Development of a Hybrid Simulator for Underwater Vehicles With Manipulators

Matteo Razzanelli, Simona Casini, Mario Innocenti, *Life Member, IEEE*, and Lorenzo Pollini, *Member, IEEE*

***Abstract*—This article describes a hybrid simulation approach meant to facilitate the realization of a simulator for underwater vehicles with one or more manipulators capable of simulating the interaction of the vehicle with objects and structures of the environment. The hybrid simulation approach is first described and motivated analytically, then an analysis of simulation accuracy is proposed, where, in particular, the implications of added mass simulation are discussed. Then, a possible implementation of the proposed architecture is shown, where a robotic simulator of articulated bodies, capable of stable and accurate simulation of contact forces, although unfit to simulate any serious hydrodynamic model, is tightly interfaced with a general purpose dynamic systems simulator that is used to simulate the hydrodynamic forces, the vehicle guidance, navigation, and control system, and also a man–machine interface. Software details and the technicalities needed to interface the two simulators are also briefly presented. Finally, the results of the simulation of three operational scenarios are proposed as qualitative assessment of the simulator capabilities.**

***Index Terms*—Remotely operated vehicle, robotic intervention, simulation, underwater vehicle manipulator systems.**

I. Introduction

RESEARCHERS have been studying unmanned underwater vehicles (UUVs) and their applications for decades. Thanks to a wide range of possible activities that UUVs are able to perform in subsea environment, significant efforts have been invested in this field [1], [2], also motivated by diverse potential applications, such as underwater archaeology, oceanography, and offshore industries [3], [4]. In particular, remotely operated vehicles (ROVs) are considered as an essential tool not only for the oil and gas sector, but also for scientific research and defense, and constitute an important market sector [5]–[7]. Nowadays, ROVs in operational use possess a low level of autonomy and are essentially, as their name suggests, connected via a tether to a support ship where trained operators remotely operate them. Depending on the type of operation, the operational cost of a work-class ROV, which requires about a dozen of people on 24 h, ranges from 100 000 to 300 000 € per day [8]. At the same time, a paradigm shift is taking place since increasing levels of autonomy are sought for ROVs involved in intervention tasks: ROVs are expected to become more and more autonomous in terms of duration of the missions and capability to perform complex tasks without human intervention or with minimal supervision [9]. Thus, today scientific research is facing new challenges connected to the realization of new autonomous functions that will allow the automation of a number of specific tasks, such as navigation among obstacles, precise positioning with respect to underwater structures, detection, recognition and grasping of objects, operations of valves and connectors, and automatic docking and undocking.

Unfortunately, innovation in the control systems of an ROV requires extensive experimentation, which is difficult and extremely expensive to perform, especially when the vehicle to be used is currently in operation. For this reason, simulation represents an opportunity to prototype and pre-evaluate algorithms, control schemes, perception systems (such as vision- or acoustic-based navigation systems), and safety and effectiveness of the system, using a virtual environment, by avoiding the risks and the costs involved in real-world experimentation.

Development of the desired autonomous functions for today's ROVs requires extensive simulation capabilities that go beyond simulation of the single vehicle and requires simulation of the interaction with the complex subsea environment to simulate situations such as operation of underwater valves with manipulator grippers, grasp and lifting of bodies, and collision with structures. Clearly, very accurate and reliable simulation of contact with other bodies of the environment is necessary in all those intervention scenarios where the UUV must, for example, operate a valve, dock to a platform, and move bodies (e.g., other machines), use tools (e.g., a cathodic probe or a drill), or use a skid equipped with some actuators.

Although the dynamic equations for underwater vehicles, and underwater vehicles with manipulators, are well known in the literature, simulation of hydrodynamics and interaction with the environment is much more complex; researchers tackle this problem in several ways, usually with solutions customized to the specific task to be simulated, obtaining different levels of accuracy. The differences between existing simulators are mainly due to how environmental forces, that is hydrodynamic and contact forces, are modeled [10]–[18].

What is proposed in this article is a hybrid simulation architecture for UUVs applications and robotic intervention missions, where the word "hybrid" indicates that the simulator is build by two main and different components, tightly interfaced the one with the other: a robotic simulator of articulated bodies capable of stable and accurate simulation of contact forces

Manuscript received May 16, 2018; revised March 29, 2019; accepted August 12, 2019. *(Corresponding author: Lorenzo Pollini.)*

Associate Editor: B. Buckham

The authors are with the Dipartimento di Ingegneria dell'Informazione, University of Pisa, 56122 Pisa, Italy (e-mail: matteo.razzanelli@ing.unipi.it; simona.casini89@gmail.com; mario.innocenti@unipi.it; lorenzo.pollini@unipi.it).

Digital Object Identifier 10.1109/JOE.2019.2935801

(although unfit to simulate any serious hydrodynamic model), and a general purpose dynamic systems simulator (Mathworks' Simulink in our case) that is used to simulate the hydrodynamic forces and, additionally, any guidance, navigation, and control (GNC) system. Furthermore, the simulator graphical front end can be used to simulate perception processes such as vision-based navigation and visual servoing. This approach blends the advantages of two worlds: accurate and efficient real-time photorealistic simulation of articulated rigid bodies with multiple simultaneous contacts between bodies of arbitrary shapes, with the ease of use and flexibility of MATLAB and Simulink, probably the most widely used tools for simulation and development of GNC systems in both academia and industry. Interfacing MATLAB and Simulink with the articulated rigid body simulator allowed us, in our opinion, the greatest possible flexibility ever to simulate hydrodynamics effects using the same tool that is usually used to process experimental data and to create and validate hydrodynamic models, together with the possibility to include and prototype, in the same environment, the simulation of sensors, actuators, and GNC components.

This article is organized as follows: Section II presents the dynamic equation of an underwater vehicle with one or more manipulators and proposes the mathematical foundations of our proposed hybrid approach. Section III, after a brief overview of articulated rigid body simulators and of the working principles of the underlying physics engines (PEs), shows how the proposed hybrid approach can be implemented using two commercial software (V-Rep and MATLAB); simulation approximations and their impacts on simulation fidelity are also discussed. Finally, Section IV presents a qualitative validation of the proposed hybrid simulator using three different scenarios. Conclusions are presented in Section V.

II. Underwater Vehicle Dynamics

This section presents the nonlinear dynamic equations of motion of an underwater vehicle with one or more manipulators. Equations are derived using the Newton–Euler formulation, then the concept of hybrid simulation is introduced and a proof of the validity of the approach is proposed.

A. Underwater Vehicle Dynamics: Preliminaries

When analyzing the motion of marine vehicles in six degrees of freedom (DOFs), it is convenient to define at least two reference frames: the body frame and the inertial frame. Fig. 1 shows the body-fixed coordinate system with the z-axis directed from top to bottom and the x-axis pointing in the surge direction. The origin of the reference frame may be chosen to coincide or not with the center of gravity.

