A Behavioral Planning Framework for Autonomous Driving

Junqing Wei, Jarrod M. Snider, Tianyu Gu, John M. Dolan and Bakhtiar Litkouhi

***Abstract*— In this paper, we propose a novel planning framework that can greatly improve the level of intelligence and driving quality of autonomous vehicles. A reference planning layer first generates kinematically and dynamically feasible paths assuming no obstacles on the road, then a behavioral planning layer takes static and dynamic obstacles into account. Instead of directly commanding a desired trajectory, it searches for the best directives for the controller, such as lateral bias and distance keeping aggressiveness. It also considers the social cooperation between the autonomous vehicle and surrounding cars. Based on experimental results from both simulation and a real autonomous vehicle platform, the proposed behavioral planning architecture improves the driving quality considerably, with a 90.3% reduction of required computation time in representative scenarios.**

I. Introduction

Autonomous driving has been a promising research area since the 1990s. With recent advances in computer and sensor engineering, partially autonomous vehicles may be produced and widely used for travelling in the near future with fully automated vehicles to follow.

There are over 1,200,000 deaths and 50,000,000 injuries each year, worldwide [11], [1]. Autonomous vehicles have great potential to avoid or reduce the severity of most of such traffic accidents which are mainly caused by human driver errors. They will also have the potential to enhance the overall performance of the current traffic system by efficiently increasing highway capacity and decreasing traffic congestion in cities [20]. Fully autonomous driving is able to spare people from the task of driving while commuting.

Since the 1980s, autonomous vehicle intelligence has increased from lane centering to actually driving on public roads with lane-changing capability [3], [12], [15], [7], [2], [19]. A few research platforms have shown great capability in driving on public roads. However, autonomous vehicle sensing, decision making and motion planning intelligence have not achieved the same level as that of good human drivers.

In the architecture of an autonomous vehicle, motion planning is a key component that affects the autonomous vehicles performance. The motion planners role is to generate a kinematically and dynamically feasible path and velocity profile for the robot to execute. Due to computational constraints, the motion planner usually simplifies the problem by assuming that the surrounding moving vehicles will keep constant speed [15], [8]. The inaccurate prediction and lack of understanding of human drivers intentions prevent autonomous vehicles from cooperating with human-driven vehicles as human drivers intuitively do with one another.

Most motion planners in autonomous vehicles also assume the robot will execute the planners command with no delay or error [21], [18]. Such assumptions cause the autonomous vehicle either to fail to execute the command, leading to potential danger, or overcompensate for the control error, potentially decreasing the comfort level.

This work was supported by NSF & GM

Junqing Wei, Jarrod M. Snider and Tianyu Gu are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA junqingw@cmu.edu

John M. Dolan is with the Department of Electrical and Computer Engineering and Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Bakhtiar Litkouhi is with the General Motor R&D, Warren, MI 48090-0955, USA

II. Related Work

1) Motion Planner: Motion planning for autonomous vehicles has been developed and substantially improved in recent decades [14]. Most fully autonomous concept vehicles have their own motion planning to directly command a desired trajectory, which includes both desired path and speed. In the Urban Challenge, CMUs Tartan Racing Teams motion planner considered kinematic and dynamic constraints of the vehicle, as well as all moving and static obstacles [15]. It generated 20 to 50 feasible trajectories from a given sample point with different offsets from the center of the road 10 to 30 meters away from the vehicle and then selected the best one. This mechanism proved to be an efficient way to avoid static obstacles. However, due to the short planning horizon, it was difficult for it to perform smooth lane changes or circumvent dynamic obstacles, especially at highway speeds.

To overcome the shortcomings of this planner, McNaughton et al. proposed concatenating multiple layers of trajectories to produce a longer horizon and search more candidate trajectories [8]. Benefiting from the larger number of possible trajectories, this achieves better performance in obstacle avoidance and circumventing slowly moving obstacles. However, the planner still only chooses from a very limited number of candidate trajectories. Their endpoints have different lateral offsets, but the same heading as the road. In cases with sharp turns and in obstacle avoidance maneuvers, sometimes no trajectory provides smooth and human-like performance.

An alternative approach to sample-based planners [15], [8] is based on path optimization [21]. The trajectory is defined as a function of a few control points. It is initialized to be the center line of the road. Then the parameters of the control points are optimized based on multiple cost terms, including curvature, length of the trajectory, and lateral acceleration. However, this kind of optimization is vulnerable to local optima, which can cause very poor performance. It is also computationally quite intensive, with non-deterministic computation time.

