On-Road Motion Planning for Autonomous Vehicles

Tianyu Gu* and John M. Dolan**

Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, 15213, PA, USA
tianyu@cmu.edu, jmd@cs.cmu.edu

Abstract. We present a motion planner for autonomous on-road driving, especially on highways. It adapts the idea of a on-road state lattice. A focused search is performed in the previously identified region in which the optimal trajectory is most likely to exist. The main contribution of this paper is a computationally efficient planner which handles dynamic environments generically. The Dynamic Programming algorithm is used to explore in spatiotemporal space and find a coarse trajectory solution first that encodes desirable maneuvers. Then a focused trajectory search is conducted using the "generate-and-test" approach, and the best trajectory is selected based on the smoothness of the trajectory. Analysis shows that our scheme provides a principled way to focus trajectory sampling, thus greatly reduces the search space. Simulation results show robust performance in several challenging scenarios.

Keywords: Motion Planning, Dynamic Programming, On-road Autonomous Driving.

1 Introduction

1.1 Motivation

In the last few decades, both industry and academia have put effort into developing technologies for autonomous driving. Many believe that autonomous driving will dramatically enhance driving safety, improve transportation efficiency, and even revolutionize the entire automobile industry.

Motion Planning (MP) for autonomous on-road driving is a challenging problem: (1) The optimal solution (trajectory) exists in high-dimensional space, yet real-time constraints must be met in finding it; (2) Trajectory solutions must adapt to complex and unpredictable traffic; (3) Perception data, which are critical to high-speed driving, are partially observed, noisy, and lagging.

* Tianyu Gu is with the Department of Electrical and Computer Engineering.
** John M. Dolan is with Dept. ECE and Robotics Institute, School of Computer Science.

C.-Y. Su, S. Rakheja, H. Liu (Eds.): ICIRA 2012, Part III, LNAI 7508, pp. 588–597, 2012.

1.2 Related Work

Much research has been conducted on motion planning of various robots [9][5]. Dijkstra's Algorithm, the A* Algorithm and their derivatives[12][8] have been used intensively for path planning on a grid-like space. The resulting paths, however, do not satisfy the non-holonomic constraints of a car-like vehicle. To address this, [1] introduced a non-holonomic path generation method.

To inspire the development of autonomous driving technologies, DARPA organized the Urban Challenge in 2007. To deal with on-road driving, many teams [2][3] performed lane-based trajectory generation by rolling out trajectories based on lateral shifts from the lane centerline. This scheme worked well in the low-density, low-speed (up to 30 mph) competition environment, but was too naive for realistic on-road driving in complex dynamic environments.

Several on-road motion planners have used an on-road state lattice. Artificial heuristics were developed in a few works to narrow down the exploration region in the lattice. [13] proposed a method that connected lattice nodes to generate paths that complied with certain speed heuristics. A directed acyclic graph search algorithm was used to search for the shortest path on the grid. [6] proposed an on-road planner that solved optimal lateral and longitudinal control problems in a Frenet Frame. Different heuristic functionals were devised for maneuvers like road following and lane merging. The disadvantage of heuristics-based approaches is that it is unrealistic to find a complete set of heuristics that are applicable in all cases.

An alternative solution is to exhaustively iterate over all possible solutions by conducting dense trajectory sampling. [4] proposed a planner that sampled trajectories on a road lattice. To prevent exponential blowup of trajectories, the author adopted a scheme to trim trajectories that ended at a similar vehicle state. Based on this work, [7] reduced the computation by sampling fewer paths but post-optimizating the trajectories. The disadvantage of sampling-based approaches is that effort is wasted, since most of the trajectories generated will eventually be discarded. Our proposed approach addresses these issues. A sequence of high-level actions that encrypts desirable maneuvers is found first and serves as guidance to a focused yet modest amount of trajectory sampling and search.

The rest of this paper is structured as follows. Section 2 presents a few assumptions to focus our work on motion planning. Section 3 introduces an action-based coarse trajectory-planning scheme amenable to Dynamic Programming. Section 4 explains focused trajectory search for fine trajectory planning. Section 5 explains the implementation details and compares our algorithm with state-of-the-art alternatives. Section 6 presents simulation results in our test scenarios.