The motion of the body-fixed frame is described relatively to the inertial reference frame. When simulating slowly moving marine vehicles, it is usually assumed that the accelerations of a point on the surface of the earth can be neglected, thus the inertial frame can be chosen to coincide with a point fixed with respect to the earth surface. This suggests that the position and orientation of the vehicle should be described relatively to the inertial reference frame, while the linear and angular velocities

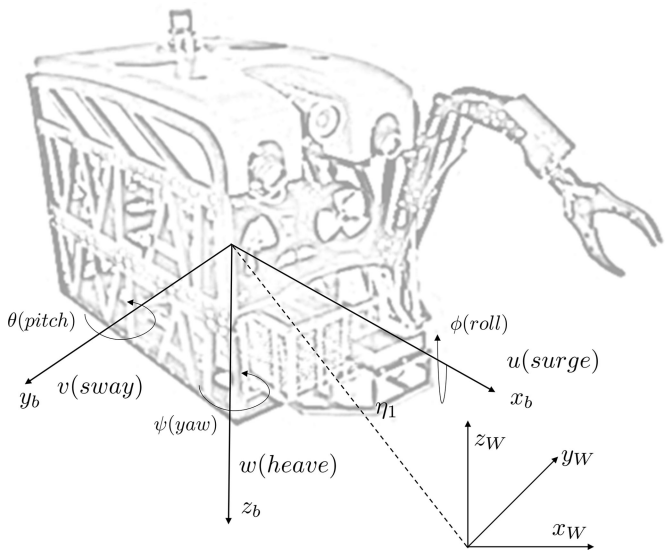

Fig. 1. Body-fixed reference frame.

of the vehicle should be expressed in the body-fixed coordinate system.

Equations of the six-DOFs dynamics for translation and rotational motions can be obtained by means of the Newton–Euler formulation (see also [19]) and are reported here as introduction and basis for the successive discussion.

Let $\boldsymbol{v}_0$ be the vehicle linear velocity vector (expressed in body axes), $\boldsymbol{\omega}$ the angular velocity vector (expressed in body axes), and $\boldsymbol{f}_0$ and $\boldsymbol{m}_0$ the applied forces and torques vectors, respectively (in body axes). Let m be the mass of a vehicle and $\boldsymbol{r}_G$ the position of the center of gravity in the body reference frame, then it can be easily shown [19] that the following holds:

$$m\left(\mathring{\boldsymbol{v}}_0 + \boldsymbol{\omega} \times \boldsymbol{v}_0 + \mathring{\boldsymbol{\omega}} \times \boldsymbol{r}_G + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \boldsymbol{r}_G)\right) = \boldsymbol{f}_0. \quad (1)$$

Equation (1) clearly represents the translation dynamics of a rigid body. By using the same notation, the rotational dynamics can be expressed as follows:

$$\boldsymbol{I}_0 \mathring{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{I}_0 \boldsymbol{\omega}) + m\boldsymbol{r}_G \times (\mathring{\boldsymbol{v}}_0 + \boldsymbol{\omega} \times \boldsymbol{v}_0) = \boldsymbol{m}_0 \quad (2)$$

where $\boldsymbol{I}_0$ is the body's inertia tensor referred to the body-fixed frame, thus usually constant. Note that to derive the equations of motion for an arbitrary origin in a local body-fixed rotating coordinate system, as in (1) and (2), use of the following identity is necessary:

$$\dot{\boldsymbol{c}} = \mathring{\boldsymbol{c}} + \boldsymbol{\omega} \times \boldsymbol{c} \quad (3)$$

which relates the time derivatives of an arbitrary vector $\boldsymbol{c}$ in the inertial ($\dot{\boldsymbol{c}}$ is the time derivative in the earth-fixed frame) and body reference frames ($\mathring{\boldsymbol{c}}$ is the time derivative in the moving reference frame). Note that also the following identity holds:

$$\dot{\boldsymbol{w}} = \mathring{\boldsymbol{w}} + \boldsymbol{\omega} \times \boldsymbol{\omega} = \mathring{\boldsymbol{w}}. \quad (4)$$

By using the SNAME notation [20] and letting

$$\boldsymbol{\nu}_1 \triangleq \boldsymbol{v}_0, \quad \boldsymbol{\nu}_2 \triangleq \boldsymbol{\omega}, \quad \boldsymbol{\tau}_1 \triangleq \boldsymbol{f}_0, \quad \boldsymbol{\tau}_2 \triangleq \boldsymbol{m}_0 \quad (5)$$

then translation and rotational equations can be expressed in a more compact form as follows:

$$M_{\text{RB}}\dot{\boldsymbol{\nu}} + C_{\text{RB}}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{\text{RB}} \tag{6}$$

where M_{RB} and $C_{\text{RB}}(\boldsymbol{\nu})$ are appropriate matrices [19] and the $\boldsymbol{\nu} = [\boldsymbol{\nu}_1, \boldsymbol{\nu}_2]^T$ vector encapsulates the linear ($\boldsymbol{\nu_1}$) and angular ($\boldsymbol{\nu}_2$) velocity vectors of the body-fixed frame with respect to the origin of the inertial frame expressed in the body-fixed frame, and $\boldsymbol{\tau}_{\text{RB}} = [\boldsymbol{\tau}_1, \boldsymbol{\tau}_2]^T$ is the vector representing forces $\boldsymbol{\tau}_1$ and moments $\boldsymbol{\tau}_2$ acting on the vehicle expressed in the body-fixed frame.

Furthermore, it can be shown that, by using suitable Jacobian matrices $J_1(\boldsymbol{\eta}_2)$ and $J_2(\boldsymbol{\eta}_2)$, it is possible [19, pp. 7–12] to express the time derivatives of the position of the vehicle (i.e. the origin of the body frame) with respect to the inertial frame, $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$; the attitudes of the body reference frame with respect to the inertial frames (expressed by Euler angles) $\dot{\boldsymbol{\eta}}_1$ and $\dot{\boldsymbol{\eta}}_2$ as a function of linear and angular velocities $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ are given as follows:

$$\dot{\boldsymbol{\eta}}_1 = J_1(\boldsymbol{\eta}_2)\boldsymbol{\nu}_1 \qquad \dot{\boldsymbol{\eta}}_2 = J_2(\boldsymbol{\eta}_2)\boldsymbol{\nu}_2 \tag{7}$$

with

$$J_1(\boldsymbol{\eta}_2) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}$$

and

$$J_2(\boldsymbol{\eta}_2) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} \tag{8}$$

where $c(\cdot)$, $s(\cdot)$, and $t(\cdot)$ are sine, cosine, and tangent of the Euler angles ϕ (roll), θ (pitch), and ψ (yaw).

Modeling an underwater vehicle requires to add, to (6), a set of terms that model the interaction with the fluid: hydrodynamic potential damping and radiation-induced forces. These latter forces are applied on the vehicle when it is forced to oscillate with the waves or current excitation frequency and there are no incident waves.

Discussing in detail the hydrodynamics effects is outside the scope of this article; nonetheless, it is necessary to introduce them here to show how these can be effectively taken apart and simulated separately from rigid body dynamics.

The radiation-induced forces and moments can be identified as the sum of three components:

1) *added mass* due to the inertia of the surrounding fluid that is accelerated by the vehicle motion when the vehicle itself accelerates;
2) *radiation-induced potential damping* due to the energy carried away by generated surface waves;
3) *restoring forces* due to the presence of weight and buoyancy forces.

