# Runtime-Bounded Tunable Motion Planning for Autonomous Driving

Tianyu Gu[1], John M. Dolan[2] and Jin-Woo Lee[3]

***Abstract*— Trajectory planning methods for on-road autonomous driving are commonly formulated to optimize a Single Objective calculated by accumulating Multiple Weighted Feature terms (SOMWF). Such formulation typically suffers from the lack of planning tunability. Two main causes are the lack of physical intuition and relative feature prioritization due to the complexity of SOMWF, especially when the number of features is big. This paper addresses this issue by proposing a framework with multiple tunable phases of planning, along with two novel techniques:**

- **Optimization-free trajectory smoothing/nudging.**
- **Sampling-based trajectory search with cascaded ranking.**

## I. Introduction

Autonomous Passenger Vehicles (APV) have demonstrated their potential in removing human errors from tedious driving and creating a safer and more efficient transportation system. The trajectory planner (TP) plays an important part in an APV system since it determines the exact motion of the host vehicle. The common goal of a TP is to find a trajectory that optimizes a single objective defined as accumulating multiple weighted feature terms, i.e., a Single-Objective-Multiple-Weighted-Feature (SOMWF) formulation.

However, the tuning of a SOMWF objective is difficult due to the lack of physical intuition as the number of features increases. It is also hard to justify a particular weighting configuration, especially when it is necessary to provide a semantic interpretation and/or explicitly specify a certain priority among features. We are motivated to address these issues with a framework of three tunable planning phases, along with two novel techniques:

- Iteration-free trajectory smoothing/nudging with augmented graph.
- Trajectory search with primitive sampling and cascaded ranking.

## II. Related Work

Trajectory planners can be roughly classified as *Optimization-based* or *Search-based*. *Optimization-based* methods iteratively deform a trajectory to optimize a cost objective function [1], [2], [3], [4], [5]. *Search-based* methods build graphs to search for a path or trajectory with application-specific sampling patterns. Based on the nature of sampling, planners can be further classified as *randomized* or *predefined*. *Randomized* methods explore the environment in a stochastic fashion, such as the Probabilistic Road-Map (PRM) [6], [7] and Rapidly-exploring Random Tree (RRT) [8], [7] algorithms. *Predefined* methods explore in fixed patterns, such as local Dynamic Window Approach (DWA) [9] or a state-lattice [10]. Once the graphs are constructed, heuristic search algorithms like A*/D* [11], [12] can be used to retrieve the optimal trajectory. As an exception, RRT-type algorithms tend to combine the graph construction and searching process.

The two planning methodologies have been applied in autonomous driving, Tran [13] and Kelly [14] both used *optimization-based* trajectory planners for a race car by adapting sequential convex programming to achieve time-optimality. Ziegler [4] developed an *optimization-based* trajectory planner by optimizing a five-feature objective function subject to trajectories' internal and external constraints. Li [15] proposed a planner for on-road navigation using support vector machine. On the other hand, on-road trajectory planners developed by Urmson [16] and Montemerlo [17] in the 2007 DARPA Urban Challenge used *predetermined search-based* methods inspired by DWA [9] by generating short-horizon local trajectory candidates to follow lane centerlines while avoiding road-side obstacles. Recent on-road planners [18], [19], [20], [21], [22] construct a spatiotemporal search space conformal to the road structure. Chen [23] suggested a guided heuristic search method by an explore-then-search strategy to reduce planner's search space.

One major requirement of planners for autonomous driving is a guarantee on algorithm runtime. For *optimization-based* methods, the continuous nature of an optimization routine does not guarantee a time-bounded convergence to the optimal solution. For *search-based* methods, their runtime depends very much on the quality of admissible heuristics, which can be environment-dependent. We want a runtime-predetermined algorithm that would terminate within a fixed time bound, even with only a resolution-complete result[1].

Prior methods aim to minimize an optimality objective defined by summing up multiple weighted feature terms, i.e., the SOMWF-formulation. For example, Ziegler [4] optimizes an objective that takes into account centerline-offset, speed difference, acceleration, jerkiness and yaw-rate while enforcing collision-free behaviors as an external constraint. Another example by McNaughton [19] optimizes

[1] Tianyu Gu, *IEEE Student Member*, is with Department of Electrical & Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA Tianyu@cmu.edu

[2] John M. Dolan is with the Robotics Institute and the Department of Electrical & Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA jmd@cs.cmu.edu

[3] Jin-Woo Lee is with the Research & Development, General Motors, Warren, MI, USA jin-woo.lee@gm.com

[1]The algorithm is guaranteed to return an optimal result given the sampling resolution.

an objective composed of thirteen feature cost terms. The lack of physical intuition in SOMWF definition makes it difficult to tune intuitively. The existence of local minima further creates difficulties for both *optimization-based* and *search-based* solvers. For the former, local minima makes the solution highly dependent on the initial optimization conditions. For the latter, the lack of admissible heuristics makes the search process less efficient and sometimes yields non-optimal solution.