2 Assumptions

In order to focus our work on motion planning, we will make the following assumptions.

Assumption 1. *Perfect Perception*

Perception is perfect in the sense that static obstacles are stable, and dynamic obstacles can be precisely predicted.

We use the road state lattice for our planer. It is convenient to use a road coordinate system, so that every interesting point can be indexed by station and latitude. Moreover, vehicle shape has been convolved to the map, so that we can plan a trajectory for a fixed point on the car body without evaluating the entire trajectory for collision checking of the sides and front/rear bumper.

Assumption 2. *Perfect Tracker*

A low-level tracking module that perfectly executes the planned trajectory is assumed, so that the safety is guaranteed if the trajectory is safe.

3 Action-Based Coarse Trajectory Planning

A human driver doesn't have a precise trajectory in mind when driving; instead, s/he would normally have a rough idea about how to avoid an obstacle, or how fast to overtake the vehicle in front. Based on this insight, we would like our planner to find a sequence of actions that describes a rough maneuver first.

Time-dimension discretization naturally gives us stages of the planning process at each time increment. The state space (manifold) is defined that describes the vehicle's state at every stage. The process now only demands choosing an action at each stage. The above two characteristics (staged & action-based) satisfy the requirements of applying Dynamic Programming (DP) algorithms.

3.1 State and Action Space

The state space includes station (s) and latitude (l) dimensions to represent the vehicle's location in road coordinates, Fig.1. Given the discretized centerline,

$$\left[x_c(s)\ y_c(s)\ \theta_c(s)\ \kappa_c(s)\right]$$

the following equations are used to construct the on-road state lattice according to the centerline:

$$\begin{aligned} x(s,l) &= x_c(s) + l \cdot cos(\theta_c(s)) \\ y(s,l) &= y_c(s) + l \cdot sin(\theta_c(s)) \\ \theta(s,l) &= \theta_c(l) \\ \kappa(s,l) &= (\kappa_c(l)^{-1} + l)^{-1} \end{aligned} \tag{1}$$

where s is the station, l is the lateral offset from the centerline.

For highway driving, the longitudinal velocity component dominates the lateral velocity component. We introduce longitudinal velocity (v_{lon}) as another dimension in state space. The addition of the velocity dimension diversifies our

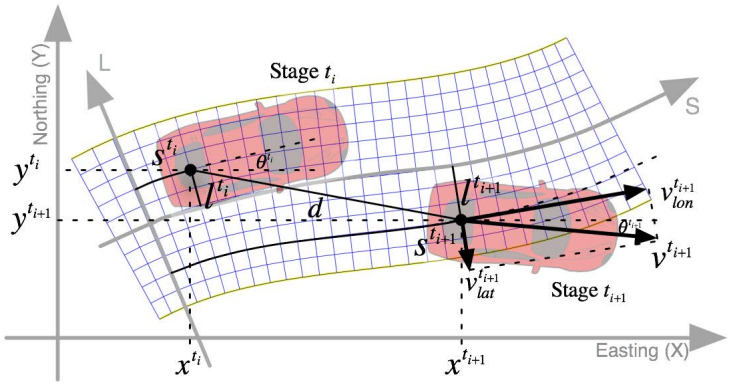


Fig. 1. Cordinates and Parameters

state space, hense The optimization process becomes more informed, since velocity serves as one of the most important indicators of the quality of driving.

Define $\mathcal{M}^{t_i}$ as a three-dimensional state manifold at stage t_i, and state X^{t_i}, where

$$X^{t_i} = \left[s^{t_i}\ l^{t_i}\ v^{t_i}_{lon}\right]^T,\ X^{t_i} \in \mathcal{M}^{t_i} \tag{2}$$

An action A^{t_i} is a function of state, $A^{t_i} \in \mathcal{U}(X^{t_i})$. It leads to a state transition, represented as $T(X^{t_i}, A^{t_i}) : X^{t_i} \xrightarrow{A^{t_i}} X^{t_{i+1}}$

3.2 Cost Functions

For each state transition $T(X^{t_i}, A^{t_i})$, a cost criterion $C(X^{t_i}, A^{t_i})$ is specified to penalize undesirable action effects. The optimality achieved by the DP algorithm is with respect to the linear addition of the cost terms in table 1:

$$C(X^{t_i}, A^{t_i}) = c_d + c_{offset} + c_{v_{lon}} + c_{a_{lon}} + c_{a_{lat}} + c_{obstacle} \tag{3}$$

A few cost terms are devised to characterize good behavior. The philosophy is to create as few and as decoupled cost terms as possible so that the tuning can be intuitive.

