Focused Trajectory Planning for Autonomous On-Road Driving

Tianyu Gu[1], Jarrod Snider[2], John M. Dolan[3] and Jin-woo Lee[4]

Abstract— **On-road motion planning for autonomous vehicles is in general a challenging problem. Past efforts have proposed solutions for urban and highway environments individually. We identify the key advantages/shortcomings of prior solutions, and propose a novel two-step motion planning system that addresses both urban and highway driving in a single framework. Reference Trajectory Planning (I) makes use of dense lattice sampling and optimization techniques to generate an easy-to-tune and human-like reference trajectory accounting for road geometry, obstacles and high-level directives. By focused sampling around the reference trajectory, Tracking Trajectory Planning (II) generates, evaluates and selects parametric trajectories that further satisfy kinodynamic constraints for execution. The described method retains most of the performance advantages of an exhaustive spatiotemporal planner while significantly reducing computation.**

I. INTRODUCTION

In the past three decades, the development of autonomous passenger vehicles has been drawing great attention from both academia and industry. Vehicle autonomy has great potential in bringing transportation systems to a new level of safety and efficiency, and will have a positive impact on people's lives.

Motion planning is a core technology for on-road autonomous driving. It must produce safe and user-acceptable trajectories in a wide range of driving scenarios. This paper presents a motion planning framework that achieves high performance in urban and highway driving settings while significantly reducing computation with respect to similarly performing methods.

II. RELATED WORK

Path generation schemes are foundational to motion planning. Arc-line[1], Bezier curve[2], B-splines[3] and quintic splines [4] have been proposed as path primitive types. However, the drawbacks of curvature discontinuity in [1] [2] and the lack of intuitive parametrization in [**?**] [4] make them less appealing for planning for passenger vehicles.

[5] and [6] proposed a real-time path planning algorithm for smooth lane changing using a high-order polynomial equation. They find a closed-form solution with second-order path continuity, but have difficulty in generating a path to avoid multiple obstacles in urban traffic situations.

*This work was supported by General Motors

[1]Tianyu Gu is with Dept. of Electrical Engineering, Carnegie Mellon University, Pittsburgh, USA tgu@andrew.cmu.edu

[2]Jarrod Snider is with Dept. of Electrical Engineering, Carnegie Mellon University, Pittsburgh, USA jmsnider@andrew.cmu.edu

[3]John M. Dolan is with Dept. of Electrical Engineering and Robotics Institute, Carnegie Mellon University, Pittsburgh, USA jmd@cs.cmu.edu

[4]Jin-woo Lee is with R&D, General Motors, Warren, USA jin-woo.lee@gm.com

[7] proposed the use of polynomial curvature spirals, which have the advantages of intuitive parameterization and computational efficiency. This method was further adapted for highway planning in [8].

Optimization techniques have been widely used in finding optimal trajectories for ground vehicles. [9] and [10] used time and lateral acceleration, respectively, as their optimization criterion. However, both works used a single optimality criterion, which lacked the tuning capability. [11] used the conjugate gradient method to optimize a path from an initial result given by the A* algorithm. [12] formalized nonlinear obstacle avoidance trajectory optimization into several steps and treated them as convex optimization subproblems. While they showed good results given enough runtime, neither [11] nor [12] is capable of meeting the real-time constraints of the on-road driving application.

Recent development of autonomous vehicles has been greatly expedited by the DARPA Urban Challenge. CMU's Boss [13] won the competition with an architectural planning framework with three subsystems: 1. Mission planner for the global route; 2. Behavior planner for rule-based reasoning; 3. Motion planner for an executable trajectory.

For on-road motion planning in Boss, one layer of fixed poses is sampled by laterally offsetting the centerline pose at a short lookahead distance, yet retaining the road heading. A quadratic curvature spiral path primitive type is used connecting the current pose with sampled poses to generate path candidates. Applying linear velocity profiles to each path generates a pool for trajectory evaluation.

