2010 IEEE Intelligent Vehicles Symposium
University of California, San Diego, CA, USA
June 21-24, 2010

A Prediction- and Cost Function-Based Algorithm for Robust Autonomous Freeway Driving

Junqing Wei, John M. Dolan and Bakhtiar Litkouhi

Abstract— **In this paper, a prediction- and cost function-based algorithm (PCB) is proposed to implement robust freeway driving in autonomous vehicles. A prediction engine is built to predict the future microscopic traffic scenarios. With the help of a human-understandable and representative cost function library, the predicted traffic scenarios are evaluated and the best control strategy is selected based on the lowest cost. The prediction- and cost function-based algorithm is verified using the simulator of the autonomous vehicle Boss from the DARPA Urban Challenge 2007. The results of both case tests and statistical tests using PCB show enhanced performance of the autonomous vehicle in performing distance keeping, lane selecting and merging on freeways.**

I. Introduction

Since the 1980s, autonomous driving has gradually become a fast-developing and promising area. The abilities of autonomous vehicles have been extended from simple cruise control, among others, to adaptive cruise control, lane-keeping, intelligent route planning, off-road navigation, and interacting with human-driven urban traffic. Autonomous driving technology has the ability to provide driver convenience and enhance safety by avoiding some accidents due to driver error. However, a highly robust and intelligent fully autonomous distance keeping and merging vehicle that interacts with human-operated traffic on freeways requires further research and development. Therefore, the development of a reliable and powerful autonomous vehicle control model is a key in reaching the goal of fully autonomous driving on freeways.

II. Related Works

In the 1990s, E.Dickmanns and his colleagues implemented an autonomous driving platform based on their 4-D approach to vision data processing [1], [2]. The NAVLAB project at Carnegie Mellon University (CMU) has built a series of experimental platforms which are also able to run autonomously on freeways [3], [4]. In 2007, the DARPA Urban Challenge provided researchers a practical scenario in which to test the latest sensors, computer technologies and artificial intelligence algorithms [5]. Basic interaction between autonomous vehicles and human-driven vehicles was proven in low-density, low-velocity traffic.

This work was supported in part by General Motors through the GM-Carnegie Mellon Autonomous Driving Collaborative Research Laboratory. The authors also would like to thank the Tartan Racing Team in the DARPA Urban Challenge, who built the vehicle control platform and the simulator.

J. Wei is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA junqingw@cmu.edu
J. M. Dolan is with the Robotics Institute and the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA jmd@cs.cmu.edu
B. Litkouhi is with GM-CMU Autonomous Driving Collaborative Research Lab, GM R&D Center, Warren, MI 48090 bakhtiar.litkouhi@gm.com

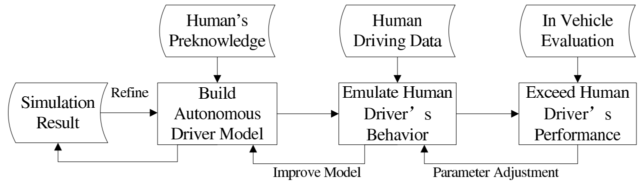

Fig. 1. Autonomous Driver Model Development Approach

In the Urban Challenge 2007, a rule-based driver model was implemented to control the autonomous vehicle Boss's behavior [6]. Though the rule-based algorithm is relatively easy to develop and verify, it is not robust enough to different traffic scenarios. It is also difficult to extend and configure, as there are a large number of rules which are correlated with one another.

Learning algorithms such as Artificial Neural Networks (ANN) have also been implemented in lane keeping and adaptive cruise control. The major constraint of learning algorithms is that they depend too much on training. Therefore, no safety criteria can be verified. They also restrict the autonomous vehicle's ability to exceed a human driver's performance.

A market-based approach is also applicable to autonomous vehicle control. Costs are computed and assigned to each potential strategy, and the algorithm outputs the strategy corresponding to the lowest cost. The market-based model makes it easy to implement, maintain and check safety criteria. However, there is no well-proven autonomous driver model based on a market-based approach due to its relatively high computational expense. In addition, most cost function-based approaches depend heavily on the heuristic cost, which is subjective and difficult to tie unambiguously to physical parameters and models.