Radiation-induced forces and moments $\boldsymbol{\tau}_R \in \mathbb{R}^6$ can be expressed as follows:

$$\boldsymbol{\tau}_R = \underbrace{-M_A\dot{\boldsymbol{\nu}} - C_A(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{Added mass}} \underbrace{-D_P(\boldsymbol{\nu})\boldsymbol{\nu}}_{\substack{\text{Potential}\\ \text{damping}}} \underbrace{-g_v(\boldsymbol{\eta})}_{\substack{\text{Restoring}\\ \text{forces}}}. \tag{9}$$

Hydrodynamic potential damping $\boldsymbol{\tau}_D$, instead, includes other damping effects, such as skin friction, wave drift damping, and damping due to the vortex shedding

$$\boldsymbol{\tau}_D = \underbrace{-D_S(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{Skin friction}} \underbrace{-D_W(\boldsymbol{\nu})\boldsymbol{\nu}}_{\substack{\text{Wave drift}\\ \text{damping}}} \underbrace{-D_M(\boldsymbol{\nu})\boldsymbol{\nu}}_{\substack{\text{Vortex}\\ \text{shedding}}}. \tag{10}$$

Considering the whole contribution of (9) and (10), all hydrodynamic forces and moments $\boldsymbol{\tau}_H$ can be written as follows:

$$\boldsymbol{\tau}_H = \boldsymbol{\tau}_R + \boldsymbol{\tau}_D = -M_A\dot{\boldsymbol{\nu}} - C_A(\boldsymbol{\nu})\boldsymbol{\nu} - D_v(\boldsymbol{\nu})\boldsymbol{\nu} - g_v(\boldsymbol{\eta}) \tag{11}$$

with the total hydrodynamic damping matrix $D(\boldsymbol{\nu})$ defined as

$$D_v(\boldsymbol{\nu}) \triangleq D_P(\boldsymbol{\nu}) + D_S(\boldsymbol{\nu}) + D_W(\boldsymbol{\nu}) + D_M(\boldsymbol{\nu}). \tag{12}$$

The right-hand-side term $\boldsymbol{\tau}_{\text{RB}}$ in (6) represents all the external forces and moments acting on the vehicle, which can be grouped as follows:

1) radiation-induced forces and hydrodynamics $\boldsymbol{\tau}_H$ [described in (9) and (10)];
2) environmental or other external forces (as ocean currents, waves, wind, contact forces) as $\boldsymbol{\tau}_v^{\text{ext}}$;
3) propulsion forces, which are thrusters or propellers and control surfaces or rudder forces, as $\boldsymbol{\tau}_v$.

Thus, letting

$$\boldsymbol{\tau}_{\text{RB}} = \boldsymbol{\tau}_H + \boldsymbol{\tau}_v^{\text{ext}} + \boldsymbol{\tau}_v \tag{13}$$

and since $M_{\text{RB}} > 0$ is a 6×6 inertia matrix defined by four 3×3 submatrices, and the fact that the Coriolis and centripetal terms can be always parameterized such that $C_{\text{RB}}(\boldsymbol{\nu})$ is skew-symmetrical [19], [21], and also thanks to linearity of the skew operator, the following holds:

$$M_v\dot{\boldsymbol{\nu}} + C_v(\boldsymbol{\nu})\boldsymbol{\nu} + D_v(\boldsymbol{\nu})\boldsymbol{\nu} + g_v(\boldsymbol{\eta}) = \boldsymbol{\tau}_v + \boldsymbol{\tau}_v^{\text{ext}} \tag{14}$$

with

$$M_v \triangleq M_{\text{RB}} + M_A; \quad C_v(\boldsymbol{\nu}) \triangleq C_{\text{RB}}(\boldsymbol{\nu}) + C_A(\boldsymbol{\nu}) \tag{15}$$

where M_v represents the inertia matrix (including the added mass) of the vehicle, $C_v(\boldsymbol{\nu})$ is the matrix of Coriolis and centripetal terms, $D_v(\boldsymbol{\nu})\boldsymbol{\nu}$ models all damping effects, vector $g_v(\boldsymbol{\eta})$ represents the restoring forces and moments due to buoyancy and weight, $\boldsymbol{\tau}_v$ are propulsion forces and moments acting on the vehicle, and $\boldsymbol{\tau}_v^{\text{ext}}$ are all other external forces and moments acting on the underwater vehicle. The subscript v is used to indicate that this equation is for the *vehicle only*. Equations (7) and (15) together represent the typical underwater vehicle dynamics model.

B. Underwater Vehicle With One or More Manipulators Dynamics

When the UUV is equipped with an n-joint manipulator, the additional degrees of freedom are usually introduced by using

the vector of n joint angles $\boldsymbol{q} \in \mathbb{R}^n$ and velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ as state variables.

The pose (position and orientation) of the manipulator end effector does not usually become part of the state vector since it can be computed by means of direct kinematic given the joint position vector $\boldsymbol{q}$; for floating base manipulators, use of the vehicle configuration $\boldsymbol{\eta}$ is also required.

A well-known procedure, usually adopted in robotics, the Denavit–Hartenberg (DH) convention [22], sets the rules for placing, for an n-link manipulator, $n+1$ frames, i.e., one for each link plus one for the base frame. Following the procedure allows to build systematically the homogeneous transformation matrices between two consecutive frames, to define the joint variables, thus the joint vector $\boldsymbol{q} \in \mathbb{R}^n$. In a manipulator with a floating base, as when a manipulator is attached to an underwater vehicle, the zero or base frame, following the DH procedure, must be attached rigidly to the vehicle. The new state vector of the vehicle manipulator system (where we are omitting the position, Euler angles, and joint angles) can be defined as follows:

$$\boldsymbol{\zeta} = \begin{bmatrix} \boldsymbol{\nu}_1 \ \boldsymbol{\nu}_2 \ \dot{\boldsymbol{q}} \end{bmatrix}^T \in \mathbb{R}^{6+n} \tag{16}$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the time derivative of the joint positions, i.e., the joint velocities.

Let us define the vector $\boldsymbol{\tau}_q \in \mathbb{R}^n$ of joint torques as

$$\boldsymbol{\tau}_q = \begin{bmatrix} \tau_{q,1} \\ \vdots \\ \tau_{q,n} \end{bmatrix} \tag{17}$$

where $\tau_{q,i}$ is the torque applied at the ith joint by the ith motor. Thus, the vector of forces and moments acting on the vehicle and the manipulator becomes $\boldsymbol{\tau} \in \mathbb{R}^{6+n}$

$$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\tau}_v \\ \boldsymbol{\tau}_q \end{bmatrix}. \tag{18}$$

Thus, it is possible to write the equation of motions of an underwater vehicle manipulator system in compact matrix form as [23] follows:

$$M(\boldsymbol{q})\dot{\boldsymbol{\zeta}} + C(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} + D(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} + g(\boldsymbol{\eta},\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}^{\text{ext}} \tag{19}$$

where $M(\boldsymbol{q}) \in \mathbb{R}^{(6+n)\times(6+n)}$ is the inertia matrix including added mass terms, $C(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} \in \mathbb{R}^{6+n}$ is the vector of Coriolis and centripetal terms, $D(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} \in \mathbb{R}^{6+n}$ is the vector of the dissipative effects, and $g(\boldsymbol{\eta},\boldsymbol{q}) \in \mathbb{R}^{6+n}$ is the vector of gravity and buoyancy effects. $\boldsymbol{\tau}^{\text{ext}} = \left[\boldsymbol{\tau}_v^{\text{ext}}, \boldsymbol{\tau}_q^{\text{ext}}\right] \in \mathbb{R}^{6+n}$ are again all forces and moments acting on the vehicle and the torques acting on manipulator joints ($\boldsymbol{\tau}_q^{\text{ext}}$).

Fig. 2 depicts a conceptual input/output model of (19).