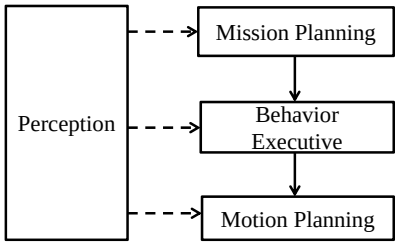


Fig. 1: Hierachical framework for an autonomous vehicle

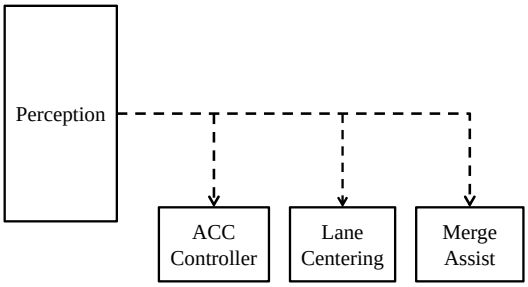


Fig. 2: Parallel framework for an autonomous freeway cruise system

2) Social Behavior: Although kinematic and dynamic constraints, smoothness of path, and obstacle avoidance are frequent concerns in motion planning, the autonomous vehicles ability to socially cooperate with surrounding moving vehicles is usually overlooked. In circumstances such as entrance ramps, lane changes, etc., the human driver will make decisions based on his/her understanding of what other human drivers are thinking or are likely to do. He can also adjust his speed to show other drivers his own intention. This kind of cooperation happens intuitively between human drivers. Not understanding this behavior can result in incorrect decision making and motion planning with undesirable outcome [18].

Wei et al. [17] proposed a Prediction- and Cost-function Based algorithm that can make the autonomous vehicle consider potential reactions of surrounding objects when making decisions. However, this approach has only been applied and tested in simulated freeway driving scenarios. Due to the computational complexity and potential explosion of the search space, it is difficult to introduce vehicle interaction and intention understanding into the planner with the current architecture of autonomous vehicles [8].

3) Hierarchical Autonomous Vehicle Architecture: To enable autonomous vehicles to finish long-term missions and reduce the workload of motion planning, a hierarchical architecture is widely used [13], [6], [10]. For each layer of the architecture, the input higher-level mission is decomposed into sub-missions and passed on to the next-lower level. Using this framework, many complicated problems becomes solvable. However, this layered architecture can cause some performance problems.

Figure 1 (CMUs Boss from the Urban Challenge) shows such a framework [9]. The perception system analyzes real-time data input from laser scanner (Lidar), radar, GPS and other sensors. Mission planning optimizes the path to achieve different checkpoints considering the arrival time, distance, or different required maneuvers. The behavior executive system makes tactical driving decisions about such things as distance keeping, lane changing, and neighboring vehicle interactions. Motion planning generates the desired trajectory of the vehicle considering the dynamic parameters and output steering and throttle pedal commands.

One of the shortcomings of this framework is that the higher-level decision making module usually does not have enough detailed information, and the lower-level layer does not have authority to reevaluate the decision. For instance, when the autonomous vehicle wants to use an oncoming lane to circumvent a slow-moving obstacle, the behavior layer makes the decision. Then it outputs the desired circumvention window and speed that the car should use to perform this behavior without interfering with traffic in the opposing lane. However, the lower-level motion planner may find out it cannot drive at the desired speed because of an upcoming sharp turn, which is not considered in the behavior layers decision making, so it has to slow down to perform the circumvention. However, because of the oncoming car, failing to keep the desired speed will cause the car to miss the circumvention time and spatial window [15].

The other shortcoming is that in this architecture, most information is processed in the motion planner, including road geometry, vehicle dynamics, and surrounding moving objects and static obstacles. This makes the planner problem very complicated and computationally expensive. The planner usually needs to sacrifice performance to meet real-time constraints [20], [8].

4) Parallel Autonomous Vehicle Architecture: There are also many autonomous or semi-autonomous vehicles based on existing Advanced Driving Assist Systems (ADAS). For example, by integrating lane centering and Adaptive Cruise Control (ACC), a basic single lane freeway autonomous driving capability can be achieved [4], [16].

