A Robust Autonomous Freeway Driving Algorithm

Junqing Wei
Department of Electrical and
Computer Engineering
Carnegie Mellon University, USA
Email: junqingw@andrew.cmu.edu

John M. Dolan
The Robotics Institute &
Department of Electrical and Computer Engineering
Carnegie Mellon University, USA
Email: jmd@cs.cmu.edu

***Abstract*—This paper introduces a robust prediction- and cost-function based algorithm for autonomous freeway driving. A prediction engine is built so that the autonomous vehicle is able to estimate human drivers' intentions. A cost function library is used to help behavior planners generate the best strategies. Finally, the algorithm is tested in a real-time vehicle simulation platform used by the Tartan Racing Team for the DARPA Urban Challenge 2007.**

I. INTRODUCTION

A. Background

Starting in the 1980s, autonomous driving has gradually become a fast-developing and promising area. The abilities of autonomous vehicles have been extended from lane-centering to intelligent route planning, off-road navigation and interacting with human urban traffic. Autonomous freeway driving is one of the most promising applications of robotics in the next few decades. Statistics show that over one million people were killed worldwide in traffic accidents in 2007. The risk of death in a freeway accident is much higher than in other traffic environments. Most accidents on freeways are caused by tired or careless drivers or mistaken maneuvers in emergency conditions. These can be efficiently avoided with an autonomous driving system. Another advantage of autonomous freeway driving is its ability to free people to perform tasks other than driving while traveling long distances. In a word, autonomous freeway driving technology has the potential to significantly improve the quality of people's lives.

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 CMU has built a series of experimental platforms which are also able to run autonomously on freeways [3], [4], but do not include lane selecting, merging behavior and the ability to blend into the human traffic environment. Rahul Sukthankar introduced evolutionary algorithms into autonomous freeway driving behavior control in the 1990s [5], [6]. A three-level lane-selecting algorithm was used by Jun Miura et al., and estimated arrival time and event-triggered computation were used to handle a simulated uncertain freeway traffic environment [7]. However, none of these systems was designed for practical road testing. Many system parameters, such as the latency, vehicle dynamic performance, input errors and sensor ranges were not fully considered. In analyzing human driver behavior, Ahmed et al. focused on building a mathematical model to simulate human driver acceleration and lane-changing behavior [8], [9]. They successfully verified this modelthrough real traffic data. However, their models' inputs are not fully compatible with the perception ability of our autonomous vehicle. And if we directly port their algorithms to our autonomous vehicle, the system may lose the potential to exceed human driving performance. In 2007, the DARPA Urban Challenge provided researchers a practical scenario in which to test the latest sensors, computer technologies and artificial intelligence algorithms [10]. Basic interaction between autonomous vehicles and human-driven vehicles was proven in low-density, low-velocity traffic. But another challenging functionality, that of freeway driving, was not included in the challenge. To finish the competition, most teams used rather conservative algorithms: their vehicles preferred to avoid difficult maneuvers in high-density traffic by stopping and waiting for a clear opening instead of interacting with it and operating the vehicle and human drivers. Compared to this previous work, our research can be summarized as follows.

1) Prediction Engine: We build an intelligent prediction engine with the ability to realizing the intentions of surrounding vehicles, which provides the autonomous vehicle a look-ahead ability similar to that of human drivers.

2) Cost Function-Based Scenario Evaluation: We construct a cost function library which is called by behavior modules to evaluate the predicted scenarios and generate best strategies.

3) Prediction- and Cost Function-based Algorithm: A prediction- and cost function-based algorithm is implemented in three behavior modules related to freeway driving ability, distance keeping, lane selecting and merge planning, which improves the performance and robustness.

4) Freeway Driving Performance Analysis: We build a hybrid performance analysis tool. It combines the advantages of both qualitative and quantitative performance evaluation.

