Scientific Research

Torpedo Trajectory Visual Simulation Based on Nonlinear Backstepping Control

Peng Hai-jun[1,2] Li Hui-zhou[3] Chen Ye[1,2]

1. Depart. of Weaponry Eng, Naval Univ. of Engineering, Wuhan 430033, China;

2. Depart. of Aeronautical Armament Fire Control, Naval Aeronautical Eng. Institute Qingdao Branch, Qingdao 266041, China;

3. College of Electronic Eng, Naval Univ. of Engineering, Wuhan 430033, China

1.penghj0383@sina.com, 2.chenye@sohu.com, 3.lihuizhou2121@sina.com

Abstract: Torpedo trajectory visual simulation has great importance on product exploitation and war field use. It's a difficult task because it contains different specialties and has a complex structure. By use of the flexible and efficient Visual C++6.0 and the specialized visual simulation platform Vega, a torpedo trajectory visual simulation system is established. Nonlinear backstepping control strategy is validated on the system. System mainframe and trajectory models are realized in VC. Scenario simulation is realized in Vega. It's shown that this simulation system is favourable. It is helpful in the design and analysis of collectivity, control system and homing system of torpedo.

Key word: visual simulation; torpedo trajectory; three dimension model; backstepping control

1 Introduction

[1]Visual simulation is the fruit of the combination of digital simulation and graphic display technology. Information from digital simulation is presented before researchers in the form of graphics and video. Its user interface (UI) is convenience. Its real-time three dimension (3D) display technology is widely used in virtual training and virtual reality. In the process of weapon system design and validation, it is not economical to test too much times. The data achieved are not enough to evaluate the weapon system completely and correctly. By means of simulation, we can analysis and optimize the digital model of torpedo. It's helpful to develop high quality torpedo at low cost in a relatively short term. Visual simulation can show the movement of weapon intuitively and vividly, and is convenient for real-time mutual operation. It's shown great importance on weapon system design and validation, weapon platform development[1].

MATLAB is brief for programming and graphic display, but it is inefficient in calculation and not compatible with scenario display platform[2]. OpenGL graphic library is widely used for visual application. Although it is versatile and ready for transplanting, the programming are complicated and inefficient[3]. By combination of the efficient programming platform, VC++6.0, and the specialized visual simulation platform, Multigen Vega, a torpedo trajectory visual simulation system is realized. On this system, nonlinear backstepping control strategy of torpedo depth trajectory is simulated and validated. This system can be used for design of torpedo trajectory, control strategy and homing rules. The system is realized on PC with PIV2.0G CPU, Ti4200 video card.

2 Function of system

Digital simulation of torpedo motion and control strategy is carried out on the simulation system. At the same time, virtual scenario is presented to researchers. Functions of the simulation system are listed as bellow:

(1) Calculation of torpedo trajectory. Data from calculation are used for real-time visual display. At the same time, these data are stored to hard disk for analysis and processing.

(2) Integrating with 3D models of torpedo and terrain, virtual scenario is constructed on visual simulation platform Vega. The movement of torpedo will demonstrate in the virtual scenario.

(3) Based on data from calculation, 3D movement of torpedo was presented before researchers. The mode of presentation, such as the distance and the direction of observation, can be controlled by user.

(4) A favorable human-computer interface is established. The process of simulation can be controlled

[1] **Foundation Item:** The National Natural Science Foundation of China (50875259)

978-1-935068-01-3

through the interface, such as starting, suspending and terminating the simulation. Before the start of simulation, parameters of torpedo, target and environment are set in parameter dialog boxes. In the progress of simulation, researchers can inspect data through diagrams.

3 Structure of system

The simulation system is realized with VC++6.0 and Multigen® Vega. VC++6.0 is an object-oriented visual program development environment which originated from C. It inherits the merits of high efficiency, flexibility and good compatibility from C. Its visual integrated development environment is efficient and powerful. So it is often used for programming of complicated software system especially those with heavy calculation and complicated structure[4]. The framework of the system is realized with VC++6.0. Vega is a high performance software environment and toolkit for real-time simulation and virtual reality applications. It consists of a graphical user interface (GUI) called LynX, Vega libraries, C callable functions and other tools[5]. Vega functionality can be extended by additional special purpose modules conveniently. MultiGen® Creator produces realistic 3D models and terrain for use in real-time applications. The torpedo and terrain 3D models are created with Creator and imported into Vega[6]. The calculation is heavy and in real-time. In order not to affect the response speed of the interface, the calculation is carried out in a single thread. Data from calculation are stored in public memory to be used by Vega. There are three modules in the simulation system:

(1) Main interface module. It manages the whole simulation system, such as setting parameters, controlling the progress of simulation, display graphics of motion status data.

