Micro ROV Simulator

Zoran Fabeković [1], Zdravko Eškinja [1], Zoran Vukić [2]
[1] Brodarski Institut, Ave. V. Holjevca 20, 10020 Zagreb, Croatia
[2] University of Zagreb, Faculty of Electrical Engineering and Computing, Unska 3, Zagreb, Croatia
E-mail: zoran.fabekovic@hrbi.hr

Abstract - *The ROV Simulator was developed as modular structure with the possibility of connecting signals received from the real ROV. Programming language VRML helped in developing a 3D model of the ROV and the interaction between the ROV and the virtual world has been achieved. The kinematics of the ROV was implemented for simulator needs. The simulator menu offers different virtual environments. The simulator has different additional possibilities such as: gripper management, realistic illumination and camera manipulations.*

Keywords – *Simulator, ROV, AUV, 3D visualization, ROV model, kinematics, Virtual Reality, VRML*

1. INTRODUCTION

In recent years, increasing attention is dedicated to the development of unmanned vehicles. Reasons for growing research efforts in this area are in the inconvenience of human work in inaccessible, and sometimes life threatening surroundings, and lower mission costs if men are not included.

Unmanned vehicles can be classified by control method and by surroundings in which they operate. By control methods there are autonomous vehicles and remotely operated vehicles. By the surroundings in which they operate there are underwater, surface and air vehicles. Unmanned vehicles find the use in different areas of human activities: military, industry, medicine, space research, underwater research, clearing mined areas etc.

For practical as well as financial reasons simulators are used during the development of unmanned vehicles. Simulators are models of real systems which contain all their important characteristics. Most often they are used for testing of control algorithms, for planning mission algorithms and behavior testing of systems in predicted and unpredicted, normal and extreme conditions. They are successfully used in training sessions for operators who control vehicles when training on real objects is too expensive or simply too hot to handle.

The fact that over 70% of Earth surface is covered by water and that most of that area is not investigated gives us enough motive for developing new unmanned underwater vehicles.

Unmanned underwater vehicles are divided by control method into autonomous unmanned vehicles (AUV) and remotely operated vehicles (ROV). Small size ROV-s are also called Micro ROV-s (Fig. 1).

Fig. 1. Micro ROV

2. SIMULATOR STRUCTURE

The simulator is developed to be modular with the possibility of enlargement of functionalities and simple parameter alternation [1].

2.1. Structure of entire simulator

Structure of the entire simulator is shown in Fig. 2. The simulator input is represented by the control part of the user interface which contains the joystick and other controls (camera, light controls etc.).

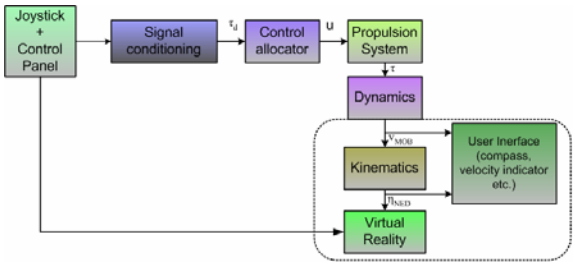

Fig. 2. Structure of the simulator

The next block is used for joystick signals conditioning. This block output is the vector of desired forces and moments $\underline{\tau}_d$. The *Control allocator* task is used to calculate control signals ($\underline{u}$) for individual actuators, such that the real forces and moments (the *Propulsion System* output) produced by actuators are equal to the desired ($\underline{\tau} = \underline{\tau}_d$). The forces and moments given by the propulsion system are used as inputs for the dynamic model (*Dynamics* block).

The dynamic model calculates ROV velocities in mobile coordinate frame ($\underline{\nu}_{MOB}$) which are used as inputs for the kinematic model – *Kinematics* block. Using these velocities and with presumption that ROV initial position is known, kinematic model gives the ROV position and orientation in the Earth-fixed coordinate frame ($\underline{\eta}_{NED}$). Position and orientation vector $\underline{\eta}_{NED}$ together with data received from control panel are used in process of creation as well as refreshing virtual reality and the indicator part of user interface.

2.2. Structure of implemented part of simulator

The implemented part of the simulator is shown in Fig. 2 surrounded by a dashed line . It consists of the kinematics and visualization which is realized using virtual reality. This part of the simulator, realized in MATLAB (The MathWorks Inc.) and adjusted for testing, is shown in Fig. 3.

