UUV Simulator: A Gazebo-based Package for Underwater Intervention and Multi-Robot Simulation

Musa Morena Marcusso Manhães,
Sebastian A. Scherer, Martin Voss
and Luiz Ricardo Douat
Corporate Sector for Research and Advance Engineering
Robert Bosch GmbH
Renningen, Germany
Email: musa.marcusso@de.bosch.com

Thomas Rauschenbach
Advanced System Technology (AST)
Fraunhofer-Institute of Optronics, System Technologies
and Image Exploitation
Ilmenau, Germany

***Abstract*—This paper describes the *Unmanned Underwater Vehicle (UUV) Simulator*, an extension of the open-source robotics simulator Gazebo to underwater scenarios, that can simulate multiple underwater robots and intervention tasks using robotic manipulators. This is achieved mainly through a set of newly implemented plugins that model underwater hydrostatic and hydrodynamic effects, thrusters, sensors, and external disturbances. In contrast to existing solutions, it reuses and extends a general-purpose robotics simulation platform to underwater environments.**

I. Introduction

The development of applications and algorithms for unmanned underwater vehicles (UUVs), such as autonomous underwater vehicles (AUVs) and remotely operated vehicles (ROVs), requires the availability of a suitable simulation platform for rapid prototyping and simulation in reproducible virtual environments. This is desirable in all robotics fields, but is especially relevant for underwater applications. Underwater vehicles usually rely on expensive hardware and their target domain can be difficult to access depending on the application. Whereas several good open-source simulation environments exist for aerial as well for ground based robots, the scenario for underwater robots is less appealing. This is mostly due to the difficulty to realistically model not only hydrodynamic forces acting on the robot itself, but also the complex environments in which the operations take place.

The EU-funded project SWARMs (Smart and Networking Underwater Robots in Cooperation Meshes) has as its main goal the development of a standard framework that allows to coordinate the cooperative behavior of underwater vehicles for monitoring, inspection and intervention missions [1]. In pre-defined use cases, heterogeneous groups of underwater vehicles will be applied for tasks such as berm building, plume tracking and inspection and maintenance of offshore wind turbines [1]. The need for a simulation environment that allows development of new missions strategies and high-level algorithms for cooperative behavior was identified as one of the challenges to be overcome in order to develop this new system. During the requirements analysis for a simulation platform, the following basic requirements were defined:

- physical fidelity for the simulation of rigid-body dynamics and collisions,
- interface for Robot Operating System (ROS) applications (the standard middleware to be used by the vehicles and systems involved in this project),
- low complexity for the setup of new world scenarios and robot models,
- extendibility for integration of additional modules,
- adequate documentation,
- regularity of updates and maintenance,
- capability of multi-robot simulation,
- open-source application with permissive license (e.g. MIT, Berkeley Software Distribution BSD, Apache).

The focus in the search for a robotics simulation tool was set for an open-source implementation with a permissive license to facilitate the dissemination of the algorithms produced within the scope of this project and the easier cooperation between partners. The first option analyzed was the ROS-based simulator developed for in the scope of the TRIDENT project, *UWSim* [2]. It offers a wide range of sensor models, provides realistic renderings of underwater environments due its graphics engine *OpenSceneGraph*[1] and has been used in several academic publications, e.g. [3]. The physics engine is used only for handling contact forces and the implementation of the vehicle dynamics, including the simulation of thruster forces, is located in one monolithic ROS node, but it could be modified to adhere to a more modular structure or interfaced with an external platform such as Matlab, if necessary. Setting up a new simulation, however, requires the configuration of the scenario, vehicles, and other objects in a single XML description file, which can make this task laborious.

[1] http://www.openscenegraph.org/

978-1-5090-1537-5/16/$31.00 ©2016 IEEE

In order to exploit existing physics engines in a more efficient way, *freefloating-gazebo* was presented as an example of a bridge between Gazebo [4] and UWSim for the simulation of underwater vehicles [5]. Gazebo is a general-purpose open-source robotics simulation platform maintained and developed by the Open-Source Robotics Foundation (OSRF). It provides interfaces to four different physics engines for the simulation of rigid-body dynamics and a graphical user interface to visualize the scenario. The *freefloating-gazebo* package includes plugins for Gazebo to allow the generation of hydrodynamic and hydrostatic forces and apply these to the underwater vehicle links while transmitting the pose of simulated objects to UWSim via ROS to exploit the underwater effects of the graphics engine. This application, however, does not include the computation of added-mass forces because of stability concerns [5]. It also allows the modular construction of a scenario on the Gazebo implementation (also in run-time), but it still requires the monolithic XML description for the complete simulation so that the UWSim visualization can be initialized.

A similar implementation involves the extension of Gazebo to the Robot Construction Kit (ROCK)[2], known as *Rock-Gazebo* [6]. In this case, the ROCK visualization tool was extended with *OpenSceneGraph* for rendering the underwater environment while the physics simulation is run in Gazebo. At this moment, it does not yet support the simulation of multiple underwater vehicles.