Based on previous research, we propose a comprehensive approach to autonomous driver model development, which is shown in Figure 1. Through this development process, our autonomous driver model will be able to better interact with human traffic by emulating human driving behavior. It also retains the autonomous vehicle's potential to exceed human driver performance.

In the first step of the development process, a prediction- and cost function-based algorithm for autonomous freeway driving was created [7]. In [7], the autonomous driver model

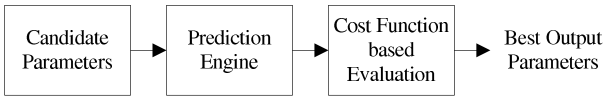

Fig. 2. Prediction- and Cost Function-Based Algorithm

was built by emulating a human driver's decision process, including prediction and evaluation. By using prediction, we reduced the difficulty of building long-term heuristic costs. The evaluation was based on a human-understandable cost function library, which was easy to extend. The algorithm was also highly reconfigurable so that it can be adjusted to perform different driving styles from conservative to aggressive.

The main advances of the current work with respect to [7] are:

- an enhanced prediction model
- more representative and human-understandable cost functions
- comprehensive statistical and case tests to verify the algorithm
- parameter adjustment based on more rigorous case tests

Through these improvements, PCB gains greater reliability, adaptablity, and robustness. In this paper, the parameters in the algorithm are still adjusted manually by experiments and some basic understanding of human driving. As the better cost function library reduces the number of parameters in PCB by 40%, it significantly simplifies both manual and learning-based parameter adjustment and performance optimization. The performance of PCB is also proven according to case tests and statistical analysis.

III. Autonomous Freeway Driving Control Modules

In PCB, we separate the freeway driving ability into three modules, see [8]. These are the distance keeper, lane selector and merge planner. The role of the distance keeper is to keep a reasonable distance from the leading vehicle. The lane selector outputs the intended lane, i.e., the lane the autonomous vehicle wants to merge into. Whenever the intended lane is different from the current lane, the merge planner will be triggered. It adjusts the vehicle's position and speed to find a best merging opportunity.

IV. Prediction- and Cost function-Based Algorithm

A. Algorithm Framework

A diagram of PCB is shown in Figure 2.

In the candidate parameter generation step, a set of candidate strategies is generated. In the distance keeper module, for example, the generator produces 20 different acceleration values that range from $-3.0m/s^2$ to $1.8m/s^2$. Then the strategy set and the map of the current moving vehicles are sent to the prediction engine, which generates a series of simulated scenarios in the following *t* seconds. The cost function-based evaluator then begins to compute the cost of each strategy for each scenario.

By using this mechanism, we successfully separate the complicated behavior strategy generation process into two relatively independent parts. The prediction engine only considers how to accurately generate simulated scenarios, while the cost function block implements and imitates a human driver's evaluation of a given scenario.

B. Prediction Engine

In general, when human drivers operate vehicles on the freeway, they can interact efficiently with one another by showing their intentions and also recognizing/predicting other vehicles' trajectories. This mechanism provides experienced human drivers enough time to prepare and make suitable maneuvers in advance. To implement this look-ahead ability in an autonomous vehicle, we built a prediction engine in our driver model.

To accurately predict the vehicles' movements, we need to consider surrounding vehicles' reactions to their own microscopic traffic environment. For instance, if one vehicle runs faster than its leader, instead of maintaining constant velocity it will slow down as it gets close to it. We therefore introduce an interactive prediction kernel with basic distance-keeping ability into this engine described in the following algorithm, in which $v(t)$ is the absolute velocity at prediction time t, $d(t)$ is the relative distance to the host vehicle and $D_{minSafetyDistance(i)}$ is a precomputed minimum safety distance to a leading vehicle for each $vehicle_i$.

```
1: Update autonomous vehicle's state
2: $v_0(t+\Delta t) = v_0(t) + a_{cmd}\Delta t$
3: Update surrounding vehicles' states
4: **for** Each surrounding vehicle **do**
5:     $d_i(t+\Delta t) = d_i(t) + (v_i(t) - v_0(t))\Delta t$
6:     Enforce restrictions
7:     $v_i(t+\Delta t) = min(v_i(t+\Delta t), SpeedLimit)$
8:     **if** $D_{i,i_{lead}}(t+\Delta t) < D_{minSafeD(i)}$ **then**
9:         $v_i(t+\Delta t) = leadingVehicleSpeed_i$
10:        $d_i(t+\Delta t) = d_{i_{lead}} - D_{minSafeD(i)}$
11:    **end if**
12: **end for**
```

