Tunable and Stable Real-Time Trajectory Planning for Urban Autonomous Driving

Tianyu Gu[†], Jason Atwood[†], Chiyu Dong[†], John M Dolan[†,‡] and Jin-Woo Lee[§]

Abstract— **This paper investigates real-time on-road motion planning algorithms for autonomous passenger vehicles (APV) in urban environments, and proposed a computational efficient planning formulation. Two key properties, tunability and stability, are emphasized when designing the proposed planner. The main contributions of this paper are:**

- **A computationally efficient decoupled space-time trajectory planning structure.**
- **The formulation of optimization-free elastic-band-based path planning and speed-constraint-based temporal planning routines in pre-determined runtime.**
- **Identification of continuity problems with previous cost-based lattice planners that caused tunability and stability issues.**

I. INTRODUCTION

Autonomous passenger vehicles (APV) have advanced significantly in both academia and industry since the 2007 DARPA Urban Challenge [1]. Subsequent efforts like Google's autonomous car project have drawn additional social attention. For a mobile robotic system like APV, the motion planning (MP) component plays a critical role.

On-road planning is a special application. It differs from other open-space low-speed motion planning tasks in its limited lateral search space (constrained by the lane width) coupled with a long longitudinal horizon (lane look-ahead). Moreover, the speed plan quality matters at least as much as that of the path plan since it plays an important role in guaranteeing driving safety and passenger comfort.

There are two distinct types of objects for autonomous driving: static and moving. For static objects, we can typically apply a decoupled trajectory planning scheme by determining the path first and generating the speed profile by applying certain speed constraints [2]. In order to handle moving objects, prior on-road planning methods [3], [4] directly incorporated the temporal dimension in the construction of the (spatiotemporal) trajectory search space. However, there are two main drawbacks. First, the number of graph edges (trajectory pieces) grows exponentially as the length of the longitudinal horizon increases (given a fixed longitudinal sampling resolution), and trajectory evaluation (e.g. collision-checking) is the most computationally expensive operation in motion planning. Practical compromises (reducing sampling resolution and/or trimming) have to be made to perform search spatiotemporally, which often causes planning sub-optimality. This problem is often referred to as the *curse of dimensionality*. Second, a straightforward spatiotemporal search scheme results in wasteful search space exploration, since much of the sampled trajectories' temporal ranges are well beyond the moving objects' trustworthy prediction horizon for any practical sensing system. Hence, our first key motivation is:

M1: Alleviate the curse of dimensionality *and wasteful search space construction by exploring mainly decoupled and locally spatiotemporal trajectory planning approaches.*

Moreover, planning tunability and stability are two important properties. Tunability matters since APV users may have differing evaluations of comfort under different situations. An easy-to-tune planner exhibits certain "continuous"[1]characteristics, which enables an engineer or even a user to easily modify the planning outcome for different scenarios and/or different subjective (user-specific) preferences. Stability of the planner characterizes the planner's dynamics in event of environmental changes. It is a critical property since jumpy plans not only cause uncomfortable driving, but also may cause damage to controller due to excessive jerkiness, even loss of vehicle control.

Prior sampling-based on-road path/trajectory planning schemes were typically formulated to find a minimum cost route on the search graph, where the cost functions were defined as a weighted sum of multiple "incommensurable" (cost) features. We demonstrate with simple point-mass holonomic path planning examples that it is possible for such a formulation to exhibit discontinuities w.r.t. parameter weighting (lack of tunability, Fig. 1) or environmental changes (lack of stability, Fig. 2), unless certain cost term constraints are met. Also, the cost constraints to satisfy each continuity can conflict with each other ($\frac{1}{3} < c < \frac{2}{3}$ and $c \geq 1$), *i.e.*, they cannot be achieved simultaneously. The illustrated discontinuity and conflict effects may be further worsened by the nonholonomic nature of the APV and/or more complicated cost terms. A "discontinuous" planner typically implies that some part of the feasible solution space is unjustifiably inaccessible due to poor design, and above all, is potentially dangerous in autonomous driving.

On the other hand, the optimization-based elastic-band path planning method [5] demonstrates both continuities, hence better tunability and stability. The shape of path plan is determined by two artificial forces — internal force to remove band slackness and repulsive force generated by an

[†] Electrical & Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA, USA 15213 tianyu@cmu.edu

[‡] Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA 15213

[§] Research & Development, General Motors, Warren, MI, USA 48093

[1]Gradually changing. But we use quote marks to emphasize that the planning is typically formulated as a deterministic-Markov-Decision-Process (dMDP) or graph search problem, which is discrete by nature.

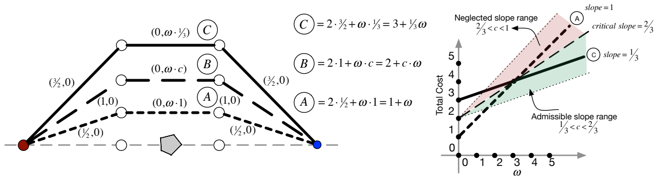


Fig. 1: Illustration of discontinuous plan change w.r.t. weight change. A holonomic path plan is modeled as a graph search problem. Each edge has two cost terms, swerve C_1 and obstacle repellence C_2, where C_2 is the product of weight ω and a cost value, represented by (C_1, C_2). The optimal plan (route) is the one with the minimum total cost, which is the sum of both cost terms over all the edges on a route. Three routes (A,B,C) are implied in this example. Route B can become admissible or neglected, depending on the cost value c which affects C_2 of the central edge. As shown in the right figure, there exists a critical value of c. The route B is only admissible when $\frac{1}{3} < c < \frac{2}{3}$, and inaccessible when c is $\frac{2}{3} < c < 1$. No such admissibility has been guaranteed in the cost definition of prior planner designs.

