3D Virtual Simulation Software for Underwater Application

Mohd Salzahrin Mohd Hamzah, Muzammer Zakaria, Mohd Fazli Izwan Abd Jalil, Kamal Zuhairi Zamli

USM Robotics Research Group, School of Electrical and Electronic Engineering
Universiti Sains Malaysia, Engineering Campus,
14300 Nibong Tebal, Seberang Perai Selatan, Pulau Pinang, Malaysia
Tel: +604-5937788 ext. 6009, Fax: +604-5941023,
E-mail:moh_salzahrin83@yahoo.com, rizal@eng.usm.my

Abstract

This paper shows the use of 3D virtual simulation, representing the actual object movement underwater. The 3D virtual environment allows a combination of real and virtual robots to work together for a system-wide study and measurement. The digital signal from controller and sensors which is mounted on the underwater vehicle generates 3D animation on the monitor. The 3D model is design by using the 3D design software with simple 3D objects such as boxes, cylinders, extrusion node etc. Each basic object can be combined to generate complex structures. This paper presents a simulation-based approach that allows a cooperative robotic system to be effectively evaluated in a virtual environment with combined real and virtual robots.

Keywords

3D, Virtual Environment, Software, Underwater, Close Loop

Introduction

The 3D virtual world is an unreal environment represented in 3D. The concept is to link both digital technology and computer vision and become a tool to carry out engineering studies, design analysis and architectural projects.

Virtual world or VR application is used to create visual effect for advertisement, video games, animation, simulator etc. The concept of virtual environment has been largely used by the technology of virtual reality (VR), which has been applied to various areas such as simulation of manufacturing plants, the planning of robotic workcells and robot operation system. The research of the VR mainly focuses on the interaction between robots and the virtual environment.

According to Xiaolin Hu and Bernand P. Ziegler [1], different configurations can be easily applied to experiment and measure the performance of the system under development. To allow simulation of robotic systems that actively interacts with an external environment, an environment model needs to be created. This environment model serves as a virtual environment to provide sensory input to robot models and to response to robots actuation. For example, a virtual obstacle that can be sensed by robot models and it responds to robot's movements by updating new sensory information to robot models.

This paper tries to emphasize the idea of using the 3D virtual world to represent the actual robot in order to monitor and inspect the movement and the condition of the robot. This paper will also describe the use of Webots software to develop a virtual simulation for underwater robot application.

Approach and Method

Webots Software

The software being use for designing the 3D virtual simulation is WEBOTS^{TM} software. Webots is a professional mobile robot simulation software package. It contains a rapid prototyping tool, allowing user to create 3D virtual worlds. Webots runs on Windows, Linux and Mac OS X and is suitable for researchers and designers interested in mobile robotics.

A world in Webots is a 3D virtual environment in which we can create objects and robots. A world is a hierarchical structure where objects can contain other objects. As to run the simulation, we need a controller. A controller in Webots is an executable binary file which is used to control a robot described in world file. The control may be native executable files (.exe in Windows) or Java binary files (.class).

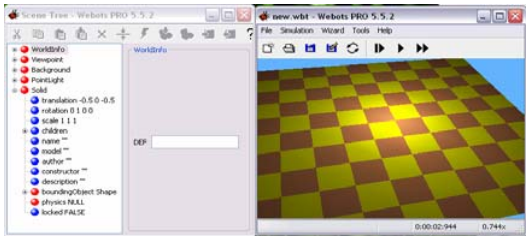

Figure 1 – Webots main window.

The main window for Webots are divided into two, on the right is the design field. The design can be view in this window. The left window is the scene tree window. This is where the designing took place. The method being used is the same like VRML methods where the object is being choose from the basic objects such as boxes, cylinders, extrusion node etc. Then by the combination of these simple objects will become a very complex 3D objects. The object can vary in sizes and shapes as it can change the values for x-axis, y-axis and z-axis for that particular object.

Simulation Controller

Inside Webots, there is a program that can help user to make a controller for their design. The controller can be either in C, C++ or Java language. For the ROV design, the controllers are in C and C++ language. At the beginning of the code, it needs to declare every component that being used in the design in order to make it work. As example, the camera must be declared first in order the camera to display its function inside the simulation.

In the main part of the code where the controller of the simulation will take place. Here, the code must be written accordingly so that the simulation will run smoothly. To connect the joystick with the controller program, a certain source code or header file for the joystick must be called into the program. This header file will make the program to detect and determine joystick that is attached to the computer. Then it will configure the joystick as to declare on how many buttons and axes that the joystick has.