Another thread of work brings physical intuition into motion planning. The classical approaches formulate the planning problem as a dynamical system, and aim to find an equilibrium of such systems as a solution, such as in artificial force fields [24], [25], [26], [27], [28]. However, the solvers for such planners are all *optimization-based*, and have runtime issues as explained above.

In this paper, we are motivated to develop a tunable and runtime-bounded trajectory planner for autonomous driving. It is composed of multiple tunable planning phases with two novel planning techniques. In what follows, section III explains the proposed planning approach, section IV presents the evaluation results, and section V concludes with further discussion.

## III. Planning Methodology

The overall planning scheme consists of three steps:

1) Generate smooth reference trajectory to improve original lane geometry.
2) Modify the reference trajectory to avoid static obstacles.
3) Search from local parametric trajectories sampled around prior reference to avoid moving obstacles.

The first two planning steps are both formulated with the same augmented graph-based technique. It is the backbone of an iteration-free solver that is able to numerically approximate the continuous *optimization-based* methods used for traditional trajectory smoothing and nudging.

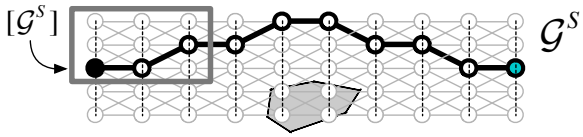
The last planning step makes use of primitive sampling-based trajectory search with a novel cascaded trajectory ranking scheme. Such a formulation makes it possible to perform prioritized trajectory selection with clear semantic interpretation of the planner's outcome.

### A. Iteration-free Solver on Augmented Graph

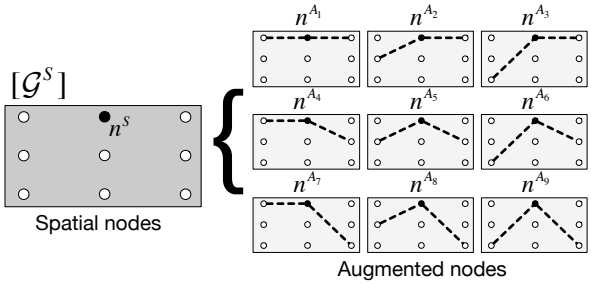
The motion planning task for on-road driving can be abstracted as planning on a corridor, naturally bounded by the lane boundaries. The configuration space in this corridor is discretized as $N$ layers of laterally sampled spatial nodes $n^S$, which formulates the spatial graph $\mathcal{G}^S$ as shown in Fig. 1a. Spatial edges $e^S$ are created by connecting nodes on neighboring layers. Each spatial edge is further evaluated to check for collision.

Rather than planning on $\mathcal{G}^S$, we augment each spatial node on $\mathcal{G}^S$ with its incoming/outgoing spatial edges to construct the augmented graph $\mathcal{G}^A$, as shown in Fig. 1b. The benefit is that each augmented node $n^A$ contains not only a spatial node, but also information about a possible combination of neighbor spatial nodes via the spatial edges. This makes it possible to calculate objectives that depend on a spatial node and its neighbors. Two augmented nodes are connected when the incoming edge of one node is the same as the outgoing edge of the other. Each augmented node is calculated with a cost $c(n^A)$. The planning problem is to find a sequence of augmented nodes $n_1^{*A} \rightarrow n_2^{*A} \rightarrow \cdots \rightarrow n_N^{*A}$ on $\mathcal{G}^A$ that minimizes the cumulative costs of each node:

$$\underset{n_1^{*A}, n_2^{*A}, \ldots, n_N^{*A}}{\arg\min} \sum_{i=1}^{N} c(n_i^A) \tag{1}$$




**(a)** Spatial graph




**(b)** Creating augmented nodes

**Fig. 1:** Spatial graph and augmented graph. Fig. 1a demonstrates the spatial graph $\mathcal{G}^S$ created from a corridor planning setup. Fig. 1b illustrates the highlighted spatial graph portion $[\mathcal{G}^S]$, and creating augmented nodes $n^{A_1}$ through $n^{A_9}$ from the spatial node $n^S$.

$\mathcal{G}^A$ is a directed acyclic graph, and therefore performing graph search with Dijkstra's algorithm is equivalent to applying dynamic programming, which is computationally efficient. Overall, this technique uses *predefined sampling-based* formulation with exhaustive search techniques, and therefore terminates in predetermined runtime.