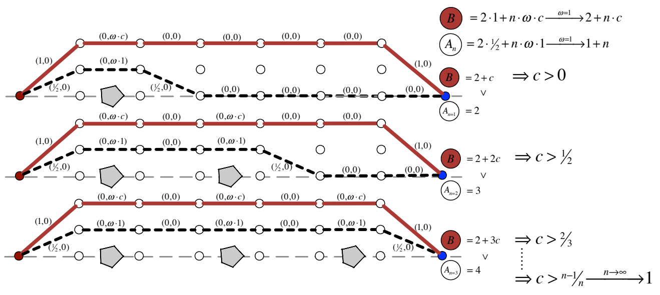


Fig. 2: Illustration of discontinuous plan change w.r.t. environment change. The formulation is the same as in Fig. 1. Environment change is modeled by adding new obstacles, which is similar to the most common scenario of on-road navigation. We typically want the planner to be tuned to avoid objects in a similar fashions regardless of the number of objects. In the three scenarios shown, we expect the plan to be route $A_{n=1}$, $A_{n=2}$ and $A_{n=3}$ respectively. That is equivalent to saying that the route Bs (red) should always be inferior to the route As by having greater cumulative costs, which implies $c > 0$, $c > \frac{1}{2}$ and $c > \frac{2}{3}$ respectively. After generalization, c must satisfy $c > \frac{n-1}{n}$. Take $n \to \infty$, $c \geq 1$.

object in the environment. A force-balanced position is found for each sampled point through an iterative optimization process. As long as the force fields exhibit certain continuous characteristics, changing the weights of each individual force component, or force field due to changing environment, can incrementally "nudge" the equilibrium positions. However, the elastic-band approach has the drawbacks of having a non-deterministic runtime and requiring a collision-free initial path. The findings above lead to our second motivation:

M2: Create an easy-to-tune trajectory planner with the aforementioned continuities and pre-determined runtime.

II. RELATED WORK

The main goal of motion planning (MP) is to find a sequence of motions to move a robot from point A to point B in an environment populated with other objects. There are two fundamental planning schemes. *Discrete schemes* treat planning as a graph search problem, and apply search algorithms such as breadth-first search (on a cyclic graph), dynamic programming (on a directed acyclic graph) or A* [6] (on a cyclic graph with an admissible and consistent heuristic term) and its variants, such as D* [7], to quickly solve the problem. Many graph-search-based planning algorithms can also be viewed as a deterministic Markov Decision Process (dMDP) if the search space is a directed acyclic graph. On the other hand, *continuous schemes* treat planning as a numerical optimization problem. For example, Quinlan [5] and Ratliff [8] proposed an elastic band planner and CHOMP. Both used gradient-based optimization techniques to generate smooth paths for mobile robots or high-DOF manipulators.

When applied to nonholonomic mobile platforms, sampling-based (discrete) approaches have been most popular, since they most easily encode kinematic constraints while constructing the search space. They also more easily allow pre-determination of the runtime by controlling the size of the discrete search space via modifying the sampling resolution, such as the state-lattice planner by Pivtoraiko [9].

The on-road navigation components in Boss [10] and Junior [1] sampled a few dynamically feasible trajectories laterally biased around the centerline up to a very short look-ahead horizon and picked the best collision-free one, but could only handle limited scenarios. Ziegler [3] and McNaughton [4] proposed sampling spatiotemporally. A directed acyclic graph was created by building and evaluating trajectories that connect multi-layer terminal states and varying accelerations. They demonstrated stronger planning capabilities in some on-road scenarios. However, the computational overhead became significantly greater, since it was challenging to retain high sampling resolution in the much higher-dimensional search space.

Gu [11] designed several cost function feature terms for on-road planning, including trajectory length, deviation from centerline, longitudinal speed, longitudinal acceleration and lateral acceleration for in-lane maneuvers. McNaughton [4] and Xu [12] both designed more than ten incommensurable feature terms to shape an overall cost for on-road driving trajectory evaluation. Abbeel [13] optimized a cost composed of trajectory length, number of motion direction changes, smoothness and distance to objects for parking lot maneuvers. The overall cost has been widely defined as a weighted linear sum of all feature terms, with the optimal plan selected such that a minimum overall cost can be achieved in the search space. Empirically, the lack of an intuitive interpretation of the sum of multiple incommensurable feature terms makes the manual tuning process difficult. While there has been work using inverse reinforcement learning (IRL) techniques to find the weights automatically [13], the design of such an overall cost can easily lead to discontinuities in the search space and the lack of tunability.

