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 propose a computationally 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 planning and speed-constraint-based temporal planning routines with pre-determined runtime.
- Identification of continuity problems with previous cost-based planners that cause 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. In order to handle moving objects, prior on-road planning methods [2], [3] 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” 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 (the planner’s dynamics in event of environmental changes) is another critical property, since jumpy plans will cause extra jerkiness in steering/braking, which leads to uncomfortable driving and unnecessary wearing of the actuation systems.

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 can 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 [4] 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 object in the environment. A force-balanced position is found

† 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

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.
planning algorithms can also be viewed as a deterministic Markov Decision Process (dMDP) if the search space is a directed acyclic graph. continuous schemes treat planning as a numerical optimization problem. Quinlan [4] and Ratliff [6] proposed an elastic band planner and Covariant Hamiltonian Optimization for Motion Planning (CHOMP). Both used gradient-based optimization techniques to generate smooth paths for mobile robots.

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 [7].

The on-road navigation components in Boss [8] 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 [2] and McNaughton [3] 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. Chen [9] proposed a two-step planning method with circle-based exploration sampling and orientation-aware heuristic search sampling. But it was pure path planning, hence more suitable for parking maneuver for its nondeterministic nature.

The overall objective has been widely defined as a cumulative cost with weighted feature terms summed up. McNaughton [3] and Xu [10] both designed more than ten incommensurable feature terms to shape an overall cost for on-road driving trajectory evaluation. Abbeel [11] optimized a cost composed of trajectory length, number of motion direction changes, smoothness and distance to objects for parking lot maneuvers. 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 [11], the design of such an overall cost can easily lead to discontinuities in the search space and the lack of tunability. To alleviate such problems, we have proposed a two-step planning method in our earlier paper [12] by decomposing the overall planning task into deliberative (reference) planning and reactive (trajectory) planning. Li [13] adopted a similar approach, but proposed to use support vector machine (SVM) for reference planning. But neither could handle dynamic objects very well.

The remainder of this paper is structured as follows. Section III explains the proposed trajectory planner, followed by a thorough evaluation in Section IV. Section V concludes with remarks and future work.

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, dynamic programming or A*/D* [5], to quickly solve the problem. Many graph-search-based
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 $M_1$, 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.

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 $G'_{CL}$: $G''_{CL}$.
- create the mask OG $G_{Mask}$ by thresholding the values of each cell in $G''_{CL}$, such that $d \leq \text{LaneWidth}$.
- the filtered obstacle OG $G_O = G'_O \text{ AND } G_{Mask}$.

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 $G'_D$ 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 $G_O$.

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.

![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 temporal prediction and overlay it on the filtered obstacle OG.](image)

![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.](image)

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:

$$\mathcal{F} = \begin{bmatrix} \omega_{\text{attract}} \\ \omega_{\text{contract}} \\ \omega_{\text{repel}} \end{bmatrix}^T \cdot \begin{bmatrix} F_{\text{attract}} \\ F_{\text{contract}}^\text{left} + F_{\text{contract}}^\text{right} \\ F_{\text{repel}} \end{bmatrix}$$

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

$$\{\tau_i^*\} = \arg\min_{\tau_i \in \Gamma} \sum_i |\mathcal{F}(\tau_i) \cdot N(\tau_i)|$$

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

Per motivation $M_2$, this cost is composed of commensurable forces (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-
The continuous path is made up of a \( \mathcal{P} \) obstacle proximity where the following constraint operators: speed limit several speed/acceleration constraints for \( \forall \) to that in [4] by finding a time-optimal speed plan under constraints, and local spatiotemporal trajectory search. Two components: suggested speed profile generation with a speed-constraint-based temporal planning approach. It has a speed function \( v \) parameterized by the arc-length \( \kappa \), where \( \kappa \) are respectively the distance to the nearest obstacle and curvature of configuration \( q \), \( v_{\text{Max}} \) is the upper bound of the speed planning, and \( \Lambda = [\lambda_{\text{obstacle}}, \lambda_{\text{LatAcc}}, \lambda_{\text{LonDec}}, \lambda_{\text{LonAcc}}] \) is the vector of tunable constraint parameters. A real-time implementation is described in [12]. Fig. 6 demonstrates a few examples.

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 \( F_{\text{attract}} \), contractive \( F_{\text{contract}} \) and repulsive \( F_{\text{repel}} \). 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.

Constraining 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 [4] 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 [14] (pure-pursuit controller [15] 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.