Based on Boss, [14] proposed a GPU-based spatiotemporal lattice highway planner, which showed great capability in several challenging scenarios. A spatial lattice is laid out conforming to the entire road for path sampling, and a temporal dimension is then appended to the spatial lattice to create a dense spatiotemporal search space. Instead of having a separate Behavior module, various cost terms inducing the desired behaviors were devised to select a desired trajectory. However, the nature of this solution is exhaustive sampling, which requires huge computation to explore a large and partially unnecessary spatiotemporal state space. Moreover, it is questionable whether the cost-function-based approach can be used to entirely replace the high-level behavior reasoning, some of which could be intrinsically rule-based.

To quell the search space blow-up, [15] attempted to make use of dynamic programming to plan discretely first, and further generate the smooth plan by focusing the search. We adapt a simplified version of this to account for static obstacles.

A. Motivation

The urban driving planning scheme in [13] and its highway variant [14] motivate our work. It is non-trivial to design a planner that naively merges these two scenarios. Nevertheless, urban and highway environments can be very similar, as shown in Fig. 1. It is desirable to have a unified motion planner for both environments.

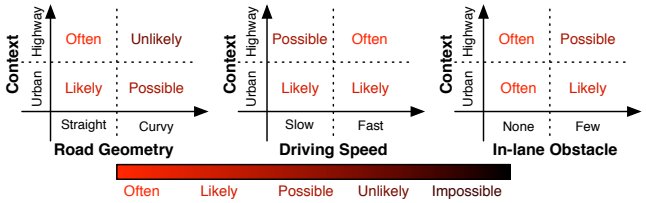


Fig. 1: Motion Planning Architecture

One common trait of all prior works mentioned above is explicit trajectory sampling and evaluation for on-road planning. Spatial (path) sampling is performed first. Three major factors determine the sampling: 1. path primitive type; 2. sampling horizon (*short* vs. *long*); 3. sampling pattern (*simple* vs. *lattice*). For the path primitive type, we use curvature polynomial spirals, as in [7] [14].

A simple approach to the sampling horizon is to relate the host vehicle speed to the horizon to prevent uncomfortable reactiveness due to being too short-sighted. This is addressed in [13] by making the lookahead horizon an empirical function of current speed. In [14], however, the horizon is always a constant long value, which is good in being less reactive, but contributes to the enormous search space. It is ideal to determine the lookahead distance based on a principled method, preferably related to the future velocity profile (an estimate), which is based on the future road geometry, road conditions and some other maneuver directives. This prevents undesirable reactiveness and an excessive search space.

For sampling patterns, simple sampling in [13] by generating paths connecting one layer of lookahead nodes is valid when the vehicle is traveling at low speed (which means a short sampling horizon) and the road geometry is not difficult. As the speed increases (which means the sampling horizon should also increase), the simple pattern will fail to conform to (complex) road geometry. Hence, the lattice approach used in [14] is used in our work to allow path sampling over long and complex road geometry.

Temporal (trajectory) sampling is typically performed after spatial (path) sampling. A direct way of building a trajectory space is by combining the spatial and temporal spaces into a high-dimensional spatiotemporal space, as in [14]. The resulting trajectory space is large enough that GPU parallel computation has to be used to achieve real-time performance.

We acknowledge the validity of spatiotemporal approaches for general motion planning in dynamic environments. It would be ideal, though, if the search space could be focused on the region where the optimal solution is likely to exist. For normal on-road planning, the spatial space is strictly constrained by the road geometry. Meanwhile, several user preference indices can greatly reduce the temporal space given the spatial space. Hence, a separate planning step that decouples spatial and temporal planning to trim the spatiotemporal space is useful.

We propose a two-step planning architecture. The Reference Trajectory Planner (I) uses multiple optimization techniques to generate a non-parametric human-like reference trajectory accounting for road geometry, obstacles and high-level directives. The Tracking Trajectory Planner (II) conducts focused spatiotemporal sampling and evaluation. By using parametric trajectories, it guarantees analytical continuity. One optimal trajectory is selected and used for execution. Like [13], we believe it is valid to have a higher-level reasoning module (Behavior module) in our planning system. The two-step planning should generate trajectories that reflect the high-level directives. Fig. 2 shows our proposed motion planning system.