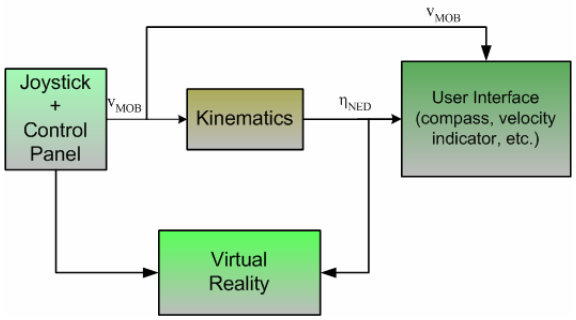

Fig. 3. Implemented part of the simulator

3. KINEMATIC MODEL

In order to develop the kinematic ROV model it is necessary to define two different coordinate frames. The first one is the Earth-fixed coordinate frame - NED (North-East-Down) which is defined in the way that the orthogonal axes N and E stretch out the tangential plane on the Earth surface, while the third axis D is directed down to the center of the Earth and together with axes N and E creates a right-oriented orthogonal coordinate frame.

The second one – mobile coordinate frame (MOB with axes x, y and z) – is connected to the ROV. At the start of the simulation its axes are matching the Earth-fixed axes: x corresponds to N, y to E and z to D. Mobile and Earth-fixed coordinate frames are shown on Fig. 4.

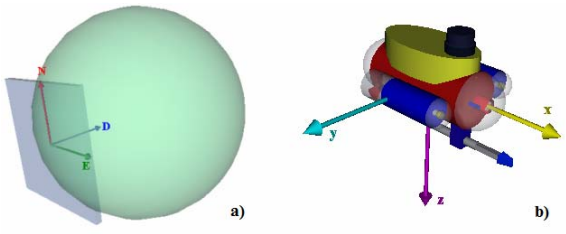

Fig. 4. Coordinate frames: a) Earth-fixed, b) Mobile

To realize the kinematic model [2, 3] a new notation of coordinate frames is needed. Position and orientation vectors are given in relation to the Earth-fixed coordinate frame:

$$\underline{\eta}_{NED} = \left[\underline{\eta}_{1NED}^T, \underline{\eta}_{2NED}^T\right], \tag{1}$$

where the ROV position vector in relation to NED is:

$$\underline{\eta}_{1\,NED}^{\,T} = \left[x\,,\, y\,,\, z\right]^T\,, \tag{2}$$

and ROV orientation vector in relation to NED is:

$$\underline{\eta}_{2NED}^T = \left[\phi, \theta, \psi\right]^T. \tag{3}$$

Velocities are expressed in relation to the Mobile coordinate frame:

$$\underline{\nu} = \left[\underline{\nu}_{1MOB}^T, \underline{\nu}_{2MOB}^T\right], \tag{4}$$

where the linear velocities vector in direction of x, y and z coordinate axes is:

$$\underline{\nu}_{1MOB}^T = \left[u, v, w\right]^T, \tag{5}$$

and the angular velocities vector arround x, y and z axis in MOB coordinate frame, in counter-clockwise (CCW) direction is:

$$\underline{\nu}_{2MOB}^T = \left[p, q, r\right]^T. \tag{6}$$

The kinematic model can now be written in vector form as:

$$\begin{bmatrix} \dot{\underline{\eta}}_{1NED} \\ \dot{\underline{\eta}}_{2NED} \end{bmatrix} = \begin{bmatrix} J_1(\underline{\eta}_{2NED}) & 0_{3\times3} \\ 0_{3\times3} & J_2(\underline{\eta}_{2NED}) \end{bmatrix} \begin{bmatrix} \nu_{1MOB} \\ \nu_{2MOB} \end{bmatrix}, \tag{7}$$

where:

$$J_1(\underline{\eta}_{2NED}) = \begin{bmatrix} c\psi\, c\theta & -s\psi\, c\theta + c\psi\, s\theta\, s\phi & s\psi\, s\phi + c\psi\, s\theta\, c\phi \\ s\psi\, c\theta & c\psi\, c\theta + s\psi\, s\theta\, s\phi & c\psi\, s\phi + s\psi\, s\theta\, c\phi \\ -s\theta & c\theta\, s\phi & c\phi\, c\theta \end{bmatrix}, \quad (8)$$

$$J_2(\underline{\eta}_{2NED}) = \begin{bmatrix} 1 & s\phi\, t\theta & c\phi\, t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi / c\theta & c\phi / c\theta \end{bmatrix}, \quad (9)$$

where: $s\cdot = \sin(\cdot)$, $c\cdot = \cos(\cdot)$ and $t\cdot = \tan(\cdot)$.

