CyberTORCS: An Intelligent Vehicles Simulation Platform for Cooperative Driving

Ming YANG, Nianfeng WAN, Bing WANG
Department of Automation, Shanghai Jiao Tong University, and
Key Laboratory of System Control and Information Processing, Ministry of Education of China,
Shanghai, 200240, China

Chunxiang WANG*
Research Institute of Robotics, Shanghai Jiao Tong University
Shanghai, 200240, China

Jianping XIE
INRIA Rocquencourt,
78153 Le Chesnay Cedex, France

Received: 10-03-2011; Accepted: 19-04-2011

Abstract

Simulation platforms play an important role in helping intelligent vehicle research, especially for the research of cooperative driving due to the high cost and risk of the real experiments. In order to ease and bring more convenience for cooperative driving tests, we introduce an intelligent vehicle simulation platform, called CyberTORCS, for the research in cooperative driving. Details of the simulator modules including vehicle body control, vehicle visualization modeling and track visualization modeling are presented. Two simulation examples are given to validate the feasibility and effectiveness of the proposed simulation platform.

Keywords: Simulation platform, cooperative driving, intelligent vehicles

1. Introduction

The development of autonomous vehicles has inspired a great trend to research of cooperative driving. Intelligent cooperative driving aims to increase the road capacity, and improve the safety for driving; accordingly, it helps to reduce car accidents and traffic jams. Notice that driver safety and comfort becomes more and more important[1], cooperative driving also improves driver experience. Despite some advanced sensor technologies and control strategies have been applied for the single vehicle system, cooperative driving requires many other technologies which may significantly increase the road capacity, and improve the safety for driving, therefore avoid accidents and traffic jam. Despite the same sensor technology and control strategy as applied for individual vehicle, cooperative driving requires many other technologies such as wireless communication, vehicle detection, self-organization, swarm intelligence[2] and so on.

In the last few years, multivehicle platforms have been developed at several universities and research labs. Some of the platforms involve unmanned ground vehicles and unmanned aerial vehicles, such as MIT's multivehicle testbed and University of Pennsylvania's multiple autonomous robots (MARS) testbed. Cornell University's RoboFlag testbed includes several small robots with local control loops that work cooperatively to achieve a common goal.

However the actual tests for cooperative driving require multi fully equipped vehicles, which a single laboratory may not possess. On the other hand, as many vehicles are involved, the risks and complexities of carrying out such experiments are significantly increased. Therefore, it is difficult to efficiently implement and evaluate our algorithms for cooperative driving research. These years

*Corresponding Author: wangcx@sjtu.edu.cn

the simulation platforms also have been developed not only in research field but also in industry. Some commercial simulation platforms are quite successful (for example, Carsim from Mechanical Simulation Corporation, Prescan from TNO). For research purpose, this kind of platforms is too expensive. In this paper, an open source intelligent vehicle simulation platform for the research of cooperative driving is proposed to address this issue. CyberTORCS, the simulation platform is designed to simulate both sensor behaviors and surrounding vehicle environments. Moreover, the simulation platform is capable of providing variety of information to help implementing and evaluating the developed algorithms. Simulation platform modules including vehicle body control, vehicle visualization modeling and track visualization modeling are discussed in detail. In order to demonstrate the capability and feasibility of the simulation platform, analysis on two simulation examples conducted based on CyberTORCS are shown.

The rest of the paper is organized as follows. Section II introduces system architecture and some modules of CyberTORCS. In Section III and Section IV, we present and discuss two experimental examples conducted based on CyberTORCS. Finally, Section V summarizes this paper with our future work.

2. System Architecture

CyberTORCS is developed based on the software TORCS (The Open Racing Car Simulator), which is an open source software for car racing. Players could drive their cars by keyboard or joystick. Also, it is very convenient to implement and test different control algorithms and driving strategies in this simulation platform. The software is written in C++, and could run on different operation system platforms.