III. SYSTEM FRAMEWORK

A. Software Platform

Because the autonomous driving systems are rather complicated and autonomous freeway driving tests can be dangerous, it is indispensable to simulate algorithms before conducting real freeway tests. In the DARPA Urban Challenge 2007,

978-1-4244-3504-3/09/$25.00 ©2009 IEEE

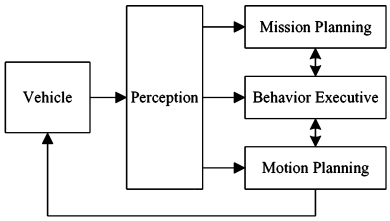

Fig. 1. TROCS Framework

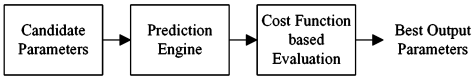

Fig. 2. Prediction and Cost Function Based Algorithm

the Tartan Racing team built a real-time vehicle controller called TROCS (Tartan Racing Operator Control System) [11]. Most of the artificial intelligence we used in the competition was developed and tested using the TROCS simulator, so its performance and accuracy have been well proven. In order to implement the new robust freeway algorithms, we decided to use the TROCS simulator for better testability and compatibility with previous work. There are four primary subsystems in TROCS, as shown in Figure 1. The perception system analyzes real-time data input from LIDAR, radar and GPS sensors. The role of mission planning is to optimize the path to achieve different checkpoints considering the arrival time and distance. In the behavior executive system we use artificial intelligence to control the vehicle's behaviors and interactions with traffic. Motion planning executes the behavior command while considering the dynamic parameters and outputting steering and throttle commands.

In developing the behavior control algorithm, we used the simulation mode of TROCS. In this mode, TROCS will simulate the perception output and also the traffic environment, so that we can test our algorithm accurately in simulation. After the simulation, the tested algorithm can be directly ported to the vehicle for practical road tests.

IV. PREDICTION- AND COST FUNCTION- BASED ALGORITHM

A. Algorithm Framework

The diagram of this algorithm is shown in Figure 2. There are three primary steps in this algorithm: candidate parameter generation, prediction, and scenario evaluation. In the candidate parameter generation step, a set of candidate output parameters is generated. In the distance keeper module, for example, the generator produces 20 different acceleration values ranging from -3.0m/s to 3.0m/s. Then the parameter set as well as 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. For instance, if we set the prediction interval to 0.3 seconds and predict for 10 steps, we will get the predicted scenario of the next 3 seconds. The cost function-based evaluation block then begins to compute the cost value of each scenario, which represents the performance of a corresponding input parameter. 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.

Fig. 3. Abstract Vehicle Map

B. Prediction Engine

While human drivers operate vehicles on the freeway, they can react and communicate efficiently with each other by showing their intentions and also recognizing other vehicles' intentions. This mechanism provides experienced human drivers enough time to prepare and make suitable maneuvers in advance. To implement this look-ahead ability in autonomous driving, we build two prediction engines into our algorithm. For better prediction performance and lower computing cost, we summarize the prediction engine's input into a single structure, the Abstract Vehicle Map (AVM), which represents the micro traffic environment. As shown in Figure 3, in this structure, the distances and velocities of the 8 vehicles around us are considered. There are three reasons of choosing this 8-vehicle Abstract Vehicle Map. First, it gives the minimum number of vehicles we should consider without adding unnecessary additional vehicles. It thus efficiently represents the micro traffic environment around a vehicle, especially on freeways. Second, in Rahul Sukthankar's research [5], [6] a similar abstract input was proven to be feasible in tactical controlling. Finally, it fits the autonomous vehicles' sensing range and ability well. After abstracting the input of the prediction engine, we use two different assumptions described below to implement the prediction engine and compute the simulated AVM results.

1) Constant-Velocity AVM: Due to the uncertainty of the traffic environment, the longer the prediction period, the larger the error. We therefore restrict the prediction time to less than five seconds, which is also similar to human driver's prediction ability. Because most drivers keep their velocities relatively constant on freeways, we just assume that the surrounding vehicles' velocities in the AVM are constant. Based on this, we build the prediction kernel equation in the prediction engine.

$$v(i+1) = v(i) + a_{cmd}\Delta t \quad (1)$$

$$d(i+1) = d(i) + v(i)\Delta t \quad (2)$$

In Equation 2, $v(i)$ is the relative velocity array of surrounding vehicles, a_{cmd} is the commanded acceleration of our autonomous vehicle, and $d(i)$ is the relative distance of the surrounding cars in the AVM. The prediction model thus assumes that our autonomous vehicle is static while other surrounding vehicles move relative to it.