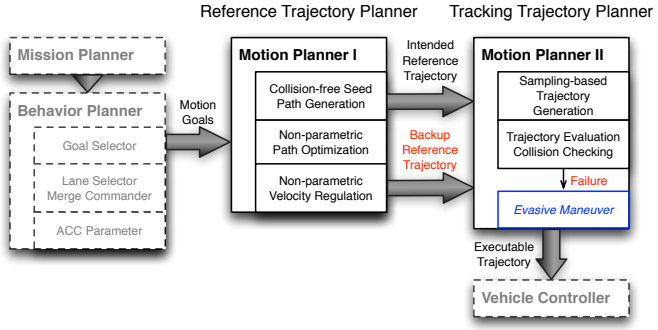


Fig. 2: Motion Planning Architecture

III. REFERENCE TRAJECTORY PLANNING

Reference trajectory planning generates a non-parametric trajectory optimized for road geometry, obstacles and higher-level motion directives.

Multiple optimization techniques are applied. The lack of an absolute criterion of trajectory quality makes it hard to devise optimality criteria. However, user preference gives a good subjective criterion. Hence it is important to formulate the optimization in a way that generates human-like trajectories and allows easy tuning to reflect individual preference.

A. Road Blockage Detection & Seeding Path Generation

Road geometry is defined by the centerline of the lane, which is sampled at a constant interval Δs to get a set of N points $P^c = \{\mathbf{p}_i^c \mid i = 0 \cdots N-1\}$; each point has an associated unit lateral vector $\overrightarrow{\mathbf{n}}_i^c$, as shown in Fig. 4.

To plan a coarse collision-free path, K layers of lookahead nodes are laid out conforming to the lane at an appropriate interval ΔL, which is related to Δs as follows:

$$\Delta L = \frac{N-1}{K-1} \cdot \Delta s$$

Edges are linear paths generated by connecting all nodes at one layer to all nodes at the next layer (blue curve in Fig. 3). Cost is assigned to each edge $e_{n_k \to n_{k+1}}$, where n_k

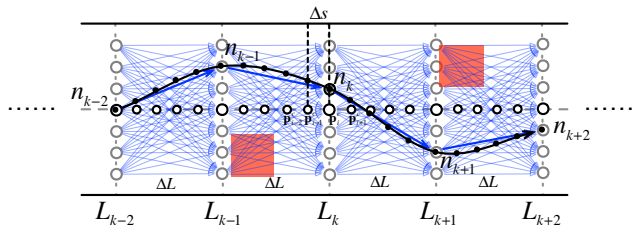


Fig. 3: Blockage Detection & Seeding Path Generation

represents the node from layer L_k and n_{k+1} the node from layer L_{k+1}:

$$C(e) = w_d \cdot d(e) + (1 - w_d) \cdot o(e) + \delta_{obstacle}(e) \quad (1)$$

In (1), $d(e)$ penalizes long edges, $o(e)$ penalizes lateral offset, and the weight w_d determines the tradeoff between these two terms; $\delta_{obstacle}(e)$ prevents collision with static obstacles and is defined by

$$\delta_{obstacle} = \begin{cases} 0 & \text{if edge is collision-free} \\ \infty & \text{otherwise} \end{cases}$$

The problem becomes a shortest-path problem, which can be quickly solved by the dynamic programming algorithm. The optimal solution is given by a sequence of moves on the K-layer network that satisfies:

$$\underset{\{e_{n_k \to n_{k+1}}\}}{\operatorname{argmin}} \sum_{k=0}^{K-1} C(e_{n_k \to n_{k+1}}) \quad (2)$$

The dynamic programming determines a decision associated with each node. If no decision exists, this node is unreachable due to prior blockage; if all nodes at one layer have infinite values, the road is blocked. Hence, blockage can be quickly detected as a by-product of the dynamic programming algorithm.