(2) Trajectory calculation module. According to parameters from interface, rules of motion, control strategy, it calculates status data of torpedo.

(3) Visual presentation module. It imports 3D models of torpedo and terrain and creates the environment which the torpedo sails in. According to torpedo status data, it demonstrates the movement of torpedo.

The structure of system is shown as figure 1:

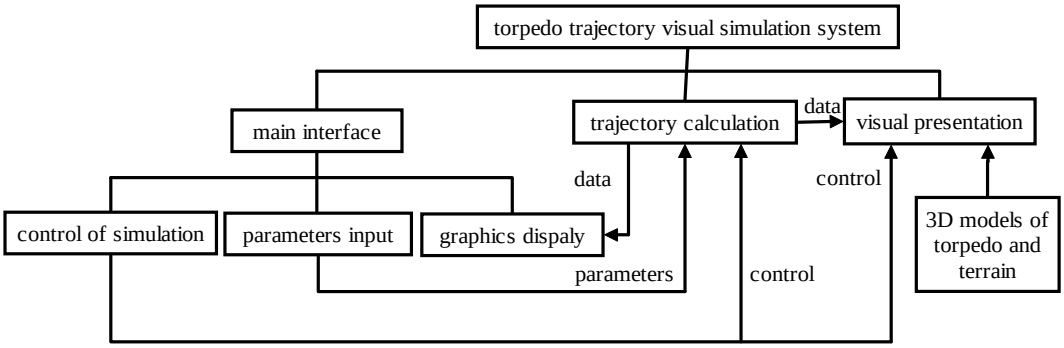

Figure 1 Structure of torpedo trajectory visual simulation system

4 Key technologies

4.1 Mathematic model of torpedo motion

The mathematic model mainly contains kinematical equations and dynamical equations. Dynamical equations describe the dynamic relationship between position, attitude of torpedo and velocity, angular speed of torpedo. Kinematical equations describe the dynamic relationship between velocity, angular speed of torpedo and outside forces, outside moments. By combining kinematical equations, dynamical equations and algebraic equations, we can get a group of equations with 15 motion status variables and 3 rudder variables[7]. This group of equations will be soluble with the supplement of 3 control equations. In control equations, rudder angles are determined by motion status variables according to control strategy. Given initial values of motion status variables, we can get the only solution.

But the group of motion equations described above is nonlinear and too complicated. It is can only be solved by numerical integration. Generally it was divided into horizontal motion and vertical motion. In this article, the

nonlinear vertical motion model is studied and the new stable zero dynamic space is found. The new input-output dynamic space was transformed into strict feedback nonlinear form. Using the lifting velocity and pitching angular speed as virtual control variables, backstepping control strategy is presented[8].

Nonlinear motion model in vertical plane is given in literature[9]:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})u \tag{1}$$

where

$\boldsymbol{x} = [v \quad \alpha \quad \omega \quad \theta \quad y]^T$;

$\boldsymbol{f}(\boldsymbol{x}) = [f_1 \quad f_2 \quad f_3 \quad f_4 \quad f_5]^T$;

$f_1 = k_{11}v^2 + k_{14}\sin(\theta-\alpha) + k_{18}$;

$f_2 = \big(k_{21}v^2\alpha + k_{22}v\omega + k_{23}\alpha + k_{24}\alpha\sin(\theta-\alpha) + k_{25}\cos(\theta-\alpha) + k_{26}\cos\theta\big)/v$;

$f_3 = k_{31}v^2\alpha + k_{32}v\omega + k_{34}\alpha\sin(\theta-\alpha) + k_{35}\cos(\theta-\alpha) + k_{36}\cos\theta$;

$f_4 = \omega$;

$f_5 = v\sin(\theta-\alpha)$;

$\boldsymbol{g}(\boldsymbol{x}) = [0 \quad k_{29}v \quad k_{39}v^2 \quad 0 \quad 0]^T$。

$u = \delta_e$, denotes input control variable. Meanings and typical values of these symbols can be found in appendix 1 of literature [9].

