Hydrodynamics and Motion Simulation of the Spherical Underwater Robot in Gazebo using ROS Tools.

○Awa TENDENG (Kagawa Univ.), Shuxiang GUO (Kagawa Univ.), Ruochen AN (Kagawa Univ.) and Liang ZHENG (Kagawa Univ.)

Abstract: In this paper, we carried out the simulation of the Spherical Underwater Robots (SUR) in a dynamic underwater environment. We configured a 3D model of the robot and settled the robot's hydrodynamics properties using Gazebo plugins. In the experiments we carried out, 4 robots were launched in the simulation and each one separately controlled. We examined the motion performances and adjusted the required parameters to achieve a realistic simulation of the robots' motion.

1. Introduction:

In recent years, many types of research have been conducted to propose and develop various kinds of autonomous underwater robots to achieve tasks in harsh environments. In the previous works, a team of amphibious spherical robots has been developed in our Lab [1][2]. Underwater is a very dynamic environment and this property makes it challenging to control and maneuver AUVs (Autonomous Underwater Vehicles) [3][4]. Therefore, it becomes essential to develop and test control algorithms before implementing them directly. Carrying out a realistic simulation, which considers the complexities inherent to the marine world, allows to prepare and anticipate the deployment work beforehand. In this research, we performed a 3D simulation of the spherical underwater robot using Gazebo software and ROS (Robot Operating System). We configured a 3D model of the robot, settled the physical parameters of the water environment such as hydrodynamics and buoyancy using Gazebo plugins, and Finally, evaluated the performances of the simulated robot.

2. Simulation software and robot model's description:

2.1. Gazebo simulator:

The Gazebo simulator is a powerful software that enables realistic 3D simulations. Gazebo comes out with a set of plugins that allow integrating into the simulation the physical characteristics of the environment in which the robot evolves (gravity, light, friction, etc.), as well as the sensors. Also, it enables the user to extend its functionality through user-defined plugins. The plugins use the simulator programming interface to access the simulation data and objects, and apply forces and torques to objects. In the case of this work, ROS, which is a set of tools for robot simulation is used with Gazebo. The tools provided by ROS allow to interact with the simulation and to send commands to the robot.

2.2. Robot's simulation model:

In previous work, a 3D model of the spherical robot was designed [5]. Thus, we modified the different 3D parts with Blender software to adapt the dimensions and made the assembly in an URDF file using Rviz for visualization. In its final version, the robot's model is composed of twelve actuators. 4 servomotors are used to steer the robot on the vertical axis, 4 others for the movement in the x, y plane and 4 motors to actuate the thrusters. In the simulation, the 3D model is used for the visual. Conversely, simple geometrical shapes are used for the collision properties. Using simple geometrical shapes for collision allows less computation cost which leads to a better simulation performance. Also, the simulation requires to know the distribution of the mass in the rigid body, the inertia tensor. That tensor was calculated for each link using the symmetry properties of the collision shape. The general expression of the inertia tensor is given on the following 3×3 matrix (eq. 1).

$$\mathrm{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{1}$$

The main body of the robot was assimilated into a sphere. Therefore, the inertial matrix is obtained from a formula depending only on the mass (m) and the radius (r). Also, by considering the symmetry plans of the body, some of the products of the tensor become equal to zero (eq. 2).

$$\mathrm{I} = \begin{bmatrix} \frac{2}{5}mr^2 & 0 & 0 \\ 0 & \frac{2}{5}mr^2 & 0 \\ 0 & 0 & \frac{2}{5}mr^2 \end{bmatrix} \tag{2}$$

The servomotors can be considered as boxes of mass m, depth d, width w, and height h. Therefore, by considering the symmetries of the body, the expression of the inertia tensor is simplified (eq. 3).

The motors and propellers were assimilated to cylinders of radius r and height h. The inertia tensor was given by the on equation 4.

SY0009/20/0000 - 1356

$$I = \begin{bmatrix} \frac{m(h^2+d^2)}{12} & 0 & 0 \\ 0 & \frac{m(w^2+h^2)}{12} & 0 \\ 0 & 0 & \frac{m(w^2+d^2)}{12} \end{bmatrix} \quad (3)$$

$$I = \begin{bmatrix} \frac{1}{12}m(h^2+3r^2) & 0 & 0 \\ 0 & \frac{1}{12}m(h^2+3r^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix} \quad (4)$$

2.3. Simulation packages:

The simulation of a robot in Gazebo requires the creation of at least one package to contain the robot's model description. Thus, two packages were created in this study: one that includes the robot model, and another one that contains the files to control the robot. The description of the robot is done using URDF, an XML format described for ROS, and compatible with Gazebo. The description package is essentially composed of the URDF files that contain the descriptions of the links, the type of joints between the links and the plugins, the Collada meshes of the links for the visual, and the launch files that describe the nodes to run and parameters to set. The control package holds, in particular, the configuration files (parameters for the plugins), some scripts to actuate the robot, and some launch files.