As shown in Fig. 1, the simulation platform contains several modules. Each vehicle in the simulator has its own identity, sensor module, vehicle dynamics module, vehicle control module, etc. Basic features of these modules are included in TORCS, and users may edit each module for their own purposes. Based on the data received from sensor perception module or communication module, the cooperation algorithm module generates behavioral intentions for all vehicles, and then sends to the vehicle control module. The vehicle control module works out the control parameters for vehicle dynamic models, and the dynamic module outputs the vehicles position and orientation information for 3D display module. The data recorder module could record all the information such as the driving trajectories, vehicle speed, and some other information in terms of txt file or animation file. Since, in the paper, the simulation platform is developed for cooperation algorithm test and evaluation, some close related modules will be discussed in more detail both in the following module introduction and simulation examples. Unfortunately, some other modules are not introduced although we did mention in the system architecture.

2.1. *Vehicle body control module*

Similar to the real car driving, vehicles in CyberTORCS are controlled by four parameters: steer, gear, accelerate and brake. The control algorithm sends some control information (e.g. desired acceleration, global orientation)

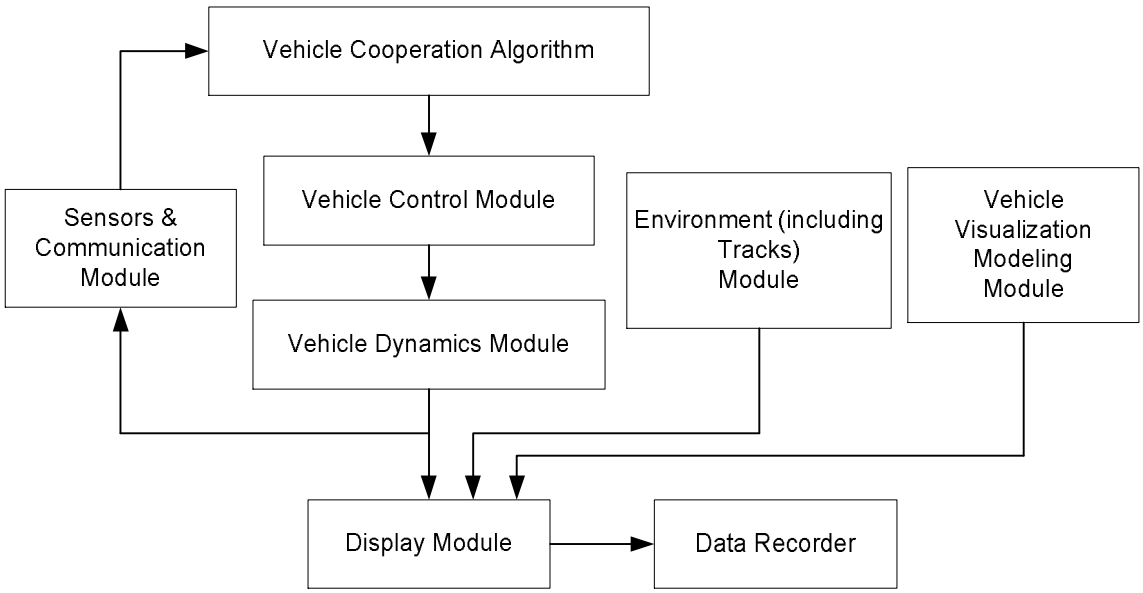


Fig.1. System Architecture

to the vehicle control module. The function of vehicle control module works out these four parameters and sends it to the vehicle dynamic mo9del. In this paper, we introduce two control examples: steering control and automatic cruise control.

In some simulation, the algorithm focuses on the longitudinal control. In this kind of situation, the steering method is to follow the middle line of the road. The steering angle (between PI and –PI) is calculated as:

$$angle = carPosition.toMiddle / currentTrackWidth$$

Where *carPosition.toMiddle* is length of the car's position to the middle line, and *currentTrackWidth* is the current track width. With this method, the car could follow the middle line of the road. More difficult steering movement such as lane changing could also use this simple method by changing the value of *carPosition.toMiddle.*

Many applications require maintaining a desired vehicle speed during the simulation. The simple automatic cruise control approach is utilized proportion control for controlling the accelerator. The input is the deviation between the current speed and the desired speed. Brake command is applied if the deviation is positive and vice versa. The gear is related to the engine RPM (revolutions per minute), in that, different speed range corresponds to different gear level.