3.3 Dynamic Programming Algorithm

The solution to a DP problem is a sequence of actions $\{A^{t_i}\}$ that minimize

$$\sum_{t=t_0}^{t_{N_{time}}} \beta^t C(X^t, A^t) \tag{4}$$

where state transitions are subject to

$$X^{t_{i+1}} = T(X^{t_i}, A^{t_i}) \tag{5}$$

Table 1. Cost Terms

Description	Parameter c	Expression
Distance	$w_d \cdot d$	$d = \sqrt{(x^{t_{i+1}} - x^{t_i})^2 + (y^{t_{i+1}} - y^{t_i})^2}$
Lateral Offset	$w_{offset} \cdot offset$	$offset = \frac{l^{t_{i+1}} + l^{t_i}}{2}$
Longitudinal Velocity	$w_{v_{lon}} \cdot v_{lon}$	$v_{lon} = v_{horizon} - \frac{v^{t_{i+1}} + v^{t_i}}{2}$
Longitudinal Acceleration	$w_{a_{lon}} \cdot a_{lon}$	$a_{lon} = a^{t_i}$
Lateral Acceleration	$w_{a_{lat}} \cdot a_{lat}$	$a_{lat} = \frac{\kappa^{t_{i+1}} + \kappa^{t_i}}{2} \cdot (\frac{v^{t_{i+1}} + v^{t_i}}{2})^2$
Obstacle	$w_{obstacle} \cdot obstacle$	$obstacle = Status(s^{t_{i+1}}, l^{t_{i+1}})$

We find it reasonable to treat planning as a stateless process over time. Again, taking the human driver as an example, s/he rarely (almost never) plans from the past, e.g considering the path s/he has travelled. Human drivers always look at the road in front and plan from the current state into the future.

This means that choosing actions in a given state is completely independent of the past states. Statelessness (the Markov Property) allows us to exploit Bellman's Principle of Optimality to solve our dynamic programming problem.

Define

$$\Omega_i = min\{ \sum_{t=t_i}^{t_{N_{time}}} \beta^t C(X^t, A^t)\}, i = 0, 1, 2, ..., t_{N_{time}} \tag{6}$$

Note that

$$\Omega_{N_{time}} = \beta^{t_{N_{time}}} C(X^{t_{N_{time}}}, A^{t_{N_{time}}}), A^{t_{N_{time}}} \text{ is NULL} \tag{7}$$

specifies the cost distribution on the manifold at stage $t_{N_{time}}$. By assigning a different distribution, we can specify the most desirable state at the final stage.

The Principle of Optimality tells us,

$$\Omega_i = min\{\beta^{t_i} C(X^{t_i}, A^{t_i}) + \sum_{t=t_{i+1}}^{t_{N_{time}}} \beta^t C(X^t, A^t)\} \tag{8}$$

$$= min(\beta^{t_i} C(X^{t_i}, A^{t_i})) + \Omega_{i+1} \tag{9}$$

After recursively finding the optimal actions for all state transitions, we can quickly backtrace a sequence of actions from A^{t_0} to $A^{t_{N_{time}-1}}$ by feeding the inital state.

3.4 Algorithm Features

Tuning parameters to get desirable maneuvers is an iterative learning process. But with decoupled cost weight terms, this process is very intuitive.

The discount factor β plays an important role in the optimization process. If the factor is small, so that transition costs from future states are very close to zero, the trajectory ends very early. The implication is that the future states are becoming untrustworthy such that the optimization would make no difference for whether to continue or not. If the factor is close to 1, on the other hand, the trajectory becomes aggressive.