Since none of the systems surveyed fully fulfill the requirements mentioned above and the effort to adapt them to our purposes would be similar to developing our own solution, it was decided to develop a new underwater robotics environment, described in the further sections in more detail, with focus on flexible and modular setup of new scenarios and vehicles. As it is also not desired to further deteriorate the already fragmented underwater simulation landscape, the extension of the already popular Gazebo simulator was evaluated as the best option. It provides a robust framework for simulation and visualization of robotics mechanisms, can be extended for new dynamics, sensors and world models through modular plugins and has a regularly updated and well-defined roadmap for new releases. Robot and world models can be added to a running simulation dynamically either programmatically or through the graphical user interface. Additionally, contributions from the Gazebo user community allow it to be regularly improved with new features. Even though Gazebo itself provides very few underwater-specific modules, the development of this package for Gazebo can count on a platform that is continuously under improvement. Additionally, the integration with ROS, also developed by OSRF, is already guaranteed through Gazebo/ROS packages.

The paper is structured as follows: Section II describes the package and its internal organization, Section III includes the description of the Gazebo simulation modules, Section IV shows an overview of applications used to interact with the simulation environment via ROS, Section V shows two example scenarios built using the UUV Simulator package, and in Section VI the modules planned for future releases and conclusion are discussed.

[2]http://rock-robotics.org/stable/

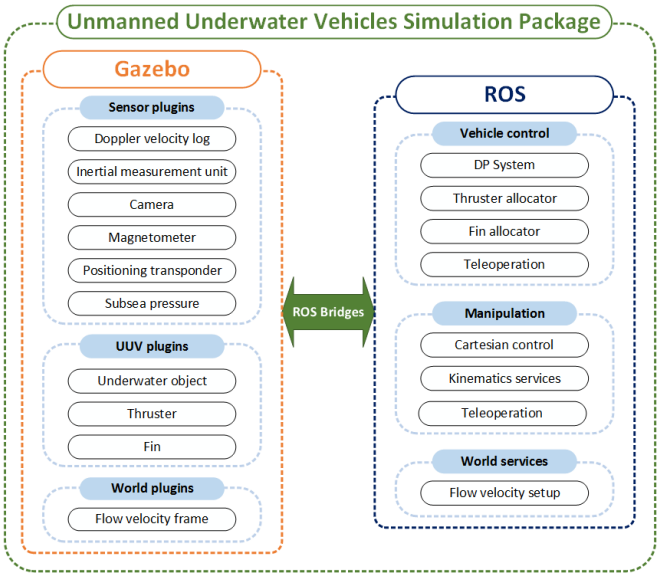


Fig. 1. UUV Simulator - Software structure

II. Software Description

One of the requirements for this simulation environment in the scope of the SWARMs project is to make it compatible with ROS so that the applications developed for the vehicles can be tested seamlessly with it. However, in order to develop a simulator that can be used in the future with other middlewares since it is to be released as an open-source application to the marine robotics community, the modules in the underwater simulation package can be divided in two groups. The models for vehicles, sensors and environmental loads are implemented as plugins extending the functionalities of Gazebo through its API. These modules are unaware of ROS and can therefore be extended to allow using different middlewares. Applications and services that command and communicate with entities in the simulation were developed with ROS and can be connected to the simulation environment by using its ROS bridges, i.e. inherited Gazebo plugins that publish and receive simulation data through ROS topics. An overview of the modules included in this package within the scope in that each one was developed is depicted in Fig. 1.

The simulation currently includes typical underwater sensor models (e.g. Doppler velocity log and subsea pressure sensor), actuator dynamic models (e.g. thruster and fins), hydrostatic and hydrodynamic forces and a constant flow velocity added to the world simulation. Additionally, the vehicles can be initialized with a robotic manipulator arm for the simulation of underwater intervention tasks. Further details about this package are described in the next sections.

Vehicle system functions are implemented using ROS and currently provide simple dynamic positioning controllers,

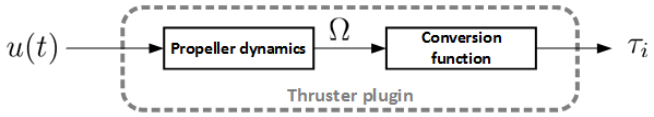


Fig. 2. Thruster plugin model

thruster allocation services, teleoperation modules (e.g. a joystick interface) and, in the case of robotic arms, Cartesian control and kinematics solver services. This set of modules provides a basic structure to interact with the simulated vehicles and the Gazebo simulation environment.