Compared with the hierarchical framework, modules in this system are relatively independent and work in parallel, as shown in Figure 2. There are dedicated sensors and actuation mechanisms for each of the controllers. For example, the lane centering module focuses on using the steering wheel to keep the car in the center of a lane, while the ACC controller controls the throttle and brake to make the car follow its leader. The benefits of this architecture could be: first, the controllers are running at a high frequency and are proven to be safe and stable; second, the controllers have achieved a high level of smoothness and performance; third, computational cost is low because there is no complicated motion planning being used. However, in some complicated cases needing cooperation, this framework may not perform well. For example, when the autonomous vehicle wants to merge into a neighboring lane with slower-moving traffic, it may not be able to properly slow down because the speed controller typically does not know the cars intended lateral movements. In contrast, some motion planner algorithms are able to slow down the vehicle and find the best opening to merge into.

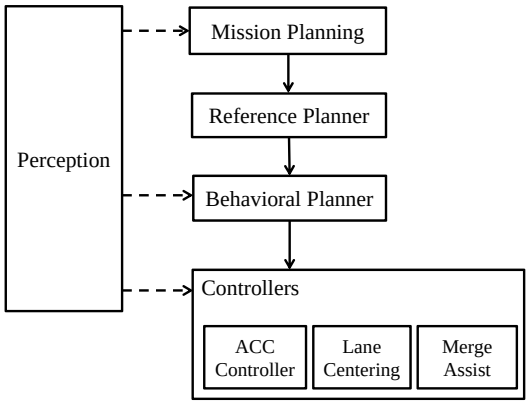


Fig. 3: The proposed behavioral planning framework

III. Behavioral Planning Framework

As discussed in Section II, there are shortcomings of both the current hierarchical and parallel robot decision making architectures. Due to real-time constraints, the motion planner cannot consider the effects of imperfect vehicle controllers or cooperation between cars. In this paper, we propose a novel behavioral planning framework that combines the strengths of the hierarchical and parallel architectures. It is based on the hierarchical architecture so that fully autonomous driving with high-level intelligence can be achieved. However, it also uses the independent controllers as in the parallel autonomous vehicle architecture to ensure basic performance and driving quality. The design greatly reduces the necessary search space for the motion planner without sacrificing any performance due to coarser granularity. The proposed framework is shown in Figure 3.

A. Mission Planning

The mission planning module takes charge of decomposing driving missions such as "go from point A to point B" into lane-level sub-missions such as which lane we should be in, whether we need to stop at an intersection, etc. It takes a human drivers desired destination and computes the shortest path to it from the robots current position. It outputs a set of future lane-level sub-missions which describe the desired lane and turn of the vehicle at each intersection. In addition, this module also controls the transition of goals. When the vehicle completes the current lane-level sub-mission, it will automatically send the next set of goals to the lower layers. When there are intersections with stop signs, traffic lights or yielding requirements, this module directly talks to the perception system to decide whether the car can proceed to the next sub-mission.

B. Traffic-free Reference Planning

The reference planning layer takes the lane-level sub-missions, and outputs a path and speed profile for the autonomous vehicle to drive [5]. In this layer, the planner assumes there is no traffic on the road. It uses non-linear optimization to find a smooth and human-like path and speed with consideration of the kinematics and dynamics of the autonomous vehicle, as well as the geometry of the road.

Fig. 4: Reference planners result: the desired path cut corners at the turn to minimize path length and generate better handling

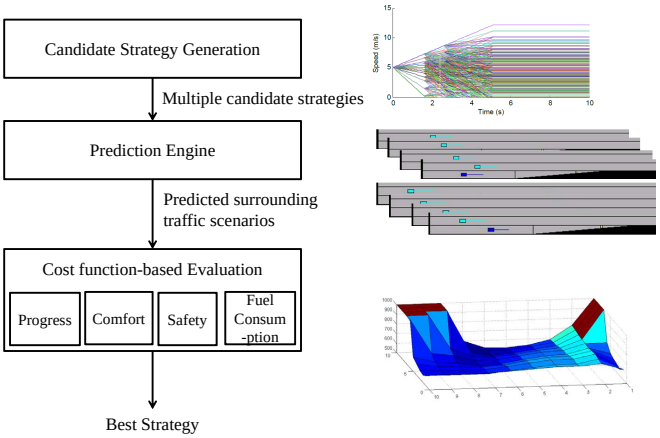


Fig. 5: The block diagram of Prediction- and Cost-function Based algorithm (PCB)

For example, if the road is not straight, as shown in Figure 4, instead of driving exactly in the center of the road, a human driver will drive slightly offset from the center line to minimize the required steering maneuver, which is emulated in this module. In addition, traffic rules such as speed limits are applied in this layer.