3. Physics and motion parameters of the robot:

In the robot's description files, a set of plugins were incorporated to describe the physics of the robot, its interactions with the simulation world as well as to provide an interface for controlling the robot's motion.

3.1. Buoyancy:

When a solid body is partially or fully immersed in water or even floating, a buoyancy force, proportional to his volume and mass is applied to it. Gazebo enables to simulate the buoyancy of a solid using the BuoyancyPlugin. The volume and center of mass are provided in the robot description file. As a result, a buoyancy force (eq. 5), opposite to gravity is generated at the center of mass of the robot.

$$F = \rho g v \quad (5)$$

With ρ: density of the liquid, v: volume of the robot's boundary box, g: gravity

3.2. Hydrodynamics:

The hydrodynamics model of the underwater robot is based on Fossen's equations of hydrodynamics. Those equations enable simulating the influence of marine forces and disturbances on an underwater vehicle. For a linear system of 6 DOF, the equation of motion is given by the equation 6.

$$(M_{RB} + M_A)\,\dot{\nu} + (C_{RB}(\nu) + C_A(\nu))\nu + D(\nu)\nu + g_0 + g(\eta) = \tau \quad (6)$$

With M_{RB} , M_A : respectively the rigid-body mass and add mass matrix, D: the damping forces matrix, C_{RB} , C_A: respectively the rigid-body Coriolis-centripetal matrix, and linear Coriolis-centripetal forces due to the movement of water. $\nu, \dot{\nu}$: the velocity vector and acceleration vectors respectively, g_0 , $g(\eta)$: the restoring forces of respectively gravity and buoyancy, $\eta = (x, y, z, \varphi, \theta, \psi)^T$: the vehicle coordinates in the NED-frame, τ: the propulsive forces (the control forces and torques that result from the vehicle actuators), and environmental forces (wind, courant). Gazebo software already integrates the rigid body equation of motion in a case such the user only has to provide the external forces matrix τ_g. The rigid body equation in Gazebo is expressed as follow (eq.7):

$$M_{RB} + C_{RB}(\nu)\nu + g_0 = \tau_g \quad (7)$$

Therefore, we used as expression of τ_g matrix the following solution (eq. 8):

$$\tau_g = -M_A\,\dot{\nu} - C_A(\nu)\nu - D(\nu)\nu - g_0 - g(\eta) \quad (8)$$

Also, we considered the overall robot as a sphere on the equation of motion to simplify the computation. This consideration is possible because the system's mass and inertia are principally concentrated on the main body (sphere). In the implementation, we used the Gazebo hydrodynamics plugin to compute the value of τ_g.

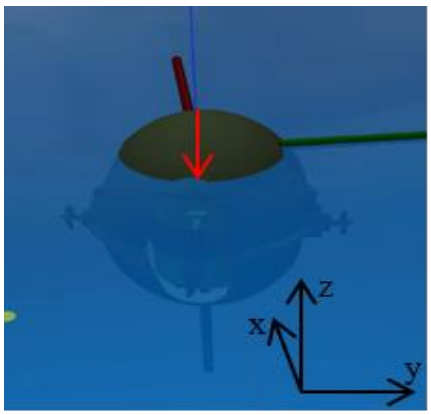
3.3. Motion's control system:

A generic joint controller is already implemented in ROS. So, we used the gazebo_ros_control plugin to provide a control interface for the robot through ROS. The controller takes as input the state of a joint and the desired position of that joint. The output is settled using a close loop feedback mechanism, more specifically a generic PID controller. We stored the PID values in a configuration file so they will be uploaded anytime the robot is launched. The right PID values were obtained through the simulation. We used the rqt-gui framework to access the PID tuner of the controller and tested several values of PID to determine the optimal values.