III. Gazebo simulation package

The Gazebo simulator gives the user the possibility to extend it with user-defined plugins. By using its API, the plugin has access to the simulation objects and data, can transmit information via topics by using Protocol Buffer[3] messages and apply torques and forces to objects in the scenario. The plugins must be initialized with a robot, sensor or world models as described in their respective SDF[4] file, an XML format designed for Gazebo. For applications using ROS, the robot descriptions are specified in URDF[5], a similar file format.

The vehicle and sensor robot description must also contain a geometry file. Visual geometries used by the rendering engine are provided in COLLADA format and the collision geometries in STL format. This separation is important to allow simplified collision geometries to be used since very complex meshes might slow down collision computations considerably.

Gazebo is already capable of rigid-body dynamics simulation due to its integration with four physics engines, one of which must be initialized and configured within the world model description. In underwater environments, hydrodynamic and hydrostatic forces and moments must also be taken into account, along with the appropriate actuator and sensor models. This is achieved using additional plugins implemented as part of the *UUV Simulator* package, which can divided into the following categories:

- actuator models (e.g. fins and thrusters)
- underwater sensors
- hydrodynamic and hydrostatic forces and moments
- underwater worlds and environmental loads.

A. Thruster and fin plugins

Thruster dynamics has a highly nonlinear behavior in real systems, which is often a source of poor performance of positioning control of underwater vehicles, especially at low speeds [7]. Several models for the thruster dynamics are available in the literature (e.g. [7], [8]) but none can be considered state-of-the-art. The model usually describes the dynamic behavior of the propeller with the angular velocity as the state variable, which is converted to the thrust force through an empirical steady-state relation between a function of angular velocity and output thrust force.

[3]https://developers.google.com/protocol-buffers/
[4]http://sdformat.org/
[5]http://wiki.ros.org/urdf

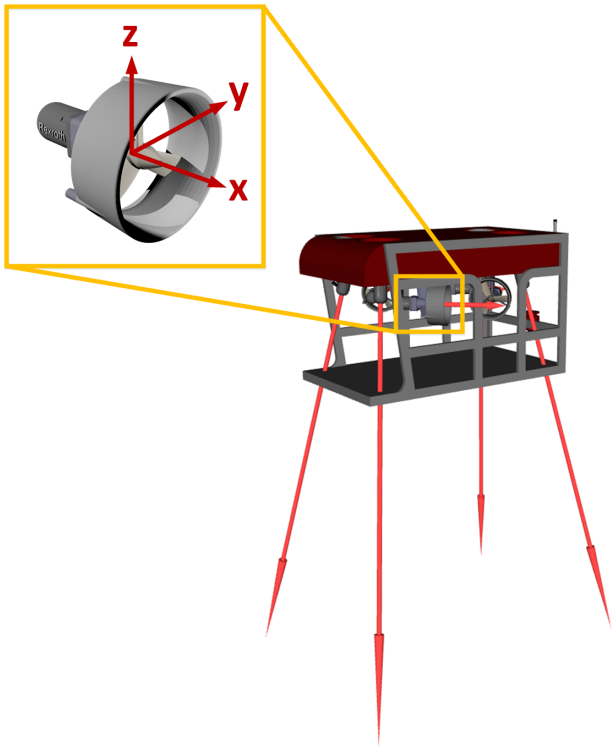


Fig. 3. Visualization of the individual thrust forces (red arrows) during simulation of station keeping of an ROV with the detailed composition of an individual thruster frame

To cover the multitude of thruster models to be used in the simulated vehicles, the dynamics model and the steady-state curve are implemented as two generic blocks (see Fig. 2). The dynamics models currently available include zero-order model, first-order model, Yoerger's model for propeller dynamic [7] and Bessa's extended model published in [8]. Steady-state curves for thrust force conversion can be modeled as a constant gain [7], constant gain with a dead-zone nonlinearity [8], or via linear interpolation between input and output pairs. The last option allows the user directly use the manufacturer's data from Bollard pull tests often included in the thruster's datasheet.

One instance of the thruster plugin is created in the scope of each thruster link on the vehicle with the force generated on the direction of the x-axis as seen in Fig. 3. Each force τ_i, i being the thruster index, will be independently applied to the thruster frame.

This separation of vehicle and actuators allows the thruster allocation matrix to be computed during the simulation without additional input parameters from the user, allowing also the setup of different thruster configurations for the same vehicle model with little effort. Fig. 3 shows the individual thrust forces generated by this plugin installed on a simulated ve-

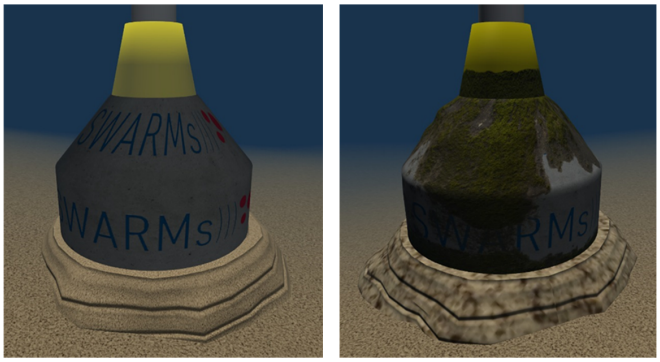


Fig. 4. View of a wind turbine foundation in mint condition (left) and one with biofouling (right) (models provided by ACCIONA [11])