As it is shown in equation 9, there is a problem in the realization of this kinematic model because there exists possibility of dividing by zero in the case of a 90° twist around y axis (pitch angle = ±90°). This problem is usually ignored because most ROV-s, including this one, cannot move in such way.

4. USER INTERFACE

The user interface is designed in two functionally different parts: control user interface and visual user interface.

4.1. Control User interface

A set of various simulator controls, including joystick controls and initial adjustments, can be found on the control user interface.

The joystick and its controls are shown in Fig. 5, while Table 1 presents joystick axes and buttons with assigned functions.

Table 1.

Control			***Function***
Axes	*X-axis*		*Forward / Backward*
	Y –axis		*Roll*
	Z – axis (Throttle)		*Depth control*
Buttons	*Button 2*		*Coordinate axes – on/off*
	Button 3		*Perspective (view point) control*
	Button 4		*Perspective – RESET*
	Button 5		*Gripper – close*
	Button 6		*Gripper – open*
	Button 7		*Light intensity - down*
	Button 8		*Light intensity - up*
	POV (Point Of View)	0°	*Tilt Up*
		90°	*Zoom In*
		180°	*Tilt Down*
		270°	*Zoom Out*

Various initial adjustments are available, such as joystick axes sensitivity, dead-zones for each axis, propeller rotation direction (CCW, CW).

Also, it is possible to choose between the three surroundings shown in Fig 6.

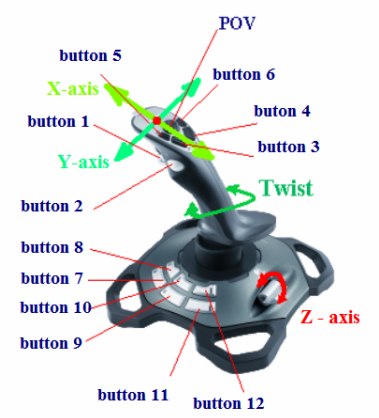

Fig. 5. Joystick axes and buttons

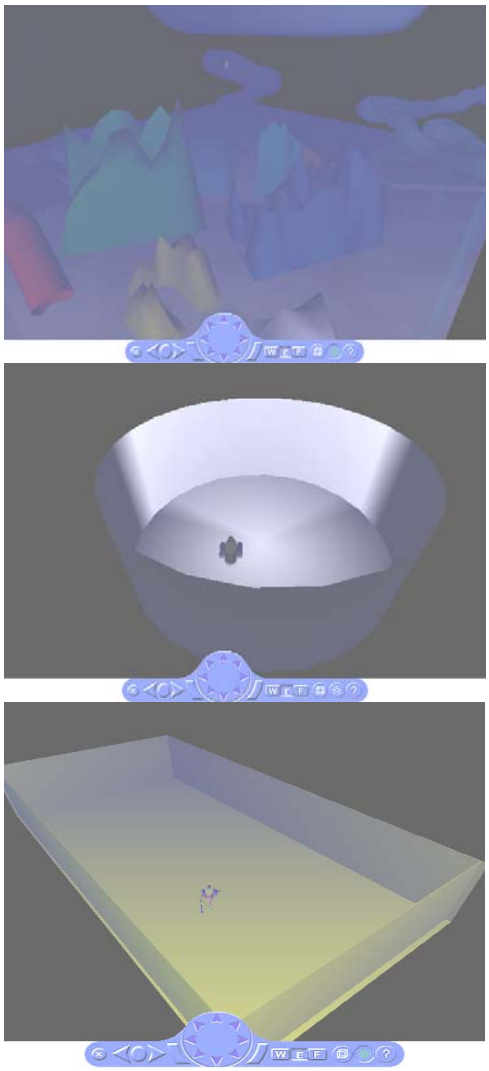

Fig. 6. Selectable surroundings

4.2. Visual user interface

The visual user interface consists of two parts: the indicator user interface and virtual reality.

