doi:10.11835/j.issn.1671-8224.2017.02.04

To cite this article: LUO Yong, ZHAO Xue, CAO Yu-feng, XIE Xiao-hong. Development of driver-vehicle-road closed-loop semi-physical simulation system [J]. J Chongqing Univ Eng Ed [ISSN 1671-8224], 2017, 16(2): 79-86.

Development of driver-vehicle-road closed-loop semi-physical simulation system *

LUO Yong [1,2,†], ZHAO Xue [1], CAO Yu-feng [1], XIE Xiao-hong [1]

[1] *College of Vehicle Engineering, Chongqing University of Technology, Chongqing 400054, P.R. China*

[2] *Key Laboratory of Advanced Manufacturing Technology for Automobile Parts Under the Ministry of Education, Chongqing University of Technology, Chongqing 400054, P.R. China*

Received 20 October 2016; received in revised form 13 January 2017

Abstract: As hybrid vehicles introduced the motor, the vehicle structure has a significant change in the power matching. A driver-vehicle-road closed-loop semi-physical simulation system, which makes real driving parts together with the simulation car, will bring convenience to the new car design. We used the computer software to simulate the road with a slope, curve and some other features based on the actual road condition, and analyzed the whole road scene in addition to geometry and physical characteristics. Analyzing and constructing the vehicle dynamics basic template, appropriate changes to the template can obtain the desired vehicle dynamics model with an external device to control the model vehicle. It combined the physical operation system with visual display, which gave us real driving feelings and increased the vehicle design predictive accuracy.

Keywords: HWIL; drive simulator; road scene; vehicle dynamics

CLC number: TP399 Document code: A

1 Introduction

The driver-vehicle-road closed-loop semi-physical simulation system has many components. We focused on the key parts. The first category gives the system structure. The second category shows the XPC system which ensures the model run in real-time simulation. The third category shows the building of vehicle dynamic model. The fourth category shows the visual display including the simulation road and collision detection. The last is the data interview port. Using the computer simulation to research vehicle performance can save the test cost and time. The vehicle model is the indispensable part of the entire automotive driving simulator, which decides the simulation veracity directly. The use of visual simulator provides us analysis means with intuitionally and high-efficiently. The model mainly contains tire, body and power train system.

† Corresponding author, LUO Yong (罗勇): 876421149@qq.com.

* Funded by the National Natural Science Foundation of China (No. 51305475), Chongqing Research Program of Basic Research and Frontier Technology (No. cstc2013jcyjA60004), and the Scientific and Technological Research Program of Chongqing Municipal Education Commission (No. KJ1500927).

2 Overall system design

To build this system, we analyzed the structure of the simulator driving system, put forward the need for dynamics model, and ensured the input and output parameters of the dynamic model. After the introduction of the vehicle model fundamental characteristics, we built the inner structure of the model. According to the need of the automotive dynamic model, we built a callable visual display model combined VS2008 and the open graphical interface (Open GI)**.** Finally, the xPC system transfered the body direction to the visual display model and drove the vehicle to flash display. Fig. 1 shows the basic structure.

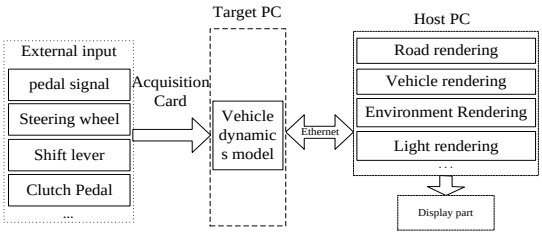

Fig. 1 Basic structure of system

3 HWIL (Hardware-in-the-loop) system

3.1 xPC target real time simulation

xPC target is a type of HWIL system. It also can be called "twin-computer" solution way, which means xPC needs two computers. The host computer runs the Simulink, and the target computer is used to carry out the generated code. The target computer runs a highly-compact real-time operating inner core. This real-time inner core is applied the 32bit protected mode. The communication between the host computer and target is achieved through an Ethernet network interface or serial interface connection. In the environment of xPC target, users can treat computers which are installed MATLAB/Simulink as the host computer. Simulink is used to create the model that users' needed. Then the RTW code generator and C compiler will produce the executable code. Then the second compatible PC runs the real time code [1]. The hardware which supports MATLAB will call the driving S function directly. As for the board card, which doesn't support MATLAB, users can write their own driving functions [2]. The entire system architecture is as shown in Fig. 2.