It can be shown that both $M(\boldsymbol{q})$ and $C(\boldsymbol{q},\boldsymbol{\zeta})$ can be split, as already seen for the vehicle only [see (15)], in a sum of rigid body only and added masses inertia matrix only

$$\begin{aligned} M(\boldsymbol{q}) &= M_{\text{RB}}(\boldsymbol{q}) + M_A(\boldsymbol{q}) \\ C(\boldsymbol{q},\boldsymbol{\zeta}) &= C_{\text{RB}}(\boldsymbol{q},\boldsymbol{\zeta}) + C_A(\boldsymbol{q},\boldsymbol{\zeta}). \end{aligned} \tag{20}$$

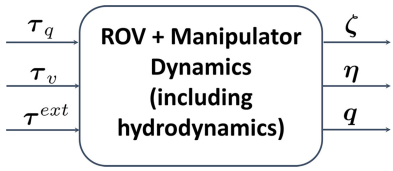

Fig. 2. Conceptual input/output model of (19).

The proof follows directly from [23, eq. (2.62) and (2.77)].

Note that matrices $M(\boldsymbol{q})$, $C(\boldsymbol{q},\boldsymbol{\zeta})$, $D(\boldsymbol{q},\boldsymbol{\zeta})$, and $g(\boldsymbol{\eta},\boldsymbol{q})$ depend on manipulator pose $\boldsymbol{q}$, and they are quite complex to derive. It is even more complex to compute M_{RB}, M_A, C_{RB}, and C_A separately. Note also that, when using this model, all external forces and moments acting on the manipulator, including all hydrodynamic effects, must be formulated as n equivalent joint torques $\boldsymbol{\tau}_{\text{ext}}^q$.

The model in (19) appears compact and extremely mathematically elegant, but its major drawback is that the computation of the model matrices can be extremely complex. Furthermore, if simulation of interaction with other bodies of the environment is sought (e.g., grasping a body, lifting a load, or simulation of contact with ground or other structures), it becomes necessary to transform all interaction forces and moments into forces and moments acting on the vehicle body ($\boldsymbol{\tau}_v$) and equivalent joint torques ($\boldsymbol{\tau}_q$); this latter job can be extremely cumbersome. The remainder of this article proposes a method to simplify the modeling efforts without losing simulation accuracy and fidelity, and adding the capability to manage complex whole body interactions with parts of the environment.

C. Introduction to Hybrid Simulation

Many different numerical solvers of dynamic equations of serial (and parallel) chains of rigid bodies exist [24]. These do not consider the equations of the whole system as in (19), but rather treat each body independently and manage connections between them by imposing and guaranteeing numerically all contact constraints; thus, these can be employed efficiently to solve iteratively the equations of articulated rigid bodies. Equation (19), when setting $M_A(\boldsymbol{q}) = 0$, $C_A(\boldsymbol{q},\boldsymbol{\zeta}) = 0$, $D(\boldsymbol{q},\boldsymbol{\zeta}) = 0$, and $g(\boldsymbol{\eta},\boldsymbol{q}) = 0$ (at least for its buoyancy component), actually represents that of an articulated body and could be solved with one of the aforementioned solvers. The hybrid simulation approach proposed in this article consists in splitting the simulator into two parts: one simulating only the articulated body dynamics, and the other simulating all hydrodynamics effects and propulsion forces. To apply this approach to simulation of an underwater robot–manipulator system, it is necessary to split (19) into two parts: one combining all rigid body components, and the other all components due to the interaction with the fluid. This second part has the dimension of an external force in the rigid body equations.

Let us start this process from the underwater vehicle model described in (14) with the assumptions (15): Added mass, hydrodynamic effect, and gravity effect (including buoyancy) contributions can be separated from rigid body terms (exactly

as they were originally introduced) as follows:

$$M_{\text{RB}}\dot{\boldsymbol{\nu}} + C_{\text{RB}}(\boldsymbol{\nu})\boldsymbol{\nu} = -M_A\dot{\boldsymbol{\nu}} - C_A(\boldsymbol{\nu})\boldsymbol{\nu} - D_v(\boldsymbol{\nu})\boldsymbol{\nu} - g_v(\boldsymbol{\eta}) + \boldsymbol{\tau}_v + \boldsymbol{\tau}_{\text{ext}}. \quad (21)$$

After this transformation, all right-hand-side terms can be considered "external" forces that are applied to the rigid body. However, note that the term containing M_A multiplies $\dot{\boldsymbol{\nu}}$, and this makes (21) a not well-posed differential equation. Clearly, if the right-hand side of (21) is treated as an external force to be applied to rigid body dynamics (the left-hand side), the vector $\dot{\boldsymbol{\nu}}$ on the right-hand side cannot be the same as the left-hand-side one. In other words, during simulation, it is not possible to compute exactly the external forces and moments that affect rigid body linear and angular accelerations at the same time of linear and angular accelerations themselves; this is what is commonly known as an algebraic loop. A practical approach to solve this issue is to introduce the approximated acceleration variable $\dot{\tilde{\boldsymbol{\nu}}}$, and use it to compute the right-hand side of (19) while checking that excessive simulation errors are not generated obtaining

$$M_{\text{RB}}\dot{\boldsymbol{\nu}} + C_{\text{RB}}(\boldsymbol{\nu})\boldsymbol{\nu} = -M_A\dot{\tilde{\boldsymbol{\nu}}} - C_A(\boldsymbol{\nu})\boldsymbol{\nu} - D_v(\boldsymbol{\nu})\boldsymbol{\nu} - g_v(\boldsymbol{\eta}) + \boldsymbol{\tau}_v + \boldsymbol{\tau}_{\text{ext}}. \quad (22)$$

The same technique can be applied to an underwater vehicle with a manipulator system: From (19), and by considering (20), all rigid body elements can be separated from all other contributions

$$M_{\text{RB}}(\boldsymbol{q})\dot{\boldsymbol{\zeta}} + C_{\text{RB}}(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} = -M_A(\boldsymbol{q})\dot{\boldsymbol{\zeta}} - C_A(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} - D(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} - g(\boldsymbol{\eta},\boldsymbol{q}) + \boldsymbol{\tau}^{\text{ext}} + \boldsymbol{\tau}. \quad (23)$$

Note that, again as in (21), $\dot{\boldsymbol{\zeta}}$ appears also on the right-hand side of (23). This means that (23) is not a well-posed differential equation. Again, the introduction of the approximated acceleration $\dot{\tilde{\boldsymbol{\zeta}}}$ variable, as in the previous case, yields the following:

$$M_{\text{RB}}(\boldsymbol{q})\dot{\boldsymbol{\zeta}} + C_{\text{RB}}(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} = -M_A(\boldsymbol{q})\dot{\tilde{\boldsymbol{\zeta}}} - C_A(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} - D(\boldsymbol{q},\boldsymbol{\zeta})\boldsymbol{\zeta} - g(\boldsymbol{\eta},\boldsymbol{q}) + \boldsymbol{\tau}^{\text{ext}} + \boldsymbol{\tau} \quad (24)$$

where the right-hand side can be considered the sum of all external forces (also due to the hydrodynamics) acting on the rigid body. Note that, anyway, all terms on the right-hand side of (24) are quite complex to compute. In addition, while similar terms in (21) can be easily computed provided that the added mass matrix is known [see [19, Property 2.5] for computing $C_A(\boldsymbol{\nu})$], dependency from joint angles vector $\boldsymbol{q}$ of the terms in (24) makes the mathematical burden very high; as a matter of fact, those terms embed a conversion of forces and moments acting on the bodies of each single manipulator link (due to velocity and acceleration of each single link and interaction with the fluid) into equivalent joint torques. This can be asserted by observing that $M_A(\boldsymbol{q})\dot{\boldsymbol{\zeta}}$ is a $(6+n)\times 1$ vector, whereas each of the n links has its own 6×6 added mass matrix.