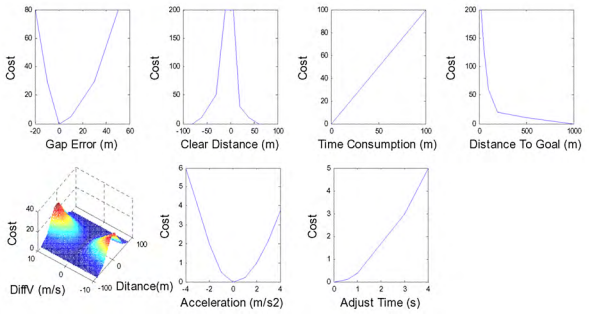

Fig. 4. Cost Function Library

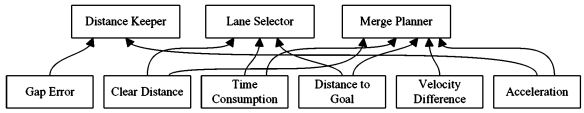

Fig. 5. Cost Function Library Dependencies

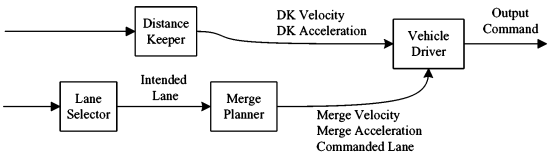

Fig. 6. Freeway Driving Modules

2) Interactive AVM: Though the constant-velocity assumption is effective in most freeway scenarios, to accurately predict the vehicles' movement we also need to consider surrounding vehicles' reactions to their own micro traffic environment. For instance, if one vehicle runs faster than the vehicle in front of it, it will slow down when close to it instead of maintaining constant velocity. We therefore introduce an interactive prediction kernel with basic distance-keeping ability into this engine. In Equation 3, $f(\Delta d, \Delta v)$ represents a velocity reduction factor related to the clearance gap and velocity differences between each vehicle in the AVM and its corresponding lead vehicle.

$$v(i+1) = v(i) + a_{cmd}\Delta t + f(\Delta d, \Delta v) \quad (3)$$

$$d(i+1) = d(i) + v(i)\Delta t \quad (4)$$

This look-ahead model, to some extent, serves as a differential controller in the traditional PID sense. However, it will be very complicated to model the interaction and interference between vehicles in traditional PID controllers. Compared to a traditional PID controller, our prediction engine mechanism is not only robust, but straightforward to implement, as well. By using the prediction engine, we successfully implement prediction ability similar to that of human drivers.

C. Cost Function Library

In scenario evaluation, the choice of cost functions is an important factor that influences the system's performance and robustness. We use seven kinds of base cost function to quantify human drivers' scenario evaluation. Based on these seven base cost functions, a highly reusable and reconfigurable cost function library shared by different behavior modules is built.

1) Gap Error: The gap error cost is mainly used in keeping a desired distance while following a lead vehicle. When the gap error is smaller than zero, which means the current distance to the lead vehicle is smaller than desired, the cost increases significantly. When the gap error is between 0 and 10 meters, the gap is deemed satisfactory, and the cost is quite low.

2) Clear Distance: The clear distance cost penalizes moving too close to surrounding vehicles. It is set to zero when all other vehicles are above safe distances.

3) Arrival Time Cost: Time cost is an important factor in scenario evaluation, since we are always trying to reach the destination as quickly as possible, all other things being equal.

4) Distance to Goal: The cost of distance to goal is used in some maneuvers, such as merging into the right-most lane when the vehicle is close to the exit of a freeway. This cost is only in effect when the autonomous vehicle is close to the goal. Figure 4 shows that when the distance to goal is small, the cost becomes very big.

5) Merge Safety: To evaluate whether a merging maneuver is safe enough, both the clear distance and the velocity difference should be considered, so this cost function is related to both of these factors. Figure 4 shows this two-dimensional cost function.

6) Acceleration: When driving a car, experienced human drivers will try to avoid large acceleration to ensure passenger comfort. The acceleration cost function represents this logic, as shown in Figure 4. In general, drivers also prefer not to brake hard in order to save gas and make driving smooth, so the gain of deceleration is larger than that of acceleration.