The remainder of this paper is structured as follows. Section III explains the overall planning sequence, and the proposed decoupled trajectory planner. Section IV evaluates the proposed planner in several challenging on-road scenarios, discusses the computational overhead and demonstrates the tunability and stability of the proposed path planner. Section V summarizes this paper and points to future work.

III. PLANNING

An important characteristic of on-road motion planning is that the possible (socially accepted) maneuvers are limited. Moreover, several maneuvers may be feasible at the same time. For example, we can avoid a moving bicyclist on the roadside either by distance keeping, lane changing or swerving. We assume that these types of decisions are made by the upper-level behavioral component and are accessible.

Per motivation *M1*, we adopt a decoupled space-time trajectory planning approach (Fig. 3). The separation of path and speed planning (local spatiotemporal planning) creates two easily-tractable planning problems.

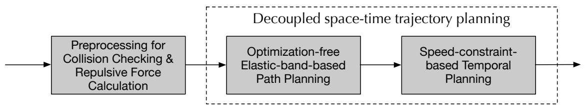

Decoupled space-time trajectory planning
Preprocessing for Collision Checking & Repulsive Force Calculation
Optimization-free Elastic-band-based Path Planning
Speed-constraint-based Temporal Planning

Fig. 3: Diagram of the overall planning sequence. The preprocessing module prepares occupancy grid maps. The decoupled space-time trajectory planning performs path planning with an elastic-band-inspired approach, and speed planning with local spatiotemporal planning after generating suggested speed profiles. The path planning generates geometric curves that avoid static objects and certain moving objects (that require swerving); the speed planning generates a trajectory plan based on the path plan and avoids the moving objects by forward-simulating their future movements.

A. Pre-processing

The planning module takes environment input from perception modules to perform trajectory evaluations (collision-checking and cost function calculation). Occupancy grid (OG) maps are used to model the static environment, while polygonal and circular shapes are used to represent moving targets. Pre-processing is needed to create necessary maps for the proposed path planner.

Let the raw obstacle OG be G'_O. For on-road driving, it is important to filter irrelevant objects out from G'_O to prevent them from affecting the planning outcomes: e.g., obstacles on the sidewalk should never cause any nudging behavior. We use the following method to keep only the on-road objects:

- build a Boolean centerline OG map G_{CL} whose only "ON" cells are the piecewise linear approximations of the lane centerlines.
- build a distance-transformed map of G_{CL}: G^D_{CL}.
- create the mask OG G_{Mask} by thresholding the values of each cell in G^D_{CL}, such that $d \leq \frac{LaneWidth}{2}$.
- the filtered obstacle OG $G_O = G'_O$ **AND** G_{Mask}.

In the decoupled space-time planning formulation, to allow the path planning to account for the moving objects, we create a sweep-volume for each object of interest up to a temporal prediction and overlay it on the filtered obstacle OG G_O. We further generate a distance field OG G^D_O that can be used to infer a direction of the repulsive force.

To efficiently perform collision checking, vehicle kernel OG $G_V^{[\theta_i,\theta_{i+1}]}$ is created for each sampled vehicle orientation range $[\theta_i, \theta_{i+1}]$. Sliced configuration OG maps $G_C^{[\theta_i,\theta_{i+1}]}$ are then created by convolving the corresponding vehicle kernel onto the filtered obstacle OG G_O. FFT-based algorithms [14] have been developed to efficiently perform convolution.

B. Decoupled Space-Time Trajectory Planning

The decoupled space-time trajectory planning module first performs path planning, then speed planning. We investigate novel planning formulations for both problems with focus on tunability and stability.

Path planning: We propose an optimization-free elastic-band(EB)-based approach for path planning with the benefits described in Section I. To address the disadvantages described in the same section, we construct a discrete search space by building the so-called elastic nodes and edges (Fig. 4). An elastic node is defined by augmenting the spatial node with an IN edge and an OUT edge that connect the neighboring spatial nodes, and two elastic nodes are connected by an elastic edge. A graph search problem is then formulated to approximate the continuous EB problem.

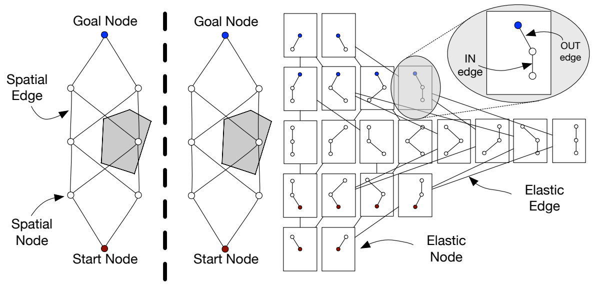


Fig. 4: Illustration of elastic nodes and edges. The left figure shows the sampling pattern and the spatial nodes and edges. The right figure highlights the elastic nodes and edges, where the elastic node extends the original node with one IN edge and one OUT edge, and one elastic edge connects two elastic nodes.