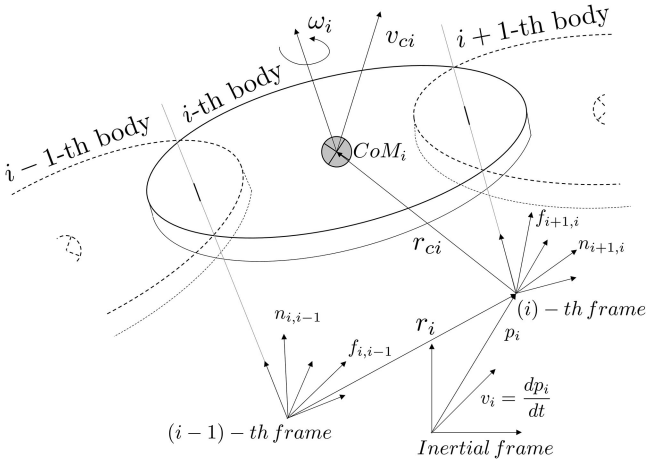

Fig. 3. Forces and moments interaction between generic linked bodies.

It is quite clear how formulation of the above-mentioned equations can be extremely complex. Usually, iterative procedures, such as the articulated body or Featherstone's algorithm [25], are employed to obtain algorithmically (19) or to compute the motion of the articulated body numerically. Section II-D proposes and motivates analytically our modeling approach that represents an alternative to computing explicitly the right-hand side of (23).

D. From General Newton–Euler to Fossen Formulation and Hybrid Simulation

An underwater vehicle with manipulators can be modeled as a chain of bodies (or links) connected by articulated joints. Fig. 3 represents a generic chain of bodies showing forces and moments involved in the following equations. The dynamics of a multibody system can be obtained by first defining inertia forces ${}^i\boldsymbol{f}_i^*$ and moments ${}^i\boldsymbol{n}_i^*$ of the ith body or link as applied at the origin of frame i (each link has its own frame attached), see [24] for details

$${}^i\boldsymbol{f}_i^* = -m_i\underbrace{\left({}^i\dot{\boldsymbol{v}}_i + {}^i\dot{\boldsymbol{\omega}}_i \times {}^i\boldsymbol{r}_{ci} + {}^i\boldsymbol{\omega}_i \times \left({}^i\boldsymbol{\omega}_i \times {}^i\boldsymbol{r}_{ci}\right)\right)}_{\dot{v}_{ci}} \quad (25)$$

$${}^i\boldsymbol{n}_i^* = -{}^iI_i\,{}^i\dot{\boldsymbol{\omega}}_i - {}^i\boldsymbol{\omega}_i \times \left({}^iI_i\,{}^i\boldsymbol{\omega}_i\right) \quad (26)$$

where ${}^i\dot{\boldsymbol{v}}_i$ is the linear acceleration of link i (right subscript), which is of the origin of frame i, expressed in frame i (left superscript) obtained by differentiating linear velocity ${}^i\boldsymbol{v}_i$ in the inertial reference frame, ${}^i\dot{\boldsymbol{\omega}}_i$ is the angular acceleration of link i (right subscript) expressed in frame i (left superscript) obtained by differentiating angular velocity ${}^i\boldsymbol{\omega}_i$ in the inertial reference frame, ${}^i\boldsymbol{r}_{ci}$ is the vector connecting the origin of frame i to the center of mass (CoM_i) of link i (subscript ci) expressed in link i reference frame (left superscript), and ${}^iI_i$ is the inertia tensor of link i (right subscript) expressed in frame i (left superscript).

Then, dynamic equilibrium equations can be written by using the first and second Euler axioms for link i as follows:

$${}^i\boldsymbol{f}_i^* + {}^i\boldsymbol{f}_{i,i-1} - {}^i\boldsymbol{f}_{i+1,i} + m_i\,{}^i\boldsymbol{g} + {}^i\boldsymbol{f}_{\text{ext}} = 0$$

$$ {}^{i}\boldsymbol{n}_{i}^{*} + {}^{i}\boldsymbol{n}_{i,i-1} - {}^{i}\boldsymbol{n}_{i+1,i} - \left({}^{i}\boldsymbol{r}_{i} + {}^{i}\boldsymbol{r}_{ci}\right) \times {}^{i}\boldsymbol{f}_{i,i-1} + {}^{i}\boldsymbol{r}_{ci} \times {}^{i}\boldsymbol{f}_{i+1,i} + {}^{i}\boldsymbol{n}_{\text{ext}} = 0 \tag{27} $$

where ${}^{i}\boldsymbol{f}_{i,i-1}$, ${}^{i}\boldsymbol{f}_{i+1,i}$, ${}^{i}\boldsymbol{n}_{i,i-1}$, and ${}^{i}\boldsymbol{n}_{i+1,i}$ are forces and pure moments due to interaction with adjacent links, and ${}^{i}\boldsymbol{r}_{i}$ is the distance between $(i-1)$th and ith reference frames

Just for exemplification, ${}^{i}\boldsymbol{f}_{i,i-1}$ represents the force that link $i-1$ produces on link i at the joint connecting them; hence, the term $-({}^{i}\boldsymbol{r}_{i} + {}^{i}\boldsymbol{r}_{ci}) \times {}^{i}\boldsymbol{f}_{i,i-1}$ is the moment due to contact force with respect to center of mass for link i. More details can be found in [24]. Note that, in this formulation, torques produced by motors mounted on the joints are considered in ${}^{i}\boldsymbol{n}_{\text{ext}}$. Equations (25)–(27), one for each link of the chain, together with position and attitude kinematics

$$ \begin{aligned} \dot{\boldsymbol{p}}_{i} &= C_{i}^{I}\, {}^{i}\boldsymbol{v}_{i} \\ \dot{C}_{I}^{i}\left(\phi,\, \theta,\, \psi\right) &= -\left[{}^{i}\boldsymbol{\omega}_{i}\times\right] C_{I}^{i}\left(\phi,\, \theta,\, \psi\right) \end{aligned} \tag{28} $$

may be used to simulate an articulated body provided that means exist for computing the external forces and moments applied to each single link. The term C_{I}^{i} is the rotation matrix from the inertial frame (superscript I) to the ith link reference frame; note that the matrix $C_{i}^{I} = C_{I}^{i\,T}$ is the same as that of matrix $J_{1}\left(\boldsymbol{\eta}_{2}\right)$ in (8) and can be parameterized with Euler angles or quaternions for instance. External forces and moments derive, in the general case, from interaction with the environment, including contact with other bodies and hydrodynamic effects.

Note that (21) and (24) are based on (1) and (2) that use derivatives in body frames; Equations (25)–(27), instead, are written using derivatives in the inertial axis. Luckily, these can be easily converted to use derivatives in body frame as well using (3) and (4) obtaining

$$ \boldsymbol{f}_{i}^{*} = m_{i}(\mathring{\boldsymbol{v}}_{i} + \boldsymbol{\omega}_{i} \times \boldsymbol{v}_{i} + \mathring{\boldsymbol{\omega}}_{i} \times \boldsymbol{r}_{ci} + \boldsymbol{\omega}_{i} \times (\boldsymbol{\omega}_{i} \times \boldsymbol{r}_{ci})) \tag{29} $$

$$ \boldsymbol{n}_{i}^{*} = I_{i}\mathring{\boldsymbol{\omega}}_{i} + \boldsymbol{\omega}_{i} \times (I_{i}\boldsymbol{\omega}_{i}) + m\boldsymbol{r}_{ci} \times (\mathring{\boldsymbol{v}}_{i} + \boldsymbol{\omega} \times \boldsymbol{v}_{i})\,. \tag{30} $$

These equations, together with (27) and (28), are virtually identical to (6) and (7), provided r_{ci} is replaced with r_{G} and the superscript i (referring to ith link of the multibody system) is removed.