7) Adjust Time: In the merging process, we need a adjustment period for autonomous vehicles to get into a feasible position for merging. This non-linear cost function shows that the preferred adjustment time is less than or equal to 0.8 seconds, which is also similar to a human's adjustment behavior before a merge.

The library utilization and dependencies are shown in Figure 5. Many modules share the same cost functions. The differences are in the x-axis scales and the cost weights. The cost weights represent how important a factor is in the scenario evaluation. The x-axis scaling also enhances cost-function library reuse and reconfigurability in different modules.

D. Implementation

There are three modules related to freeway driving ability that need to be implemented or modified in the behavior executive. They are the distance keeper, lane selector and merge planner. The data flow is depicted in Figure 6. The role of the distance keeper is to keep a reasonable distance from the lead vehicle and it has two outputs, desired acceleration and desired velocity. The role of the lane selector is to output the intended lane that the autonomous vehicle wants to merge into.

When the intended lane is different from the current lane we are driving in, the merge planner will be triggered. It computes the feasibility of merging and chooses the best opportunity to merge. Three outputs are used to implement this. They are merge acceleration, merge velocity and commanded lane.

1) Distance Keeper: We first implement our distance keeper based on the prediction- and cost function-based algorithm framework. The objective of this module is to keep a reasonable distance from the leading vehicle. The distance $d_{desired}$ is computed according to the current velocity as shown in Equation 5.

$$D_{desired} = D_{min} + kV_{vehicle} \tag{5}$$

In this module, we use two cost functions for evaluating and selecting the best result for the gap error ($\Delta D = D_{current} - D_{desired}$) and the cost of acceleration, as shown in Equation 6.

$$C_{scenario} = \mu_1 C_{\Delta D} + \mu_2 C_{acc} \tag{6}$$

For each generated prediction scenario, we compute its corresponding $C_{\Delta D}$, C_{acc} and then get $C_{scenario}$. As shown in Equation 7, C_{total} is the average cost of all the predicted scenarios, and n is the number of prediction steps.

$$C_{total} = \Sigma(C_{scenario}/n) \tag{7}$$

The acceleration and velocity corresponding to the lowest C_{total} is the module's output. The performance will be shown and analyzed in Section V.

2) Freeway Lane Selector: The role of the lane selector is to output the ID of the intended lane for the vehicle to merge into. The previous lane-selecting algorithm is rather conservative. It always tries to merge into the goal lane and then keep in that lane no matter how far the destination is, except that the lead vehicle is stopped or of abnormal low speed. Our prediction- and cost function-based lane selector provides a more intelligent and robust ability on selecting intended lanes. One of the main component of the lane selection cost is the estimated arrival time to the goal. However, since the traffic environment is uncertain, it is impossible to compute the arrival time with only local sensor data. We therefore use a virtual goal instead. The estimated arrival time of virtual goal consists of three parts as shown in Equation 8.

$$C_{arrival} = C_{merge} + C_{catch} + C_{follow} \tag{8}$$

C_{merge} represents the estimated time to perform the merging maneuver; if the vehicle stays in the current lane, then $C_{merge} = 0$. C_{catch} represents the estimated time for autonomous vehicle to catch up with a lead vehicle, if there is one. C_{follow} represents the remaining time cost after merging and catching up maneuvers to reach the virtual goal.

For better adjustable and robustness to input noise, we compute three estimated arrival time for virtual goals 450m, 350m and 250m down the road. We then weight the three time cost and compute the sum fo them for the final evaluation, as shown in Equation 9.

$$C_{ave} = \mu_3 C_{arrival450} + \mu_4 C_{arrival350} + \mu_5 C_{arrival250} \tag{9}$$

After computing the estimated arrival time, merge risk penalty and distance to goal penalty are added as shown in Equation 10

$$C_{lane} = C_{ave} + \mu_6 C_{risk} + \mu_7 C_{closeToGoal} \tag{10}$$

The merging risk uses the clear distance cost in the library. It reflects a strategy preference of merging into lanes with larger gaps. When the vehicle is close to goal, we add penalty times to the estimated arrival time of the lanes other than the goal lane, so that the lane selecting algorithm will not cause the vehicle to miss destinations. Then this module will output the lane corresponding to the lowest C_{lane}.