If there is a stable zero dynamic space, we need only to design control strategy of input-output dynamic space. By this control strategy, we can control the depth of torpedo trajectory stably. It was proved that vertical motion model has a stable zero dynamic space:

$$\begin{cases} \dot{v} = k_{11}v^2 + k_{18} \\ \dot{\theta} = 0 \end{cases} \tag{2}$$

Using depth y, lifting velocity v_y, pitching angular speed ω as status variables of input-output dynamic space, this nonlinear space can be transformed into standard strict feedback nonlinear form:

$$\begin{cases} \dot{y} = v_y \\ \dot{v}_y = \boldsymbol{\theta}_2^T\boldsymbol{\varphi}_2 + g_2\omega \\ \dot{\omega} = \boldsymbol{\theta}_3^T\boldsymbol{\varphi}_3 + g_3u \end{cases} \tag{3}$$

Where

$\boldsymbol{\theta}_2 = [k_{11} \quad k_{14} \quad k_{18} \quad k_{21} \quad k_{23} \quad k_{24} \quad k_{25} \quad k_{26}]^T$;

$\boldsymbol{\varphi}_2 = [v^2\sin(\theta-\alpha) \quad \sin^2(\theta-\alpha) \quad \sin(\theta-\alpha) \quad -v^2\alpha\cos(\theta-\alpha) \quad -\alpha\cos(\theta-\alpha) \quad -\alpha\sin(\theta-\alpha)\cos(\theta-\alpha) \quad -\cos^2(\theta-\alpha) \quad -\cos(\theta-\alpha)\cos\theta]^T$;

$g_2 = (1-k_{22})v\cos(\theta-\alpha)$;

$\boldsymbol{\theta}_3 = [k_{31} \quad k_{32} \quad k_{34} \quad k_{35} \quad k_{36}]^T$;

$\boldsymbol{\varphi}_3 = [v^2\alpha \quad v\omega \quad \alpha\sin(\theta-\alpha) \quad \cos(\theta-\alpha) \quad \cos\theta]^T$;

$g_3 = k_{39}v^2$;

The backstepping control strategy is designed with follow steps:

(1) We assume $z_1 = y - y_c$. It denotes the error of depth tracking. Where y_c is depth instruction. The Lyapunov function V_1 is

$$V_1 = \frac{1}{2}z_1^2 \tag{4}$$

(2) We define $z_2 = v_y - v_{yc}$, where $v_{yc} = -c_1z_1$. v_{yc} denotes the virtual lifting velocity instruction. The Lyapunov function V_2 is

$$V_2 = \frac{1}{2}z_1^2 + \frac{1}{2}z_2^2 \tag{5}$$

(3) We define $z_3 = \omega - \omega_c$, where $\omega_c = -\boldsymbol{\theta}_2^T\boldsymbol{\varphi}_2 / g_2$. ω_c denotes the virtual pitching angular speed instruction. The Lyapunov function V_3 is

$$V_3 = \frac{1}{2}(z_1^2 + z_2^2 + z_3^2 + \boldsymbol{\theta}_2^T\boldsymbol{\theta}_2 + \boldsymbol{\theta}_3^T\boldsymbol{\theta}_3) \tag{6}$$

(4) Then we can design nonlinear backstepping control strategy:

$$u = -\frac{1}{g_3}\boldsymbol{\theta}_3^T\boldsymbol{\varphi}_3 \tag{7}$$

4.2 3D models of entities and terrain

MultiGen Creator is a software package designed specifically for the creation of real-time 3D models for visual simulation. Traditional 3D modeling software packages emphasize the visual effect, so models take CPU a lot of time to render. They are not suitable for real-time simulation. Creator is base on OpenFlight data format, which was designed specifically for real-time visual application. OpenFlight has layered data structure and manages data with nodes. These make real-time rendering an easy work[6].

We create 3D model of torpedo following these steps:

(1) Geometric size of the torpedo was collected.

(2) OpenFlight database file is created. Then unit and root node of the database is set.

(3) Branch nodes are created below the root node. These nodes manage certain parts of torpedo model separately, such as models of body, fin, rudder and screw.

(4) Light sources and materials are attached to torpedo model, this makes the torpedo more realistic.

(5) Texture is applied to surface of model to make the model vivid. Trifling details are replaced by texture, so the real-time application will run more smoothly and use less memory.

At the end, we get the torpedo model shown in figure 2. The structure of database is show in figure 3.