Multibody numerical simulators solve (25)–(28) using numerical algorithms, such as the Featherstone's algorithm [25], and compute contact forces with other bodies of the environment using a large variety of contact simulation models; hydrodynamic effects, instead, are usually not considered or are only roughly approximated. Since "additional" forces and moments acting on each link of the multibody structures (note that, in this formulation, the ROV body becomes one of these links) are modeled by ${}^{i}\boldsymbol{f}_{\text{ext}}$ and ${}^{i}\boldsymbol{n}_{\text{ext}}$, hydrodynamic effects can be inserted, for each link separately, using these terms.

The forces ${}^{i}\boldsymbol{f}_{\text{ext}}$ and moments ${}^{i}\boldsymbol{n}_{\text{ext}}$ are forces and moments acting on the ith link expressed in the ith link reference frame; thus they are equivalent to the vector $\boldsymbol{\tau}_{\text{RB}}$ in (6), and, consequently, all the terms on the right-hand side of (21) could be computed outside the multibody simulator, and used as ${}^{i}\boldsymbol{f}_{\text{ext}}$ and ${}^{i}\boldsymbol{n}_{\text{ext}}$ to add hydrodynamic effects to the multibody simulation. This approach allows us to define all forces and moments due to the underwater environment as each single link of the articulated body (including the ROV body as one of these) is a separate body. This fact results in a dramatic simplification of nonrigid body terms, since explicit dependency from $\boldsymbol{q}$ is vanished. Expressions for added mass, potential damping, and restoring forces on the right-hand side of (22) can be used directly instead of computing the terms on the right-hand side of (24). Thus, the explicit conversion of external forces acting on the links into equivalent joint torques needed to build the matrices of the right-hand side of (24) is not necessary. Note that, as already highlighted before, this equation presents an approximation issue due to the presence of $\dot{\boldsymbol{\nu}}$ on the right-hand side.

Since the ROV body and the manipulators are treated as links of the articulated body structure, it is necessary to implement the calculation of hydrodynamic effects as in (22) for each one of them; the external forces and moments acting on the ith link are as follows:

$$ \begin{bmatrix} {}^{i}\boldsymbol{f}_{\text{ext}} \\ {}^{i}\boldsymbol{n}_{\text{ext}} \end{bmatrix} = -M_{A}\dot{\tilde{\boldsymbol{\nu}}}_{i} - C_{A}\left(\boldsymbol{\nu}_{i}\right)\boldsymbol{\nu}_{i} - D_{v}\left(\boldsymbol{\nu}_{i}\right)\boldsymbol{\nu}_{i} - g_{v}\left(\boldsymbol{\eta}_{i}\right) + \boldsymbol{\tau}_{i} + \boldsymbol{\tau}_{i}^{\text{ext}} \tag{31} $$

where $\boldsymbol{\nu}_{i} = \left[{}^{i}\boldsymbol{v}_{i},\, {}^{i}\boldsymbol{\omega}_{i}\right]$, $\boldsymbol{\eta}_{i} = [\boldsymbol{p}_{i},\, \boldsymbol{\epsilon}]$ (with $\boldsymbol{\epsilon}$ a parameterization of the rotation matrix C_{i}^{I}), $\dot{\tilde{\boldsymbol{\nu}}}_{i} = (d/dt)({}^{i}\boldsymbol{v}_{i})$, and $\boldsymbol{\tau}_{i}$ and $\boldsymbol{\tau}_{i}^{\text{ext}}$ are the vectors of applied forces and moments.

For the ROV body, $\boldsymbol{\tau}_{i}$ is actually the propulsion vector $\boldsymbol{\tau}_{v}$, and $\boldsymbol{\tau}_{i}^{\text{ext}}$ can be used to model external disturbances; for the manipulator links instead, $\boldsymbol{\tau}_{i}$ should be computed from the joint torques $\boldsymbol{\tau}_{q,j}$ in (17) (note that the jth manipulator link corresponds to the ith body of the multibody structure) applied by the motor mounted at the joint, whereas $\boldsymbol{\tau}_{i}^{\text{ext}}$ has a similar function as for the ROV body. Multibody simulators, in general, allow to specify directly the joint torque, thus the explicit conversion from $\boldsymbol{\tau}_{q,j}$ to $\boldsymbol{\tau}_{i}$ is usually not necessary as will be shown in Section III.

In conclusion, the dynamics described by (24) can be implemented by simulating the articulated body dynamics of the ROV and attached manipulators (with a total of $6+n$ degrees of freedom) using a multibody simulator [that actually implements the left-hand side of (24)] and adding all hydrodynamic effects using a set of $n+1$ equations such as (31), one for each link of the manipulator, plus one for the ROV body. Finally, it is possible to build the state vector of the ROV plus manipulator system in (24): $[\boldsymbol{\zeta},\, \boldsymbol{\eta},\, \boldsymbol{q}]$ directly from the state vector of the multibody simulator. Fig. 4 depicts the structure of the proposed hybrid simulator where the two main components are well identified.

Section III describes how to implement this concept using a commercial multibody simulator software, and discusses the effects, in terms of simulation accuracy, of using the approximated acceleration for computing the added mass terms.

III. Hybrid Simulator Development

Several multibody dynamics simulators exist; they usually provide a graphic environment where robots, vehicles, etc., can be assembled: As a matter of fact, such multibody systems are modeled as articulated open or closed chains of rigid bodies that are connected together by joints (passive or actuated, linear or

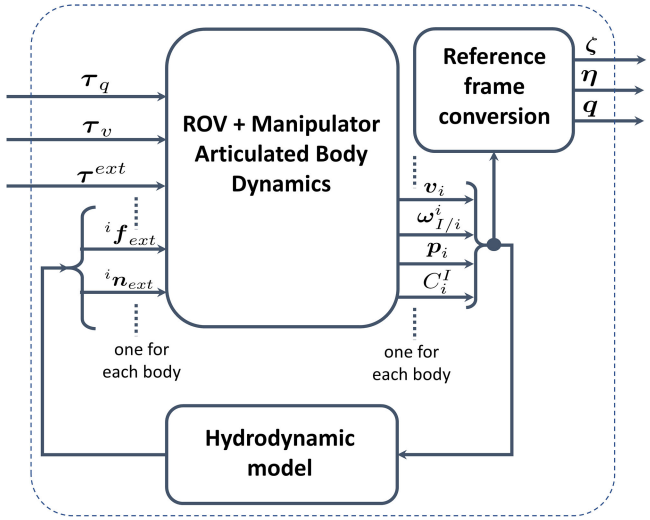

Fig. 4. Conceptual input/output scheme showing implementation of (19) through the use of (24) and (31).

rotary) that block or allow relative motion along specific degrees of freedom. The entire model is then simulated by an underlying PE. The PE is essentially the software that solves the differential equations [in practice (4)–(28)] associated with the model under study, enforcing the constraints posed by the degrees of freedom of the joints, and that manages the interaction between the model and the environment (usually composed of other bodies subject to the same laws of physics). Interaction with the environment is achieved by employing contact models [26]–[28] to compute forces and moments that each body of the model exerts on each body of the environment and vice versa.