2.2. *Vehicle appearance module*

The appearance modeling is an important part in simulation process, not only because it brings a better visual performance, but also accurately shows the information of each part of a vehicle, thus makes it possible to verify sensor navigation algorithms or other control strategies, or access some important position and orientation information precisely before the actual experiment. CyberTORCS simulator allows users to define a car's appearances, colors, materials and so on. It also provides some interfaces with other modeling software.

In this paper, we take the CyberC3 vehicle platform[3] depicted in Fig. 2(a) as the simulation target. The correspondent CyberC3 vehicle model in CyberTORCS is shown in Fig. 2(b) after the visualization modeling. As seen from Fig. 2, the vehicle visualization modeling model is able to build a very precise vehicle model.

(a) CyberC3 vehicle

(b) CyberC3 model

Fig.2. Comparison between the actual vehicle and the model

2.3. *Track modeling*

Track modeling or road modeling is usually more important. Sometimes intelligent vehicles would demonstrate in a particular region. If the track model in that region is precise enough, researcher could verify the algorithm in the lab and predict the performance accurately. CyberTORCS stores track information in XML form, and it provides an edit tool called track editor, which allows users to build and edit their own tracks. However, manually editing leads to errors. This paper presents a method for track modeling based on the road border GPS information with negligible errors.

In track editor, a track is considered as the connection of several sections. And sections are defined as two main types: straight ones and arc ones. A straight section is determined by its start point and its length, and an arc section is determined by its start point, radius and angel. It is common to use clothoids to construct roads in reality, however, when track editor builds clothoids, the errors would significantly increase. So we choose to use

only straight line and arcs to match real GPS points more accurately. Given a series GPS points, the method is to fit them to several straight sections and circle arc sections end to end to minimize the error.
The main idea is to divide the points into groups. Points in each group could be fitted as a straight line or a circle arc. There, thus, should be a start points, an end point and a direction. The steps of dividing points into groups are as follows:

Step 1: find the first point, and make it as point A;
Step 2: connect point A, point A+i (i = 2 at first), name this line L;
Step 3: calculate the sum of squares of the distance from the points between A and A+i to the line L;
Step 4: if the sum is large than a constant C (determined by users), the straight line group is from point A to point A+i, else i+1 and go back to *Step 2.*
Step 5: if a straight line group is generated, points A+i should then be the end points of the straight line, use the method of generating straight line to generate a line.
Step 6: make point A+i as the start point of the next circle arc, use the method of generating arc to calculate which points should be in the next circle arc group. For instance, the group contains points from A+i to A+j.
Step 7: use the method of generating arc to calculate the arc radius and angle.
Step 8: go back to step 2 until all points are divided into groups.

Following the method above, points are divided into groups, and lines are worked out from those straight lines groups. Here then the track is filled with several separated lines. The next step is to connect those lines with circle arcs. The method of generating circle arcs between the lines is as follows:

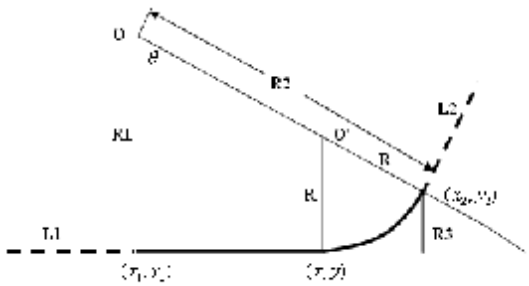


Fig.3. Method of Generating Circle Arcs

As it is shown in figure 3, (x_1, y_1) is the end point of L_1, (x_2, y_2) is the start point of L_2. The steps of the method are as follows:

Step 1: calculate the intersection point o of the public vertical lines of L_1 and L_2.