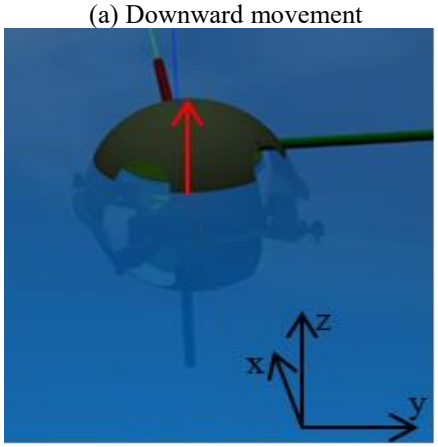
4. Simulation Results:

4.1. Evaluation of hydrodynamics properties

First, a single robot is launched in an underwater world. As expected, it is immersed in water at approximately 70%. When non-actuated, the robot tends to move slowly because of the water dynamics. The effects of the hydrodynamics plugin are visually noticeable. On figure 1 we have two pictures taken at different moments. We can distinctly see the robot's small oscillations along the vertical axis and rotation in the horizontal plane.

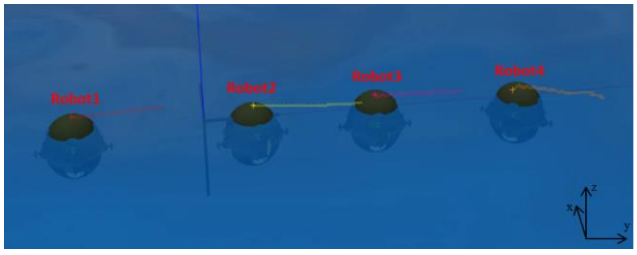


(a) Downward movement

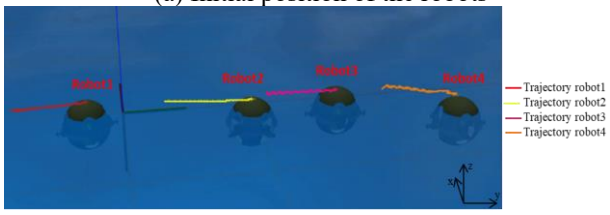


(b) Upward movement

Figure 1 Effect of hydrodynamics properties on the SUR



(a) Initial position of the robots



(b) Final position of the robots

Figure 2 Motion Simulation of multiples robots

4.2. Evaluation of the motion characteristics

Four robots were launched at the final stage of the simulation, each one with a different namespace. The namespace is used to identify in a unique way each robot. We wrote a python script to move all the robots in the same direction, at the same velocity. In figure 2, we have the pose of the robots at the initial stage and at the final one, after 25 seconds of motion. The simulation shows that the robots can be effectively maneuvered using a script.

5. Conclusion

This study investigates the simulation implementation and settings of the spherical underwater robot. We designed the simulation model and parameters in Gazebo using ROS resources. The physical properties of the simulation were based on the equations of hydrodynamics of linear 6 DOF systems. The results of the simulation show the influence of those properties on the robot motion. In our future work, we plan to extend this research by developing a collaboration architecture to coordinate the motions of the robots.

References

[1] Jian Guo, Chunying Li, Shuxiang Guo, “Path Optimization Method for the Spherical Underwater Robot in Unknown Environment”, Journal of Bionic Engineering, Vol.17, No.5, pp.944-958, DOI: 10.1007/s42235-020-0079-3, 2020.

[2] Guo, Chunying Li, Shuxiang Guo, “A Novel Step Optimal Path Planning Algorithm for the Spherical Mobile Robot Based on Fuzzy Control”, IEEE Access, Vol.8, pp.1394-1405, DOI: 10.1109/ACCESS.2019.2962074, 201

[3] Liang Zheng, Shuxiang Guo, Yan Piao, Shuoxin Gu, Ruochen An, “Collaboration and Task planning of Turtle-Inspired Multiple Amphibious Spherical Robots”, Micromachines, Vol.11, No.1, DOI:10.3390/mi11010071, 2020.

[4] Xihuan Hou, Shuxiang Guo, Liwei Shi, Huiming Xing, Yu Liu, Huikang Liu,Yao Hu, Debin Xia and Zan Li, “Hydrodynamic Analysis-Based Modeling and Experimental Verification of a NewWater-Jet Thrusterfor an Amphibious Spherical Robot”, Sensors, Vol.19, No.2, DOI: 10.3390/s19020259, 2019.

[5] Shuoxin Gu, Shuxiang Guo, Liang Zheng, “A highly stable and efficient spherical underwater robot with hybrid propulsion devices”, Autonomous Robots, Vol.44, No.5, pp.759-771, DOI: 10.1007/s10514-019-09895-8, 2020.