C. Behavioral Planning

Since the road geometry has been considered in the previous layer, the behavior planner focuses on handling on-road traffic, including moving obstacles and static objects. It takes the traffic-free reference, moving obstacles and all road blockages as input. It outputs controller directives including lateral driving bias, the desired leading vehicle we should follow, the aggressiveness of distance keeping, and maximum speed. In this module, a Prediction- and Cost-function Based (PCB) algorithm is implemented [17]. There are three primary steps: candidate strategy generation, prediction, and cost function-based evaluation, as shown in Figure 5.

In the PCB algorithm, the world is abstracted as shown in Equation 1. VS_{host} is the state vector for the host vehicle and surrounding vehicles, with station (longitudinal distance along the reference path), velocity, acceleration and lateral offset information. VS_{other} is the state vector for each surrounding vehicle. Compared with VS_{host}, it has additional information about the intention and probability of intention. S_{road} is the state vector of the reference path with maximum speed information at each of M stations. S_{PCB} is the state vector for the behavioral planner.

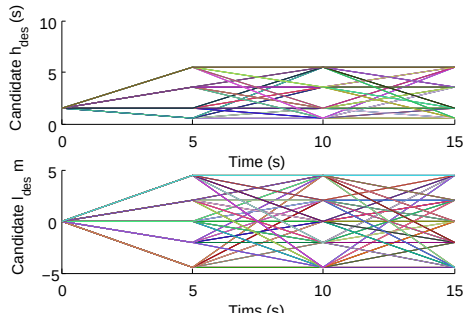


Fig. 6: Candidate strategies in the form of controller directives for the next 15 seconds: desired timing headway h_{des} profile (top) and desired lateral offset l_{des} profile (bottom)

$$
\begin{aligned}
VS_{host} &= [s, v, a, l] \\
VS_{other} &= [s, v, a, l, i, p(i)] \\
S_{road} &= [s_{0,1,2,...,M}, v_{max,0,1,2,...,M}] \\
S_{PCB} &= [VS_{host}, VS_{other=1,2,3,...,N}, S_{road}]
\end{aligned} \tag{1}
$$

1) Candidate Strategy Generation: In the candidate decision generation step, a certain number of candidate directives for the controllers are generated. Figure 6 shows the candidate strategies for longitudinal (h_{des}) and lateral movement (l_{des}) of the car, respectively, where l_{des} is the desired lateral offset, and h_{des} is the desired time headway, which is related to the distance the host vehicle wants to keep from its leader. By pairing an l_{des} profile with a h_{des} profile, a candidate strategy is created. The following steps will convert each candidate strategy profile into trajectories for the behavioral planner to evaluate.

2) Prediction Engine: All the potential controller directives are sent to the prediction engine, which forward-simulates the controllers to generate a trajectory for the car to drive. There are two controllers that are forward-simulated in the behavioral planning, the Adaptive Cruise Controller (ACC) and lateral controller. In the prediction engine, each candidate strategy is forward-simulated based on the ACC controller and lateral controller models to get the trajectory of the host vehicle. At every time step, all vehicles behaviors including their reaction to the host vehicles trajectory are also predicted.

a) Forward-Simulate ACC Controller: The ACC controller takes charge of the longitudinal speed control of the vehicle. The state of the ACC controller S_{ACC} is described in Equation 2, in which h_{des} comes from the candidate strategy generated in the first step of the PCB algorithm.

$$S_{ACC} = [v_{max}, d_{lead}, v_{host}, v_{lead}, h_{des}] \tag{2}$$

The desired distance can be computed using Equation 3, in which d_{min} is the minimum distance when both the leading and following cars are static.

$$d_{desired} = d_{min} + h_{des} v_{lead} \tag{3}$$

Other parameters in the ACC controller state can be calculated from VS_{host}, VS_{other} and S_{road} using the following mechanism.