Step 2: calculate the distance between o and (x_1, y_1), which is named R_1, and calculate the distance between o and (x_2, y_2), which is named R_2. Compare R_1 and R_2. Here may wish to set $R_1 < R_2$ (if not, just exchange L_1 and L_2)

Step 3: calculate the angle of L_1 and L_2, which is named q.

Step 4: move o on the vertical line of L_2, to the point o', which has the same distance to L_1 and L_2.

Step 5: calculate the distance from the point (x_2, y_2) to line L_1, which is named R_3, thus from the figure,

$$R = \frac{R_3(R_1 - R_2\cos q)}{R_1 - (R_2 + R_3)\cos q} \qquad (1)$$

Step 6: based on o' and R, work out the point (x, y), replace (x_1, y_1) with (x, y).

Here the angle q and the radius R are worked out, and it is easy to identify the circle arc needed. With all parameters required are worked out, the track could be defined in CyberTORCS. The paper takes the road in Oriental Land, Shanghai, China as an example[1]. The following figures show that the track model generated by using the method while the error standard variation is 4cm. Figure 4 indicates the GPS map and the track model generated. It is shown that the result has high accuracy and little errors.

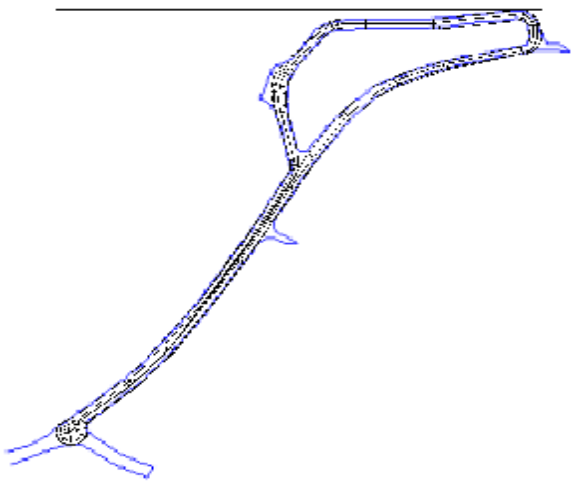


Fig.4. Comparison of the Model and GPS map

3. Overtaking Simulation Example

Overtaking issue is a typical cooperative driving scenario. The overtaking experiment requires high steering accuracy and stable communication between vehicles[4,5].

Moreover, the vehicles' driving trajectories should be recorded for analysis[6]. In this example, CyberTORCS shows its abilities in such aspects.

In general, the overtaking process includes three steps: 1. changes lane to the adjacent lane; 2. passes the overtaken vehicle; 3. returns to the original lane.

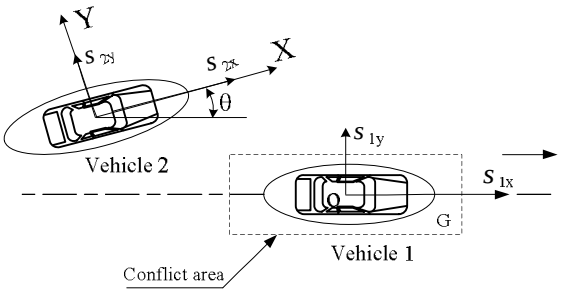


Fig.5. Vehicle overtaking

In the paper[7], the author proposes an overtaking control method based on the estimation of the conflict probability. In the process of overtaking, the longitudinal and lateral position errors of both the overtaking vehicle and the overtaken vehicle are normally distributed with zero means and covariances that have eigenvectors in the along-track and cross-track directions. The position-error covariances can be estimated by intelligent vehicles in online or offline way. Refer to Fig. 5, vehicle 1 is the overtaken vehicle and vehicle 2 is the overtaking vehicle. The ellipses around vehicles are the position-error ellipses and the maximum axes of these ellipses are aligned with the x axes of the vehicle bodies. The rectangular region around the overtaken vehicle is the predefined conflict area. The position-errors of the overtaking vehicle and the overtaken vehicle are independent. In the vehicle-fixed coordinate, the position-error covariance matrix of the overtaken vehicle is