Connecting K points (the dark blue arrows in Fig. 3) by interpolating between any two points with cubic spline produces a curvature-continuous path (the black curve in Fig. 3 and 4), on which N points are sampled to generate a non-parametric seeding path (the black dots in Fig. 3 and 4). For the purpose of subsequent optimization, the sampling finds the lateral offset (given by o_i^{seed} in Fig. 4) indicated by the spline function at Δs intervals

$$P^{seed} = \{\mathbf{p}_i^{seed} = \mathbf{p}_i^c + o_i^{seed} \cdot \overrightarrow{\mathbf{n}}_i^c \mid i = 0 \cdots N-1\}$$

B. Non-parametric Path Optimization

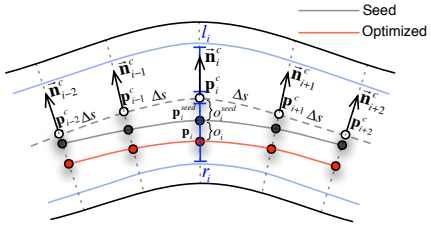


Fig. 4: Road Model & Non-parametric Path Optimization

A non-parametric path under optimization is given with respect to the seeding path P^{seed}

$$P = \{\mathbf{p}_i \mid i = 0 \cdots N-1\}$$

where

$$\begin{aligned} \mathbf{p}_i &= \mathbf{p}_i^{seed} + o_i \cdot \overrightarrow{\mathbf{n}}_i^c \\ &= \mathbf{p}_i^c + (o_i + o_i^{seed}) \cdot \overrightarrow{\mathbf{n}}_i^c \end{aligned}$$

and o_i is the nudging scalar to be determined.

The blue curves in Fig. 4 with value pairs of l_i and r_i set the bounds of optimization

$$o_i + o_i^{seed} \in [r_i, l_i]$$

Since the path points are close to each other, the linear distance Δs_i can be used to approximate the actual path distance between two points:

$$\Delta s_i = \|\mathbf{p}_{i+1} - \mathbf{p}_i\| \quad (3)$$

The unit heading vector $\mathbf{u}_i$ is given by

$$\mathbf{u}_i = \frac{(\mathbf{p}_{i+1} - \mathbf{p}_i)}{\Delta s_i} \quad (4)$$

The change of unit heading vector can be calculated accordingly as a measurement of path curvature $\boldsymbol{\kappa}_i$

$$\boldsymbol{\kappa}_i = \frac{(\mathbf{u}_{i+1} - \mathbf{u}_i)}{\Delta s_i} \quad (5)$$

The optimization criterion is

$$\underset{\{o_i\}}{\operatorname{argmin}} \left(\sum_{i=0}^{N-1} w_\kappa \cdot \|\boldsymbol{\kappa}_i\| + (1 - w_\kappa) \cdot |o_i| \right) \quad (6)$$

where the weight w_κ trades off between minimizing cumulative curvature and minimizing cumulative lateral offset.

The Levenberg-Marquardt algorithm is used in the optimization. Like most nonlinear programming techniques, this algorithm only returns local minima unless the objective function is convex. The $\|\boldsymbol{\kappa}_i\|$ term in Equation (6) is not convex, so the optimization result is a local optimum.

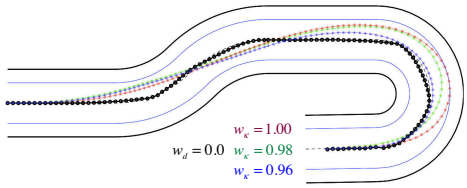
The result is influenced by the initialization, and both the smoothness and the position of the seeding path matter. Empirical experience shows that the algorithm takes a long time to converge if the seeding path is curvature-discontinuous. This is the reason for using the cubic interpolation generating the seeding path in the previous subsection. Meanwhile, the position of the seeding path affects the local minima to which the algorithm converges.

In sum, there are two parameters that affect the optimization result: w_d in Equation (1) from the seeding path generation and w_κ in Equation (6) from the nonlinear optimization. We have the tuning capability to capture different user preferences by varying these parameters, as shown in Fig. 5(a) and Fig. 5(b). The optimized path is given by

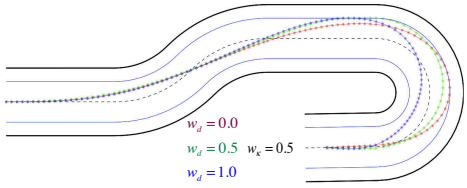
$$P^{opt} = \{\mathbf{p}_i^{opt} \mid i = 0 \cdots N-1\}$$

where

$$\mathbf{p}_i^{opt} = \mathbf{p}_i^c + (o_i^{opt} + o_i^{seed}) \cdot \overrightarrow{\mathbf{n}}_i^c$$