The planning outcome is a sequence of augmented nodes on $\mathcal{G}^A$. Connecting the spatial nodes projected back to $\mathcal{G}^S$, we obtain a piecewise linear path consisting of a sequence of spatial edges. In order to further generate a smooth reference, a vehicle model consisting of a kinematic half-car model and a pure pursuit controller is used to forward-evaluate the process of tracking this piecewise-linear path and record the simulated vehicle state. After obtaining a kinematically feasible smooth path, a constraints-based speed profile generation scheme can be further used to generate a reference trajectory [29]. In the following subsection, two planners are developed with different definitions of the augmented nodes' cost.

*1) Reference path smoothing:* The original lane centerline can be further smoothed taking advantage of the full width of the lane boundaries. As shown in Fig. 2, for each augmented node $n^A$, its cost is calculated as:

$$c(n^A) = \omega_l \cdot l + \omega_{\Delta h} \cdot \Delta h$$

where $l$ is the lateral offset from the centerline of the spatial node $n_C$ corresponding to $n^A$, $\Delta h$ is the heading change of $n^A$ between the incoming edge connecting $n_L$ and outgoing edge connecting $n_R$:

$$\Delta h = \| \frac{\boldsymbol{n_C} - \boldsymbol{n_L}}{|\boldsymbol{n_C} - \boldsymbol{n_L}|} \cdot \frac{\boldsymbol{n_R} - \boldsymbol{n_C}}{|\boldsymbol{n_R} - \boldsymbol{n_C}|} \|$$

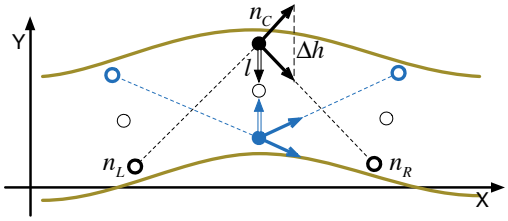



**Fig. 2:** Cost definition for path smoothing. Two augmented nodes in blue and black colors are depicted. The black node is used to demonstrate the calculation of feature $l$ and $\Delta h$.

*2) Reference nudging:* The reference path should be modified, or nudged, if static obstacles are observed to interfere with normal on-road driving. A nudging routine, inspired by the elastic band method [24], [30], can be formulated on the augmented graph. As shown in Fig. 3, three forces are calculated for each augmented (elastic) node $n^A$ in a warped road coordinate system. Their joint force is defined as the residual force $\boldsymbol{f}_{res}$ of $n^A$. Conceptually, larger $\boldsymbol{f}_{res}$ makes $n^A$ less stable as in the original elastic-band formulation:

$$\boldsymbol{f}_{res} = \omega_{\boldsymbol{f}_a} \cdot \boldsymbol{f}_a + \omega_{\boldsymbol{f}_c} \cdot \boldsymbol{f}_c + \omega_{\boldsymbol{f}_r} \cdot \boldsymbol{f}_r$$

where $\boldsymbol{f}_a$ and $\boldsymbol{f}_r$ are the attraction and repulsion forces obtained by artificial force fields as explained in [30]. $\boldsymbol{f}_c$ calculates the contraction forces of each augmented node,

$$\boldsymbol{f}_c = (\boldsymbol{n_L} + \boldsymbol{n_R} - 2 * \boldsymbol{n_C}) \cdot \rho$$

where $\rho$ is the artificial spring constant to calculate the contractive force. Therefore, the overall cost for each $n^A$ is defined as the norm of the residual force:

$$c(n^A) = \|\boldsymbol{f}_{res}\|$$

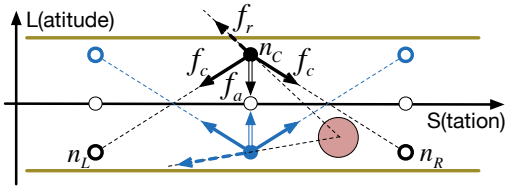



**Fig. 3:** Force/cost definition for path nudging. Two augmented nodes in blue and black colors are depicted. The black node demonstrates the calculation of forces $\boldsymbol{f}_a$, $\boldsymbol{f}_r$ and $\boldsymbol{f}_c$.

### B. Local Parametric Trajectory Planning

The local planner generates executable trajectories that control the motion of the host vehicle. The design of an executable motion primitive, sampling pattern, and a novel cascaded ranking scheme are explained.