$$C_1 = \begin{bmatrix} s_{1x}^2 & \\ & s_{1y}^2 \end{bmatrix} \tag{2}$$

where s_{1x} and s_{1y} are the standard deviations of the longitudinal and lateral position errors of the overtaken vehicle, respectively. Similarly, in the vehicle-fixed coordinate, the position-error covariance matrix of the overtaking vehicle is

$$C_2 = \begin{bmatrix} s_{2x}^2 & \\ & s_{2y}^2 \end{bmatrix} \tag{3}$$

where s_{2x} and s_{2y} are the standard deviations of the longitudinal and lateral position errors of the overtaking vehicle, respectively. By using coordinate transformation, the position-error covariance matrix of the overtaken vehicle can be converted to the coordinate that fixed on the overtaking vehicle, that is

$$C_1' = RC_1R^T \tag{4}$$

where

$$R = \begin{bmatrix} \cos q & -\sin q \\ \sin q & \cos q \end{bmatrix} \tag{5}$$

θ is the azimuth angle of the overtaking vehicle.

The position-error covariance of the overtaking vehicle and that of the overtaken vehicle can be combined in the coordinate that fixed on the overtaking vehicle, and the relative position-error covariance matrix can be presented by

$$C = C_1' + C_2 \tag{6}$$

Assume that in a moment, the instantaneous relative longitudinal and lateral positions of the two vehicles are s_x and s_y in the coordinate fixed on the overtaking vehicle. Then the instantaneous conflict probability density function can be expressed as

$$f(s_x, s_y) = \frac{1}{2p|C|^{1/2}} \exp\{-\frac{1}{2}(X-m)^T C^{-1}(X-m)\} \tag{7}$$

where $X=[s_x, s_y]^T$ and $\mu=[0, 0]^T$.

The conflict probability between the overtaking vehicle and the overtaken vehicle is the integral of the relative position-error probability density over the conflict area. Therefore the instantaneous conflict probability is

$$P(t) = \iint_G f(s_x, s_y)dxdy \tag{8}$$

where G is the conflict area.

For the overtaking intelligent vehicle, both current and future safety conditions should be considered in the process of overtaking, so the model predictive control can be applied in this process. The prediction model of the overtaking control is used to predict the future conflict probability corresponding to the possible control input. This model includes the relative position prediction model and the conflict probability prediction model. The conflict probability prediction model uses the estimated relative position to predict the future instantaneous conflict probability

In order to evaluate the effectiveness, simulation tests are implemented on CyberTORCS[8]. The simulation platform provides a two-lane traffic situation, in which the high-speed vehicle may generate the overtaking intention when it detects the preceding low-speed vehicle. The sizes of the overtaking vehicle and the overtaken vehicle are 4.8 m×1.9 m and 5.0m× 2.0 m,

respectively. The initial intervehicle distances in all the tests are 32 m.

Two different overtaking control methods have been compared in the same scenes. One is the proposed conflict-probability-estimation-based overtaking control method; the other method has been described in[9]. Corresponding to different situations, the speed of the overtaken vehicle is kept at 60 km/h and that of the overtaking vehicle is 80km/h, 85km/h, and 90km/h respectively.

Firstly, the simulation platform recorded the overtaking trajectories, as shown in Fig.6. Here the predefined safe conflict probability is 1.0×10^{-4} and the conflict area is Ca=25m×5m in these tests.

Secondly the platform illustrated the variation of the closest distance for several scenarios to find out the relationship between the distances and the parameters.