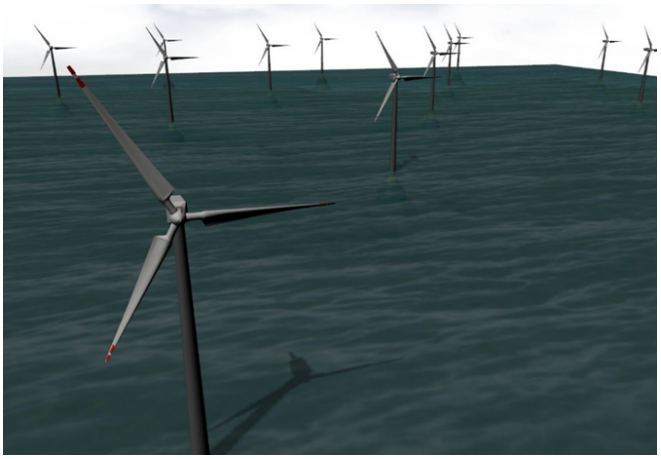


Fig. 5. A wind park scenario simulated in Gazebo (wind turbine models provided by ACCIONA [11]))

hicle.

A fin plugin has a similar modular structure to the thrusters. The dynamics block has the angle of attack of the fin as a state variable, which is then used by a model to compute lift and drag forces acting on the fin. Two models for lift and drag are currently implemented: the quadratic model described in [9] and the default Gazebo lift and drag model plugin[6].

B. Underwater scenarios and environmental loads

The dynamics model for underwater vehicles (see Section III-D) takes into account the relative velocity $\boldsymbol{\nu}_r$ of the vehicle with respect to its surrounding fluid. As explained in [10], this dynamics model is suited for underwater vehicles operating outside of the wave zone, which leads to the assumption that the hydrodynamic coefficients remain constant.

The relative velocity used further on equations of motion of the vehicle is $\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c^B$, being $\boldsymbol{\nu}_c^B$ the current velocity in the vehicle's body frame, which is transformed from $\boldsymbol{\nu}_c^I = (u_c, v_c, w_c, 0, 0, 0)^T$ in the North-East-Down (NED) frame. The constant flow velocity can be assigned to a frame with user-defined orientation and its plugin will broadcast the flow velocity vector.

In addition to physical aspects, setting up a visually realistic underwater scenario is also an important task in this implementation. It affects tasks involving, for example, visual servoing and realistically reproducing the visual feedback received by a ROV operator. Examples of underwater environments with wind turbine foundations can be seen in Fig. 4.

As stated in Section I, Gazebo does not easily allow the construction of underwater environments per default. UUV Simulator includes a world model named *ocean_box* that consists of a water surface plane with texturized waves (see Fig. 5), a sand seabed and light attenuation provided by Gazebo's fog module, which can be configured for color and density. These features will be expanded in the future to improve visual effects.

C. Underwater sensors

The deployment of underwater vehicles requires a slightly different set of sensors than the modules provided by Gazebo, which are more suitable for robots traveling on the ground or through air. A number of sensor modules were created or adapted from existing implementations for this work, their parameters being configurable by the user. Some of these sensor plugins are described below.

Although an inertial measurement unit (IMU) is already available in Gazebo, it neglects important aspects as the drifting bias. For that reason, the IMU used is the model implemented for rotors simulator[7] described in [12]. Its model includes zero-mean Gaussian noise and sensor biases that follow a random walk with a zero-mean white Gaussian noise as their rate.

A magnetometer is also available to measure the heading of the vehicle. This task is performed by the compass plugin, a module based on the magnetometer provided in the *Hector Quadrotor* simulation package[8] [13].

Furthermore, the Doppler velocity log (DVL)'s four sonar sensors are modeled as Gazebo's already existing ray sensor to provide a visualization to the user. The output measured linear velocity v_m is given by the sum of the true linear velocity v of the sensor frame and a zero-mean Gaussian noise n_v.

Multi-beam echo sounders are also essential to simulate the generation of bathymetric maps or obstacle avoidance. In the easiest case they can be simulated similarly to 2D laser range finders already available in Gazebo. Fig. 6 shows the resulting topographic measurements generated by range sensor configured with the specifications of a commercial 2D multi-beam echo sounder.