To mitigate the "Curse of Dimensionality", our formulation constructs state space with relatively low dimensionality and coarse resolution, and also a modest action space. Yet it retains enough diversity to represent desirable on-road maneuvers.

4 Focused Fine Trajectory Planning

The result of the previous section is a global plan in the form of a sequence of desirable maneuvers, and a sequence of safe vehicle poses. Once this is given, we need to generate one dynamically feasible smooth trajectory for the vehicle to execute.

4.1 Path Generation

A path that satisfies nonholonomic constraints is given by

$$\begin{aligned} x(\tilde{s}) &= x(0) + \int_0^{\tilde{s}} cos(\theta(\tau))d\tau \\ y(\tilde{s}) &= y(0) + \int_0^{\tilde{s}} sin(\theta(\tau))d\tau \\ \theta(\tilde{s}) &= \theta(0) + \dot{\kappa}(\tilde{s}) \\ \kappa(\tilde{s}) &= p_0 + p_1\tilde{s} + p_2\tilde{s}^2 + p_3\tilde{s}^3(+p_4\tilde{s}^4 + p_5\tilde{s}^4) \end{aligned} \tag{10}$$

where $\tilde{s}$ is the arc-length of the path, and the unknown parameters $p_0 ... p_5$ and s_f. To solve the unknowns, we use the method proposed in [1].

[11] proved that quintic polynomial curvature guarantees the continuity of both the curvature's rate of change and its derivative, which leads to smooth robot motions. While quintic polynomial paths are suitable for high-speed trajectories, cubic polynomials are sufficient, even ideal, for low-speed trajectories in that they will result in paths that are quicker in turning [10].

4.2 Velocity Profile Generation

Instead of using linear velocity profiles, as do many prior works, we use a cubic function of time, which is smoother.

$$v(t) = q_0 + q_1 t + q_2 t^2 + q_3 t^3 \tag{11}$$

This relation naturally gives us analytical expressions for both acceleration and length by differentiation and integration respectively. Given the travel time t_f, start velocity v_0, start acceleration a_0, end velocity v_f and path length s_f, we can analytically express the remaining unknowns.

4.3 Focused Trajectory Sampling and Evaluation

Unlike prior work [4][7], we don't want to generate a large number of trajectories that eventually will be discarded, nor do we want to generate trajectories that are too long, and will not have the chance to be executed, since the planner is replanning very fast.

A sampling center that guides the focused trajectory sampling must be determined. A sampling center is chosen as any of the states in the sequence of state transitions solved by previous planning. We have two rules in choosing:

(1) The trajectory should last at least T seconds. T should be greater than the planner's replanning period, so that we will always have a safe trajectory. We pick T = 1sec.

(2) The trajectory should be at least S meters long. S should be long enough so that the path does not have undesirable features, e.g. the curvature and the derivative of curvature may increase dramatically in the middle of a too-short path. We pick S = 5m.

Once the sampling center is picked, we conduct a random path and velocity profile sampling and evaluation within this small region, and pick the best trajectory with the minimum integral of the squared jerk.

5 Implementation and Analysis

5.1 Implementation

As explained in section 3, the states contain three components: station, lateral offset and speed.

States are discretized to adapt the need for mimicking on-road driving maneuvers. $\Delta T, \Delta S, \Delta L, \Delta V$, are the units of our system discretization. Their values need to be carefully specified. Starting with ΔT, we believe a second-level discretization will be enough for a coarse on-road trajectory plan, $\Delta T = 1s$. ΔV is the minimum speed difference of sampled speeds. Any $v_{lon} = n \cdot \Delta V$, where integer $n \in [0, N_V)$. The finer ΔV is, the more accurate speed we can express. For our purpose, we found that $\Delta V = 3m/s$ is a reasonable value. To decide ΔS, we notice that for any $v_{lon}^{t_i} = n_1 \cdot \Delta V$, $v_{lon}^{t_{i+1}} = n_2 \cdot \Delta V$, where n_1, n_2 are integers. the difference $\|v_{lon}^{t_{i+1}} - v_{lon}^{t_i}\| = \|n_1 - n_2\| \cdot \Delta V$ is always a multiple of ΔV. Thus the minimum traversing station (other than zero) between two stages is $\Delta S = \frac{\Delta V \cdot \Delta T}{2} = 1.5m$. For on-road driving, the lateral speed is much smaller than the longitudinal component. We assume the maximum lateral velocity to be 0.5m/s, thus set $\Delta L = 0.5 \cdot \Delta T = 0.5m$.