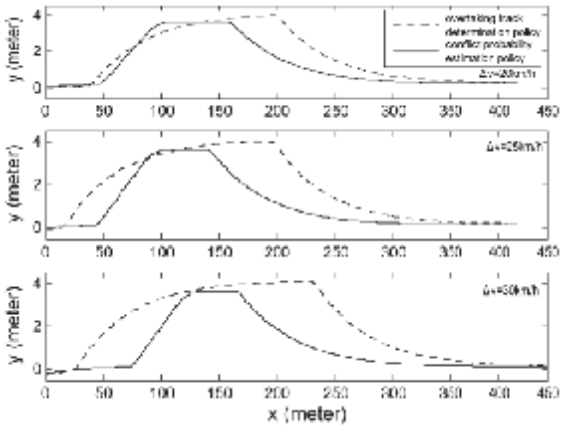


Fig.6. Overtaking tracks in different situations.

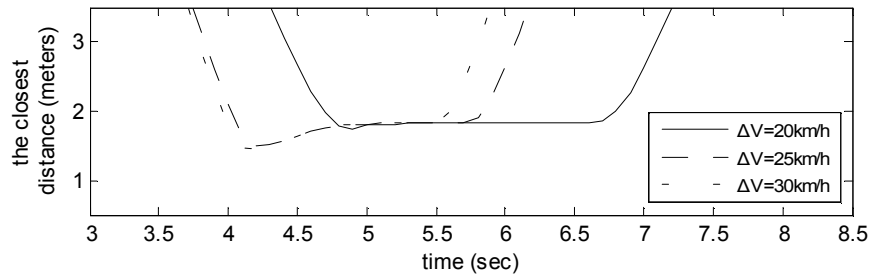


Fig.7. The variation of the closest distance (scenarios with different relative speed)

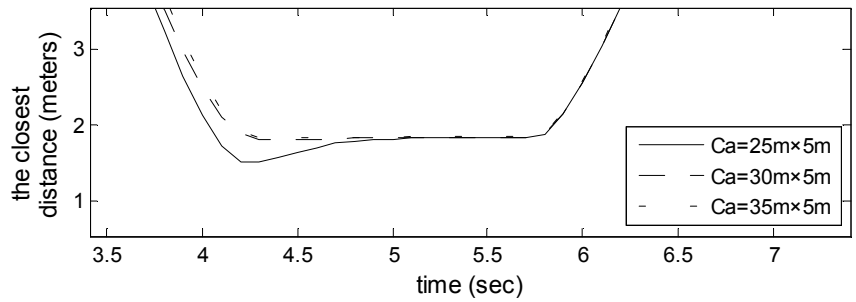


Fig.8. The variation of the closest distance (scenarios with different conflict area)

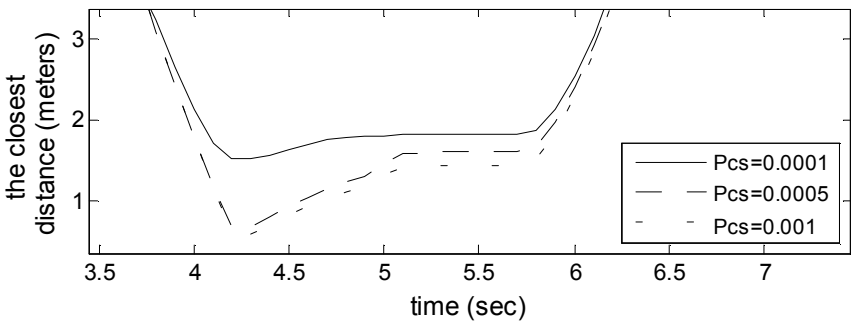


Fig.9. The variation of the closest distance (scenarios with different safe conflict probability)

The result from Fig. 7 shows that a higher relative speed corresponds to an earlier initiation of the overtaking, but the closest distance hasn't varied greatly in different scenarios. The result from Fig. 8 shows that with the increase in the conflict area, the closest distance hasn't decreased remarkably. Therefore the variation of conflict area in a certain scope doesn't affect the overtaking safety significantly. The result from Fig. 9 shows that the closest distance decreases with the increase in the safe conflict probability. Based on the accurate simulation results, the author's conclusion is convincing.

4. Platoon Simulation Example

Platoon is another important cooperative driving scenario. A string of vehicles drives with constant velocity in a line may increase the capacity of the road. How to keep them with constant position and how to reduce position and velocity error is the main problem of this issue. Some research uses adaptive tracking control method for solving these problems[9].