The prediction engine simplifies the modeling complexity for producing agile reactions to complicated and hard-to-predict traffic. By using the prediction engine, human-like prediction ability is emulated in PCB, which helps select better control strategies.

C. Cost Function-Based Evaluation

After predicting a number of scenario sequences corresponding to these strategies, a cost is computed for each of them. In the computation, all the predicted scenarios from $t = 0$ to $t = t_{predictLength}$ are evaluated, as shown in Equation 1.

$$C_{strategy(i)} = \sum_{t=0}^{t_{PredictLength}} C_{scenario(i,t)} \tag{1}$$

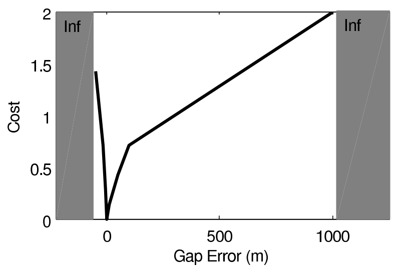

Fig. 3. Distance keeper progress cost: $C_{DKpro}(\Delta d_{gap})$, with vertices (-25,1.5), (-15,0.9), (-5,0.14), (0,0), (10,0.14), (50,0.43), (100,0.7), (1000,2)

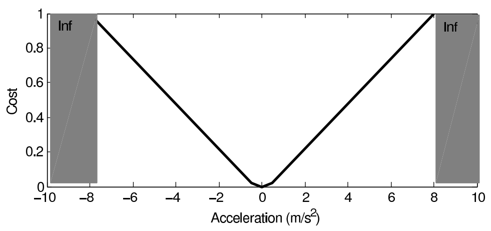

Fig. 4. Comfort cost: $C_{comfort}(acc)$, with vertices (-8,1), (-0.5,0.02), (0,0), (0.5,0.02), (8,1)

Sometimes, the surrounding vehicles' actual behaviors are different from the prediction engines estimate. Using this mechanism, the strategy is selected according to not only the predicted results, but current safety conditions, as well. Therefore, it improves PCB's robustness to prediction errors. In most scenarios, the prediction engine helps improve the performance of the control model without degrading the safety.

1) Cost Function Parameterization: There are multiple ways of cost function representation, including linear and polynomial models, etc. In PCB, we use a vertices-based cost function parameterization. For each cost function, a set of coordinates $(x_0, y_0), (x_1, y_1)...(x_n, y_n)$ is defined, in which $x_0 < x_1 < ... < x_n$. The cost function is then linearly interpolated between neighboring vertices. With this representation x_0 and x_n restrict the available input range, which can be used as a safety criteria check. In the implementation, x_0 and x_n are determined by both the physical constraints and the safety rules of the vehicle. The cost for an unacceptable input will be infinite, and its corresponding strategy will not be used. If there is no available strategy for selecting, the autonomous driver model will command the vehicle to slow down.

There are three kinds of costs built to evaluate the strategies. They are the progress cost, comfort cost and safety cost.

2) Progress Cost: The progress cost represents how well a strategy does in finishing a given task by penalizing those strategies which take longer to finish the task.