3) Merge Planner: The merge planner computes the merge commands, including desired velocity, acceleration and merge status. There are mainly two statuses in a typical merging maneuver, adjusting and merging. The rule-based merge planner strategy we used in the Urban Challenge tries to make this maneuver as safe as possible. But on freeways its adjustment strategy, such as slowing down or even stopping to wait for merging gaps, is not proper and can be dangerous. We therefore implemented a smarter merge planner with better performance on freeways. In our prediction- and cost function-based implementation, there are four parameters that represent a merging strategy, adjustment time, adjustment acceleration, merging time and merging acceleration. But since the merging time is controlled by the lower-level motion planning subsystem, we set it to a constant. We then send different combination of the three inputs into the prediction engine. As shown in Equation 11, the cost of a merge strategy consists of two parts, adjustment cost and merging cost.

$$Cost_{total} = \mu_8 C_{adj} + \mu_9 C_{merge} \tag{11}$$

$$Cost_{adj} = \mu_{10}\Sigma C_{clear} + \mu_{11} C_{adjT} + \mu_{12} C_{acc} \tag{12}$$

$$Cost_{merge} = \mu_{13}\Sigma(C_{clear}) + \mu_{14}\Sigma(C_{\Delta V}) + \mu_{15} C_{acc} \tag{13}$$

Adjustment cost considers the clear distance, adjustment time and acceleration only in the current lane. And the merging cost considers the clear distance, acceleration, and velocity difference of both current and intended lanes, as shown in Equation 12 and 13.

V. Performance Evaluation

A. Testing Scenario Generator

To test the performance of our algorithm, the first step is to build a freeway scenario in simulation. The scenario should be random, but statistically repeatable, and should be similar to practical freeway environments.

1) Normal Freeway Traffic: The freeway traffic environment is generated through a Gaussian random number generator. Both the distances between vehicles and velocities have a Gaussian distribution. Therefore, by adjusting the four parameters, d_{ave}, d_σ, v_{ave} and v_σ, we get different traffic environments.

2) Checkpoints: The testing autonomous vehicle enters the road in the center lane and its destination is in the same lane. Currently, the more complicated traffic features near exits or entrances are not included.

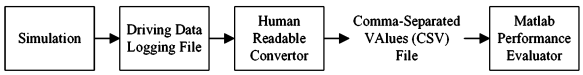

Fig. 7. Logging File Analyzing Mechanism

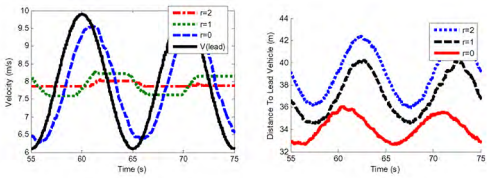

Fig. 8. Performance Improvements of Distance Keeper

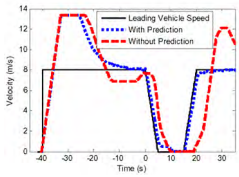

Fig. 9. Vehicle Following Performance

3) Road Length: In this experiment, the length of the test road is set to 20,000m, so that it is long enough for repeatability.

B. Performance Analyzing

Because human evaluations of the driving skills are mainly qualitative, there is no universally accepted way to compare different algorithms or set an optimization goal. Some previous work in strategy optimization uses a global cost function to evaluate the performances [5], [6]. However, there are many factors in driving ability, so a single value is not representative enough. Other researches use the time-maneuver plots to show the performance improvements [7]. But this qualitative evaluator is not able to show global performance improvements. We therefore propose a logging file analysis mechanism, as shown in Figure 7. The evaluator can summarize our simulation results using the following features: arrival time, number of maneuvers, histogram of velocity, acceleration and clear distances. It is then much easier for people to use their preliminary knowledge to analyze the algorithms' the advantages and disadvantages. In our research, both the statistical features and explanation of specific examples will be used to show performance improvements.

C. Experiment Results

1) Distance Keeper: Figure 8 shows that due to the cost of the acceleration, the vehicle no longer strictly follows the lead vehicle's speed and keeps a constant distance from it. Instead, it considers the need of following and also the smoothness of driving at the same time. Figure 8 also shows that the ratio $r = \mu 2/\mu 1$ represent the strategy generators' preference of strict following or smooth driving. As shown is Figure 9, the latency of reaction is smaller using this algorithm in following lead vehicle. With prediction the vehicle is able to accelerate or decelerate more resonablely. All these features show that the prediction- and cost function-based algorithm makes the distance-keeping strategy more similar to that of the human drivers.