The original EB planner only modeled two forces: a repulsive force generated by the external obstacle and contractive forces generated by the neighboring points. For on-road planning, we introduce another "attract" force modeling the desirability of getting back to the road centerline. Note that the artificial forces are applied to elastic nodes, as shown in Fig. 5. The total force of each elastic node is the weighted sum of all force vectors:

$$\mathscr{F} = \begin{bmatrix} \omega_{attract} \\ \omega_{contract} \\ \omega_{repel} \end{bmatrix}^T \cdot \begin{bmatrix} F_{attract} \\ F^{left}_{contract} + F^{right}_{contract} \\ F_{repel} \end{bmatrix} \tag{1}$$

where weights $\Omega = [\omega_{attract}, \omega_{contract}, \omega_{repel}]$ are tunable parameters. The path planner then aims to find the sequence of elastic nodes $\{\tau_i^*\}$ such that:

$$\{\tau_i^*\} = \underset{\tau_i \in \Gamma_i,\, i=1,2,\ldots}{argmin} \sum_i |\mathscr{F}(\tau_i) \cdot N(\tau_i)^T| \tag{2}$$

where $N(\tau_i)$ represents the lateral (normal) direction of τ_i.

Per motivation *M2*, this cost is composed of commensurable features (force components). In fact, it also inherits the core optimization idea of the original EB planner. By minimizing this value (the equilibrium positions have the minimum 0), the routine finds a best estimate of the EB method subject to the constraints of its discrete formulation.

As a graph search formulation with predefined resolution, the proposed planner can also guarantee deterministic runtime. Spatial edges are evaluated for collision when con-

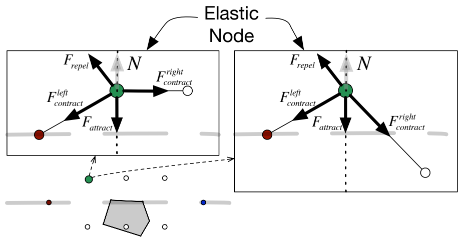


Fig. 5: Illustration of artificial forces on two elastic nodes. The two rectangles represent the two elastic nodes created from one of the spatial nodes (green). Three artificial forces are created: center-attractive $F_{attract}$, contractive $F_{contract}$ and repulsive F_{repel}. The lateral (normal) direction is defined by N. For both elastic nodes, the repulsive and center-attractive forces are the same since they are only related to the spatial node. However, the contractive forces are not the same since the IN/OUT edges are different.

structing the elastic nodes, which removes infeasible edges when constructing the search graph.

Now that the plan is a sequence of nodes, the line segments between neighboring nodes are guaranteed to be collision-free. However, this piecewise-linear path is non-smooth in curvature, which is inadmissible for speed planning. B-splines [5] have been used to generate smooth paths, but do not easily guarantee kinematic feasibility. Moreover, smoothing may cause the collision-free guarantee to be voided. We use a coarse model of the vehicle similar to [15] (pure-pursuit controller [16] and kinematic vehicle model) to generate kinematically feasible paths by forward evaluation. By sampling a set of pure-pursuit controllers with different look-ahead parameters, we can easily manipulate how closely the forward-evaluated path tracks the original piecewise-linear path. The maximum lateral deviations after evaluations are controlled by a threshold, so that they still preserve the essence of the path plan. Collision checks on the forward evaluated paths are efficiently re-examined with the aforementioned sliced configuration OG maps.

Speed planning: The continuous path is made up of a configuration functional $q(s)$ parameterized by the arc-length s. The goal of speed planning is to determine how s is parameterized by time t, which is equivalent to finding the scalar speed function v parameterized by s, *i.e.*, $v(s)$. We propose a speed-constraint-based temporal planning approach. It has two components: suggested speed profile generation with constraints, and local spatiotemporal trajectory search.

The suggested speed profile $\hat{v}(s)$ is generated similarly to that in [5] by finding a time-optimal speed plan under several speed/acceleration constraints for $\forall s \in [0, s^{max}]$, where s^{max} is the total length of the path plan. We define the following constraint operators: speed limit $\wp_{limit}$, obstacle proximity $\wp_{obstacle}$, lateral acceleration $\wp_{LatAcc}$, and longitudinal acceleration/deceleration $\wp_{LonAcc}$, such that:

$$\begin{aligned}
\wp_{limit}(s, v^{Max}) &: \hat{v}(s) \le v^{Max} \\
\wp_{obstacle}(s, q(s), \lambda_{obstacle}) &: \frac{\hat{v}(s)}{d(q)} \le \lambda_{obstacle} \\
\wp_{LatAcc}(s, \lambda_{LatAcc}) &: \kappa(q)\hat{v}^2(s) \le \lambda_{LatAcc} \\
\wp_{LonAcc}(s, \lambda_{LonDec}, \lambda_{LonAcc}) &: \lambda_{LonDec} \le \dot{\hat{v}}(s) \le \lambda_{LonAcc}
\end{aligned}$$

where d and κ are respectively the distance to the nearest obstacle and curvature of configuration q, v^{Max} is the upper bound of the speed planning, and $\Lambda = [\lambda_{obstacle}, \lambda_{LatAcc}, \lambda_{LonDec}, \lambda_{LonAcc}]$ is the vector of tunable constraint parameters. A real-time implementation is described in [2]. Fig. 6 demonstrates a few examples.