*1) Trajectory Primitive:* a pair of cubic curvature and linear acceleration (quadratic speed profile) polynomials are used to concisely parameterize the trajectory primitive:

$$\begin{aligned} \boldsymbol{\kappa_c}(\boldsymbol{s}, \mathbf{P}) &= p_0 + p_1 \cdot s + p_2 \cdot s^2 + p_3 \cdot s^3 \\ \boldsymbol{a_c}(\boldsymbol{t}, \mathbf{Q}) &= q_0 + q_1 \cdot t \end{aligned} \tag{2}$$

where $\kappa$ is the curvature, $a$ is acceleration, curvature polynomial $\mathbf{P} = \{p_0, p_1, p_2, p_3\}$ and acceleration polynomial $\mathbf{Q} = \{q_0, q_1, q_2\}$. The primitive is interpreted as a time-stamped geometric curve. A kinematic bicycle model (3) is used to evaluate the trajectory primitive:

$$\dot{\mathcal{X}}(t; \mathbf{P}, \mathbf{Q}) := \begin{cases} \dot{x}(s) = cos\theta(s) \\ \dot{y}(s) = sin\theta(s) \\ \dot{\theta}(s) = \boldsymbol{\kappa_c}(\boldsymbol{s}, \mathbf{P}) \\ \dot{s}(t) = v(t) \\ \dot{v}(t) = \boldsymbol{a_c}(\boldsymbol{t}, \mathbf{Q}) \end{cases} \tag{3}$$

where $x$, $y$ and $\theta$ are the global pose, $v$ is the scalar speed, and $s$ is the arc-length.

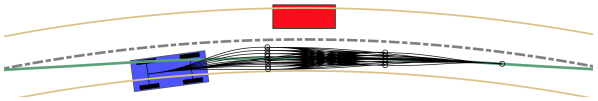
Trajectory generation solves a boundary-satisfaction problem, where $\mathbf{P}$ and $\mathbf{Q}$ are found such that the trajectory starts from a start state $\mathcal{X}^s$ and ends at a goal state $\mathcal{X}^g$:

$$\{\mathbf{P}^*, \mathbf{Q}^*, t_f^*\} = \underset{\mathbf{P}, \mathbf{Q}, t_f}{argmin} \, \|\mathcal{X}^g - \mathcal{X}^s\| \tag{4}$$

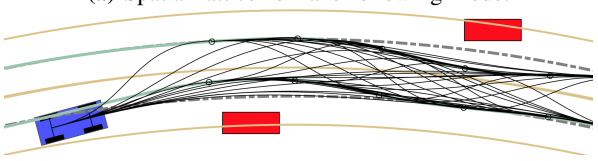
where the dynamics are given by:

$$\mathcal{X}^g = \mathcal{X}^s + \int_0^{t_f} \dot{\mathcal{X}}(t; \mathbf{P}, \mathbf{Q}) \cdot dt$$

*2) Spatiotemporal Sampling Pattern:* Sampling-based spatiotemporal planning generates a set of candidate trajectories to be evaluated. Each trajectory is composed of a sequence of path segments that specify the spatial curve and a speed profile that specifies the velocity along the path. Sampling is decoupled into separate spatial and temporal sampling.



**(a)** Spatial lattice for lane-following mode.



**(b)** Spatial lattice for lane-changing mode.

**Fig. 4:** Spatial lattice sampling pattern. For either mode, the goal is to generate a set of path sequences connecting the start node and terminal node on the path lattice.

Spatial sampling is performed focused around the previously generated reference path by making use of the state-lattice. Two sampling modes are used for both lane-following and lane-changing scenarios. For lane-following (Fig. 4a), a lattice is built consisting of layers of sampled nodes in front

of the host vehicle with decreasing lateral samples around a selected deliberative reference. For lane-changing (Fig. 4b), a lattice is built consisting of nodes along the deliberative references in both the depart and target lanes. A complete spatial plan is a sequence of paths consisting of multiple path segments on the lattice from the start node to the terminal node layer.

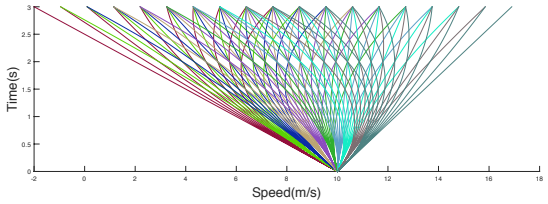



**Fig. 5:** Speed profile sampling pattern. In this example, a set of profiles start from the current speed of the host vehicle at 10 m/s, with start and terminal acceleration ranging from $-3m/s^2$ to $2m/s^2$ at the resolution of $0.5m/s^2$ and a duration of 3 seconds.

For each path sequence, a set of speed profiles are sampled to turn a path into a trajectory by uniformly varying the **Q** in equation 4. The linear acceleration speed parameterization enables sampling a rich set of near-term speed profiles with steady/increasing/decreasing accelerations (Fig. 5).