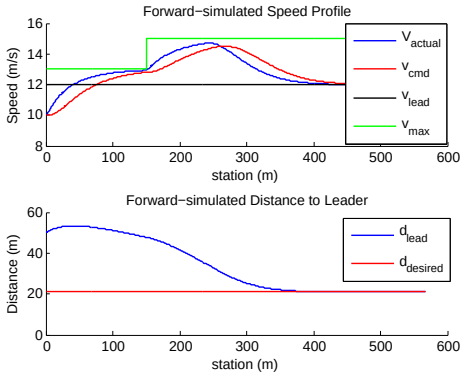


Fig. 7: Result of forward-simulating ACC controller with initial condition $d_{lead} = 50m$, $v_{host} = 10m/s$, $v_{lead} = 12m/s$, $d_{min} = 3.0m$, $h_{des} = 1.5s$

$d_{lead} \leftarrow Inf$
for surrounding moving cars $VS_{other}(i)$ **do**
  if $(abs(l_i - l_{host}) < w_{lane}$ & $s_i - s_{host} < d_{lead}$ & $s_i - s_{host} > 0)$ **then**
    Set $d_{lead} \leftarrow s_i - s_{host}$
    Set $v_{lead} \leftarrow VS_{other}(i).v$
  end if
end for

An LQR controller with gain scheduling is developed to generate the acceleration command for the host vehicle. The desired velocity is generated by adding the acceleration command to the last commanded velocity, as shown in Equation 4.

$$
\begin{aligned}
\Delta d &= d_{lead} - d_{desired} \\
\Delta v &= v_{lead} - v_{host} \\
acc_{cmd} &= k_d \Delta d + k_v \Delta v \\
v_{cmd}(t) &= v_{cmd}(t - \Delta t_{ACC}) + a_{cmd} \Delta t_{ACC}
\end{aligned} \tag{4}
$$

The kinematics of the host vehicle are simplified and modeled as a first-order system with pure time delay. Therefore, the discretized model for simulated speed of the host vehicle is computed using Equation 5.

$$v_{host}(t) = (1 - \tau) v_{host}(t) + \tau v_{cmd}(t - t_{delay}) \tag{5}$$

The delay and parameter of the first-order system are identified from data collected using a real autonomous vehicle, which will be discussed in Section IV-A. Figure 7 shows an example of the forward-simulated ACC controller, which also uses the maximum speed information from S_{road} to constrain the vehicle's maximum speed.

Using this model, for every time step, we can get a prediction of the host vehicle's speed given a certain candidate strategy.

b) Forward-Simulate Vehicle Lateral Controller: The autonomous vehicle lateral controller's rule is to minimize the error between current lateral offset l_{host} and desired lateral offset $l_{desired}$, which is generated in the candidate strategy generation step. The system's response can be simulated using Equation 6.

$$
\begin{aligned}
\Delta l &= max(-0.5, min(0.5, (l_{host} - l_{desired}))) \\
l_{host}(t) &= l_{host}(t - \Delta t_{lat}) + k_{lat} \Delta l \Delta t_{lat}
\end{aligned} \tag{6}
$$

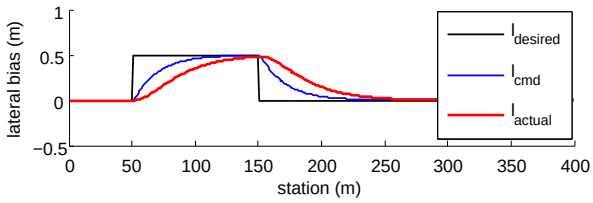


Fig. 8: Result of forward-simulating lateral controller of an autonomous vehicle with a square wave desired lateral offset strategy

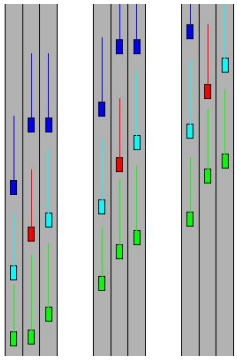


Fig. 9: Results of predicting all surrounding vehicles' movements including their reactions to the host vehicle's behavior: from left to right $t = 0.0, 4.0, 8.0$

Figure 8 shows an example of the forward-simulated ACC controller, which also uses the maximum speed information from S_{road} to constrain the vehicle's maximum speed. Although the input candidate strategy is discretized and not smooth, the forward-simulation creates a practical path the car will execute.

c) Predict Surrounding Vehicles' Reaction: For every step of the simulation, all the surrounding moving obstacles are also predicted based on the assumptions that they will tend to keep their current speed and be able to perform distance keeping to their leading vehicle. The prediction from $S_{PCB}(t)$ to $S_{PCB}(t + \Delta t)$ is described as below:

for moving cars $VS_{other}i$ **do**
  if (found the closest leader) **then**
    Simulate distance keeper
    Generate desired speed command
  else (no leading vehicle)
    Keep constant speed
  end if
  Update $VS_{other}i$
end for
Update V_{host}
  Forward-simulate ACC controller
  Forward-simulate lateral controller

By iteratively running the prediction, a series of simulated scenarios of S_{PCB} including VS_{host} and VS_{other} covering the prediction horizon $t_{predict}$ can be generated, as shown in Figure 9.