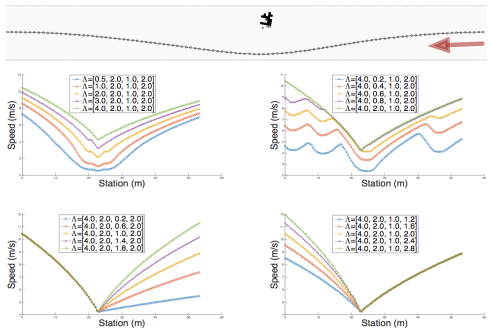


Fig. 6: The generation of suggested speed profile. The top figure shows a path plan on which the speed planning is performed. The bottom four plots show the different speed plans as each of the speed parameters changes (the other three parameters remain fixed). The plots demonstrate both flexible and intuitive tuning capability of the constraint-based approach.

Compared with [5], [2], the novelties are two-fold. First, an obstacle-related speed constraint is designed. This is useful to limit the speed when the planned path has to pass close by the obstacles, e.g. narrow passage. Second, the tunable constraint vector Λ is not determined by the APV's actuation limits, but by user preference.

Given the path plan $q(s)$ and the suggested speed profile $\hat{v}(s)$, a spatiotemporal dynamically feasible trajectory search (Fig. 7) is performed. It is guaranteed collision-free and evaluates the trajectory plan for moving objects. Sampled trajectories are created by connecting from the current vehicle state (pose, curvature, speed) to all feasible terminal states at the next station via parametric path spirals and linear speed profiles:

$$\begin{aligned}
\rho &: \kappa(s) = p_0 + p_1 \cdot s + p_2 \cdot s^2 + p_3 \cdot s^3 \\
\nu &: v(t) = q_0 + q_1 \cdot t
\end{aligned}$$

where $\kappa(s)$ represents a cubic path spiral whose curvature κ is parameterized by arc-length s. $v(t)$ represents a linear speed profile whose speed v is parameterized by time t.

Trajectories must be evaluated against all static[2] and moving objects (by forward simulating their future movements) to guarantee being collision-free. All feasible trajectories are evaluated at uniformly spaced evaluation poses:

$$\{\xi_1^*, ..., \xi_n^*\} = \underset{\text{All } \{\xi\}}{argmin} \sum_{i=1}^{n} c_{\Delta v}(\xi_i) + \omega \cdot c_d(\xi_i)$$

[2]We need to re-check static objects for collision since the parametric trajectory is newly generated.

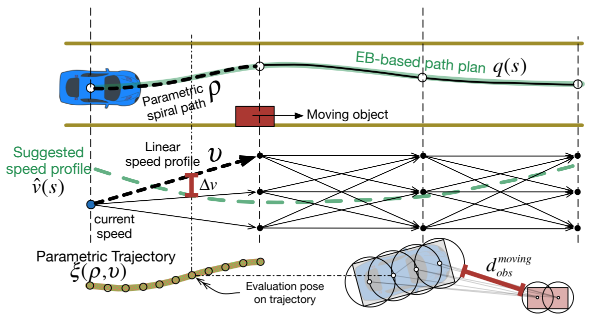

EB-based path plan $q(s)$
Parametric spiral path ρ
Moving object
Suggested speed profile $\hat{v}(s)$
Linear speed profile υ
Δv
current speed
Parametric Trajectory $\xi(\rho,\upsilon)$
Evaluation pose on trajectory
d_{obs}^{moving}

Fig. 7: Illustration of local spatiotemporal planning. The top plot demonstrates sampling parametric spiral paths ρ on three longitudinal look-ahead layers based on the elastic-band path plan. The middle plot illustrates the sampling of linear speed profiles ν between longitudinal layers. The bottom plot shows the calculation of two feature (cost) values, deviation from the suggested speed profile Δv and distance to moving object d_{obs}^{moving}, that are used in trajectory ξ evaluation.

$$c_{\Delta v}(\xi_i) = \sum_{j=0,1,2,\ldots} |v(s_j) - \hat{v}(s_j)|^2$$

$$c_d(\xi_i) = \sum_{j=0,1,2,\ldots} d_{obs}^{moving}(q(s_j))$$

where $\xi_1^*, ..., \xi_n^*$ are the optimal trajectory pieces of each sampled layer. $c_{\Delta v}$ penalizes deviation from the suggested speed profile, and c_d penalizes excessive proximity to moving objects. Note that the static objects are not involved in the cost function design, since the trajectories are sampled based on the EB-path plan, which already accounts for static obstacles with desirable swerve maneuvers.

Careful selection of weight ω becomes crucial. When commanded to distance-keep, the vehicle ends up following the leading vehicle too closely if ω is too small. When commanded to overtake by swerving, the trajectory plan can be too conservative to accomplish if ω is too big, since the cost penalizes proximity too much, which prevents catching up with the moving objects. While it empirically works well (see Section IV), the two cost terms are "incommensurable" by nature, and the above-mentioned "discontinuity" effect potentially exists. More work needs be done to investigate principled cost function design methods in such cases, where incommensurable costs are needed to account for distinct factors in a single planning formulation.

IV. RESULTS

A. Evaluation in Challenging Dynamic Environments

The proposed planner was implemented in the autonomous driving test platform developed at CMU [17]. To highlight the capability of the proposed planning approach, a challenging on-road situation is demonstrated. In this example, three types of common on-road objects, static blockages, pedestrians and bicyclists are encountered. To successfully handle this scenario, the planner must generate a swerve-to-avoid maneuver for static obstacles, and slow-down / distance keeping maneuver and (on-demand) swerve-to-overtake maneuvers for moving objects. Fig. 8 depicts the trajectory plan for the entire planning sequence.