- Distance Keeper
  The goal of the distance keeper is to keep the distance as close as possible to $d_{desired}$, which is computed in Equation 2, in which v is the current velocity of the vehicle.
  $$d_{desired} = D_{min} + k_{vgain}v \tag{2}$$
  The progress cost is then computed according to the difference between the current distance to the leading vehicle ($d_{current}$) and the desired distance to it ($d_{desired}$), which is Δd_{gap}. The distance keeper progress cost C_{DKpro} is shown in Figure 3, in which the gray area has infinite cost. To make the distance keeper work when the leading vehicle does not exist, a virtual vehicle running at the speed limit is inserted at 150 meters away if there is no leader in the lane. This prevents the progress cost from jumping a lot when the leading vehicle is close to our maximum sensing range and is not well detected.
- Lane Selector
  Its progress cost is based on the prediction of potential time benefits from a merging maneuver. A virtual destination is placed 200 meters away for us to compute the estimated arrival time $t_{arrival}$. An alternative lane keeping prediction engine is built to compute $t_{arrival}$, which is described as the following algorithm.

```
1: while distance traveled $< 200m$ do
2:     if leading vehicle distance $> 20m$ then
3:         Catch up with $acc = 1.8m/s^2$
4:     else
5:         Follow the leading vehicle
6:     end if
7: end while
```

  Then a linear cost is built.
  $$C_{LSpro} = t_{arrival} \tag{3}$$
- Merge Planner
  The merge planner's rule is to execute some adjustment and perform the merge at a reasonable opportunity. Therefore, we define the merge planner progress cost according to the merge execution time t_{finish} and its finishing distance d_{finish}, as shown in Equation 4.
  $$C_{MGpro} = \mu_1 t_{finish} + \mu_2 d_{finish} \tag{4}$$

3) Comfort Cost: While driving a car, experienced human drivers will try to avoid large accelerations for greater comfort. Therefore, a comfort cost $C_{comfort}$ is built to represent this logic, as shown in Figure 4.

4) Safety Cost: The safety cost of a scenario consists of two terms: the clear distance cost and the braking distance cost. The clear distance cost $C_{distance}$ penalizes moving too close to surrounding vehicles. We first perform normalization on the input clear distance, which makes the minimum clear distance smaller when the vehicle speed is low. The minimum clear distance for low speed is computed by Equation 5.

$$d_{minClear} = 2 + 0.5v \tag{5}$$

$$d_{norm} = 15/min(15, d_{minClear}) * d_{input} \tag{6}$$

The cost function of d_{norm} is shown in Figure 5.

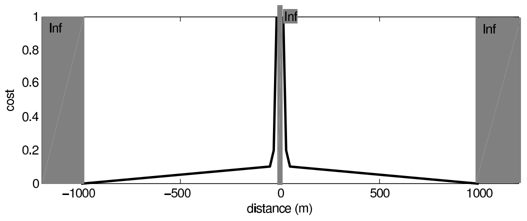

Fig. 5. Clear distance cost: $C_{distance}(distance)$, with vertices (-1000,0), (-50,0.1), (-30,0.2), (-15,1), (15,1), (30,0.2), (50,0.1), (1000 , 0)

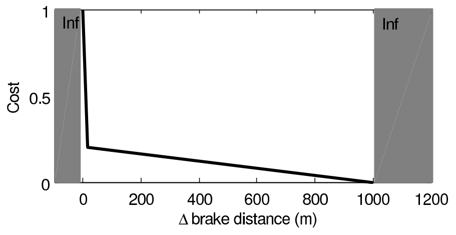

Fig. 6. Braking distance cost: $C_{brake}(brakeDistance)$, with vertices (0,1), (15,0.2), (1000,0)

However, this cost is not informative enough for us to avoid collision in some situations, since it does not consider the vehicles' velocities. Therefore, another safety cost based on the braking distance difference Δd_{brake} between two vehicles is computed.

$$\Delta d_{brake} = d_{lead} + \tfrac{1}{2}v_{lead}^2/a_{maxdec} - v_{follow} * t_{response} - \tfrac{1}{2}v_{follow}^2/a_{maxdec} \quad (7)$$

In Equation 7, the remaining distance after braking is computed. Δd_{brake} smaller than 0 means that if the leading vehicle begins to brake hard, we do not have enough space to avoid the collision, which is obviously not acceptable. The cost C_{brake} based on Δd_{brake} is shown in Figure 6. Then the overall safety cost is computed using Equation 8:

$$C_{safety} = \mu_3 C_{brake}(\Delta d_{brake}) + \mu_4 C_{distance}(d) \quad (8)$$