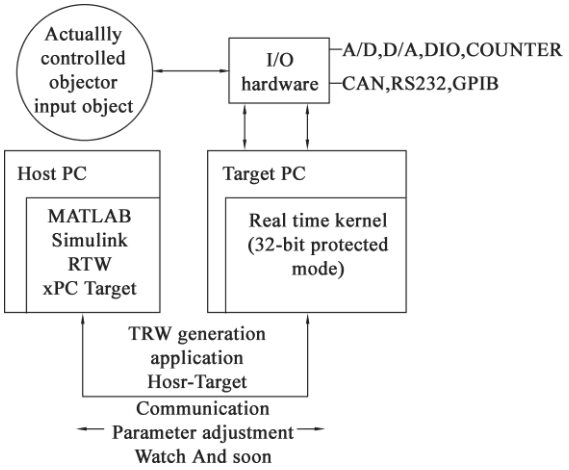

Fig. 2 Structure of xPC

3.2 Installation and debugging

The host computer can install the windows XP system, the lower MATLAB can install the 32bit version, but higher version like MATLAB 2014 need to install 64bit system. The compiler can install Microsoft Visual C/C++. More details are in Ref. [3]. When the DOS startup disk is generated, users can choose USBoot or FlashBoot. USBoot is easy, but we need to pay attention to that USBoot is only used on windows XP. Users can make USB flash disk into FDD, HDD or ZIP. ZIP supports xPC best, but the capacity of the USB flash disk is less than 1 GB. After making the USB flash disk, there are two system files: IO and MSDOS.

Input ‘xpcexplr’ in the MATLAB command. The ‘xPC TargetExplor’ window open, where users can configure Host PC Root and Target. Choose C compiler as the compiler and the installation path is the same with it. Choose TCP/IP mode in Target Tab controls Communication protocol, and then configure the IP address. Finally, choose a different driving part relied on the net card. After finishing the setting, choose DOSLoader in Target boot mode, and click ‘CreaterBooDisk’ and the generated ‘autoexec.bat’, ‘xpcboot.com’, ‘xpctgo.rbt’. That is the preparation task of xPC.

4 Vehicle dynamic modelling

According to vehicle ISO reference frames, we took the point of sprung mass as the origin of coordinates. The *x* axle is parallel with the vehicle driving direction, the *y* axle points are the left of automobile, and the *z* axle is vertical to the ground and points above. The whole car model contains the longitudinal motion which goes along with the *x* axle, the sideways movement which goes along with the *y* axle, the yawing motion which goes around with the *z* axle, and the rotary movement of four wheels. This model can be used to simulate the straight braking, the turning braking and other complicated working conditions [4]. The hypothesis foundations of the model are

1) The moving origin of automobile coordinates with the mass point.

2) Ignore the movement of suspension, without thinking of vertical movement.

3) The yawing angle around the *y* axle and that around the *x* axle are zero.

4) Four wheels will not break away from the ground, and the entire vehicle is still on the road.

5) There is no influence of road gradient. Ignore the influence of air resistance.

6) Mechanism characteristics of the front wheels and rear wheels are the same without the condition of wheel aligning torque.

Take the front wheel turning angle as the input variable, and the stress of car model is as shown in Fig. 3.

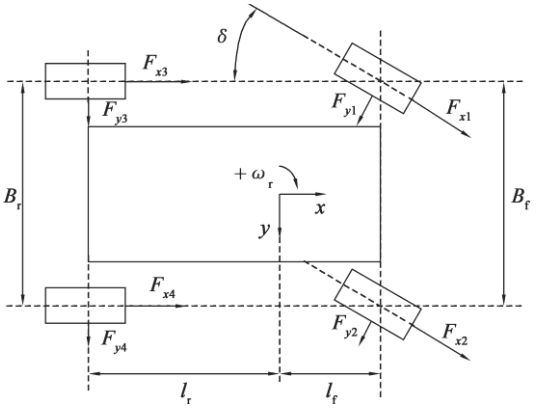

Fig. 3 Vehicle dynamics model

The longitudinal force balancing equation is

$$m(u_x - u_y\omega_\gamma) = (F_{x1} + F_{x2})\cos\delta + F_{x3} + F_{x4} - (F_{y1} + F_{y2})\sin\delta - F_w,$$

where u_x is the car’s longitudinal velocity in terms of m/s; δ is the wheel turning angle in terms of rad; u_y is the car’s horizontal velocity in terms of m/s; ω_γ is the yaw velocity in terms of m/s; F_{xi} is the wheel longitudinal force and i = 1 to 4 in terms of N; F_{yi} is the wheel yawing force and i = 1 to 4 in terms of N; and F_w is the air resistance in terms of N.