*3) Cascaded Evaluation with Bucketing:* The sampled feasible trajectories must be ranked according to some metrics and the top-ranked trajectory is the optimal to be executed. Traditional planning methods define a cumulative objective cost function with weighted feature terms (SOMWF), and use it to rank all candidate trajectories:

$$C = \sum_{i=i}^{M} \omega_{x_i} \cdot f_{x_i}$$

where $x_i$ represents the $i^{th}$ feature, $\omega_{x_i}$ and $f_{x_i}$ are the weight and cost function of feature $x_i$, and $M$ is the number of feature terms.

Tuning is difficult when $M$ is large. It is also impossible to encode prioritization among features. We therefore propose a cascaded ranking mechanism for the candidate trajectories: the set of trajectory candidates is ranked such that the trajectories whose higher-priority features have lower costs are always ranked higher regardless of the costs of lower-priority features. This prioritized ranking process endows the trajectory selection with a clear semantic interpretation.

**TABLE I:** Feature terms for trajectory evaluation/ranking

| Priority | Feature | Unit | Explanation |
|---|---|---|---|
| 1 | $f_{\mathcal{S}}$ | $m$ | Min. distance to static objs |
| 2 | $f_{\mathcal{M}}$ | $m$ | Min. distance to moving objs |
| 3 | $f_{lat}$ | $m/s^2$ | Max. lateral acceleration |
| 4 | $f_{lon}$ | $m/s^2$ | Max. longitudinal acceleration |
| 5 | $f_{\mathcal{R}_v}$ | $m/s$ | Avg. temporal diff. from ref |
| 6 | $f_{\mathcal{R}_p}$ | $m$ | Avg. spatial diff. from ref |

As shown in Table I, we identify six categorizes of feature terms related to autonomous on-road driving. The terms $f_{\mathcal{S}}$ and $f_{\mathcal{M}}$ come in the first tier of feature priority, since they are critical indicators of the impacts from surrounding static/moving objects. Note that there could further be more refined prioritization within objects, e.g., higher priority to staying away from pedestrians than from other vehicles. The terms $f_{lat}$ and $f_{lon}$ are the vehicle dynamics indicators that select smoother trajectories based on lateral and longitudinal accelerations. The terms $f_{\mathcal{R}_v}, f_{\mathcal{R}_p}$ measure temporal/spatial proximity to reference to prefer the trajectories that track the reference whenever possible.

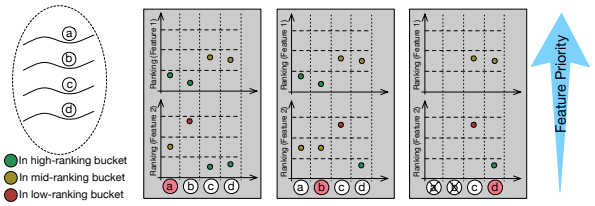



**Fig. 6:** Cascaded ranking with feature term bucketing. Suppose we have four candidate trajectories (a,b,c,d) in a two-feature planning formulation. Each feature has three ranking buckets (high, mid, low). For situation I, trajectory $a$ is the optimal trajectory: while feature-1 terms of both $a, b$ are ranked the same, the feature-2 term of $a$ is higher than that of $b$. For situation II, trajectory $b$ is the optimal trajectory: both feature rankings of $a, b$ are the same, but $b$ has a slightly higher ranking within the same bucket of feature-1 of higher priority. For situation III, trajectory $a, b$ are infeasible, feature-1 terms are ranked the same, then optimal trajectory automatically selected $d$ with the higher-ranked feature-2 term, which is in lower priority.

Since the feature terms are continuous, in order to prevent the ranking process from degenerating into sorting based on any single feature term, we make use of bucketing techniques (Fig. 6): the trajectory candidates with feature term values falling into the same bucket have the same preference. Hence, the larger the bucket size, the less significant a feature is. Meanwhile, some feature terms have infeasible regions (e.g., absolute minimum distances to objects, absolute maximum lateral accelerations). Trajectories that violate these constraints of lower-priority feature terms are also deemed infeasible regardless of the higher-priority term. This automatically allows consideration of other feasible trajectories with less desirable higher-priority feature terms.

The tuning of this ranking mechanism includes the change of priority sequence as well as the bucketing sizes. From a different perspective, the cascaded ranking mechanism is similar to a decision-tree. The difference lies in the systematic branching defined by the bucketed cost function, and the goal is not to make a decision, but to sort the candidate trajectories in a prioritized ranking pattern.