External forces and moments (namely ${}^i\boldsymbol{f}_{\text{ext}}$ and ${}^i\boldsymbol{n}_{\text{ext}}$) can be added at each body, as well as torques or forces acting on the joints (namely the torques $\boldsymbol{\tau}_{q,j}$).

The scientific literature contains several comparison exercises between various PEs [29]–[32]; however, comparing features of various PEs is outside the scope of this article. A brief overview on how a PE library works is presented in Section III-A, and a comparison with some existing simulators is presented in Section III-B with the aim of highlighting the positioning of the proposed simulation architecture with respect to other solutions that are usable/used, natively or not, for the underwater environment.

Section III-C presents, instead, how the proposed hybrid simulation approach, depicted in Fig. 4 and presented in Section II, can be effectively implemented using a robotics simulator and a PE. The simulator selected for this article, V-REP by Coppelia Robotics, together with few code excerpts using its extension application programming interfaces (APIs), is used, as an example, to highlight how such a hybrid simulation approach can be easily implemented.

Finally, simulation accuracy issues are discussed in Section III-D, with particular attention to the approximation introduced by the proposed hybrid simulation with respect to the added mass terms (see the non-well-posedness issue of (24) discussed in Section I).

A. Overview of PE Libraries

A PE is a software that provides an approximate simulation of certain physical systems, such as chains of rigid bodies, soft bodies, and, to some extents, even fluids. In general, a robotics rigid body simulator is just a graphical front end to a PE, which is employed to graphically build the interconnections between the bodies and to visualize graphically the output of the simulation, which is the motion of the bodies. Usually, a simulator may support several PEs. PEs of relevance for this article are based on the Newton–Euler equations [namely (4)–(28)]. However, PEs differ from each other in the way they represent the rigid body structure and in the method used to compute contact forces [33], [34]; in addition, sometimes, PEs accuracy with respect to precise contact simulation is questioned [35], mainly due to the adoption of the first-order Euler numerical integration. This is because accurate contact modeling remains an active research area. Details on active research areas in contact modeling and rigid body dynamics simulation can be found, for instance, in [36] and [37].

Nowadays, simulators and PEs that combine efficient recursive algorithms and are considered quite reliable by the scientific community are available. For the purpose of building a demonstrator of the hybrid simulator approach presented in this article, four PEs were considered: Open Dynamics Engine (ODE) [38], Bullet [39], Newton [40], and Vortex [41], [42].

ODE and Bullet are multipurpose rigid body PEs and open-source libraries, and are used in several simulation tools. Newton Dynamics is a cross-platform physics simulation library that implements a deterministic solver, which is not based on iterative methods, but is considered stable and reaches real-time performance. Vortex is a commercial software specialized in simulation of contact dynamics in different operating environments (e.g., terrain and water).

Several researchers attempted a field comparison of various PEs, including those considered for this article, in terms of stability, simulation speed, and physical accuracy (see, for instance, [29]–[32]). We may summarize here that the general conclusion is that none of them should be considered better than the others: The most common conclusion of scientific papers that attempt to compare them is that, given their respective weaknesses and points of strength, the best PE depends on the application that must be developed.

Since, generally speaking, no PE for articulated rigid bodies turned out to be preferable over the others, a simulator, i.e., V-REP, capable of supporting all of them was selected as exterior shell and graphical front end for our hybrid simulator. Specific motivations for selecting V-REP are also contained in Section III-B.

B. Possible Comparison With Existing Simulators for Underwater Vehicles

It is undoubtedly difficult to propose a fairly quantitative comparison of different simulators, for the underwater environment or not, in terms, for instance, of computational time, real-time performance, or, even worse, accuracy. A recent survey on existing simulators for the underwater environment [12] compared

several platforms and concluded that, even if there are many options for simulation of underwater robots, each with its own strengths and weaknesses, each one of them requires additional efforts from the user for conscious use and trust in the results. The authors also added that future development in autonomous underwater vehicle simulation should seek a higher physical fidelity in terms of end effector world interaction as well as in terms of simulated hydrodynamic and environmental forces.

Nonetheless, to better frame the proposed hybrid simulation approach in the landscape of underwater vehicle simulation, we provide, in the following, a qualitative comparison with respect to some well-known and widely used simulation tools.

Several commercial simulators exist, such as the FMC Schilling Robotics ROV Pilot Simulator [17] and Forum Energy Technologies VMax [18]. The ROV Pilot Simulator and VMax are commercial simulators intended for pilot training, and are not sufficiently open as research tools. They are considered state-of-the-art simulators and are widely used worldwide by underwater operations companies; nonetheless, it is difficult to tell to which extent the designers of such simulators favored a graphically appealing and stable simulation to a dynamically accurate one. The ROV Pilot Simulator dynamic simulation is managed by the Vortex PE, a commercially available PE that is also usable as a PE in our proposed hybrid simulator.

Several research simulators exist instead that are fully open and available to the academic community; the most widely used appear to be UWsim [14] and UUV Sim [13]. UWSim is an open-source simulator implemented using several software modules and a midlleware for interprocess communications (i.e., ROS), and is currently used in the underwater research community. UWSim implements limited contact and interaction dynamics by simulating, in parallel to vehicle dynamics simulation (using Bullet, one of the PEs considered in this article), a collision model that renders contact forces to be fed back to the vehicle dynamic simulation. Although this approach may work for very simple tasks, it is well known to be little reliable and prone to contact simulation instability (see for instance [43]). Only the Bullet PE is supported by UWSim, and although Bullet is a well-reputed PE, this represents a weakness in terms of future developments. In addition, thruster models are very simple, and changing them requires programming in C++ language and a certain knowledge of the OpenSceneGraph simulation framework.

UUVSim [13] is an underwater vehicle simulator based on Gazebo, a general-purpose robotic simulator. It is written in C++ and requires C++ programming for extending and modifications of the vehicle model. Gazebo implements simulation of rigid body dynamics using ODE or BULLET as PE, thus UUVSim introduces hydrodynamic effects similarly to what is proposed in our simulation approach, although it does not simulate added masses, as its designers state, to limit possible simulation instabilities [44]. Thrusters and actuated fins models are implemented as Gazebo plug-ins and, although this represents a modular approach to vehicle modeling, with the aim of making realization of new vehicles easy, it has the drawback of offering very simple dynamic models for these elements that make simulation fidelity low. Modification of these components requires knowledge of the ROS midlleware and programming in C++.

Other robotic simulators have been used to implement simulations of underwater vehicles; Freefloating-gazebo [44] and UW MORSE [45] are examples of such efforts. MORSE [46] is an open-source general-purpose academic robot simulator, and the underwater vehicles simulator UW MORSE [45] is built upon it. As UWSim and UUV Sim, it adopts Bullet as PE library, thus the physical fidelity is considered high. Accuracy of underwater dynamics simulation is difficult to tell since the authors do not explicitly describe how simulated underwater effects are integrated into the PE. Freefloating-gazebo, not much differently from UUVsim, exploits the PE embedded in Gazebo to simulate the rigid body dynamics and adds hydrodynamics forces externally. The author explicitly excludes the simulation of added masses, and only quadratic drag effects are simulated.

As a final remark, we should state that a comparison in terms of simulation accuracy or fidelity, meaning the capability of a simulator to reproduce closely the behavior of a specific real vehicle, would be inappropriate here: Simulation fidelity depends largely on the knowledge of the actual vehicle parameters and hydrodynamic effects that are indeed difficult to quantify. What can be done, instead, is to compare the possibility of a simulator to simulate correctly the specified dynamics given the best parameters and hydrodynamics models/maps available.