B. Computation

One of the main benefits of the proposed approach is its computational efficiency. We are able to achieve a 10Hz update rate on a single Intel Core-2 Quad Processor used by the autonomous driving test platform.

Pre-processing of occupancy grid (OG) maps is the most expensive module for the FFT operations on shape convolution and the distance transform operation to create a distance field. They are implemented as a separate post-perception task, and well-optimized algorithms exist for these operations. The main steps of planning include EB path planning (A), controller-based kinematically-feasible path smoothing (B), constraint-based suggested speed profile generation (C) and local spatiotemporal trajectory search (D).

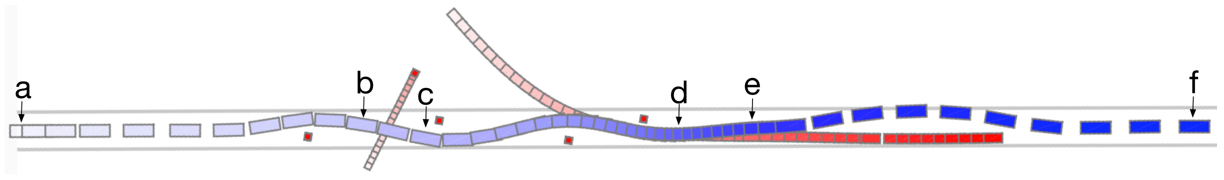
- A: The most time-consuming operation is the evaluation of spatial edges, since multiple poses on each edge must be checked for collision. Our implementation is fast, since only a single query on the corresponding sliced configuration OG map is needed per pose, rather than making queries for all grids on the obstacle OG map taken by the host vehicle.
- B: Multiple controllers are used to forward-evaluate the vehicle tracking raw EB path planning output and check for collision. A smoothed path can be obtained at the same time. Similar to A, a fast collision checking routine accelerates the evaluation process.
- C: A single scan on the generated smooth path plan is needed to apply the speed-limit, obstacle proximity and lateral acceleration constraints. Two scans (forward+backward) are needed for longitudinal acceleration/deceleration constraints. As discussed in [2], the run-time complexity is $O(N)$.
- D: The spatiotemporal trajectory search space is subject to exponential blowup [4]. The size is determined by the number of longitudinal lookahead layers n and the number of sampled terminal speeds m, *i.e.*, $O(m^n)$. Per *M1*, we limit our search space to explore "locally" at a few ($n = 3$) longitudinal look-aheads at a fine speed sampling resolution ($v_{max} = 20m/s, \Delta v = 2m/s$, so $m = 11$). The total number of trajectories to evaluate is hence on the scale of 10^3, which is tractable for CPU processors, rather than on the scale of 10^6 [3], [4].

C. On the tunability and stability of path planning

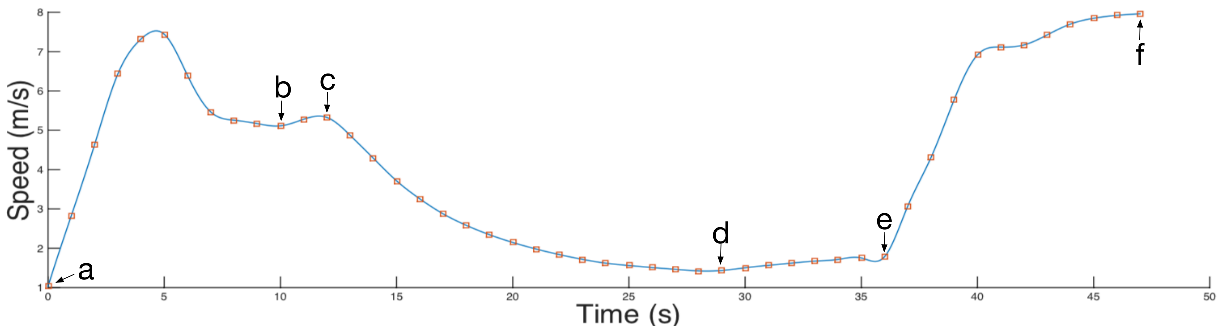
The path plan directly affects speed planning (through curvature). More importantly, path plan is directly associated with lateral control (steering command of APV), hence crucial.

We demonstrate the tunability and stability of the EB path planning by highlighting the two continuity characteristics (section I) of the proposed path planner. A prior path planner with cumulative incommensurable cost terms is implemented and compared with.

The choice of the baseline planner requires careful thought. As discussed, many prior planners perform spatiotemporal planning, which includes cost terms irrelevant to path planning. Even for the path-related terms, certain



(a) World plot



(b) Speed plot

Fig. 8: Planning results for the proposed test case. Plot (a) shows the world plot of the host vehicle and the environment. Plot (b) shows the speed plot of the host vehicle. From a to b, the host vehicle performs slow-down to avoid a pedestrian crossing the road between two static obstacles. From c to d, the host vehicle distance-keeps to a leading bicyclist. From e to f, the host vehicle performs an overtaking maneuver (upon a trigger) to circumvent the bicyclist. All maneuvers avoid static obstacles simultaneously.

terms do not exist in the EB-planning formulation, e.g., path curvature. Hence, we need to choose a planner that has the same number of feature (cost) terms of similar types.