## IV. Results

The proposed planning scheme was extensively evaluated in a simulation environment, in which both the host vehicle and surrounding objects like pedestrians, bicyclists and other vehicles, are separately modeled. In the following discussion, the proposed planner is denoted by **P**, while the two benchmarking planners are denoted by:

- **B-I**: the on-road planner for Boss [16].
- **B-II**: the spatiotemporal lattice planner developed in [19].

### A. Computational Complexity

The reference smoothing and nudging planners sample 40 spatial node layers ($N_{layer}$) with an 80-meter longitudinal horizon and a 2-meter resolution. Each layer has 20 spatial nodes sampled ($N_{\frac{s-node}{layer}}$) on a 4-meter lateral span with a 0.2-meter resolution. The maximum ratio between lateral/longitudinal offsets of any two connected spatial nodes is 0.3, hence the number of in/out degrees ($N_{degree}$) is 5. For the computationally expensive collision detection operation, $N_{edge}$ edges are evaluated:

$$N_{edge} = N_{layer} \cdot N_{s-node/layer} \cdot N_{degree} \approx 4,000$$

Therefore 25 augmented-nodes are constructed for each spatial node ($N_{\frac{a-node}{s-node}}$). The number of augmented-nodes cost calculation is given by:

$$N_{a-node} = N_{layer} \cdot N_{\frac{s-node}{layer}} \cdot N_{\frac{a-node}{s-node}} \approx 20,000$$

However, this is not computationally intensive due to its pure arithmetic calculation nature.

In the trajectory sampling and search, a three-layer local spatial lattice of trajectories is built. We use a three-layer lattice design, with the number of lateral samples in each layer decreasing from $N_{L1} = 5$ to $N_{L2} = 3$ to $N_{L3} = 1$. Then a set of path sequences are created connecting start node to the terminal node on the third layer:

$$N_{path} = N_{L1} \cdot N_{L2} \cdot N_{L3} + N_{L1} \cdot N_{L3} + N_{L2} \cdot N_{L3} + N_{L3} = 22$$

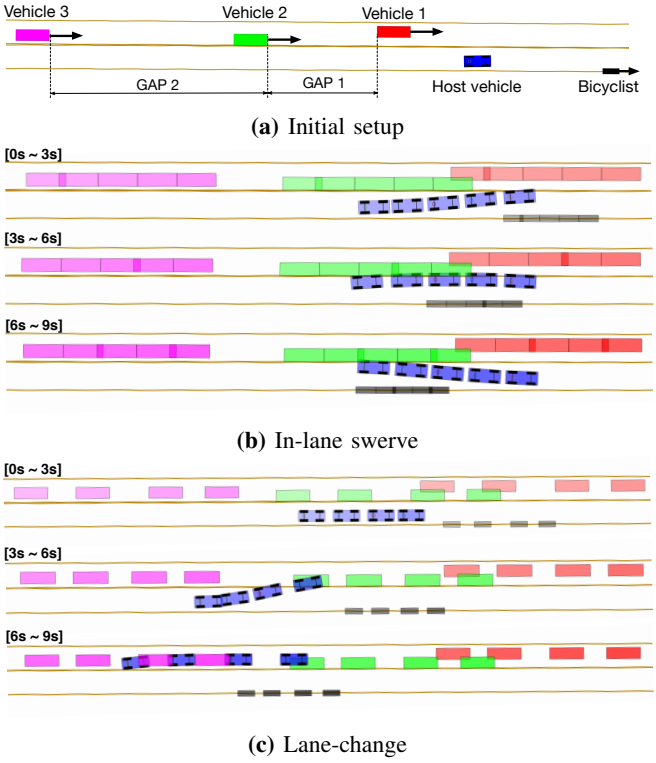
Then for each path, we uniformly sample a set of linear acceleration profiles starting and stopping from a range of accelerations from $a_{min} = -4.0$ to $a_{max} = 3.0$ at the resolution of $\Delta a = 0.5$. Hence the total number of trajectories ($N_{\frac{traj}{pathway}}$) on each path is at most 14. Therefore the total number of trajectories $N_{traj}$ is:

$$N_{traj} = N_{\frac{traj}{pathway}} \cdot N_{pathway} = 308$$

Compared to **B-I**, **P** is more computationally involved, but it does not require maneuver-level supervision from the behavior planner to trigger the basic maneuvers like vehicle following, swerving and lane-changing. Comparing to **B-II**, which requires over 200,000 explicit trajectory evaluations, **P** requires much less computation due to simpler physically intuitive formulations.