3) Cost function-based Evaluation: Exhaustive search is used to iterate through all candidate strategies and run

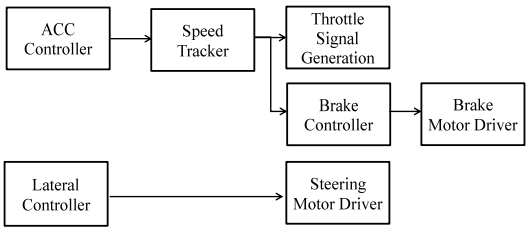


Fig. 10: The block diagram of the vehicle controller layer

prediction for each of them. For each scenario, the progress cost, comfort cost, safety cost and fuel consumption cost are calculated as shown in Equation 7.

$$C_{sce} = \mu_1 C_{progress} + \mu_2 C_{comfort} + \mu_3 C_{safety} + \mu_4 C_{fuel} \tag{7}$$

The cost for each strategy is the sum of the cost of the series of predicted scenarios, as shown in Equation 8.

$$C_{str(i)} = \sum_{t=0}^{t_{Predict}} (C_{sce(i,t)}) \tag{8}$$

The detailed descriptions of these cost terms are given in [17], [18].

One of the benefits of using the PCB architecture is that the surrounding vehicles' reactions to the host vehicle's movement are considered. The other benefit is that the vehicle controllers' reactions (including the lateral controller and distance keeping controller) and time delays are simulated. The planner can therefore take these into account in the cost function when it searches for the best strategy.

D. Vehicle Controller Layer

The last layer in the framework consists of multiple vehicle controllers running in the embedded controllers in the vehicle, including an adaptive cruise controller and vehicle lateral controller, as shown in Figure 10. Both of these controllers are modeled and forward-simulated in the behavioral planner layer.

The lateral controller takes the lateral offset output from the behavioral planner as a directive. It can therefore make the car drive along the road with desired lateral offset to perform lane changes or avoid static obstacles.

The adaptive cruise controller (ACC) uses control laws to compute a desired speed command based on the current state of the vehicle, leading vehicle information and control preferences. Then this velocity command is converted to throttle and brake pedal actuation commands by a speed tracking controller. Based on the control preferences h_{des} given by the upper-layer behavioral planner, it can perform aggressive distance keeping that keeps a closer distance to its leader or more conservative behavior that stays further away from its leader and applies milder acceleration.

IV. Testing Platform

To support future development of autonomous vehicle technology, including a more consumer-friendly appearance, human-like smooth vehicle control and intelligence, and advanced sensor fusion based on automotive-grade sensors, a

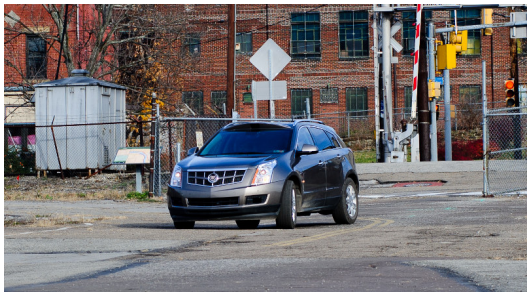


Fig. 11: The CMU autonomous vehicle research Platform in road test

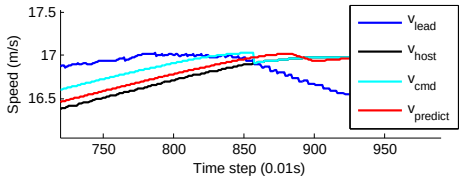


Fig. 12: Real autonomous vehicle data for system identification of the ACC controller

new autonomous driving platform has been built by Carnegie Mellon University (CMU) and is shown in Figure 11.

This autonomous driving research vehicle is a modified 2011 Cadillac SRX. The vehicle has been extensively tested on both a closed test field and public roads. The behavioral planning architecture proposed in this paper is implemented and tested on this platform in both simulation-mode and on the vehicle.

A. On Road Validation

Real vehicle testing has been done to verify the accuracy of the forward simulation in the behavioral planner, as shown in Figure 12. It can be seen that the simulated ACC model matches the actual velocity of the vehicle closely. The system identification of the vehicle's longitudinal and lateral responses to the behavioral planner's directive is shown in Table I.