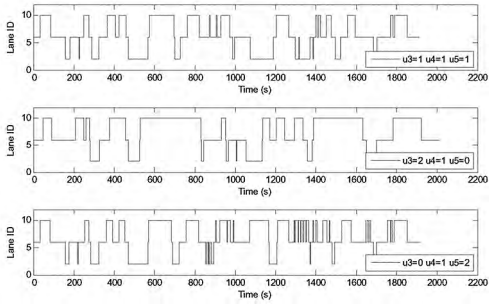

Fig. 10. Performance Comparison of Lane Selector

TABLE I
AVERAGE PERFORMANCE FROM 5 RANDOM SCENARIO TESTS

Cost Weights	$N_{maneuver}$	$t_{arrival}$ (s)
$\mu_3 = 1, \mu_4 = 1, \mu_5 = 1$	55.3 ± 3.8	1897.1 ± 73.1
$\mu_3 = 2, \mu_4 = 1, \mu_5 = 0$	38.7 ± 5.4	2049.9 ± 122.4
$\mu_3 = 0, \mu_4 = 1, \mu_5 = 2$	85 ± 9.0	1939.2 ± 69.8

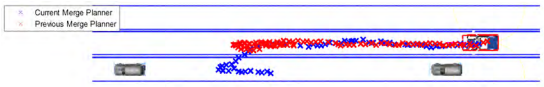

Fig. 11. Merging Between Two Vehicles

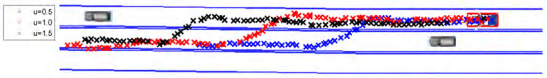

Fig. 12. Circumventing Vehicles

2) Lane Selector: Table I shows the performance improvements of using prediction- and cost function-based algorithm. We also find that parameters μ_3, μ_4 and μ_5, which represent the respective weights of $t_{arrival}$ to 250m, 350m, 450m virtual goals, influence the performance a lot. With larger μ_5 or smaller μ_3, the system emphasizes long-term time savings more than short-term advantages. However, if we consider long-term time savings too much, the possible velocity variances of surrounding vehicles will lead to inaccurate prediction and cause the vehicle to make incorrect decisions. As shown in Figure 10, with larger μ_5, there are more oscillations between lanes. In contrast, with more consideration of short-term advantages, larger μ_3, we failed to select the most time-saving lanes and the time consumption is larger.

3) Merge Planner: There are two examples show the robust and smarter merge planner we implemented, which is in Figure 11 and 12. In the first scenario, the vehicle makes a smooth merge while keeping the most reasonable distances between V1 and V2. And by adjusting cost function parameters μ_9 the merge strategy can be either conservative ($\mu = 1.5$) or aggressive ($\mu = 0.5$). The second example shows the autonomous vehicle's ability of emergency merging with the relatively optimal safety clearances instead of a merge failure.

TABLE II
AVERAGE OVERALL PERFORMANCE FROM 5 RANDOM SCENARIO TESTS
(d_ς=1, v_ς=1)

	d_{ave} (m)	v_{ave} (m/s)	$t_{arrival}$	N_{merge}
Current	150	8	2173.4 ± 130.2	36.4 ± 2.0
Previous	150	8	2985.9 ± 99.1	156.4 ± 20.1
Current	120	7.5	2220.0 ± 99.7	43.5 ± 5.3
Previous	120	7.5	2726.4 ± 100.2	146.1 ± 2.0
Current	90	7	2641.7 ± 54.2	51.0 ± 7.5
Previous	90	7	2941.0 ± 150.7	139.9 ± 19.6
Current	60	6.5	3195.5 ± 20.3	36.9 ± 2.0
Previous	60	6.5	3070.4 ± 190.5	146.0 ± 31.9

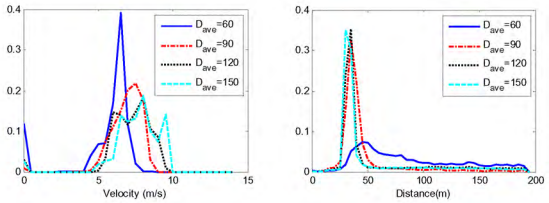

Fig. 13. Velocity and Clear Distance Histogram

This is very helpful when the vehicle is close to the objective freeway exit.