The yawing force balancing equation is

$$m(u_y + u_x\omega_\gamma) = (F_{x1} + F_{x2})\sin\delta + F_{y3} + F_{y4} + (F_{y1} + F_{y2})\cos\delta .$$

The yawing moment is

$$I_z\omega_\gamma = l_f[(F_{x1} + F_{x2})\sin\delta + (F_{y1} + F_{y2})\cos\delta] - l_r(F_{y3} + F_{y4}) + \frac{B_f}{2}[(F_{x1} - F_{x2})\cos\delta + (F_{y2} + F_{y1})\sin\delta] + \frac{B_r}{2}(F_{x3} - F_{x4}),$$

where l_f is the distance between the mass point and the

front axle; l_r is the distance between the mass point and the rear axle; B_f is the front tread; B_r is the rear tread; and I_z is the body around the z axle moment inertia (2 kg m).

5 Visual display

5.1 Introduction of real time simulation of 3D visual scene

There are many 3D visual simulator software in the market, such as Vega Prime, Vega, VRML, and OSG. The program interface of Vega Prime is C++. It owns the graphical user interface and tools package, which focus on the roads. So we chose Vega Prime as the 3D real time software [5]. Vega Prime is a professional visual development software and Fig. 4 shows its development process.

5.2 Simulation road

The road is a 3D entity. So the road model must contain plane line types, profile and cross section [6]. Road tools of Vega Prime are used to build a real time road model. When the road 3D model is built, planes and hook faces are not difficult to combine and adjust. The model which is constructed by road tools could be used in the simulation and vehicle dynamics calculation. The construction of roads has three processes: road construction, road tessellation and special data input [7]. Users could build road models quickly by using the basic model (Fig. 5).

5.3 Collision detection algorithms of Vega Prime

When vehicles run on the roads, we need to focus on two points.

Vehicles running on the roads get gravity, unlike the airplane flying in the air, so we need to change the motion to ‘MotionDrive’ modes.

Vehicles could not sink into roads, so we need to detect and touch the roads. That’s why we need to add collision detection on the roads.