In the thesis[10], the author proposes a new spring damper control method for platoon control. This method considers that all vehicles in a platoon as a group of connected spring-damper systems, which indicates that only local but bidirectional information is used. The system of a platoon of vehicles can be represented as follows:

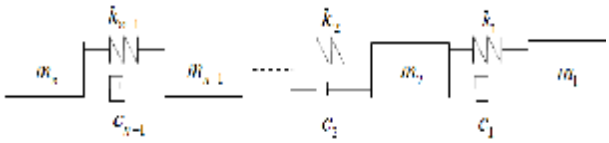


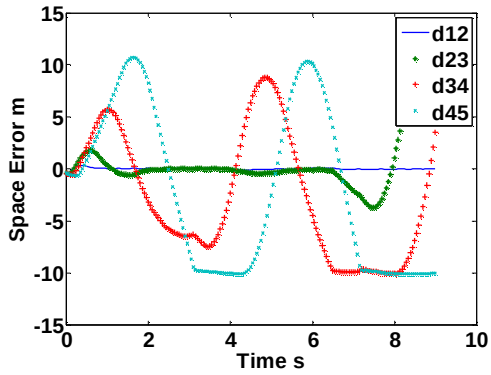
Fig.10. Spring-damper system

Then the author divides the whole system into several subsystems, each subsystem contains three vehicles. And overlapping decomposition method is used for building the new expanded system. Finally the LQ optimal control is used to work out the best coefficients for both the expanded system and the original one.

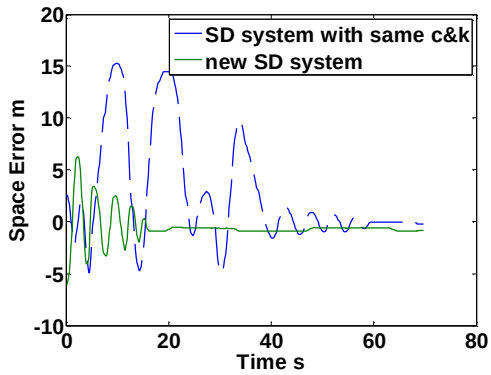
We use CyberTORCS to compare the new method with classical algorithm.

The simulation test selected an oval track for a length of 1908.3 meters, road width of 25.0 meters. The platoon is composed of five cars, the expected speed is 40 km per hour, and the vehicle spacing is 10 meters. The simulation object is to examine whether the position and velocity errors of vehicles would propagate along the platoon when the first car has a sudden acceleration and deceleration. The platoon cruise speed is 40km/h. The leading car firstly accelerates to 60km/h with the acceleration of 10m/s^2, and then decelerates to 40km/h with the acceleration of -10 m/s^2. The experiment compares the present method with the classical PD control method and a fixed spring damper coefficient method[11].

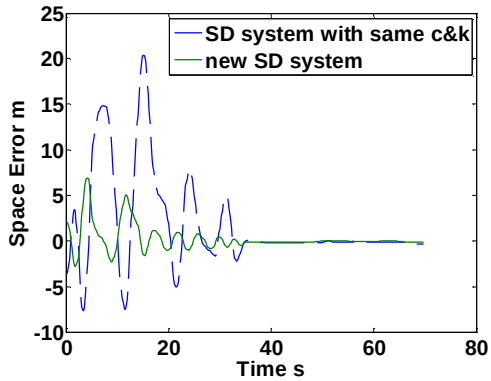
The results are shown in Figure 11(a), 11(b) and 11(c). Figure 11(a) indicates the spacing error variation using PD control. It can be seen that the spacing error between the 4th and the 5th vehicle firstly reached -10 m in the first 4 seconds, that is, the collision occurred. Similarly, the 4th and the 3rd vehicle in the first 7 seconds also



(a) spacing error variation using PD control



(b) spacing error variation between 4th and 5th vehicle



(c) spacing error variation between 3rd and 4th vehicle

Fig.11. Spacing Errors

collided. Figures 11(b) and 11(c) show the spacing error variation between the 4th and the 5th vehicle and between the 3rd and the 4th vehicle respectively, with the application of the fixed spring damper coefficient method and the present method. It is shown that both systems are convergent, while the method proposed in this paper presents a small offset, and the stabilization time is short, also the error will not propagate along the platoon. And the results show the convenience of the platform CyberTORCS compared with others[12,13].