V. Implementation

A. Distance Keeper

In the PCB distance keeper, the candidate strategies are a set of different accelerations, chosen as $\{-3.0, -2.4, -1.8, \cdots 0.0 \cdots 1.2, 1.8\}$ in this paper. For each acceleration candidate, the prediction engine is run for 10 steps with a 0.6-second step length, which provides us 6 seconds look-ahead ability. The cost for each scenario C_{sce} is computed using Equation 9.

$$C_{sce} = \mu_5 C_{DKpro} + \mu_6 C_{comfort} + \mu_7 C_{DKsafety} \quad (9)$$

In Equation 9, C_{DKpro} is computed with Equation 3, and C_{safety} is the safety cost of the leading vehicle. Then the overall cost for each strategy is computed using Equation 1.

B. Lane Selector

For the lane selector, there are three candidate output lanes: left(L), current(C) and right(R). Therefore, three strategy costs are computed using Equation 11, in which $C_{LSsafety}$ is the sum of the costs corresponding to each car in the intended direction.

$$C_{strategy} = C_{LSpro} + \mu_8 C_{LSsafety} \quad (10)$$

$$C_{LSsafety} = \sum_{i}^{vehicles\ in\{L,C,R\}} C_{safety}(vehicle(i)) \quad (11)$$

After computing the strategy costs, instead of directly comparing the cost of those strategies, a hysteresis block is built to switch between strategies. For example, when the vehicle is running straight, it will only select the left or right lane if the benefit is larger than the hysteresis's upper threshold. This limits oscillation in lane selection. The hysteresis thresholds for the left and the right lane changes are also different so that the vehicle prefers left-overtaking when both left and right lanes are available.

C. Merge Planner

The merge planner strategy consists of three parameters, a_1, a_2 and t_{adj}. Based on these three parameters, it will first perform an acceleration a_1 for $\frac{1}{2}t_{adj}$ and then a_2 for another $\frac{1}{2}t_{adj}$ seconds . For the accelerations a_1 and a_2, we are using the same candidate set as the distance keeper. Candidate set $\{0, 0.2, 0.4, \cdots 6\}$ is used for t_{adj}. Using these strategies, both the speed and the position can be adjusted while looking for the best merge opportunity. The cost for each merge strategy is computed with Equations 12 and 13, in which $C_{mergeSafety}$ is the sum of safety costs of the host vehicle and all the cars in the intended lane and C_{safety} is the cost of the leading vehicle.

$$C_{strategy} = \sum_{t=0}^{PredictLength} C_{sce(t)} + \mu_9 C_{mergeSafety} \quad (12)$$

$$C_{sce(t)} = \mu_{10} C_{MGpro} + \mu_{11} C_{comfort} + \mu_{12} C_{safety} \quad (13)$$

After adjustment, the merge planner need to trigger the actual merging manuever. The basic rule is that whenever the t_{adj} of the best strategy is smaller than a threshold, the vehicle will be commanded to perform a lane change. A hysteresis block is also used here to avoid output oscillation.

D. PCB Model Summary

As already described, 6 different cost functions are built in PCB with 17 vertices and 12 weight parameters. Compared to the previous robust freeway driving algorithm developed in [7], the number of parameters is reduced by around 40%, which efficiently simplifies the parameter adjustment. With the more human-understandable, heuristic and less redundant cost functions, PCB is more robust and easier to configure.