The simultaneous use of both interface parts gives the user better and more complete picture about real ROV position and orientation and about ROV's interaction with the environment. Fig 7 presents the indicator user interface which consists of speed, depth, position and heading indicators.

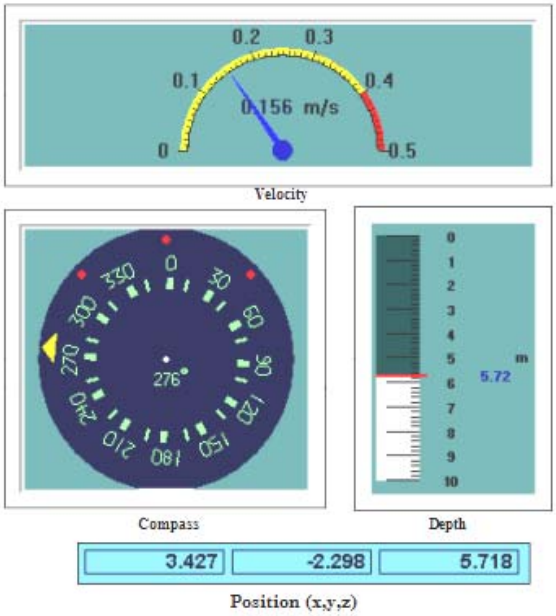

Fig. 7. Indicator user interface

ROV position is determined by the position of the mobile coordinate frame origin in the fixed coordinate frame, while the depth is just a position coordinate in D-axis direction. The shown speed is the absolute translatory speed of the ROV:

$$vel = \sqrt{u^2 + v^2 + w^2}\ . \tag{10}$$

Heading can be determined from the ROV's orientation vector in the NED coordinate frame - equation (3). If the z-axis of the mobile coordinate frame is parallel with D-axis of the fixed coordinate frame, which is true in this case, the course is the third element of the $\underline{\eta}_{2NED}$ vector.

VRML (Virtual Reality Modeling Language) is used for realization of virtual reality [4, 5]. It is a programming language which is used for 3D interactive scenes visualization, and it is primarily developed for 3D visualization over the Internet.

Using VRML it is possible to change scene objects colors, textures and transparency. It is possible to integrate animations, sounds, light effects, and various sensors into the virtual world. Program code implementation is possible (written in *Java* or *Java script*) by insertion of *Script Node*. Files created in this format have *.wrl* extension and are saved in textual format, which means that only a text editor is needed for VRML programming. Fig. 8 shows one scene in virtual reality during a simulation.

Fig. 8. Virtual reality scene – in the middle of a simulation

The blue section in the bottom part of Fig 8 is used for viewpoint control and navigation through the virtual world. This way it is possible to observe specific situations from various viewpoints, in order to get the real picture of ROV's position and orientation in its surrounding.

5. SIMULATOR VERIFICATION

The simulator implements many features like: various predefined points of view fixed to ROV and viewpoints fixed to the Earth-fixed coordinate frame, camera management (zoom in/out, tilt up/down), point of view control using joystick, lights which enable realistic scene illumination in every moment of simulation, changing ROV lights intensity, visibility decrease in relation to ROV's depth, gripper management, change of propeller rotation direction and velocity in relation to ROV direction and velocity, switching on/off of coordinate axes.

Some examples of mentioned features are shown in text and figures below. Front camera control is presented in Fig. 9. Fig. 9a to 9e show the front camera view and Fig. 9f is obtained by changing the point of view using joystick button 3 and joystick manipulations. The yellow spot in Fig 9a, 9b, 9c and 9e is the mobile coordinate frame x-axis.