TABLE I: System identification result based on data from the CMU Cadillac SRX [19]

Δt_{ACC} (s)	0.10
τ_{ACC}	0.13
t_{delay}	0.39
Δt_{lat} (s)	0.01
k_{lat}	0.10

V. Experimental Results

The framework has been implemented in the autonomous vehicle control software TROCS [9], which also works as a real-time autonomous vehicle simulator. The behavioral planner while planning at a higher level of abstraction, was more computationally efficient in pruning the search space than the previous motion planners [21], [8] that evenly sampled the spatio-temporal space. We compare the behavioral planners

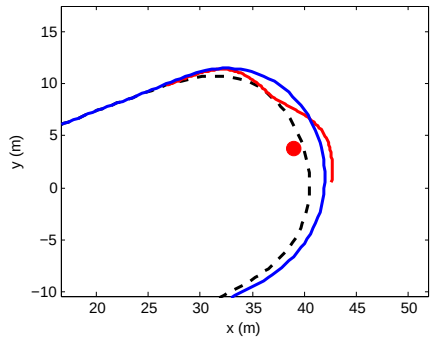


Fig. 13: Comparison of path planning results: black, center line of road; blue, behavioral planners output; red, spatio-temperal sample-based planners output

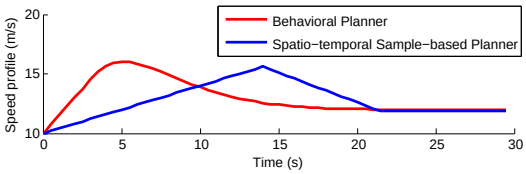


Fig. 14: Comparison between the best speed profile found by the behavioral planner and the saptio-temporal sample-based planner

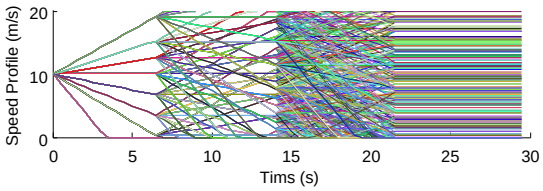
performance with a sample-based planner in the spatio-temporal space [21], [17] in two testing cases: path planning to avoid an obstacle on a curve and distance keeping.

1) Path planning: The sample-based planners path candidates usually are described by a polynomial. The heading and curvature for the end points of the trajectory need to be fixed [21] to constrain the polynomial. In cases wherein the autonomous vehicle drives on a curve and needs to perform evasive maneuvers, the constraints of the sample point limited the flexibility of candidate paths, thereby considerably affecting its performance. Instead of applying these arbitrary constraints, the behavioral planner uses lateral offset to direct the controller to avoid obstacles smoothly, as shown in Figure 13.

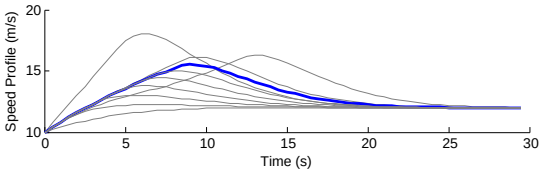
2) Velocity planning: Figure 14 shows the sample space for the normal planner applying searches on spatio-temporal grids, which covers a wide variety of possible speed profiles for the car to execute. In this scheme, the profiles, though piecewise linear, are non-smooth connections between discretized velocity sample points. Figure 15a shows the search space for the behavioral planner, which samples different candidate time headways. They have enough diversity, but more importantly, they all perform distance keeping to the leading vehicle, which is not true of all of the profiles in Figure 15a.

Table II compares the behavioral and spatio-temporal sample-based planners performances in searching for the best speed profile by computing the cost terms described in Section III-C.3 in 1,000 randomly generated traffic scenarios.

It can be seen that even with many fewer samples, the behavioral planners driving quality (the cost of the



(a) Speed profile search space for a spatio-temporal sample-based planner



(b) Speed profile search space in the behavioral planner

Fig. 15: Compared with the normal spatio-temporal planner, all the candidate profiles in behavioral planner's search space using different h_{des} are smoother and able to perform basic distance keeping behavior

TABLE II: Speed Profile Planner Performance Comparison

	Sample-based Planner	Behavioral Planner
Number of Samples	512	10
Feasible Samples (%)	73.40 ± 26.84	97.23 ± 3.23
Average Sample Cost	1349.91 ± 84.61	839.53 ± 480.76
Best Sample Cost	625.75 ± 285.04	608.15 ± 502.48
Computing Time (ms)	325.5 ± 23.1	31.6 ± 4.3

best sample) outperforms the spatio-temporal sample-based planners by 2.9%. The overall computational cost is also dramatically reduced, by over 90%. That is because the proposed behavioral planner does not need to search through the huge number of candidate profiles that the other planner does.