For example, when the joystick is moved forward, the code will read the data sent by the joystick and put it into a memory. Next, the data needs to be declared also. So as if the data for forward is from 0 to +1, the 3D ROV simulation will show forward movement when it receive data or value that is between 0 and +1.

Results

Figure 2 shows the result of the ROV design using Webots. As it can be seen, the design is the combination of those simple objects with different sizes. The thrusters are being represented by cylinders. The body of the ROV is the combination of boxes and the frame is the combination of extrusion node.

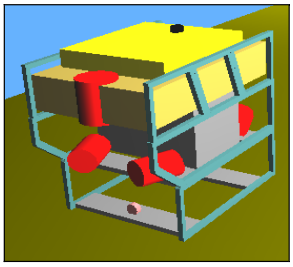

Figure 2 - 3D ROV visualization using Webots.

Figure 3 shows the design of the virtual environment as the condition is underwater. The design for the sea-bottom is made from extrusion node. Then there are some boulders made from the combination of objects. It can be seen in the figure, where the ROV is doing some inspection on the pipeline. The pipeline is made from different sizes of cylinders.

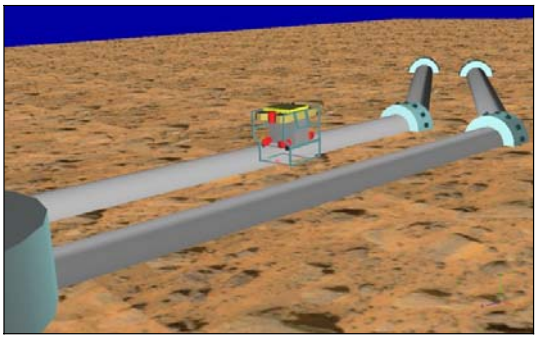

Figure 3 – 3D virtual environment design

The 3D design has 5 DOF (Degree of Freedom), same with the real ROV design. The simulation can move forward, backward, sway left and right, heave up and down and it can rotate or yaw to left or right and roll.

The "differential_wheel" node can make the simulation to move forward, backward and it can show the rotation of the simulation. The differential_wheel robot will automatically search for and take control wheel objects within its children. That is why the robot can do movements just like a differential two-wheel steering even though the ROV design doesn't have wheels. The speed for both the wheel; "right wheel" and "left wheel" can be specified by the **differential_wheel_set_speed( )** function. When moving forward, the speed for both wheels are

having the same positive values. While moving backward, the speeds are negative values. Then to make the sway movements, I used the "servo" node. A servo node is used to add one (active of passive) degree of freedom in a mechanical simulation. The DOF is created between the parents and children nodes (VRML hierarchy). The servo node is the children of the differential_wheel node, therefore it allow moving the servo children with the respect to the parents. The servo can be linear or rotational. I used this servo to show sway movements, so it is better to choose the linear servo as it is used to simulate a sliding motion.

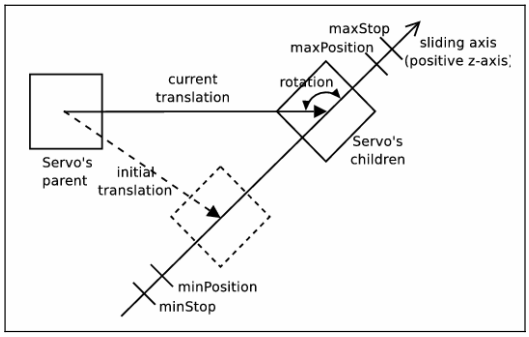

Figure 4 – Linear Servo.

Next, I use another servo node to show heave movement. The method is similar with the first servo and I make the second servo becomes the children to the first servo. The 3D simulation also display the view from the camera attached on the ROV. The use of this camera is to show the condition of the ROV. One camera will show the front of the ROV and one more will show the side view of the ROV.

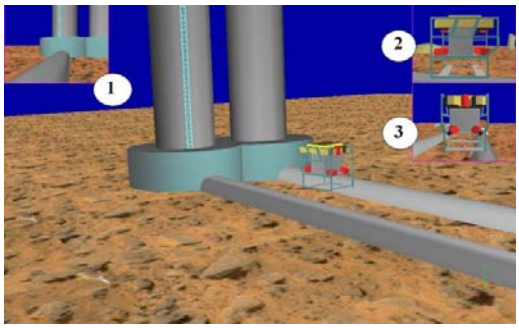

Figure 5 – camera display of the ROV

Camera 1 will display the view from the ROV camera. Camera 2 will display the left view of the ROV and camera 3 will display the front view of the ROV. The 'roll' data from the sensor will be use to display the roll condition for the ROV.