(a) Different w_κ



(b) Different w_d

Fig. 5: Path Parameter Tuning

C. Non-parametric Velocity Regulation

The non-parametric reference velocity profile is one extra dimension v_i^{reg} appended to each optimized path point $\mathbf{p}_i^{opt}$.

$$V^{reg} = \{v_i^{reg} \mid i = 0 \cdots N-1\}$$

Three legal and user preference constraints can be applied to reduce the temporal space:

Limit 1: Legal road speed

$$v_i^{reg} \leq V_{lon}^{Limit}$$

Limit 2: Maximum centripetal acceleration

$$|\kappa_i| \cdot (v_i^{reg})^2 \leq A_{lat}^{Limit}$$

Limit 3: Maximum longitudinal acceleration/deceleration

$$-D_{lon}^{Limit} \leq \dot{v}_i^{reg} \leq A_{lon}^{Limit}$$

Limit 1 can be constrained straightforwardly. To apply *Limit* 2, we must first find the curvature κ_i at each path point $\mathbf{p}_i^{opt}$ by locally (least-square) fitting a spline curve and calculating the curvature analytically. *Limit* 3 is satisfied using algorithm 1, which approximates velocity to be linear between discretized points.

D. Trajectory Optimization Result

Fig. 6(a) demonstrates a sample segment of road on which the trajectory optimizations are performed. The centerline is the dashed black curve, and the optimized path is in red. Comparing the two curvature plots in Fig. 6(b), we can see the path optimization obviously results in much smoother curvature change by leveraging the lateral width of the lane.

Fig. 6(c) and Fig. 6(d) show the velocity profile being regulated with centerline and optimized path, respectively. The black (constant), blue and red curves represent the shapes of the velocity profile after applying each of the three limits. The red-shaded areas are the truncated temporal space. Comparing the two figures, the path optimization also smooths the velocity profile overall.

Algorithm 1 Capping Acceleration/Deceleration

Require: A discrete velocity profile V
Ensure: An Acc/Dec-regulated velocity profile V^{reg}
  SCAN Velocity profile
  IDENTIFY Accelerating regions R_a
  IDENTIFY Decelerating regions R_d
  DO
  FOR EACH region r $\in R_a$
    FOR points in region r
      IF $(v_{i+1}^2 - v_i^2)/(2 \cdot s_i) \geq A_{lon}^{Limit}$
        $v_{i+1} = \sqrt{v_i^2 + 2 \cdot s_i \cdot A_{lon}^{Limit}}$.
  FOR EACH region r $\in R_d$
    FOR points in region r
      IF $(v_i^2 - v_{i+1}^2)/(2 \cdot s_i) \geq D_{lon}^{Limit}$
        $v_i = \sqrt{v_{i+1}^2 + 2 \cdot s_i \cdot D_{lon}^{Limit}}$.
  WHILE change of velocity profile $\leq$ Threshold

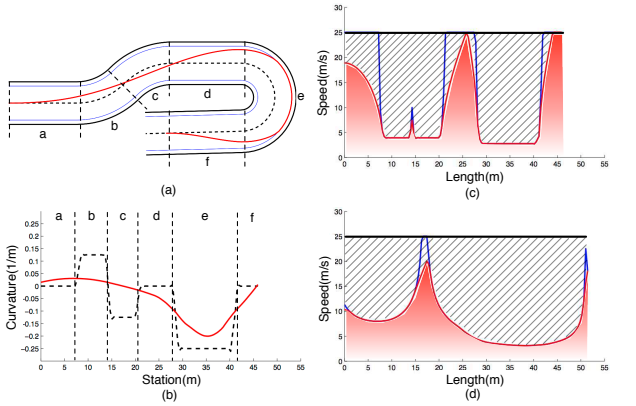


Fig. 6: Reference Trajectory Optimization Simulation Result

IV. TRACKING TRAJECTORY GENERATION

A common controller approach is to split up path tracking and velocity tracking. [16] compared geometric, kinematic and dynamic path trackers, and concluded that they all require at least curvature-continuous path reference and sometimes even first-derivative curvature continuity. The velocity tracker is commonly implemented as a PI controller that tracks the velocity reference independently. High-order continuity of the velocity profile is preferred to prevent overshoot and uncomfortable oscillation.

