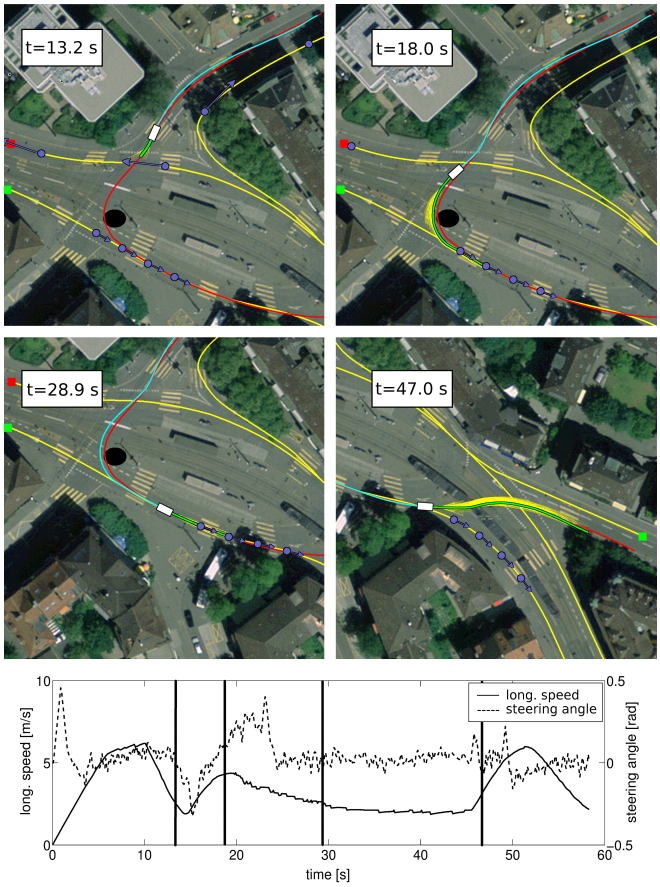

Fig. 6: *Top*: Snapshots of the run. The vehicle (odometry solution in cyan) successfully avoids dynamic obstacles (blue circles with velocity vectors) as well as static (black) ones while following the reference path (red line). From a set of explored trajectories (yellow lines), the best partial motion plan (green line) is chosen. *Bottom*: The graph shows longitudinal speed as well as steering angle of the ego vehicle. Vertical lines refer to the timestamps of the snapshots.

speed up the motion generation and collision checking steps.

operation	computation time
obstacle map dilatation	6 ms
path distance transform with Voronoi decomposition	6 ms
trajectory generation	43 ms
collision checking	11 ms

TABLE II: Computation times for the algorithm's main operations. The obstacle map contains 500×500 cells at 0.1 m resolution. Trajectory generation and collision check times are expressed per 1000 trajectories and 100 samples.

V. Conclusion

We presented a sampling-based framework for online motion planning. Through a method capable of dealing with arbitrary motion models and system constraints, the framework is widely applicable. Motion samples aligned with the reference path are generated via forward simulation of a vehicle model. A controller regulates the vehicle towards samples, representing a desired lateral offset and reference speed. The method is able to generate motions for discontinuous reference paths, where previous methods often failed. Due to the difficulty in controlling combustion engine cars at low speed, future research will focus on the identification of a system model for the real platform, capturing a probabilistic input-output map. Incorporating a probabilistic treatment of the ego motion uncertainty is intended.

References

[1] D. Pomerleau and T. Jochem, "Rapidly adapting machine vision for automated vehicle steering," *IEEE Expert*, vol. 11, pp. 19 –27, apr 1996.

[2] A. Broggi et al., "The argo autonomous vehicles vision and control systems," *International Journal of Intelligent Control and Systems*, pp. 409–441, 1999.

[3] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 278–288, 1991.

[4] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics Automation Magazine, IEEE*, vol. 4, pp. 23–33, Mar. 1997.

[5] M. Pivtoraiko and A. Kelly, "Constrained Motion Planning in Discrete State Spaces," in *Field and Service Robotics*, pp. 269–280, 2005.

[6] M. Rufli and R. Siegwart, "On the design of deformable input- / state-lattice graphs," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3071–3077, May 2010.

[7] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Tech. Rep. 98-11, Dept. of Computer Science, Iowa State University, 1998.

[8] M. Werling et al., "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *The International Journal of Robotics Research*, vol. 31, pp. 346–359, Dec. 2011.

[9] A. Bacha et al., "Odin: Team VictorTango's entry in the DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.

[10] M. Montemerlo et al., "Junior: The Stanford entry in the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[11] J. Bohren et al., "Little Ben: The Ben Franklin Racing Team's entry in the 2007 DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 598–614, 2008.

[12] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," *Intelligent Robots and Systems (IROS)*, 2009.

[13] C. Urmson et al., "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[14] T. M. Howard and A. Kelly, "Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots," *The International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007.

[15] C. Samson and K. Ait-Abderrahim, "Feedback control of a nonholonomic wheeled cart in cartesian space," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pp. 1136 –1141 vol.2, apr 1991.

[16] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," tech. rep., Cornell Computing and Information Science, 2004.

[17] G. Borgefors, "Distance transformations in digital images," *Comput. Vision Graph. Image Process.*, vol. 34, pp. 344–371, June 1986.