VI. Performance Evaluation

Two different testing mechanisms were implemented to evaluate the PCB's performance. First, five case tests that emulated different special traffic scenarios were built. The case tests were designed to verify the PCB's functionality qualitatively and help us preliminarily adjust those parameters. Then, an integration test combination of the case tests on a 30 kilometer-long three-lane freeway enabled us to gain

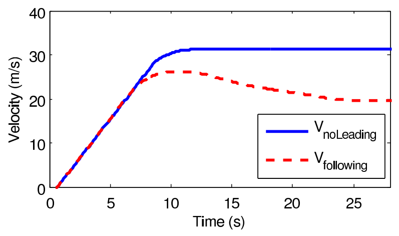

Fig. 7. Lane Keeping with/without Leading vehicle, Speed Limit is 31.3m/s, $V_{lead} = 27.6m/s$

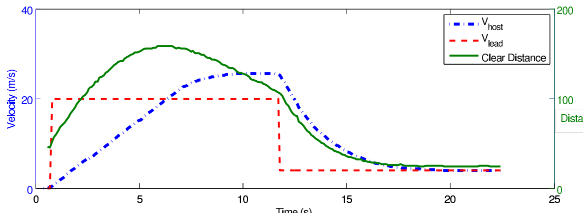

Fig. 8. Emergency Brake

statistical performance data on PCB. The statistics of velocities, accelerations, distance to leading vehicle, etc. were extracted from these test runs for a performance evaluation.

The parameters used in the tests were:

$\Delta t = 0.6$	$N_{predictStep} = 10$	
$D_{min} = 5.0$	$k_{vgain} = 1.14$	$\mu_1 = 8.0$
$\mu_2 = 2.0$	$\mu_3 = 1.0$	$\mu_4 = 2.0$
$\mu_5 = 50.0$	$\mu_6 = 10.0$	$\mu_7 = 10.0$
$\mu_8 = 0.01$	$\mu_9 = 200.0$	$\mu_{10} = 1.0$
$\mu_{11} = 1.0$	$\mu_{12} = 100.0$	

A. Case Tests

1) Lane Keeping with and without Leading Vehicle: In this test, the autonomous vehicle starts stationary and speeds up gradually to keep a safe distance to the leader or achieve the speed limit of the lane when there is no leading vehicle, as shown in Figure 7.

2) Emergency Brake: This scenario is built to test the autonomous vehicle's response when the leading vehicle does not behave as the prediction engine predicted. As shown in Figure 8, the leading vehicle keeps constant velocity for a few seconds, then suddenly brakes with its largest deceleration. The velocity of the host vehicle and the distance to the leader is shown in Figure 8. Through this test case, PCB's collision avoidance behavior is shown to be functional.

3) Forced Lane Change: This scenario is to test PCB's strategy when the vehicle is commanded to merge into the other lane as soon as possible. The merge progress cost C_{MGpro} will push the vehicle to perform a lane change with minimum adjustments. In the test scenario, the host vehicle is moving at high speed in its own lane and is commanded to merge into the neighboring lane, which has slow and dense traffic. This scenario emulates, for example, the conditions when we are close to our destination exit on a freeway. The coordinate of Figure 9 is bound to traffic vehicles which are of same and constant speed, so the circles show the

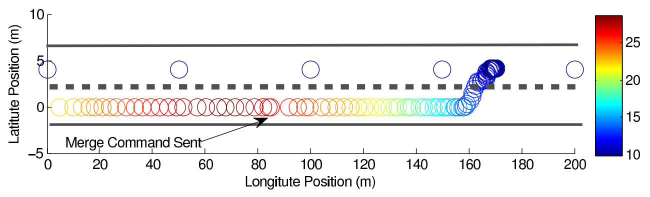

Fig. 9. Forced Lane Change

autonomous vehicle's relative position, the color of which represents the speed. PCB's merge strategy is to decelerate to a proper speed while looking for the best opportunity of merging, which is reasonable and performs well.

4) Freeway Entrance : This test is to verify the algorithm's strategy of looking for an opportunity and entering a freeway. The simulated vehicles in the neighboring lane are of the same speed. The distances between them are of Gaussian distribution. PCB causes the host vehicle to first predict the potential gaps for it to merge into and then accelerate to a proper speed to perform a smooth lane change. To the extent that an autonomous vehicle's sensors are good at identifying the speed, position and dynamics of vehicles in the neighboring lanes, its performance of entering a freeway could be better than that of human drivers .

5) Vehicle Merge In/Out: In this scenario, the host vehicle is running on a straight road with a few simulated vehicles merging into/out of its lane in front of it. With the prediction ability, the host vehicle is able to slow down properly before the vehicle enters its lane and accelerate immediately when it exits the lane.