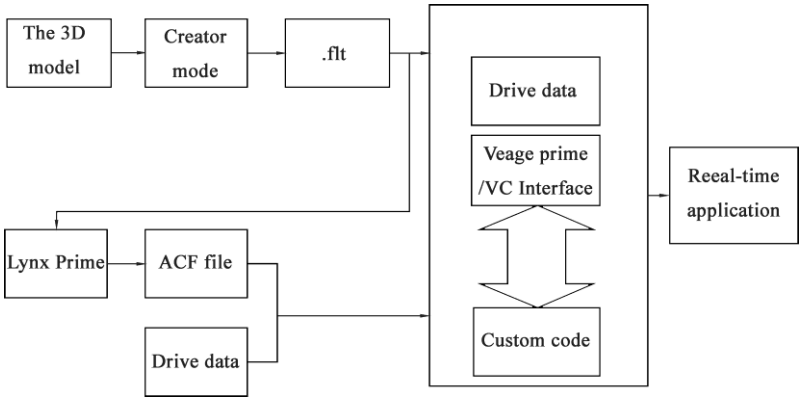

Fig. 4 The development process of Vega Prime

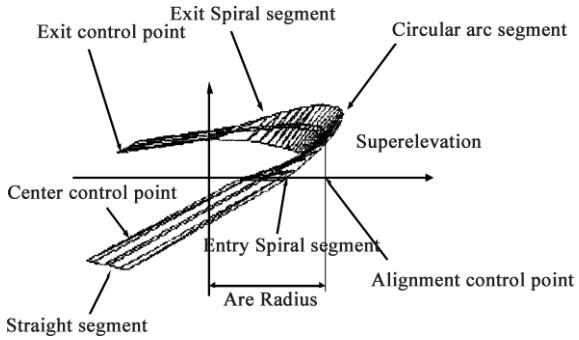

Fig. 5 Basic model of road

Detection algorithms can be divided into discrete points of collision detection and continuous collision detection. The discrete collision detection is to specify a time *T* for the collision body to find out whether there is overlap between them. If there is no overlap, it returns their closest point distance; if the overlap exists, it returns the overlap depth and the overlap direction. The continuous collision detection is specified separately for collision body positions in time T_1 and time T_2 to find out whether there is a collision from T_1 to T_2. If the collision exists, it returns the first collision point position and normal. The continuous collision detection is the most natural collision detection. It can greatly facilitate the writing of the collision response logic and easily avoid objects overlap or cross. Although it is the most natural way, its implementation is very complicated and its operation is expensive. So most mature physics engines and collision detection engines are adopted based on the discrete points of collision detection. To avoid the deep object overlap or crossing each other, the simulation step size is smaller [8].

The Vega Prime uses the discrete collision detection. It uses a cube or a spherical body (box) wraping the 3D object (or major part). A whole distance description according to the box and the location information can calculate whether a collision exists.

Lsectors is the touch detection in Vega Prime. Some touch detections need a lot of complicated calculation. According to the detection types, users could use C++ to write the reaction action. The following are the common used modes: Bump, six lines, which go along with the positive and the negative ways of *x*, *y*, and *z* axles bomb; Los, single sight lines, which go along with the *y* axle to radiate, and concentrated in the front data; HAT, single line, which radiates along with the *z* axle, calculating the height of the whole terrain; XYZPR, calculating the angle and steering; and Z, calculating the bomb point on *z* axle. The vehicle with collision is as shown in Fig. 6.

Fig. 6 The vehicle in the scenario with collision

6 Interface of MATLAB and open GI

6.1 Data interfaces and map

The 3D visual simulator model could run real-time only by data driven. We used two ways of data transmission. The first one is loading data files.mat; the second one is dynamic-link libraries (DLL). They own a conventional operating mechanism, which initializes, loads data and then releases the data. We used a factory mode for extending and protecting data [9]. Define an abstract CarData category, which owns three same objects, then define a structural body CarDataS, and provide the detail usage of the category. The relationship between categories is as shown in Fig. 7.

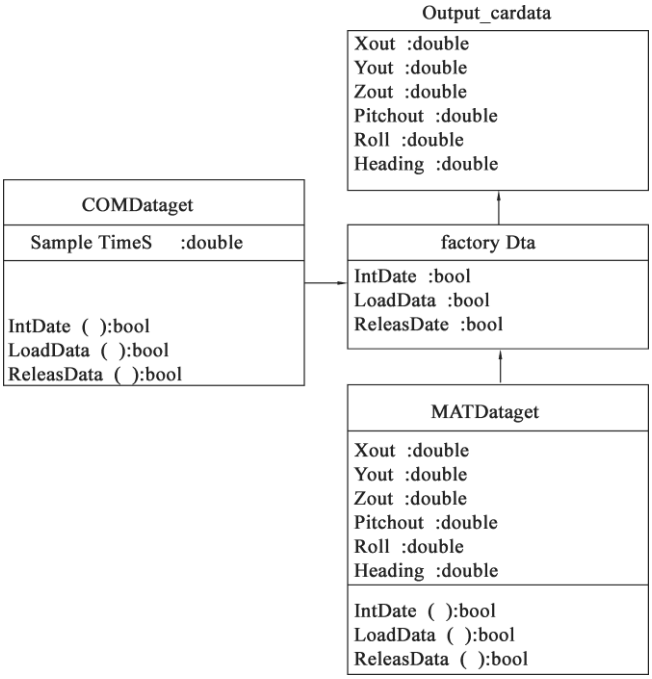

Fig. 7 Data receiving module

6.2 Reading data files

In the testing period of vehicle simulation, vehicle data storage relied on MAT file. .mat is a special purpose binary system file storage. This file could keep the driving data and then call MAT files in C program, so we could use the 3D visual simulator software easily. The common API functions are as shown in Table 1.