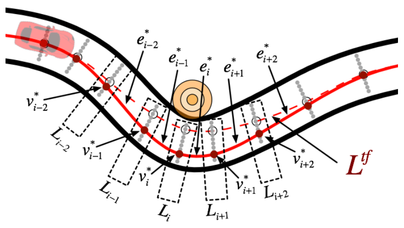


Fig. 9: Baseline path planner. An array of vertices consisting of multiple layers $\{L_i\}$ is generated based on L^{tf}. The longitudinal sampling distance between two neighboring layers is ΔS, and lateral uniformly sampled vertices on each layer are spaced by ΔL. Edge $\boldsymbol{e}_i$ connects a vertex $\boldsymbol{v_i}$ in layer L_i to a vertex $\boldsymbol{v_{i+1}}$ in layer L_{i+1}. The lateral shift of edge $\boldsymbol{e_i}$ and the lateral offset of vertex v_{i+1} from the centerline are multiples of ΔL.

The baseline planner [18] samples the spatial nodes and edges similarly to the proposed as in Fig. 9; however, it doesn't further create elastic nodes and edges. The optimal sequence of vertices $\{\boldsymbol{v_i^*}\}$ is defined below. Let $\Omega' = [\omega_{deviation}, \omega_{swerve}, \omega_{obstacle}]$ represent the tunable parameters of the baseline planner. Now the baseline and proposed path planners each have three tunable parameters Ω' and Ω.

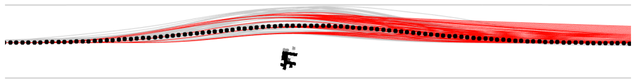
$$\{\boldsymbol{v_i^*}\} = \underset{\{\boldsymbol{v_i} \in L_i\}}{\operatorname{argmin}} \sum_{i=0}^{N-1} C_{swerve} + C_{deviation} + C_{obstacle}$$

where

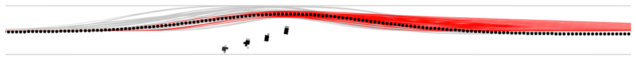
$$C_{swerve} = \omega_{swerve} \cdot e^{\frac{|l(\boldsymbol{e_i})|}{\Delta L}}$$

$$C_{deviation} = \omega_{deviation} \cdot \frac{|l(\boldsymbol{v_{i+1}}) - l^{tf}(\boldsymbol{v_{i+1}})|}{\Delta L}$$

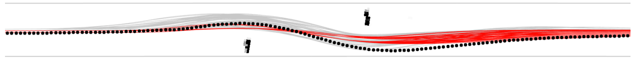
$$C_{obstacle} = \omega_{deviation} \cdot \begin{cases} 1/d & \text{d} > 0 \\ \infty & \text{d} = 0 \end{cases}$$



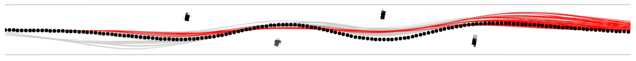
(a) Scenario I



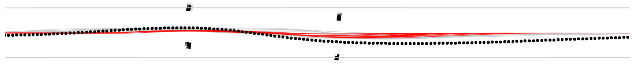
(b) Scenario II



(c) Scenario III



(d) Scenario IV

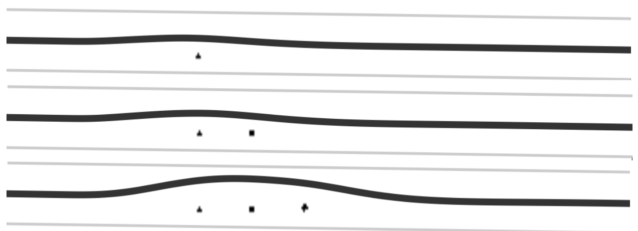


(e) Scenario V

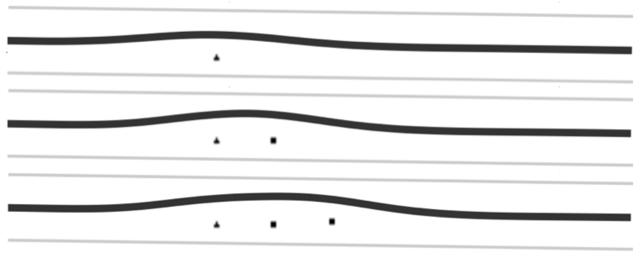
Fig. 10: Comparison of continuity w.r.t. weight change (tunability). Five obstacle avoidance scenarios in a single lane are set up. The car moves in the direction from left to right. The thick grey parallel lines mark the lane boundary. The black cells represent the static obstacles in the occupancy grid map. The thin grey curves show the outcomes (961 trajectories) of the proposed planner in sampled parameter space. The thin red curves show the outcomes (961 trajectories) of the baseline planner in sampled parameter space.