Controller

As in ROV, digital control is used to control the thruster driver which is being used to control the ROV position and heading. For now, the vehicle is controlled using joystick and SIMULINK software. For the computers communication, the data are sent and receive through Power Line Carrier (PLC) using User Datagram Protocol to communicate.

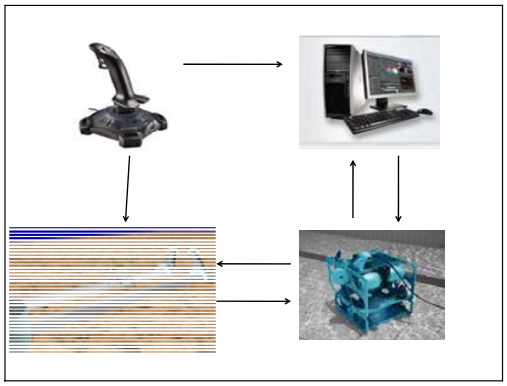

Figure 6 - Communication line diagram

The 3D simulation simulates according to joystick control, so the 3D simulation program and digital control program will run at the same time. This is an open loop system.

Table 1 – ROV Condition Table

ROV CONDITION	*T1*	*T2*	*T3*	*T4*	*T5*	*T6*
Stop (initial)	*0*	*0*	*0*	*0*	*0*	*0*
Forward	*+1*	*+1*	*-1*	*-1*	*0*	*0*
Backward	*-1*	*-1*	*+1*	*+1*	*0*	*0*
Shift-L (sway)	*+1*	*-1*	*-1*	*+1*	*0*	*0*
Shift-R(sway)	*-1*	*+1*	*+1*	*-1*	*0*	*0*
Spin-L (yaw)	*+1*	*-1*	*+1*	*-1*	*0*	*0*
Spin-R (yaw)	*-1*	*+1*	*-1*	*+1*	*0*	*0*
Up	*0*	*0*	*0*	*0*	*+1*	*+1*
Down	*0*	*0*	*0*	*0*	*-1*	*-1*

Legend:
+1 = clockwise turn (forward force)
-1 = counterclockwise turn (reverse force)

Table 1 shows the ROV condition table shows the movement of the ROV and thrusters activation for each condition. Compass-gyro sensors are used to sense the actual condition of the ROV in the water. The sensors data will be sending from the ROV to the surface computer through UDP communication. Sensors data are displayed on the Guide User Interface (GUI) to verify the 3D animation. The data are compass (heading), pitch, roll and depth.

Comparison

There are many simulations software being builds especially for the underwater application such as the simulation from Marine Simulation [4]. The company has developed the ROVsim Undersea Pilot Series. The system uses the latest technology in real-time 3D visualization software. It is able to simulate a various missions.

The system features accurate physics simulation with good visualization and almost realistic environment.

Figure 7 – ROVsim mission window.

As can be seen, there are differences if compared to this simulation where the simulation have a several of useful information plus it also display sonar. The video monitor 1 is the main window where this is the view from the ROV camera. The video monitor 2 is the view of the sonar and camera 3 will display the view of the ROV from behind. The Vehicle Main Operational Controls is at the bottom of the main window. In the panel contains the ROV data such as heading, depth, altitude and speed.

The software that I build will show the movement of the ROV in real-time. This is because I used the close-loop control application in the design. The main controller is the joystick but in the same time, the software also receives data from the sensors on the ROV such as roll, pitch, heading etc. These data will alter the condition and movement of the ROV in the simulation.

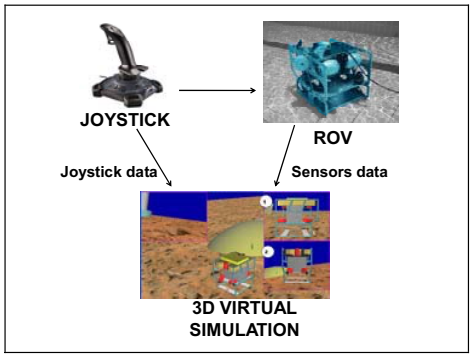

Figure 8 – Close Loop Control

Discussion

The connection between Webots software and sensor is by using the serial port. The compiler of the Webots needs a header file for the serial port. After the header file is included and all the declarations have been made, then we can receive the data from the sensors.