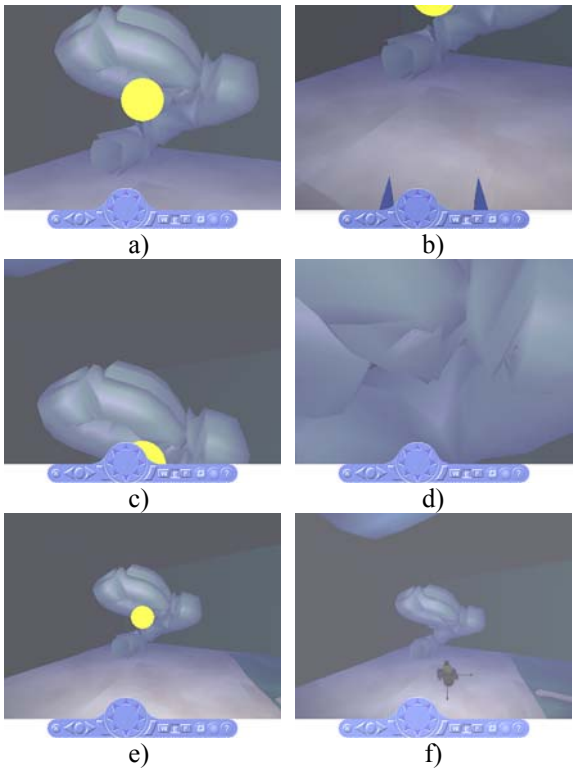

Fig. 9. Front camera and point of view control using joystick: a) initial position, b) tilt down, c) tilt up, d) zoom in, e) zoom out, f) changing point of view

Gripper and light intensity management is shown in Fig. 10a and Fig 10b. The front camera view with low light intensity and with gripper opened just a bit is shown in Fig 10a, while Fig 10b presents fully opened gripper and maximum light intensity. Fig. 10c presents the point of view fixed to Earth-fixed coordinate frame, while coordinate axes are switched on. Collision detection problem is shown in Fig 10d – ROV is passing through the wall.

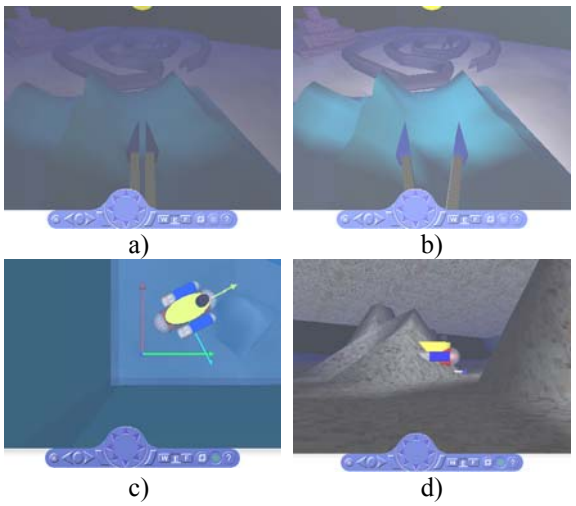

Fig. 10. Lights and gripper management

6. CONCLUSION

Although the implemented ROV simulator does not contain ROV dynamics, it has many other features. The simulator is modular in structure and it supports connecting a ROV dynamics module and other upgrade modules. Also, it supports connecting real ROV signals to the simulator visualization block. This feature allows many new possibilities such as ROV visualization while the operator cannot directly see the ROV. The kinematic model and the whole visualization are implemented in the virtual reality. A big lack of this simulator is nonexistence of collision detection between ROV and other objects in the virtual scene. The virtual model implements adjustable dynamic lights which move together with the ROV and in this way contributes to even more realistic visualization. The visibility change in relation to ROV depth contributes to the same realistic feeling. Gripper management is also implemented. Some added simulator features like front camera zoom and coordinate axes on/off switch, are not present on this actual ROV – VideoRay Pro II. Three types of environment are offered. Complex area is used for operator training, while the circle and rectangular pool are used for laboratory research needs. Rotation direction selection of each propeller is offered by menu.

The implementation of ROV dynamics is needed for future research. In order to make the simulator closer to reality, it is needed to implement modules which simulate disturbances like sea waves and sea current [6], and to solve the collision detection problem.

ACKNOWLEDGEMENT

We would like to thank Edin Omerdic, Nikola Mišković and Marin Stipanov for their support and helpful comments during development of this simulator.

REFERENCES

[1] Z. Fabeković, *VideoRay Pro II Simulator*, Master Thesis, University of Zagreb FER, Croatia 2006. (in Croatian)

[2] Thor I. Fossen, *Guidance and Control of Ocean Vehicles*, John Wiley & Sons, 1994.

[3] Z. Kovačić, S. Bogdan, V. Krajči: Robotics Basics, Graphis, Zagreb 2002. (in Croatian)

[4] VRML 2.0 specification, Available at http://graphcomp.com/info/specs/sgi/vrml/spec

[5] VideoRay Pro II technical specifications

[6] Z. Vukić, Lj. Kuljača, *Automatic Control - linear systems analysis*, Kigen, Zagreb 2005. (in Croatian)