For continuity w.r.t. weights changes, the baseline planner sometimes yields split clustered planning outcomes, while the proposed planner generates well-distributed candidate paths, such as in Fig. 10a. Meanwhile, parts of the baseline planning results tend to unjustifiably converge to the centerline in extra-long paths, while the proposed planner yields expected symmetric plans, such as Fig. 10a to 10d. For tightly constrained scenarios like Fig. 10c, 10d and 10e, the proposed method consistently demonstrates better (wider region) of choices.

For continuity w.r.t. environment changes, we compare the planning outputs of the baseline and proposed path planner in a situation where obstacle gradually appear as the vehicle moves closer the obstacle (Fig. 11). In Fig. 11a, the baseline planner yields a jumpy plan from the detection of the second obstacle to the detection of the third obstacle. This generally causes planning instability which results in excessive jerk, even dangerous behavior. In Fig. 11b, the proposed planner can generate incremental nudging behaviors as expected, which in general results in smoother and consistent path plan.



(a) Baseline planner performance



(b) Optimization-free elastic-band planner performance

Fig. 11: Comparison of continuity w.r.t. environment change (stability). We simulate a receding detection of static objects, which is a common case for on-road autonomous driving due to limited sensing range.

From these experiments and our extensive on-vehicle autonomous driving tests, a clear superior in tunability and stability has been evaluated of the proposed path planner.

V. CONCLUSIONS

In this paper, we proposed a novel decoupled space-time trajectory planner with an emphasis on improving computational efficiency and planning tunability/stability. Optimization-free elastic-band path planning is performed to account for static objects and selected moving objects for swerve maneuvers. Constraint-based local spatiotemporal trajectory planing mainly follows the path plan, but modulates the speed plan to further accounts for constraint factors and moving objects. It eventually generates a dynamically feasible trajectory for vehicle control. Compared with spatiotemporal trajectory planning approaches which suffer from sub-optimality due to *the curse of dimensionality*, our decoupled approach, while not spatiotemporally complete, allows finer sampling on a lower-dimensional search space, and overall better tunability and stability.

Another contribution of this paper is to identify the continuity problems (discontinuity w.r.t. weight/environment change) with previous cost-based lattice planners that caused tunability and stability issues. We further evaluate the tunability and stability of the proposed path planner design, and demonstrate its superiority over a prior on-road path planning algorithm for APV.

For future work, the planner should account for static/moving objects of varying types, which is important for realistic on-road autonomous driving. We will also explore on-line learning methods to allow designers/passengers to customize the autonomous driving behavior in real-time.

REFERENCES

[1] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, and B. Huhnke, “Junior: The stanford entry in the urban challenge,” *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[2] T. Gu, J. Snider, J. M. Dolan, and J.-W. Lee, “Focused Trajectory Planning for autonomous on-road driving,” *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pp. 547–552, 2013.

[3] J. Ziegler and C. Stiller, “Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios,” *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 1879–1884, 2009.

[4] M. McNaughton, C. Urmson, and J. M. Dolan, “Motion planning for autonomous driving with a conformal spatiotemporal lattice,” *… and Automation (ICRA …*, 2011.

[5] S. Quinlan, “Real-time modification of collision-free paths,” tech. rep., Stanford, CA, USA, 1995.

[6] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, pp. 100–107, July 1968.

[7] D. Ferguson and A. Stentz, “Field D*: An Interpolation-based Path Planner and Replanner,” pp. 1–10, Aug. 2005.

[8] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, “CHOMP: Gradient optimization techniques for efficient motion planning,” in *Robotics and Automation, 2009. ICRA ’09. IEEE International Conference on*, pp. 489–494, IEEE, 2009.

[9] M. Pivtoraiko and A. Kelly, “Efficient constrained path planning via search in state lattices,” *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2005.

[10] C. Urmson, C. Baker, J. Dolan, P. Rybski, B. Salesky, W. Whittaker, D. Ferguson, and M. Darms, “Autonomous driving in traffic: Boss and the urban challenge,” *AI magazine*, vol. 30, no. 2, p. 17, 2009.

[11] T. Gu and J. M. Dolan, “On-Road Motion Planning for Autonomous Vehicles,” in *Intelligent Robotics and Applications*, pp. 588–597, Berlin, Heidelberg: Springer Berlin Heidelberg, Jan. 2012.

[12] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, “A real-time motion planner with trajectory optimization for autonomous vehicles,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 2061–2067, 2012.

[13] P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, “Apprenticeship learning for motion planning with application to parking lot navigation,” in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1083–1090, IEEE, 2008.

[14] L. Kavraki, “Computation of configuration-space obstacles using the fast Fourier transform,” in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pp. 255–261, IEEE Comput. Soc. Press, 1993.

[15] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. How, “Real-Time Motion Planning With Applications to Autonomous Urban Driving,” *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY*, vol. 17, pp. 1105–1118, Aug. 2009.

[16] J. M. Snider, “Automatic steering methods for autonomous automobile path tracking,” *Robotics Institute, Carnegie Mellon University, technical CMU-RI-TR-09-08*, 2009.

[17] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, “Towards a viable autonomous driving research platform,” *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 763–770, 2013.

[18] T. Gu, J. Dolan, and J.-W. Lee, “On-Road Trajectory Planning for General Autonomous Driving with Enhanced Tunability,” *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, vol. 25, pp. 325–345, Feb. 2014.