With the reference trajectory as input, this section explains the details of parametric trajectory generation, focused spatiotemporal sampling and evaluation.

A. Parametric Path Generation

Choosing the polynomial curvature spiral as the path primitive type, the path generation problem is formulated as: given starting vehicle state X_0 and desired final vehicle state X_f, find the desired parameter vector P that generates a path connecting X_0 and X_f satisfying the path model

$$X_f = X_0 + F(P) \tag{7}$$

where F abstracts the kinodynamic model in [7], given by

$$\begin{aligned} x(s) &= \int_0^{s_f} cos(\theta(s)) \cdot ds \\ y(s) &= \int_0^{s_f} sin(\theta(s)) \cdot ds \\ \theta(s) &= \int_0^{s_f} \kappa(s) \cdot ds \\ \kappa(s) &= p_0 + p_1 \cdot s + p_2 \cdot s^2 + ... \end{aligned} \tag{8}$$

In the cubic spiral case, the terms in $\kappa(s)$ stop at $p_3 \cdot s^3$. Hence, the unknowns to solve for are $P = [s_f, p_0, p_1, p_2, p_3]^T$; The inputs are starting vehicle state $X_0 = [x_0, y_0, \theta_0, \boldsymbol{\kappa_0}]^T$ and goal vehicle state $X_f = [\boldsymbol{x_f}, \boldsymbol{y_f}, \boldsymbol{\theta_f}, \boldsymbol{\kappa_f}]^T$. (x, y), θ and κ specify position, heading and curvature (implies its steering angle) respectively.

The gradient-based shooting method proposed in [7] can solve for the unknown parameters efficiently. Note that the number of explicit constraints (bold parameters) must equal the number of unknown parameters in P to have a unique solution.

B. Parametric Velocity Generation

Instead of using linear velocity profiles, as did many prior works, we specify velocity as a cubic function of time to be capable of generating an acceleration-continuous profile, which is important for smooth low-speed driving.

$$v(t) = q_0 + q_1 t + q_2 t^2 + q_3 t^3 \tag{9}$$

Given the start velocity v_0 and acceleration a_0, final velocity v_f and acceleration a_f, and the path length s_f, the unknown parameters $[t_f, q_0, q_1, q_2, q_3]^T$ of velocity are given analytically by solving the following equations.

$$\begin{aligned} v(0) &= v_0 = q_0 \\ a(0) &= a_0 = q_1 \\ v(t_f) &= v_f = q_0 + q_1 t_f + q_2 t_f^2 + q_3 t_f^3 \\ a(t_f) &= a_f = q_1 + 2 q_2 t_f + 3 q_3 t_f^2 \\ s(t_f) &= s_f = q_0 t_f + \tfrac{1}{2} q_1 t_f^2 + \tfrac{1}{3} q_2 t_f^3 + \tfrac{1}{4} q_3 t_f^4 \end{aligned} \tag{10}$$

Note that we still retain the capability of generating a linear velocity profile (e.g. urgent stop) by enforcing $q_2 = q_3 = 0$.

C. Focused Trajectory Sampling and Evaluation

1) Sampling: With the path/velocity generation schemes described above, the reference trajectory is used to conduct focused spatiotemporal sampling. Three factors determine the sampling process: lookahead horizon, path endpoints and velocity endpoints. Fig. 7 illustrates the idea.

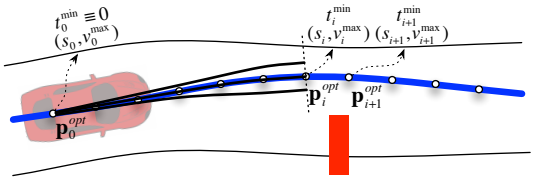


Fig. 7: Trajectory Sampling near Reference Trajectory

Approximating the velocity to be linear between points, we can estimate the earliest arrival time (EAT) t_i^{min} of each point $\mathbf{p}_i^{opt}$ by sequentially calculating

$$t_{i+1}^{min} = t_i^{min} + \frac{2 \cdot (s_{i+1} - s_i)}{(v_{i+1}^{max} + v_i^{max})} \tag{11}$$

Given the lookahead time T, the trajectory horizon is chosen at the point whose EAT is the closest to T. This provides a principled way of sampling that guarantees (at least) T seconds of validity. Following determination of the horizon, the path set is sampled by selecting several end points slightly laterally offset to either side of the lookahead point $\mathbf{p}_i^{opt}$ on the reference trajectory. At last, for each sampled path, a few velocity profiles are sampled by ranging the end velocity from 0 to the maximum reference velocity v_i^{max} at point $\mathbf{p}_i^{opt}$.