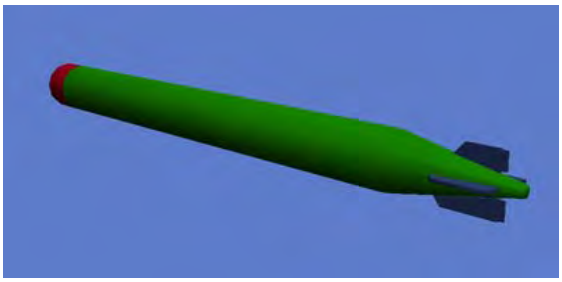

Figure 2 3D model of torpedo

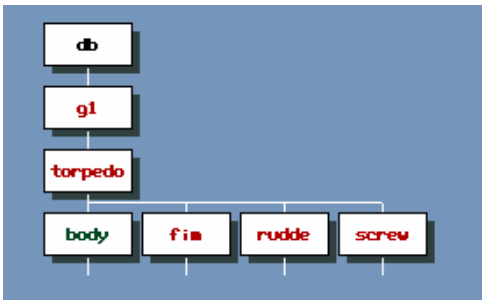

Figure 3 Structure of torpedo model database

Weapon platform and target are modeled in the same way.

Just like entity models, terrain is stored as 3D models. Terrain models are composed of polygons too. Textures and feature data are added into terrain model. Then it can be put into database and driven by real-time application.

The process of modeling sea floor is listed as bellow:

(1) Altitude data files in all kinds of format are converted into “DED” files, which can be edited by Creator. Creator has toolkits to do this work.

(2) DED files and default parameters are loaded into Creator. Then you can select the area your database will cover.

(3) In order to save time of processor, some details of terrain beyond certain distance will be concealed. According to the degree you conceal the details, critical distances are specified.

(4) Feature data are added to terrain.

(5) Selecting algorithm to convert altitude data to triangle data of terrain.

(6) At last, terrain database with feature data is established.

3D model of terrain is shown in figure 4. Structure of terrain model database is shown in figure 5. Each branch node represents a certain area of terrain.

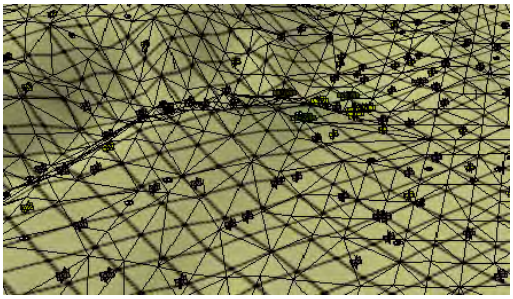

Figure 4 3D model of terrain

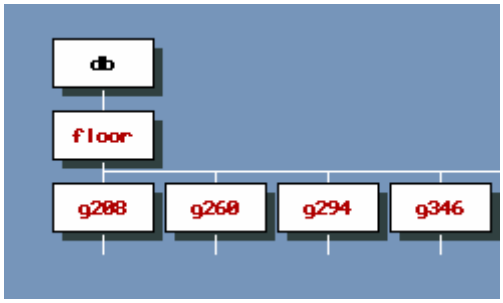

Figure 5 Structure of terrain model database

4.3 Integrating Vega with VC mainframe

This simulation system is constructed with the Document/View mainframe of VC. But the View of mainframe is not derived from class CView of VC[4]. A new class zsVegaView is derived from CView, and functions of Vega are added into zsVegaView. So zsVegaView has all functions of CView and Vega. The

View of mainframe is derived from zsVegaView. So this mainframe has all functions of the one provided by VC, and can make use of Vega.

It is zsVegaView that integrates Vega and VC. Its functions are list as bellow:

(1) At the beginning of simulation, zsVegaView starts Vega thread. Because the scenario needs to be refreshed continually and response to windows messages from users, so the job of Vega is carried out in a single thread. The call-back of the thread is callbackVegaThread ( ).

(2) Attaching the handle of CView's window to Vega system.

(3) Initiating Vega system and setting parameters of Vega.

(4) Selecting mode of observation. In this system, the trajectory zone is very big and the torpedo is relatively rather small, so tether mode is selected. Under this mode, observer will follow the torpedo with a certain distance and angle. User can change the distance and angle by mouse.

(5) The mode of motion is set to user-defined. So the motion and attitude of torpedo are completely decided by real-time input data.

(6) Refreshing the scenario continually.

4.4 Driving the scenario with data from trajectory calculation

The motion of entity in Vega is managed by motion model. There are several motion models in Vega, such as spinning model, UFO model and so on. But all these models are rather simple and difficult to improve. There are not suitable for complex system[5]. By selecting user-defined model, we can replace it with the mathematic model we are studying. Runge-Kutta method is adopted to solve the nonlinear motion model of torpedo. Data produced are provided to Vega for visual presentation.