B. Integration Tests

Several three-lane freeways were built to test PCB's overall performance. Simulated human traffic capable of simple distance keeping and lane changes was also added into the simulator. Both velocity and distance between simulated vehicles are of Gaussian distribution with parameters: $\mu_v, \sigma_v, \mu_d, \sigma_d$.

We first compared PCB's performance with the previous robust freeway driving algorithm (RFD) [7] and the algorithm we used in the Urban Challenge 2007 (UC07) [8]. The two previous algorithms are designed for driving less than 30 mph , which is also the speed limit of the Urban Challenge. We built a 20 kilometer-long road with a 30 mph speed limit. For each algorithm, we ran the simulation with $\mu_d = 40, 80, 120$ and $\sigma_d = 0.3\mu_d$. For each μ_d, the tests were run five times to get the statistical result. The speed parameters of the simulated traffic were $\mu_v = 10, \sigma_v = 3$. PCB performance in different scenarios and the comparison between the current PCB, the previous PCB algorithm developed and the algorithm from Urban Challenge 2007 are shown in Table I, in which $N_{laneChange}$ is the number of lane changes, acc_{ave} is the average acceleration, $t_{arrival}$ is the arrival time , $t_{D<10m}$ is the amount of time that the distance to the leading vehicle is less than 10 meters and $t_{brakeD<10m}$ is the amount of time the braking distance is less than 10 meters.

TABLE I
SIMULATION RESULTS WITH $\mu_v = 12m/s$

	UC07	RFD	PCB
$\mu_d = 40, \sigma_d = 12$			
$N_{LaneChange}$	151.0 ± 72.1	30.4 ± 2.9	33 ± 4.3
$acc_{ave}(m/s^2)$	0.80 ± 1.15	0.10 ± 0.02	0.10 ± 0.01
$t_{arrival}(s)$	2476 ± 66	1964 ± 105	1980 ± 216
$t_{D<10m}(s)$	0.65 ± 0.92	4.70 ± 1.55	0.00 ± 0.00
$t_{brakeD<10m}(s)$	0.00 ± 0.00	8.82 ± 0.42	0.00 ± 0.00
$\mu_d = 80, \sigma_d = 24$			
$N_{LaneChange}$	93.2 ± 35.4	28.8 ± 2.8	38.0 ± 5.7
$acc_{ave}(m/s^2)$	0.75 ± 1.10	0.09 ± 0.34	0.11 ± 0.35
$t_{arrival}(m/s)$	2199 ± 328	1924 ± 150	1736 ± 77
$t_{D<10m}(s)$	0.05 ± 0.71	6.23 ± 6.64	0.00 ± 0.00
$t_{brakeD<10m}(s)$	0.00 ± 0.00	17.32 ± 11.03	0.40 ± 0.42
$\mu_d = 120, \sigma_d = 36$			
$N_{LaneChange}$	124.6 ± 48.1	19.4 ± 1.5	29.4 ± 0.8
$acc_{ave}(m/s^2)$	0.77 ± 1.05	0.06 ± 0.24	0.07 ± 0.25
$t_{arrival}(m/s)$	1992 ± 151	1853 ± 207	1684 ± 125
$t_{D<10m}(s)$	0.00 ± 0.00	0.95 ± 0.49	0.00 ± 0.00
$t_{brakeD<10m}(s)$	0.42 ± 0.56	8.50 ± 4.67	0.55 ± 0.21

Table I shows that, compared to the Urban Challenge algorithm (UC07), RFD and PCB provide higher average speed with significantly fewer lane changes, which means the lane change decision is smarter. Additionally, due to the comfort cost, RFD and PCB prefer to output lower acceleration while driving. In PCB, the number of parameters is reduced by 40% by using understandable and more informative cost functions; however, the performance is not sacrificed compared with RFD . In those tests with $\mu_d = 80$ and $\mu_d = 120$, PCB performs about 10 more lane changes and arrives at the destination about 5 minutes earlier than RFD, which means that PCB is able to circumvent low-speed vehicles by performing more intelligent lane changes and merging maneuvers. In the previous freeway driving algorithm, there was no braking distance computed in the safety cost. Therefore, there are about 30 seconds in the test with braking distance less than 10 meters, which could be dangerous. In PCB with consideration of braking distance, these situations are effectively eliminated.