The details are listed in Table 2.

An action on states takes effect on all three components. Particularly, a_1 affects longitudinal velocity, a_2 lateral offset, and a_3 longitudinal velocity.

$$\begin{aligned} &T(v_{lon}^{t_i}, A^{t_i}) : v_{lon}^{t_{i+1}} = v_{lon}^{t_i} + a_1^{t_i} \cdot \Delta T \\ &T(l^{t_i}, A^{t_i}) : l^{t_{i+1}} = l^{t_i} + a_2^{t_i} \cdot \Delta T \\ &T(s^{t_i}, A^{t_i}) : s^{t_{i+1}} = s^{t_i} + a_3^{t_i} \cdot \Delta T \end{aligned} \tag{12}$$

Table 2. Dimension Discretization List

Dimensions	**T**ime(s)	**S**tation(m)	**L**attitude(m)	**V**elocity(m/s)
Horizon	$H_T = 10$	$H_S = 60$	$H_L = 5$	$H_V = 27$
Discretization	$\Delta T = 1$	$\Delta S = 1.5$	$\Delta L = 0.5$	$\Delta V = 3$
Increments	$N_T = 10$	$N_S = 40$	$N_L = 10$	$N_V = 10$

To constrain the action space, we let $a_1^{t_i} = n_1 \frac{\Delta V}{\Delta T}$, where integer $n_1 \in [-2, 1]$ and $a_2^{t_i} = n_2 \frac{\Delta L}{\Delta T}$, where integer $n_2 \in [-1, 1]$ and $a_3^{t_i} = \frac{v_{lon}^{t_{i+1}} + v_{lon}^{t_i}}{2}$.

Let P represent the number of possible state transitions from each state. We use approximate equality, since these actions are not available to all states.

$$P \approx num(n_1) \cdot num(n_2) = 12 \tag{13}$$

5.2 Analysis

For the heuristics-based approaches [13][6], it is hard to perform a direct comparison on computation, since the authors did not provide a detailed computation cost. On the other hand, we can compare to the sampling-based approaches [4] [7], since the authors have provided the number of trajectories they evaluated for each planning cycle.

Typically, trajectory evaluation is conducted in the following steps: (1) sample on the trajectory; (2) perform collision checking for each sampled point; (3) calculate the cost for each of the sampled points; (4) accumulate the cost for each of the sampled points. For a fair comparison, we assume the same discretization resolution, and suppose a realistic 10 points/trajectory sampling.

[7] specified the full search space at every cycle, thus suffered the "Curse of Dimensionality" with our resolution: 1,000,000 trajectories/cycle = 10,000,000 points/cycle.

[4] used a clever trimming scheme that constrains the search space while the search proceeds, so that the search space does not blow up. Still, the author had to maintain a complex data structure and had to evaluate about: 400,000 trajectories/cycle = 4,000,000 points/cycle.

For our approach, the focused fine planning only selects a fixed and small number of trajectory samples (about 100), which is a trivial overhead. The major computation occurs in calculating state transition in the action-based coarse planning. The number of state transitions is given by $[(N_S \cdot N_L \cdot N_V) \cdot P] \cdot N_T$ = 480,000 transitions/cycle.

The computation required to calculate a state transition is similar to that of conducting collision checking and calculating cost for a point. Comparing to [4] and [7], we have a 8.3X and 20.8X speed-up respectively.

Actually, it is nearly as efficient as if we were doing trajectory evaluation with only one sample point, that is saving $\frac{N-1}{N} \cdot 100\%$ computations for each trajectory evaluation, where N stands for the number of sampling points. In this sense, our approach obviously wins out over the brute force sampling approaches.

6 Simulation Result

Four on-road situations were tested in simulation (Fig. 2).