2) Evaluation: The sampled trajectory is first checked against both static and obstacles explicitly to guarantee safety. The evaluation is then performed measuring both spatial and temporal closeness of the candidate trajectories to the reference. The trajectory with the minimum cost is selected:

$$cost = w_{spatial} \cdot C_{spatial} + w_{temporal} \cdot C_{temporal}$$

where $C_{spatial}$ is the cumulative lateral distance offset with respect to the reference path, and $C_{temporal}$ is the cumulative time offset with respect to the reference trajectory. The weights $w_{spatial}$ and $w_{temporal}$ can be adjusted to give the relative importance of spatial and temporal closeness.

V. RESULTS

Experiments on a challenging route were conducted in our high-fidelity simulation environment.

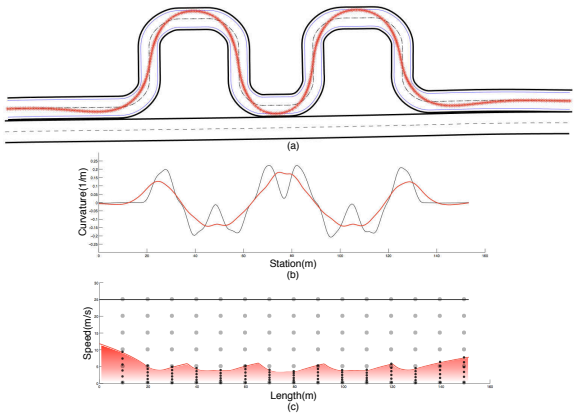


Fig. 8: Tight Turns

Fig. 8(a) shows the optimized path for the snake-like road segment. The result is a smoother, more human-like path, which is easier to track compared to those methods [17] that directly follow the centerline. In Fig. 8(b), the curvature plots of the centerline (black) and the optimized path (red) are shown. The optimized curvature curve is more tightly bounded, and smoother.

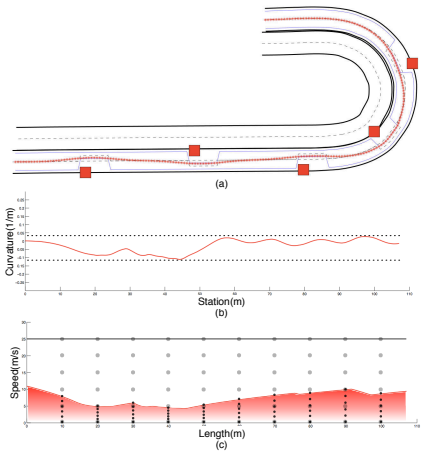


Fig. 9: Obstacle Avoidance

An obstacle avoidance path was also generated in a challenging scenario with both turns and straight road (Fig. 9(a)). The curvature plot of the optimized path in Fig. 9(b) demonstrates smooth avoidance maneuvers.

It can be seen in both Fig. 8(c) and Fig. 9(c) that since there is a sequence of lateral swerves in front, the optimized velocity profile (red blended region) reflects this oncoming driving context and reduces the temporal sampling space. This benefits the trajectory sampling and evaluation process by focusing the velocity sampling. [13], [14] conduct uniformly-spaced velocity sampling up to the speed limit as demonstrated by the light grey circles, while our approach samples the same number of points, but focuses them in the allowable regions, as shown by the dark dots. This results in less invalid velocity sampling and overall more human-like trajectories.