5. Conclusion

The current simulation platform CyberTORCS utilizes several ways to obtain high accuracy in multi vehicle experiments. According to examples, with benefits in reducing cost and maintaining safety, such platform with high accuracy would play a more important role in the research of cooperative driving in the future. Researchers could use this platform to test their algorithms and revise them without huge expense. In fact, the platform has been used to test several algorithms for vehicle cooperation and other aspects.
In addition, the following aspects have been taken into account in future works.
1) The data would be analyzed after the experiment is done. The next step is to create a real-time data display module, which may help researchers to monitor each parameter in time.
2) Several sensor models are developing, which is not mentioned in this paper. And interface development with other simulation platform is also a good direction.
3) The hardware-in-the-loop module would help the platform enlarge the ability in testing and evaluating.

Acknowledgements

The work was supported by the Shanghai Science and Technology Action Program for World Expo (10dz0581100). The authors would also like to thank all scientists who build and update TORCS, their examples and documentations on the website help this work a lot (http://torcs.sourceforge.net/).

References

1. W.H. Wang, F.G Huo, H.C.Tan, H. Bubb, "A Framework for Function Allocation in Intelligent Driver Interface Design for Comfort and Safety". *International Journal of Computational Intelligence Systems*. 3(5) (2010)531-541.
2. Y.X.Shen, G.Y.Wang, C.M.Tao, "Particle swarm optimization with novel processing strategy and its application". *International Journal of Computational Intelligence Systems*. 4(1) (2011)100-111.
3. T. Xia, M. Yang, and R. Yang, "CyberC3: A Prototype Cybernetic Transportation System for Urban Applications" IEEE Trans. on Intelligent Transportation Systems VOL. 11, 142-152, 2010
4. G. Hegeman, R. Horst, K. A. Brookhuis, and S. P. Hoogendoorn, "Functioning and acceptance of overtaking assistant design tested in 332driving simulator experiment," Trans. Res. Rec., vol. 2018, pp. 45–52, 2007.
5. J. Guldner, V. I, Utkin, and J. Ackermann, "A Sliding Mode Control Approach to Automatic car Steering," in Proc. American Control Conf., Baltimore, MD, Jun. 1994, pp. 1969-1973.
6. Doug Orrin, "Simulation Development" Vehicle Dynamics International, Issue 1, 2009.
7. F Wang, M Yang, R Yang. "Conflict-probability-estimation-based overtaking for intelligent vehicles" IEEE Transactions on Intelligent Transportation System: VOL. 10, 366-370, 2009.
8. Fenghui Wang, "Study on Multi Vehicle Cooperation for the local intelligent transporation system", *Phd degree thesis*, Shanghai Jiaotong University,2009.
9. Mou Chen, Bin Jiang, Jie Zou, Xing Feng. "Robust Adaptive Tracking Control of the Underwater Robot with Input Nonlinearity Using Neural Networks" *International Journal of Computational Intelligence Systems*, Vol.3, No. 5 (October, 2010), 646-655
10. Nianfeng Wan, "Study on Cooperation Algorithm for Multi-intelligent Vehicles", *Master degree thesis* , Shanghai Jiaotong University,2009
11. Soo-Yeong Yi, Kil-To Chong. "Impedance control for a vehicle platoon system". *Mechatronics* 15 627–638, 2005
12. Doug Orrin, "Vehicle Dynamics Software-Today and Tomorrow" *Vehicle Dynamics International magazine* 2006
13. B. Guvenc, E. Kural, "Adaptive Cruise Control Simulator" *IEEE Control System Magazine 42-55,* June 2006