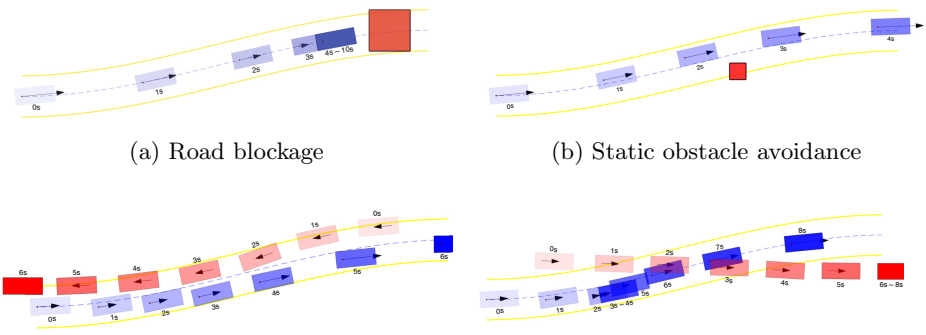
Road Blockage: The car can reach a full stop just in time to avoid collision with the blocking obstacle.

Static Obstacle Avoidance: The car will slightly nudge to the left, and decrease the speed a little bit to avoid collision.

Oncoming Vehicle Avoidance: The car will veer slightly to the right, and meanwhile decrease the speed until the oncoming vehicle drives away.

Aggressive Merging Vehicle Avoidance: This scenario shows a rogue vehicle trying to cross our lane. Our car comes to a stop smoothly, and gets back to on-road driving when the moving vehicle is out of the way.

All sub-figures in Fig. 2 came from the same setting of the cost weights and discount parameter.



(a) Road blockage (b) Static obstacle avoidance

(c) Oncoming vehicle avoidance (d) Aggressive merging vehicle avoidance

Fig. 2. Simulation Results with Time Steps Indicated

7 Conclusion

Most prior on-road motion planners have wasted a large amount of computation on arbitrary and unfocused sampling of trajectories. We provide a two-step scheme that plans coarsely first, attempting to capture the gist of how human drivers drive, namely not knowing the precise plan, but having a global sense of

how they should drive. Simulation has shown that our method can robustly handle different dynamic on-road driving scenarios, some of which are challenging even to human drivers.

Our immediate next step is to implement and test our planner on a real vehicle, then robustify the scheme by making it capable of handling more complex and realistic scenarios, for example, planning lane changes.

References

1. Kelly, A., et al.: Reactive nonholonomic trajectory generation via parametric optimal control. International Journal of Robotics Research 22(7), 583–601 (2003)
2. Urmson, C., et al.: Autonomous driving in urban environments: Boss and the urban challenge. J. Field Robotics 25(8), 425–466 (2008)
3. Montemerlo, M., et al.: Junior: The stanford entry in the urban challenge. J. Field Robotics 25(9), 569–597 (2008)
4. McNaughton, M., et al.: Motion Planning for Autonomous Driving with a Conformal Spatiotemporal Lattice. In: IEEE International Conference on Robotics and Automation, vol. 1, pp. 4889–4895 (2011)
5. Pivtoraiko, M., et al.: Differentially constrained mobile robot motion planning in state lattices. Journal of Field Robotics 26(3), 308–333 (2009)
6. Werling, M., et al.: Optimal trajectory generation for dynamic street scenarios in a frenét frame. In: ICRA, pp. 987–993 (2010)
7. Xu, W., et al.: A real-time motion planner with trajectory optimization for autonomous vehicles. In: ICRA (2012)
8. Koenig, S., Likhachev, M., Furcy, D.: Lifelong planning a*. Artif. Intell. 155(1-2), 93–146 (2004)
9. LaValle, S.M.: Planning algorithms. Cambridge University Press, Cambridge (2006), `http://planning.cs.uiuc.edu/`
10. McNaughton, M.: Parallel algorithms for real-time motion planning. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (July 2011)
11. Piazzi, A., Bianco, C.G.L.: Quintic G2-splines for trajectory planning of autonomous vehicles. In: IEEE Intelligent Vehicles Symposium (2000)
12. Stentz, A.: The focussed d* algorithm for real-time replanning (1995)
13. Ziegler, J., Stiller, C.: Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In: The International Conference on Intelligent Robots and Systems (2009)