VI. Summary

In this paper, we propose a novel behavioral planning framework for fully autonomous vehicles. Using the framework, an autonomous vehicle can achieve functionalities such as lane keeping and lane changing; distance keeping; handling traffic lights, stop signs and yield signs; and avoiding obstacles. The key component of this framework is the behavioral planner, which uses the PCB algorithm to coordinate the ACC controller and lateral controller of the vehicle to perform high-quality distance keeping, lane changing and obstacle avoidance behaviors. For path planning, the behavioral planner does not need to use polynomial paths, as do spatio-temporal sample-based planners. Therefore, it generates much smoother paths in some complicated cases. In speed profile planning, a statistical test shows that the behavioral planner reduces computing time by 90.3% while achieving 5.1% higher quality with respect to smoothness, safety, and fuel consumption costs.

The framework has been fully implemented on an autonomous vehicle platform. Preliminary road testing shows the accuracy of the behavioral planners forward simulation of the autonomous vehicles movement. However, more intensive road testing and statistical analysis need to be performed to fully verify fully the frameworks implementation and performance .

References

[1] Traffic safety facts. *National Highway Traffic Safety Administration (NHTSA)*, 2008.

[2] M. Bertozzi, L. Bombini, A. Broggi, M. Buzzoni, E. Cardarelli, S. Cattani, P. Cerri, A. Coati, S. Debattisti, A. Falzoni, et al. Viac: An out of ordinary experiment. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 175–180. IEEE, 2011.

[3] E. D. Dickmanns. Vehicles capable of dynamic vision: a new breed of technical beings? *Artificial Intelligence*, 103(1-2):49–76, Aug. 1998.

[4] GM Press. Self-Driving Car in Cadillac's Future. `http://media.gm.com/media/us/en/cadillac/news.detail.html/content/Pages/news/us/en/2012/Apr/0420_cadillac.html` as of Oct 5, 2012.

[5] T. Gu, J. Snider, J. M. Dolan, and J.-w. Lee. Focused trajectory planning for autonomous on-road driving. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 547–552. IEEE, 2013.

[6] J. Leonard, D. Barrett, J. How, S. Teller, M. Antone, S. Campbell, A. Epstein, G. Fiore, L. Fletcher, E. Frazzoli, et al. Team mit urban challenge technical report. 2007.

[7] J. Markoff. Google cars drive themselves, in traffic. *The New York Times*, 10:A1, 2010.

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

[9] M. McNaughton, C. R. Baker , T. Galatali, B. Salesky, C. Urmson, and J. Ziglar. Software infrastructure for an autonomous ground vehicle. *Journal of Aerospace Computing, Information, and Communication*, 5(1):491 – 505, December 2008.

[10] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 25(9):569–597, 2008.

[11] M. Peden, R. Scurfield, D. Sleet, D. Mohan, A. A. Hyder, E. Jarawan, C. D. Mathers, et al. World report on road traffic injury prevention, 2004.

[12] D. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems 1*. Morgan Kaufmann, 1989.

[13] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, and J. O'Sullivan. A layered architecture for office delivery robots. In *Proceedings of the first international conference on Autonomous agents*, pages 245–252. ACM, 1997.

[14] S. Thrun, W. Burgard, D. Fox, et al. *Probabilistic robotics*, volume 1. MIT press Cambridge, 2005.

[15] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(1):425–466, June 2008.

[16] V. Vijayenthiran. Volkswagen Shows Off Self-Driving Auto Pilot Technology For Cars. MotorAuthority, 2011.

[17] J. Wei, J. Dolan, and B. Litkouhi. A prediction-and cost function-based algorithm for robust autonomous freeway driving. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 512–517. IEEE, 2010.

[18] J. Wei, J. Dolan, and B. Litkouhi. Autonomous vehicle social behavior for highway entrance ramp management. 2013.

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

[20] Q. Xu, K. Hedrick, R. Sengupta, and J. VanderWerf. Effects of vehicle-vehicle/roadside-vehicle communication on adaptive cruise controlled highway systems. pages 1249–1253, 2002.

[21] 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*, pages 2061–2067. IEEE, 2012.