Finally, one of the drawbacks of Gazebo is the lack of tools to realistically represent underwater environments. A special camera module has been designed to further improve the light attenuation with an image post-processing step. This process imposes an exponential attenuation [14] of the pixel intensity on the image acquired by the camera as follows:

[6]http://gazebosim.org/tutorials?tut=aerodynamics&cat=physics

[7]https://github.com/ethz-asl/rotors_simulator/wiki

[8]https://github.com/tu-darmstadt-ros-pkg/hector_quadrotor

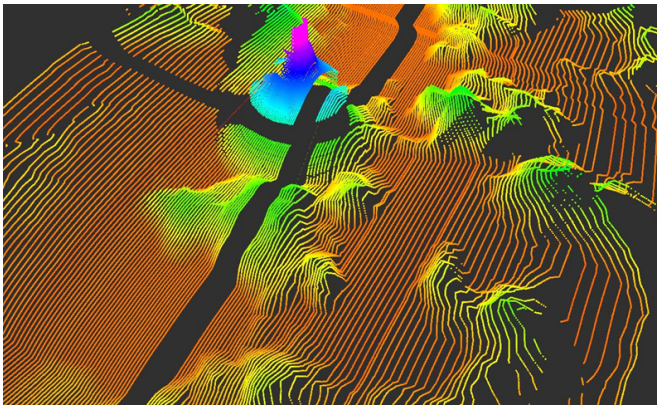


Fig. 6. Bathymetric map generated with simulated multibeam echo sounders installed on an AUV (color map represents the height value) measuring the seabed of the wind park scenario.

$$i_{c,m} = i_c e^{-r a_c} + \left(1 - e^{-r a_c}\right) b_c, \forall c \in \{R, G, B\}, \quad (1)$$

where i_c is the original intensity value for the colour channel c, $i_{c,m}$ is the measured intensity value, a_c is a colour-dependent attenuation factor, b_c is the background intensity of the channel c and r is the distance to the object projected on the pixel computed from its visual depth d and the direction of the ray for the corresponding pixel. A comparison between the original and processed image is shown in Fig. 7.

D. Hydrodynamic and hydrostatic forces

For pure rigid-body dynamics simulation, Gazebo has to be provided with the robot description (e.g. mass, inertial tensor, collision geometry, joint parameters). Due to their roots in the robotics community, Gazebo and ROS use the more prevalent ENU (East-North-Up) convention, whereas NED (North-East-Down) is widely-used for marine vessels.

In order to support both conventions, vehicles are created with a *base_link* frame at the center of gravity following the ENU conventions as is common in Gazebo and ROS. A second NED link is added at the same origin with a fixed transformation. This allows to easily transform data to either reference frame.

Computation of the interaction of a fluid with a submerged body is still not feasible for an off-the-shelf computer for complex geometries like underwater vehicles in real-time. The best known model that is suitable for underwater vehicle simulation is Fossen's equations of motion [10] in the SNAME notation [15] shown in Equation 2,

$$\begin{aligned}(\mathbf{M}_{RB} + \mathbf{M}_A)\dot{\boldsymbol{\nu}}_r + (\mathbf{C}_{RB}(\boldsymbol{\nu}_r) + \mathbf{C}_A(\boldsymbol{\nu}_r))\boldsymbol{\nu}_r + \\ \mathbf{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \mathbf{g}_0 + \mathbf{g}(\boldsymbol{\eta}) = \boldsymbol{\tau}.\end{aligned} \quad (2)$$

Here $\mathbf{M}_A$ is the added-mass matrix and $\mathbf{C}_A(\boldsymbol{\nu}_r)$ its Coriolis and centripetal matrix, $\mathbf{D}(\boldsymbol{\nu}_r)$ is the damping matrix, $\mathbf{g}_0$ and $\mathbf{g}(\boldsymbol{\eta})$ are the restoring forces of gravity and buoyancy, respectively, and $\boldsymbol{\tau}$ are the control forces and torques. The vehicle pose in the inertial NED-frame is $\boldsymbol{\eta} = (x, y, z, \phi, \theta, \psi)^T$.

Fig. 7. Comparison between the original image acquired during the simulation with Gazebo's from a wind turbine foundation with the default camera plugin (left) and the result with exponential attenuation with the addition of fog effects (right).

In contrast, Gazebo integrates the default rigid body equations of motion with six degrees of freedom:

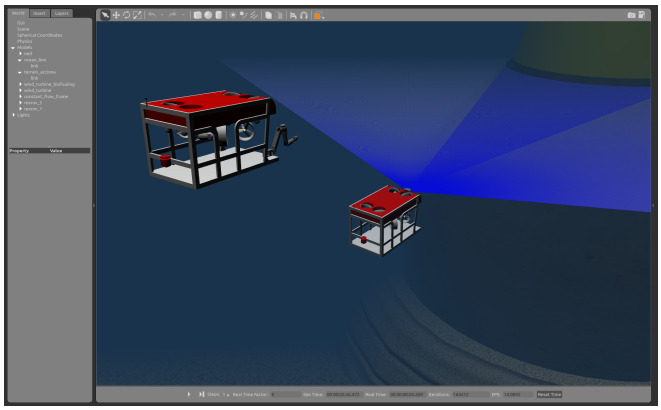
$$\mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{g}_0 = \boldsymbol{\tau}_g \quad (3)$$

where $\boldsymbol{\tau}_g$ are external forces and torques that can be computed for example within plugins. Since Gazebo itself relies on one out of four supported open-source physics engines to integrate the equations of motion above, it is difficult to change it to adhere to Eq. 2. If we instead move all necessary terms of Eq. 2 to the right-hand side and compare it with Eq. 3, it follows that we can simulate underwater effects by setting the external forces and torques to:

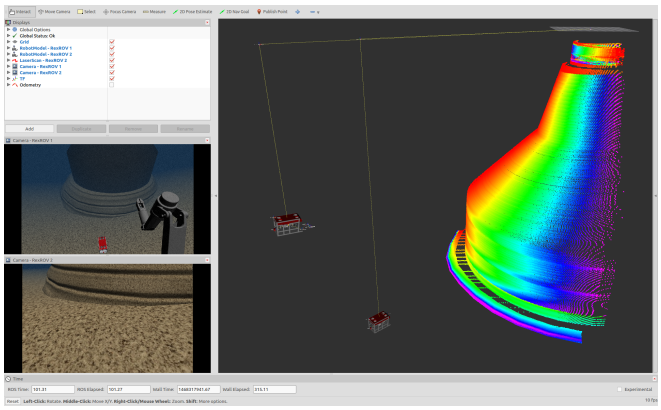
$$\boldsymbol{\tau}_g = -\mathbf{M}_A\dot{\boldsymbol{\nu}}_r - \mathbf{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r - \mathbf{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r - \mathbf{g}(\boldsymbol{\eta}). \quad (4)$$

In order to add hydrodynamic and hydrostatic forces, $\boldsymbol{\tau}_g$ is applied at each iteration of the simulation to each submerged link. Note that $\boldsymbol{\tau}_g$ does not include forces and torques from thrusters and fins, since they are applied in independently as described in Section III-A.

As also mentioned in [5], the inclusion of added-mass as described above can lead to simulation instability depending on the relation between mass m and added-mass matrix coefficients in $\mathbf{M_A}$. This is due to the fact that the acceleration is needed in Eq. 4 as input to compute hydrodynamic forces and torques, which are then used by the physics engine to solve the equations of motion to compute the current acceleration, velocity, and pose. Only the previous accelerations can, of course, be used when computing the latest acceleration. But using its previous value to compute the current one can cause the system to become unstable if the added-mass coefficient is larger than the actual body mass, since an acceleration at the previous time step can lead to a slightly bigger reaction due to the added-mass at the current time step and an increasingly bigger value at the subsequent time step. The current solution to avoid unstable simulations is the use of a low-pass filtered version of previous accelerations to compute hydrodynamic forces and torques according to Eq. 4. On further releases, this issue will be studied in more detail to improve the simulation results.



(a) Visualization of the simulation environment in Gazebo in a offshore wind park scenario.



(b) Visualization in RViz, camera output for each vehicle on the left panel, point cloud output from the simulated multibeam sonar depicted with a colormap on the main panel on the right.

Fig. 8. Simulation of inspection of a wind turbine foundation using two ROVs.

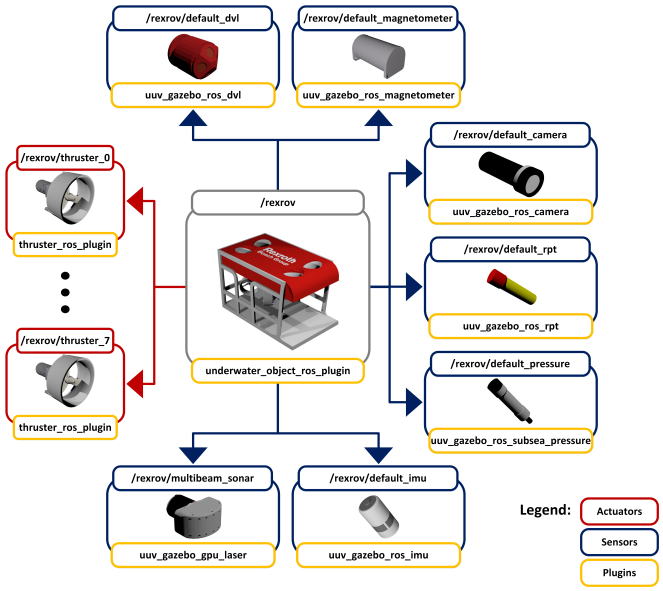


Fig. 9. RexROV test vehicle with its constituting modules

IV. ROS MODULES

The ROS-specific modules are subdivided into teleoperation, manipulator control, vehicle control and Gazebo-ROS plugins. These bridged plugins are available to all Gazebo plugins presented and consist on an inheritance to a Gazebo module that will add features that will make it visible to a ROS client node or software. Further modules are described in the following sections.