6.3 Interface functions

MATLAB provides the calling of C/C++ and FORTRAN language in windows and Linux environment. Microsoft pushes COM module which can strengthen the universalization of MATLAB models. MATLAB could translate the simulator platform into COM module in 32 bit dynamic linked libraries. The following gives users the function interfaces that are supplied by linked libraries.

1) The modelIniPoint function prototype is void modelIniPoint (double *x*, double *y*, double *z*). Three parameters representing the displacement of *x*, *y*, and *z* directions.

2) The modeangle function prototype is void mode angle (double Roll double Pitch double Heading).

The modeIInit function prototype is void modeIInit.

Table 1 Dynamic link library API functions

Function name	Meaning
modelIniPoint	The initial position setting
modeangle	The angle position setting
modeInit	The model initialization
SetSampleTime	Set solution step
getOneStep	Get dynamic data
modelRelease	The model release
…	

3) This will initialize all the data, including the parameters of the car and four resistances of the car, to provide the initial state of the car to start.

4) The SetSampleTime function prototype is void SetSample (double sampTime). This function will set the step size.

5) The GetOneStep function prototype is dataType oneStep (). The function is settled and returns a set of data.

6) The modelRelease function prototype is void modelRelease. It releases the model after the end of the simulation.

The mode of Interface calls InitData () to execute functions (1) to (4) and set the model before solving all the Data loading function. LoadData() will call the getoneStep function to obtain the dynamic data, and then save the dataStore in the m_ cardata structure. The ReleaseData () function will release the model when the simulation is stopped.

7 Conclusions

We introduced dynamic models of automotive, 3D modelling and interfaces, and built a real time driving simulation system based on xPC. The system achieved external data input and instantaneity through xPC. The mix programming of MATLAB and C++ realized that the dynamic model of automotive can drive the 3D visual model. Readers could extend and make simulation analysis on this platform easily, which promote the designing, verification and analysis of automobiles. We just introduced the core part of the design of the driving simulation platform [10].

Acknowledgement

This project is supported by the National Natural Science Foundation of China (No. 51305475), Chongqing Research Program of Basic Research and Frontier Technology (No. cstc2013jcyjA60004), and the Scientific and Technological Research Program of Chongqing Municipal Education Commission (No. KJ1500927).

References

[1] SHIAKOLAS P S, VAN SCHENCK S R, FRANGESKOU I. A real-time digital control environment based on MATLAB, xPC-target along with a magnetic levitation device for neural network control law implementation and verification [C]//Dynamic Systems and Control. ASME 2003 International Mechanical Engineering Congress and Exposition. Washington, DC: ASME, 2003: 1293-1301.

[2] SHI Y Z, WANG X Y, ZHANG H W. Using MEX S C function to write the xPC environment of the study of the driving module [J]. Measurement and Control Technology, 2006, 25(7): 59-60.

[3] MATH W. Real-time workshop for user with SIMULINK [Z]. USA: The Match Works Inc, 2010.

[4] 安丽华.汽车电子稳定性程序(ESP)控制方法及联合仿真研究[D].南京:南京理工大学,2009.
AN L H. Automotive electrical stability program (ESP) controlling method and united simulator [D]. Nanjing: Nanjing University of Science and Technology, 2009. (In Chinese).

[5] PENG P F, WEI L. Research on the vehicle model building technology based on multigen creato [J]. Hebei Traffic Science and Technology, 2010, 7(2): 58-60.

[6] 北京华力创通科技有限公司.Vega Prime培训教材[M].[S. l.: s. n.],2003.
HWA CREATE. Vega Prime training materials [M]. [S. l.: s. n.], 2003. (In Chinese).

[7] CRISP D, INGERSOLL A P, HILDEBRANDET C E, et al. VEGA Balloon meteorological measurements [J].

Advances in Space Research, 1990, 10(5): 109-124.

[8] 严治河.三维设计在公路工程中的应用[J].中外公路,2002,22(4):66-67.
YAN Z H. The application of 3D design in highway engineering [J]. Chinese and Foreign Highway, 2002, 22(4): 66-67. (In Chinese).

[9] MENG X M, LIU W Q. Creator multiGen tutorial [M]. Beijing: National Defense Industry Press. 2005: 161-181.

[10] DONG W G. MATLAB 7.X hybrid programming [M]. Beijing: Mechanical industry Press, 2006: 225-280.