### B. Experiments

Fig. 7b demonstrates the initial setup of the scenario. The host vehicle is in the right lane with a slow-moving leading bicyclist. In the left lane, three fast-moving vehicles remain at a constant speed with two gaps with gap-1 smaller than gap-2. Fig. 7b shows the lane following case, where the planner is able to take advantage of the shorter gap-1 to swerve around the slow leading bicyclist. In the lane change case shown in Fig. 7c, the planner takes advantage of the longer gap-2 to lane change to the target lane. It made this choice because the lane change planner is aware that in the terminal state, the vehicle needs to maintain a safe distance from the leading vehicle, which will not be satisfied by taking gap 1.

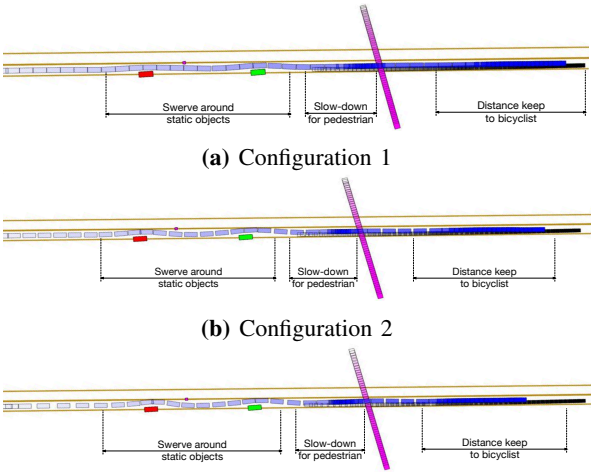



**(a)** Initial setup

**(b)** In-lane swerve

**(c)** Lane-change

**Fig. 7:** Experiments of highway planning scenarios. 7a shows the initial setup on a two-lane road segment. 7b and 7c show maneuvers performed by APV with or without lane-changing being allowed.




**(a)** Configuration 1

**(b)** Configuration 2

**(c)** Configuration 3

**Fig. 8:** Experiments in a challenging urban scenario. Configuration 1 in Fig. 8a tends to stay closer to the parked cars and trash bins, slows down the latest to react to the crossing pedestrian, and keeps a steady 5m-gap with the leading bicyclist. On the contrary, configuration 3 in Fig. 8c tends to stay further away from the parked cars and trash bins, slows down the earliest to react to the crossing pedestrian, and keeps a steady 20m-gap with the leading bicyclist. Configuration 2 in Fig. 8c yielded a resulting plan between configuration 1 & 3.

A challenging single-lane scenario (Fig. 8) is further evaluated: the host vehicle navigates within a single lane in an environment where multiple surrounding objects exist, including two parked cars, a trash bin, a pedestrian crossing the street and a slow leading bicyclist (5 m/s) moving along

the lane. Three configurations of objective function parameters (Table II) are evaluated. Fig. 8a through 8c demonstrate the planners in all three configurations performing a complex maneuver sequence: swerve around static objects, slow down for a pedestrian and distance keep to a leading bicyclist. Note that the vehicle is not guaranteed to swerve at the desired margin. For example, the car has to agilely navigate through the region with tightly spaced static objects, where the maximum margin is below the desired distance, or due to lateral constraints, the car is not allowed to swerve too much into the neighboring lanes.

**P** first evaluates each feature independently, preserving its original meaning, then uses cascaded ranking, guaranteeing prioritization among features. In comparison, **B-I** and **B-II** use weights to adjust the behavior of the planner, whereas **P** uses feature individual parameter and bucketing sizes. This formulation provides a clear semantic interpretation to achieve a straightforward tuning outcome.

**TABLE II:** Three configurations

| Feature | Unit | Config. 1 | Config. 2 | Config. 3 |
|---|---|---|---|---|
| $f^*_{\mathcal{O}_{bin}}$ | m | $[0.2, \infty)$ | $[0.4, \infty)$ | $[0.8, \infty)$ |
| $f^*_{\mathcal{O}_{pcar}}$ | m | $[0.2, \infty)$ | $[0.4, \infty)$ | $[0.8, \infty)$ |
| $f^*_{\mathcal{O}_{ped}}$ | m | $[2.0, \infty)$ | $[4.0, \infty)$ | $[8.0, \infty)$ |
| $f^*_{\mathcal{O}_{bike}}$ | m | $[5.0, \infty)$ | $[10.0, \infty)$ | $[20.0, \infty)$ |
| $f^*_{lat}$ | $m/s^2$ | $[0, 0.5)$ | $[0, 0.5)$ | $[0, 0.5)$ |
| $f^*_{lon}$ | $m/s^2$ | $[0, 1.0)$ | $[0, 1.0)$ | $[0, 1.0)$ |
| $f^*_{\mathcal{R}_v}$ | $\frac{m/s}{m}$ | $[0, 1.0)$ | $[0, 1.0)$ | $[0, 1.0)$ |
| $f^*_{\mathcal{R}_p}$ | $\frac{m}{m}$ | $[0, 0.2)$ | $[0, 0.2)$ | $[0, 0.2)$ |

## V. Conclusion

In this paper, we proposed a unable planning formulation. The three key contributions are: the three-phase tunable planning structure, the iteration-free reference planning with augmented graph and a novel local trajectory planner with cascaded-ranking to achieve clear tuning semantics.