\textbf{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 [4] by finding a time-optimal speed plan under several speed/accleration constraints for \( \forall s \in [0, s_{\text{max}}] \), where \( s_{\text{max}} \) is the total length of the path plan. We define the following constraint operators: speed limit \( \varphi_{\text{limit}} \), obstacle proximity \( \varphi_{\text{obstacle}} \), lateral acceleration \( \varphi_{\text{LatAcc}} \), and longitudinal acceleration/deceleration \( \varphi_{\text{LonAcc}} \), such that:

\begin{align*}
\varphi_{\text{limit}}(s, v_{\text{Max}}) : \hat{v}(s) &\leq v_{\text{Max}} \\
\varphi_{\text{obstacle}}(s, q(s), \lambda_{\text{obstacle}}) : \frac{\hat{v}(s)}{d(q)} &\leq \lambda_{\text{obstacle}} \\
\varphi_{\text{LatAcc}}(s, \lambda_{\text{LatAcc}}) : \kappa(q) \hat{v}^2(s) &\leq \lambda_{\text{LatAcc}} \\
\varphi_{\text{LonAcc}}(s, \lambda_{\text{LonDec}}, \lambda_{\text{LonAcc}}) : \lambda_{\text{LonDec}} \leq \hat{v}(s) &\leq \lambda_{\text{LonAcc}}
\end{align*}

(3)

\[ \rho : \kappa(s) = p_0 + p_1 \cdot s + p_2 \cdot s^2 + p_3 \cdot s^3 \]

\[ \nu : v(t) = q_0 + q_1 \cdot t \]

(4)

where \( \kappa(s) \) represents a cubic path spiral whose curvature \( \kappa \) 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:

\[ \{\xi_i^1, ..., \xi_i^n\} = \arg \min_{\text{All } \xi} \sum_{i=1}^{n} c_{\Delta v}(\xi_i) + \omega \cdot c_d(\xi_i) \]

(5)

where \( \xi_i^1, ..., \xi_i^n \) 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 tomov-
speed profile

Suggested v̇ (s)

ρ

Moving object

v

Parametric Trajectory ξ(p, 0)

Moving object

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}, that are used in trajectory ξ evaluation.

ing objects:

\[ c_{Δv}(ξ_i) = \sum_{j=0,1,2,...} (v(s_j) - ˙v(s_j))^2 \]

\[ c_d(ξ_i) = \sum_{j=0,1,2,...} d_{moving}^{obs}(q(s_j)) \]

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 to 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 [16]. 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, slow-down/distance-keeping 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 path evaluation against static obstacle OG map, since multiple poses on each edge (path) 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. The run-time complexity is \(O(N)\).
- D: The spatiotemporal trajectory search space is subject to exponential blowup [3]. 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 \(MI\), we limit our search space to explore “locally” at a few \((n = 3)\) longitudinal look-aheads at a fine speed sampling resolution \((v_{max} = 20\text{m/s}, \Delta v = 2\text{m/s})\), so \(m = 11\). The total number of trajectories to evaluate is hence on the scale of \(10^9\), which is tractable for CPU processors, rather than on the scale of \(10^6\) [2], [3].

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 for comparison.

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 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.

The baseline planner [17] 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 \( \{v^*_i\} \) is defined below. Let \( \Omega' = \{\omega_{\text{deviation}}, \omega_{\text{swerve}}, \omega_{\text{obstacle}}\} \) represent the tunable parameters of the baseline planner. Now the baseline and proposed path planners each have three tunable parameters \( \Omega' \) and \( \Omega \).

\[
\{v^*_i\} = \arg\min_{\{v_i \in L_i\}} \sum_{i=0}^{N-1} C_{\text{swerve}} + C_{\text{deviation}} + C_{\text{obstacle}}
\]

where

\[
C_{\text{swerve}} = \omega_{\text{swerve}} \cdot e^{\frac{||\epsilon_i||}{\Delta L}}
\]

\[
C_{\text{deviation}} = \omega_{\text{deviation}} \cdot \frac{d|f(v_{i+1}) - f(v_i)|}{\Delta L}
\]

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

We investigate the continuity w.r.t. weight changes by comparing the planning outcomes with a great set of parameter samples. Only two out of the three parameters in \( \Omega' \) and \( \Omega \) each are independent, since only relative scale matters. Now we have a two-dimensional (continuous) parameter space. We discretize the parameter space, and enumerate all possible parameter combinations. Both the parameter space region and the discretization resolution have to be manually determined. Two empirical guidelines are: avoid too-small parameter space, which results in obviously insufficient planning coverage, and avoid too-coarse discretization resolution, which may cause jumpy plans. In our setup, we perform uniform discretization 0 : 1 : 30 of the two independent parameters \( \omega_{\text{swerve}}, \omega_{\text{obstacle}} \) in \( \Omega' \), and 0 : 0.5 : 15 for the two independent parameters \( \omega_{\text{contract}}, \omega_{\text{repel}} \) in \( \Omega \). As a
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.

In 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**