The approach can be divided into following steps:

(1) Calculating motion status of torpedo. The calculation thread is start at the beginning of simulation. Its call-back is callbackTorpedoCalcThread ( ). Data produced are stored in public memory to be used by Vega.

(2) Implementing motion model of torpedo. First, a user-defined model is registered in Vega system. That is, a call-back callbackTorpedoMotion ( ) is registered. This call-back takes charge of the initiating, updating of position and terminating of motion. Then the motion model is attached to torpedo.

(3) According to the process of simulation, status data of torpedo is passed to torpedo model. Vega uses these data to refresh visual scenario.

Scene of simulation is shown as figure 6.

Figure 6 Scene of simulation

5 Result of simulation

Step length of calculation is 0.001s. Initial status is given as $x_{e0}=0$, $y_{e0}=-10m$, $\theta_0=0°$, $\omega_0=0°/s$, $v_{y0}=0$, $v_{x0}=18m/s$. Depth instruction y_c is set to -25m and -50m. According to motion model (Equation 1) and control strategy (Equation 7), we can get curves of depth data of torpedo trajectory, as shown in figure 7.

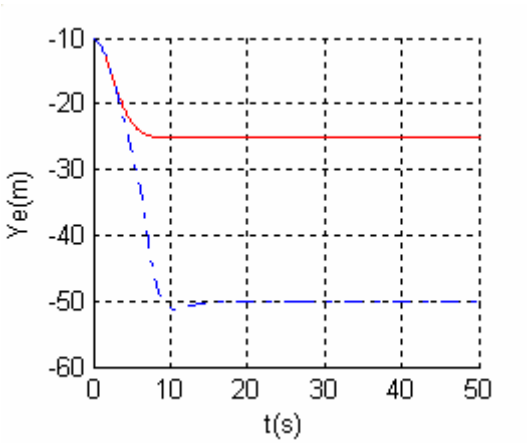

Figure 7 Curves of depth data of torpedo trajectory

It is seen that the depth of torpedo levels off after 10s and 15s.

From the result of simulation, we can find that the

978-1-935068-01-3 © 2009 SciRes.

motion model agrees well with reality and the nonlinear backstepping control strategy is effective.

6 Conclusions

Based on VC++6.0 and visual simulation environment Vega, torpedo trajectory visual simulation system is realized. Considering the nonlinear property of torpedo vertical motion, nonlinear backstepping control strategy is presented and validated on this system. In this simulation system, motion status of torpedo is calculated in real time. At the same time virtual scenario is presented to user. The simulation system characterizes the motion and control strategy of torpedo. Using efficient developing platform, the simulation system saves a lot of CPU time, which is beneficial to extend the system.

Supplemented with signal detection model, homing model, motion model of lunch, this simulation system can be extended into full trajectory simulation system. Methods used in this article are useful for visual simulation of other weapons and weapon platforms such as missile, submarine, and so on[10].

Acknowledgements

The authors are especially grateful for the support and encouragement of Dr. Wang De-shi at Naval University of Engineering.

References

[1] Kang Fen-ju, Yang Hui-zhen, Gao Li-e. Technology and Application of Modern Simulation[M]. Beijing: National Defense Industry Press, 2006.
[2] Chai Lin, Fang Qun. The Simulation of Torpedo's Guidance Trajectory Based on MATLAB / Simulink[J]. Journal of System Simulation, 2003, 15(2): 231-234
[3] Yuan Xu-long, Zhang Yu-wen. The Design and Implementation of Air-Drop Torpedo Trajectory Visual Simulation System[J]. Science and Technology of Vessels. 2002, 24(3): 24-26
[4] Beck Zaratian. Visual C++6.0 Programmer's Guide[M]. Beijing: National Defense Industry Press, 1998.
[5] Wang Cheng. Real-Time Visual Simulation with Vega[M]. Huazhong University of Science and Technology Press, 2005
[6] Meng Xiao-mei. Multigen Creator Tutorial[M]. Beijing: National Defense Industry Press, 2005.
[7] Fan Feng-tong. Dynamics of Torpedo Voyage[M]. Wuhan: Naval University of Engineering Press, 1981.
[8] Chen Ye, Wang De-shi. Adaptive Backstepping Control of Nonlinear Torpedo Attitude System[J]. Journal of Naval University of Engineering, 2008, 20(4): 87-90
[9] Xu De-min. Automatic Control System of Torpedo[M]. Xi'an: Northwestern Polytechnical University Press, 1991.
[10] Qian Xin-fang. Dynamics of Missile Aviation[M]. Beijing: Beijing Institute of Technology Press, 2006.