A. Underwater manipulators

Underwater manipulators can currently only be operated using ROS modules. They are available as units that can be added to the vehicle's body frame using pre-defined macros. This is done to allow the generation of several configurations for the same vehicle by only adding an extra unit to the robot description.

The manipulators are initialized with ROS controllers for each joint and can be operated by using a joystick to command the reference velocity for the end-effector. A Jacobian-transpose Cartesian controller allows control of the end-effector's pose. Each manipulator and its respective ROS nodes (e.g. controllers, kinematics solvers services) are initialized within a unique ROS namespace, allowing multiple manipulator units (on one or multiple vehicles) to be initialized simultaneously.

B. Vehicle system services and control systems

Even though the first phase of development of this software was focused on simulation, some service and control modules are already available to allow a basic operation of the vehicle models. These modules communicate with the extended versions of the plugins described in Section III to ROS.

A thruster allocator can be initialized to read the relative pose and orientation of a specific vehicle and generate the thruster allocation matrix at run-time. It receives forces and torques to be applied to the vehicle and transmits the command to each thruster unit described in Section III-A.

V. EXAMPLE USE-CASES

As an example application, a vehicle has been integrated in the simulation environment based on the parameters for the Sperre SF 30k ROV published in [16]. The thruster configuration has been altered to include eight thrusters and integrated in a model description named *RexROV*. Two instances of the ROV have been initialized, one with a simplified model of a Schilling Robotics Orion 7P robotic manipulator. The diagram seen in Fig. 9 shows the independent modules initialized with the vehicle are labeled according to the unit's link name and their respective operating plugins.

Each vehicle's velocity reference is given by the input of an joystick and cascaded PID controllers control the vehicle's

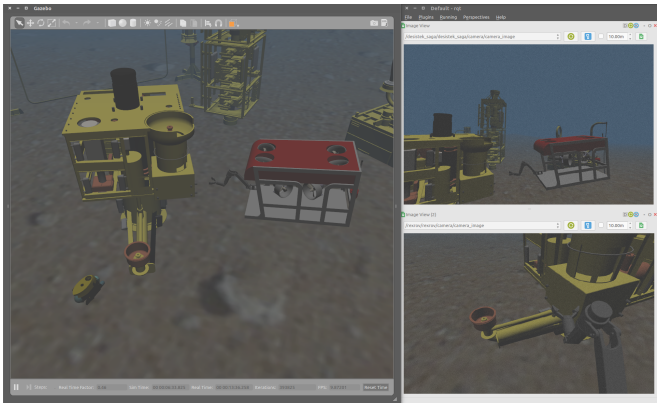


Fig. 10. RexROV and Desistek SAGA mini-ROV in a subsea factory scenario. The window on the left side shows the Gazebo environment and on the right the output of the vehicles' cameras (Desistek SAGA on the top, RexROV on the bottom). Desistek SAGA 3D model provided by Desistek[10].

velocity. The manipulator arm is provided with individual joint effort controllers, a kinematics service and a Cartesian controller. Fig. 8a shows the dynamic simulation in the Gazebo environment in the wind turbine scenario. One ROV equiped with a forward multibeam sonar measures the structure, while the other vehicle is in place to execute any maintenance task necessary. In Fig. 8b, the output of the simulated sonar can be seen in the form of a point cloud with a color map along with the current frames from the cameras installed on both vehicles. As can be seen, both vehicles are setup with different sensor and actuator configurations, which can be later chosen through user-defined arguments by the initialization of the vehicle instance. The instances of world and UUVs are independently uploaded in Gazebo in run-time, making the construction of a new scenario very straightforward.

In another example of simulation of multiple robots shown in Fig. 10, a RexROV and a Desistek SAGA mini-ROV are uploaded in a subsea factory scenario. The mini-ROV is used to monitor the work-class ROV while the manipulator is being deployed.

VI. Conclusion and Future Work

In this paper a package for simulation of unmanned underwater vehicles was presented. It extends the general-purpose robotics simulator Gazebo for application in underwater environments and provides an interface for operation of the simulated vehicles through ROS. Vehicle, actuator and sensor models are implemented with the Gazebo API and can be extended to other robotics middlewares, if desired. This package allows the setup of new simulation scenarios in run-time by allowing world and robot models to be initialized dynamically, diminishing the effort to test, build and develop new applications and mission strategies. Methods to validate the dynamic simulation implemented are currently being studied. This application is open-source and is available at *https://github.com/uuvsimulator/uuv_simulator*.