Serial Port Communication

The protocol of serial port communication can be done by certain procedure. The serial port can be called in order the user to read/write data from/to the serial port. By using C/C++ programming in Webots, calling serial port can be done.

First of all, the compiler for the programmer must recognize the coding for serial port. This can be done by adding some information in the 'makefile'. This file is the main compiler for a Webots C++ controller.

Here how it should been written:

SOURCE_INCLUDES = js.h, Tserial.h
CPP_SOURCES = Joystick.cpp

"Tserial.h" is the header file for recognizing the RS232 serial port. Next, in the main controller programming part, the "Tserial.h" file must be defined so that the controller will make a link with the file. This is not the only file that we must define. One more is the "tserial.cpp" file. These two files are related to each other. Here how it should be written in the controller programming part.

#include "Tserial.h"
#include "tserial.cpp"

After we include the file with the main controller, next we need to manually define the definition that we need to use for calling the serial data from the port.

*Tserial *com;*

"Tserial" is the header file that we defined earlier. "com" will be the definition that we will use in the entire code to represent the COM1 of the serial port. Next we will connect the serial port and call the data from the serial port.

com = new Tserial ();
com->connect ("COM1", 19200, spNONE);

"COM1" is the port argument and "COM1" is the default connection for serial port. The value 19200 is the rate argument of the serial port. The spNONE is the serial parity of that port.

As for the device is concern, the sensor data is in a string of characters, which is mean the data is send in a set of data. So if we try to call one bit at a time, we will only get only one bit of the entire string. For example, one string of data can have at almost 50 single bits of data. So here, we need to call for the entire set of data or we called it as a string.

```
com->getArray(buffer,50);

    for(int i=0; i<50; i++)
    {
     robot_console_printf("%c",buffer[i]);
    }
    robot_console_printf("\n");
```

"getArray" is used to get a set of data that have been placed in a buffer. A buffer can have a set of data, like in this case; our string data produces up to 48 to 50 bits of data. So the value 50 in the bracket after the "getArray" is the value of bits of data that we wanted to get from the serial port. The '**for**' loop is use to get the whole data in one set and then it will be displayed in the log window of the Webots. If we wanted the log window to display the data continuously, so we need an infinity loop.

So after the data have been displayed continuously, we can use these data as for data logging. The data will differ according to the movement of our sensor and it will be update for every milliseconds.

The data from the sensor is a series of string data, which is mean it is not a numerical value. First it needs to be separate into five sets of data. The data for heading, depth, roll, pitch and temperature. Next the data must be converted into an integer so that the values can be use in the simulation.

Close Loop Control

The communication for close loop control must meet in real time. The real time communication avoiding the controller loses control and easy to avoid the underwater vehicle hitting unexpected object under water especially fish and rock.

The virtual 3D animation was easily control by the joystick because it generates binary data rather than compass-gyro sensor which it generates string data in NMEA code. Binary data transfer is much faster than NMEA data transfer. The NMEA data need to be separated to five kinds of data. The five data's are heading, pitch, roll, temperature and depth. Each data delimited by an alphabet. Example: $C328.3P28.4R-12.4T21.1D21.01. Between 'C' and 'P' is the value of heading, between 'P' and 'R' is the value of pitch and so on. This separation made the process slower and cannot meet the real time communication.

Conclusion

This paper has shown how the 3D virtual software can be used to simulate the underwater vehicle. The 3D virtual software is showing real condition of the underwater vehicle even cannot be seen in real world. Furthermore, this kind of virtual simulator allows the user to interact with the underwater vehicle comfortably. It also allows the user to check the best equipment to carry out future projects.

Acknowledgments

This project is sponsored by National Oceanographic Directorate (NOD) which is from Ministry of Science and Innovation (MOSTI).

References

[1] Hu, X., and Ziegler, B.P., "*Measuring Cooperative Robotic Systems Using Simulation-Based Virtual Environment*", Computer Science Department, Georgia State University, Atlanta GA, USA 30303

[2] Michel, O. / Cyberbotics Ltd – Webots™: Professional Mobile Robot Simulation, pp. 39-42, *International Journal of Advanced Robotic Systems*, Volume 1 Number 1 (2004)

[3] Cyberbotics, October 8th, 2007, Webots™ 5 User Guide. URL: http://www.cyberbotics.com

[4] Marine Simulation LLC (2008) – ROVsim® Underwater Pilot Series™. URL: http://marinesimulation.com