The high-speed performance of the PCB algorithm was also evaluated with another integration test. A 40 kilometer-long road with a 65 mph speed limit was built to simulate the highway traffic. We also ran these tests with three different traffic environments: $\mu_d = 40, 80, 120, \sigma_d = 0.3\mu_d, \mu_v = 20, \sigma_v = 6$. The results are shown in Table II.

Neither the UC07 nor RFD is able to pass the high-speed driving test. The algorithm in Urban Challenge does not have efficient prediction ability, so it can not slow down early enough to avoid collisions at high speed. The RFD does not take the velocity of vehicles into account while evaluating the safety. Therefore, when the speed limit increase from $30miles/hour$ to $65miles/hour$ algorithms, they are not functional anymore. From Table II, we see that PCB can be used for high-speed freeway driving with different traffic distributions .

TABLE II
PCB ALGORITHM SIMULATION RESULTS WITH $\mu_v = 20m/s$

	$\mu_d = 40$	$\mu_d = 80$	$\mu_d = 120$
$N_{LaneChange}$	28.5 ± 9.2	40.0 ± 5.7	46.5 ± 3.5
$acc_{ave}(m/s^2)$	0.28 ± 0.50	0.34 ± 0.50	0.36 ± 0.60
$t_{arrival}(s)$	1966 ± 221	1974 ± 95	1761 ± 39
$t_{D<10m}(s)$	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
$t_{brakeD<10m}(s)$	59.3 ± 74.3	141.2 ± 24.6	95.4 ± 55.9

VII. CONCLUSION

In this paper we proposed an extended Prediction- and Cost function-Based algorithm (PCB) and showed its ability to perform freeway driving in autonomous vehicles. With a more efficient prediction engine and human-understandable and informative cost functions, the overall performance of freeway driving is improved compared to previous algorithms. This algorithm is also able to control the vehicle at high speed on freeways. The algorithm was tested in five case tests designed to verify its functionality and around 500 kilometers of integration tests with different speeds and traffic environments to evaluate its performance. The tests show that PCB has better robustness, smoothness and intelligence in controlling autonomous vehicles behaviors on freeways. In future work, we will compare PCB's performance with that of human drivers and test it using autonomous vehicle. Based on that, further improvements and optimization can be made allowing the autonomous driver exceeds human driver performance with the help of autonomous controller's smaller response time and better sensing ability. In summary, PCB has strong potential in implementing and improving autonomous vehicle freeway driving ability.

REFERENCES

[1] E. D. Dickmanns, "Vehicles capable of dynamic vision: a new breed of technical beings?" *Artificial Intelligence*, vol. 103, no. 1-2, pp. 49–76, Aug. 1998.

[2] R. Gregor *et al.*, "Ems-vision: a perceptual system for autonomous vehicles," *IEEE Journal of ITS*, vol. 3, no. 1, pp. 48–59, March 2002.

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

[4] C. Thorpe, M. Herbert, T. Kanade, and S. Shafer, "Toward autonomous driving: the cmu navlab. i. perception," *IEEE Expert*, vol. 6, no. 4, pp. 31–42, Aug. 1991.

[5] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[6] C. Baker and J. Dolan, "Traffic interaction in the Urban Challenge: Putting boss on its best behavior," in *International Conference on Intelligent Robots and Systems (IROS 2008)*, 2008, pp. 1752–1758.

[7] J. Wei and J. M. Dolan, "A robust autonomous freeway driving algorithm," *2009 IEEE Intelligent Vehicle Symposium (IV2009)*, 2009.

[8] C. R. Baker and J. M. Dolan, "A case study in behavioral subsystem engineering for the Uuban Challenge," *IEEE RAM Special Issue on Software Engineering in Robotics*, 2008.