[10]http://www.desistek.com.tr/

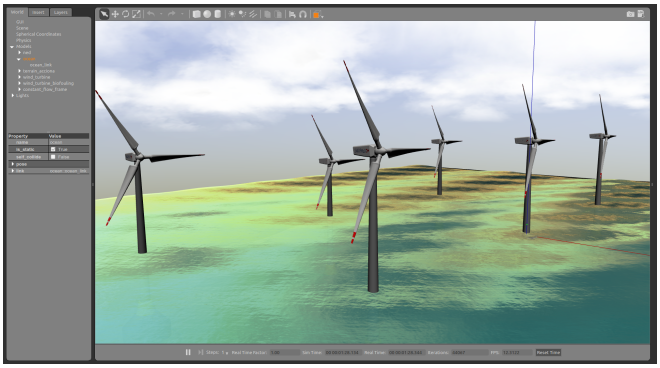


Fig. 11. Rendering waves in an offshore wind park scenario.

Regarding future work within this project, one of the most noticeable features missing in this underwater robotics simulator for the Gazebo environment is the lack of characteristic visual effects as floating particles and proper light damping as a function of water depth, along with the generation of waves on the sea surface. The implementation of such effects with the support of Gazebo's graphics engine, OGRE, are currently under development since it plays an important role in the simulation of visual servoing applications and on the control teleoperated vehicles with visual feedback. A sample of the current stage of integration of waves in Gazebo can be seen in Fig. 11.

The implementation of an umbilical plugin is also a desired feature to improve the simulation of ROVs. Although the umbilical forces can be simulated, the simulation of contact forces on the tether will require the geometrical representation of cables in Gazebo, an implementation still not existent in this platform.

Furthermore, as seen in the examples, the vehicles are used for the simulation of missions on offshore wind parks, often in the wave zone, which requires the simulation of environmental loads originated from waves and turbulence around the wind turbines, another desired feature for future releases.

Finally, configuration tools to allow users not familiar with Gazebo/ROS development to use this tool are also included in the roadmap. They will allow creation of new vehicles and scenarios through a graphical user interface, preventing errors originated when editing the configuration files manually.

Acknowledgment

This software has been developed as part of the work undertaken for the EU-funded research project SWARMs (Smart and Networking Underwater Robots in Cooperation Meshes), ECSEL project number 662107. Thanks to DFKI Robotics Innovation Center, Bremen, for providing the 3D meshes of the Schilling Robotics Orion 7P Manipulator.

References

[1] (2015) SWARMs project website. [Online]. Available: http://www.swarms.eu/

[2] M. Prats, J. Pérez, J. J. Fernandéz, and P. J. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Institute of Electrical & Electronics Engineers (IEEE), Oct 2012.

[3] P. Kormushev and D. G. Caldwell, "Towards improved AUV control through learning of periodic signals," in *2013 OCEANS - San Diego*, Sept 2013, pp. 1–4.

[4] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Institute of Electrical & Electronics Engineers (IEEE), 2004.

[5] O. Kermorgant, "A dynamic simulator for underwater vehicle-manipulators," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer Science + Business Media, 2014, pp. 25–36.

[6] T. Watanabe, G. Neves, R. Cerqueira, T. Trocoli, M. Reis, S. Joyeux, and J. C. Albiez, "The Rock-Gazebo integration and real-time AUV simulation," in *12th Latin American Robotics Symposium and 2015 Third Brazilian Symposium on Robotics*, 2015.

[7] D. R. Yoerger, J. G. Cooke, and J.-J. E. Slotine, "The influence of thruster dynamics on underwater vehicle behavior and their incorporation into control system design," *IEEE Journal of Oceanic Engineering*, vol. 15, no. 3, pp. 167–178, jul 1990.

[8] W. M. Bessa, M. S. Dutra, and E. Kreuzer, "Thruster dynamics compensation for the positioning of underwater robotic vehicles through a fuzzy sliding mode based approach," in *ABCM Symposium Series in Mechatronics*, vol. 2, 2006, pp. 605–612.

[9] Ø. Engelhardtsen, "3D AUV Collision Avoidance," Master's Thesis, Norwegian University of Science and Technology, Department of Engineering Cybernetics, 2007.

[10] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 2011.

[11] (2016, May) ACCIONA homepage. [Online]. Available: http://www.acciona.com/

[12] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS—a modular gazebo MAV simulator framework," in *Studies in Computational Intelligence*. Springer Science + Business Media, 2016, pp. 595–625.

[13] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and gazebo," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer Science + Business Media, 2012, pp. 400–411.

[14] Y. Y. Schechner and N. Karpel, "Clear underwater vision," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, June 2004, pp. I–536–I–543 Vol.1.

[15] *Nomenclature for Treating the Motion of a Submerged Body Through a Fluid: Report of the American Towing Tank Conference*. Society of Naval Architects and Marine Engineers, 1950.

[16] F. Dukan, "ROV Motion Control Systems," Ph.D. dissertation, Norwegian University of Science and Technology, Department of Marine Technology, 2014.