4) Cooperation of the Three Modules: After separately testing the three modules, we integrated them into our TROCS platform together. Table II shows the statistical results extracted from tests in different scenarios. Compared to the previous strategy, the vehicle using the new algorithm arrives 20% faster. The improvements are more obvious in not so heavy traffic, since there are more circumventing opportunities. We also analyze the histograms of velocities and clear distances, as shown in Figure 13. As you can see, when $d_{ave} > 90$, the histograms of velocities are similar. This means our algorithm is adaptive to different traffic densities. When the traffic is heavier ($d_{ave} = 60$), it takes the vehicle longer time to get into a feasible position for merging, therefore it has to keep larger distances from the lead vehicles quite often.

Through the performance evaluation, we arrive at four conclusions regarding the prediction- and cost function-based algorithm. First, the algorithm increases the human significantly. The autonomous vehicle performs more intelligently and similarly to a human driving vehicle. Second, the safety performance is satisfactory. Different parameters in behavior modules only affect the preferences of the strategy, while the safety performance is strictly ensured. Third, the algorithm is capable and robust in dealing with traffic environments with higher density and variant velocity. Finally, the algorithm provides high reconfigurability ability in behavior modules. By adjusting parameters, the strategies will perform either aggressively or conservatively.

VI. CONCLUSIONS

A. Achievements

According to the performance analyzer based on the real-time vehicle control platform TROCS, the robustness and performance of the autonomous freeway driving increase a lot. The strategy generated by the new algorithm saves about 20% times while driving on a freeway with traffic. And the maneuver numbers decreases 70%, which means that the merging and lane selecting are more reasonable and efficient. In conclusion, the prediction- and cost function-based algorithm has been proven to be functional and promising in improving autonomous vehicles' behavior abilities.

B. Further Improvements and Works

Though autonomous freeway driving ability is preliminarily proven to be satisfactory on the simulated platform, there are potential improvements. First, we plan to use machine learning and pattern recognition to refine our prediction model and extend it to a vehicle intention recognition system. Second, a parameters self-adjustment is necessary for the prediction- and cost function-based algorithm. Through this mechanism, we may be able to find a set of particular cost function library parameters that makes a greater contribution to improving travel time, while another set contributes more to lowering the number of lane changes. By intelligent switching between these parameter sets, we may achieve better performance, robustness and safety. Finally, we expect to test our algorithm at freeway speeds with human test subject soon.

ACKNOWLEDGMENT

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 TROCS.

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, M. Lutzeler, M. Pellkofer, K. H. Siedersberger, and E. D. Dickmanns, "Ems-vision: a perceptual system for autonomous vehicles," vol. 3, no. 1, pp. 48–59, March 2002.

[3] D. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," 1989.

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

[5] R. Sukthankar, J. Hancock, D. Pomerleau, and C. Thorpe, "A simulation and design system for tactical driving algorithms," *Proceedings of AI, Simulation and Planning in High Autonomy Systems, 1996.*

[6] R. Sukthankar, J. Hancock, S. Baluja, D. Pomerleau, and C. Thorpe, "Adaptive intelligent vehicle modules for tactical driving."

[7] J. Miura, M. Ito, and Y. Shirai, "A three-level control architecture for autonomous vehicle driving in a dynamic and uncertain traffic environment," *IEEE Conf. on Intelligent Transportation Systems*, pp. 706–711, 1997.

[8] K. Ahmed, E. Moshe, H. Koutsopoulos, and R. Mishalani, "Models of freeway lane changing and gap acceptance behavior," in *Proceedings of the 13th International Symposium on the Theory of Traffic Flow and Transportation*, 1996, pp. 501–515.

[9] K. I. Ahmed, "Modeling drivers' acceleration and lane changing behavior," Ph.D. dissertation, Massachusetts Institute of Technology, Feb 1999.

[10] 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.

[11] C. R. Baker and J. M. Dolan, "A case study in behavioral subsystem engineering for the urban challenge," *IEEE RAM SPECIAL ISSUE ON SOFTWARE ENGINEERING IN ROBOTICS*, 2008.