The proposed approach as currently implemented provides a roughly five-fold computation reduction compared to the method described in [14]. For the Tracking Trajectory Planner (TTP), the number of velocity samples is the same, but the number of longitudinal and lateral samples is reduced respectively from 7 to 3 and from 19 to 5, yielding an approximate nine-fold reduction. The combined required computation time for the steps in the Reference Trajectory Planner (RTP) is comparable to that of the TTP, cutting the reduction in half to roughly five-fold. The computation time can be even further reduced, given the current naive implementation of the optimization schemes in the RTP.

VI. CONCLUSIONS

We propose a novel approach addressing on-road planning by conducting focused spatiotemporal search. Our scheme retains most of the performance advantages of exhaustive sampling approaches. Furthermore, the search is focused to a reachable/desirable subset of the vast spatiotemporal space to reduce irrelevant sampling. While significantly reducing computation, the optimization techniques can also generate one human-like and easy-to-tune reference trajectory that can efficiently account for road geometry, obstacles and higher-level directives.

Future work includes increasing computation speed further by training a neural network to perform path optimization in a way that approximates the nonlinear optimization routine. Also, with the many tuning parameters designed in this paper, we can make use of machine learning techniques to train for individual preference.

The proposed approach assumes well-regulated traffic conditions. A safer planning system would also require evasive maneuver planning that deals with hazardous driving scenarios. Explicit spatiotemporal planning and a deeper synthesis of planning and control are needed.

Our proposed planning architecture has been partially implemented in our autonomous vehicle system. More tests on the methods used will be conducted on-vehicle to verify and perfect a few design choices, like the path primitive type and sampling pattern.

REFERENCES

[1] J. Horst and A. Barbera, "Trajectory Generation for an On-Road Autonomous Vehicle," *NIST IR*, Sept. 2005.

[2] J. wung Choi, R. Curry, and G. Elkaim, "Path planning based on bezier curve for autonomous ground vehicles," *World Congress on Engineering and Computer Science, Advances in Electrical and Electronics Engineering - IAENG Special Edition of the*, vol. 0, pp. 158–166, 2008.

[3] J. Connors and G. Elkaim, "Manipulating B-Spline based paths for obstacle avoidance in autonomous ground vehicles," *ION National Technical Meeting, ION NTM 2007*, pp. 1081–1088, 2007.

[4] A. Piazzi and C. Guarino Lo Bianco, "Quintic g2-splines for trajectory planning of autonomous vehicles," in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pp. 198 –203, 2000.

[5] J.-W. Lee and B. Litkouhi, "Control and Validation of Automated Lane Centering and Changing Maneuver," *ASME Dynamic Systems and Control Conference*, Oct 2009.

[6] J.-W. Lee and B. Litkouhi, "A unified framework of the automated lane centering/changing control for motion smoothness adaptation," *The 15th IEEE Intelligent Transportation Systems Conference*, 2012.

[7] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *The International Journal of Robotics Research*, vol. 22, pp. 583 – 601, July 2003.

[8] M. McNaughton, *Parallel Algorithms for Real-time Motion Planning*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2011.

[9] D. Q. Tran and M. Diehl, "An application of sequential convex programming to time optimal trajectory planning for a car motion," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pp. 4366 –4371, dec. 2009.

[10] C. Dimitrakakis, "Online statistical estimation for vehicle control," *IDIAP-RR 13, IDIAP*, 2006.

[11] D. D. et al., "Practical search techniques in path planning for autonomous driving," in *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*, (Chicago, USA), AAAI, June 2008.

[12] G. P. Bevan, H. Gollee, and J. O'Reilly, "Trajectory generation for road vehicle obstacle avoidance using convex optimization," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 224, pp. 455–473, Jan. 2010.

[13] C. U. et al., "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robotics*, vol. 25, no. 8, 2008.

[14] M. e. a. McNaughton, "Motion Planning for Autonomous Driving with a Conformal Spatiotemporal Lattice," *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 4889–4895, Sept. 2011.

[15] T. Gu and J. Dolan, "On-road motion planning for autonomous vehicles," *Intelligent Robotics and Applications*, pp. 588–597, 2012.

[16] J. M. Snider, "Automatic steering methods for autonomous automobile path tracking," Tech. Rep. CMU-RI-TR-09-08, Robotics Institute, Pittsburgh, PA, February 2009.

[17] M. e. a. Montemerlo, "Junior: The stanford entry in the urban challenge.," *J. Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.