For future work, extensive evaluation will be performed on a real vehicle in even more complicated urban driving scenarios. Along with the trend of personalized autonomous driving, we therefore focus on proposing a systematic way to quantify tunability and use machine learning techniques to distill individual-specific driving patterns.

## References


[1] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, and G. Hoffmann, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[2] F. Boyer and F. Lamiraux, "Trajectory deformation applied to kinodynamic motion planning for a realistic car model," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, pp. 487–492, IEEE, 2006.

[3] 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, no. 4, pp. 455–473, 2010.

[4] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha - A local, continuous method," in *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 450–457, 2014.

[5] J. Peng, W. Luo, W. Liu, W. Yu, and J. Wang, "A suboptimal and analytical solution to mobile robot trajectory generation amidst moving obstacles," *Autonomous Robots*, vol. 39, pp. 1–23, feb 2015.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics ...*, 2011.

[8] S. M. LaValle, "Rapidly-exploring random trees a ew tool for path planning," 1998.

[9] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[10] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[11] 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, no. 2, pp. 100–107, 1968.

[12] S. Koenig and M. Likhachev, "D* lite," in *AAAI/IAAI*, pp. 476–483.

[13] D. Q. Tran and M. Diehl, "An application of sequential convex programming to time optimal trajectory planning for a car motion," in *Proceedings of the 48th IEEE Conference on Decision and Control*, pp. 4366–4371, IEEE.

[14] D. Kelly and R. Sharp, "Time-optimal control of the race car: a numerical method to emulate the ideal driver," *Vehicle System Dynamics*, vol. 48, no. 12, pp. 1461–1474, 2010.

[15] X. Li, Z. Sun, A. Kurt, and Q. Zhu, "A sampling-based local trajectory planner for autonomous driving along a reference path," in *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 376–381, 2014.

[16] C. Urmson, J. A. Bagnell, C. R. Baker, M. Hebert, A. Kelly, R. Rajkumar, P. E. Rybski, S. Scherer, R. Simmons, and S. Singh, "Tartan racing: A multi-modal approach to the darpa urban challenge," 2007.

[17] 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.

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

[19] M. McNaughton, *Parallel Algorithms for Real-time Motion Planning*. Thesis, 2011.

[20] W. Xu, "A Real-Time Motion Planner with Trajectory Optimization for Autonomous Vehicles," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 2061–2067, IEEE, 2012.

[21] U. Schwesinger and M. Rufli, "A sampling-based partial motion planning framework for system-compliant navigation along a reference path," ... *Symposium (IV), 2013 ...*, 2013.

[22] S. Wang, *State Lattice-based Motion Planning for Autonomous On-Road Driving*. PhD thesis, 2015.

[23] C. Chen, M. Rickert, and A. Knoll, "Path Planning with Orientation-Aware Space Exploration Guided Heuristic Search for Autonomous Parking and Maneuvering," no. Iv, pp. 1148–1153, 2015.

[24] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pp. 802–807, IEEE.

[25] O. Brock and O. Khatib, "Elastic strips: A framework for integrated planning and execution," in *Experimental Robotics VI*, pp. 329–338, Springer, 2000.

[26] H. Fraichard, "From Path to Trajectory Deformation," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 159–164, IEEE, 2007.

[27] T. Brandt, T. Sattel, and J. Wallaschek, "Towards vehicle trajectory planning for collision avoidance based on elastic bands," *International Journal of Vehicle Autonomous Systems*, vol. 5, no. 1/2, p. 28, 2007.

[28] V. Delsart and T. Fraichard, "Navigating dynamic environments with trajectory deformation (teddy)," *CIT. Journal of Computing and Information Technology*, vol. 17, no. 1, pp. 27–36, 2009.

[29] T. Gu, J. Snider, J. M. Dolan, and J. W. Lee, "Focused Trajectory Planning for autonomous on-road driving," in *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 547–552, IEEE, 2013.

[30] T. Gu, J. Atwood, and C. Dong, "Tunable and stable real-time trajectory planning for urban autonomous driving," in *IROS*, 2015.