For what regards accuracy of contact and collision simulation, most of the solutions analyzed rely on the Bullet PE. Since Bullet is one of the PEs available through V-REP, the simulator adopted in this article for the implementation of the proposed hybrid simulation approach, we may conclude that there should not be any major difference in terms of collision simulation accuracy between our proposed solution and other existing simulators. The fact that V-REP supports several PEs in addition to Bullet represents an advantage of our solution in terms of future upgrade ability and sustainability of our simulator, that is, it is actually independent of any specific PE.

In addition to this, as will be shown in the following, the proposed hybrid simulation approach is designed to achieve a tight integration between the PE and the simulator part computing the necessary additional forces and moments due to the underwater environment; this guarantees the absence of any possible midlleware communication delay (that may be present in modular distributed solutions such as UWSim, UUV Sim, or UW Morse) or other specific simulator-bound issue. Actually V-Rep acts only as an exterior graphical shell to the simulation performed by the hybrid simulator.

Our proposed approach is specifically designed for GNC system design needs and allows us to create highly detailed models of any underwater vehicle component, such as actuators and sensors, to include detailed hydrodynamic maps for each single part of the vehicle with great ease directly, as anticipated, in the MATLAB/Simulink environment, and to include any dynamic system simulating GNC functionalities.

With the proposed tight integration scheme, the PE exchanges data at each simulation step with the simulator part implementing all “nonrigid body” dynamics and functionalities (hydrodynamics, control, etc.) in MATLAB/Simulink, simulating rigid body dynamics and all other components synchronously with

a common simulation time base and absolutely no simulation delay.

Finally, on the contrary to all other simulators analyzed, simulation of added masses is not a problem in our simulator, even if with a minimal simulation error as shown later in Section III-D.

Comparison in terms of computation time with respect to real time is also difficult to perform since it is largely affected by the graphical complexity of the rendered scene, and the various simulators analyzed differ significantly in terms of graphical output. The portion of computation time due to the execution of the PE instead should be very similar in all the simulators analyzed that adopt the same PE. It might appear that, at least for our proposed hybrid simulator, the simulator part computing the hydrodynamic forces and GNC components should take a considerable part of the computing time, since it is implemented in Simulink and not in C++ or other compiled language. Nonetheless, as it will be explained later, the Simulink scheme implementing all "nonrigid body" parts of the simulator is compiled and transformed into a dynamic link library (DLL), and, in a reasonably modern desktop computer, all the simulations we tried ran faster than real time, allowing for interactive simulation.

C. Sample Implementation Using V-REP

V-REP is a highly customizable simulator that supports all the four PEs that were considered for implementation of the proposed hybrid simulator.

The capability to inject external forces in the PE at run time, computed as a function of the vehicle state (i.e., the capability to simulate the presence of generic externally generated forces in the rigid body equations), is a fundamental PE requirement for the implementation of the proposed hybrid simulation approach. V-REP offers an elaborate API that supports six different programming languages, each having particular advantages (and obviously also disadvantages) over the others, but all six are mutually compatible (i.e., can be used at the same time, or even hand in hand). Implementation details will be limited to a minimum to keep the discussion general enough and applicable also to other PE–simulator couples; nonetheless, pseudocode excerpts are provided below to ease implementation of a similar hybrid simulator by interested research groups.

The simulation algorithm of the PE embedded inside V-REP may be represented by the following pseudocode:

Require: Simulation algorithm
1: Set initial conditions $X \leftarrow X_0$ and $T \leftarrow T_0$.
2: **while** simulation non finished **do**
3: $T_{k+1} \leftarrow T_k + \Delta T$
4: compute new PE state : $X_{k+1} \leftarrow f(X_k)$
5: **end while**

where the function $f(\cdot)$ is symbolizing the execution of the PE for one time step, that is, in our case, the numerical integration of the left-hand side of (24) with the vector of contact/constraint forces and moments at the right-hand side.

Simulators such as V-REP allow us to modify the algorithm flow above by inserting, before computing the new PE state, a call to external functions that may interact with the PE using software APIs. This is achieved by creating the so-called child scripts. To implement our proposed hybrid simulation approach, any additional forces and moments acting on the parts of ROV, which do not come from rigid body dynamics (that are unknown thus to the PE) as hydrodynamic forces, environmental forces, propulsion forces, etc., namely the terms at the right-hand side of (24), can be added by means of an appropriate child script that sets them before letting the PE step forward in time and compute next system state.

The simulation algorithm then becomes the following:

Require: Simulation algorithm with external forces
1: Set initial conditions $X \leftarrow X_0$ and $T = T_0$.
2: **while** simulation non finished **do**
3: $T_{k+1} \leftarrow T_k + \Delta T$
4: child scripts executes $\rightarrow$ compute all external forces: $U_k = g(X_k)$
5: compute new PE state : $X_{k+1} \leftarrow f(X_k, U_k)$
6: **end while**

where the function $g(\cdot)$ is symbolizing the computation of all additional forces and moments U_k, excluding those coming from contacts and constraints, to be applied to the simulation scene bodies.

As anticipated, APIs and child scripts can be developed in several languages; in our implementation, a very simple child script was created in Lua, a scripting language with the only duty of running an external V-REP plug-in. A plug-in is a shared library (e.g., a Windows DLL) that is automatically loaded by V-REP's main client application at program start up and can be developed in any language that is able to generate a shared library and to call exported C-functions (the V-REP APIs). With this technique, well documented in the V-REP manual, it is possible to let V-REP execute any code, synchronous with the simulation, and at any time step, that has the possibility to "interact" with the bodies managed by the PE.

The plug-in code interacts with the bodies using the APIs; many different functionalities exist, but, for the purpose of this article, it is necessary only to read all bodies state (position, attitude, and angular and linear velocities) and to apply forces and moments to them. Just for exemplification, the following code excerpt shows the V-REP C++ API functions that can be used to read the following vehicle state:

Require: Read body state vector
simGetObjectPosition(Rov,-1,Rov$_{\boldsymbol{p}}$);
simGetObjectOrientation(Rov,-1,Rov$_{\phi,\theta,\psi}$);
simGetObjectVelocity(Rov, Rov$_{\boldsymbol{v}}$, Rov$_{\boldsymbol{\omega}}$);

where Rov is an appropriate C++ object that "points" to the actual ROV body modeled in V-rep, and Rov$_{\boldsymbol{p}}$, Rov$_{\phi,\theta,\psi}$, Rov$_{\boldsymbol{v}}$, and Rov$_{\boldsymbol{\omega}}$ are appropriate C++ objects that will be filled, with current vehicle dynamic state, namely η and ν of (24). Similar code can be used to obtain dynamic state of any body that is

under control of the PE: manipulator links, thrusters, fixed or moving objects of the environment, etc. Thus V-REP allows the plug-in to access all the system state in between two successive time steps.

Once the system state is known, all forces and moments to be applied to each single body can be computed by the plug-in. Actual application of these forces to the rigid bodies of the scene happens, again, using the APIs. Just as exemplification, the following code excerpt shows the C++ API function used to apply these forces (that is actually to tell the PE that these forces must be included in its next state update step via the function):

Require: Set body forces and moments
simAddForceAndTorque(Rov, F_{hydro}, M_{hydro});

where F_{hydro} and M_{hydro} are appropriate C++ objects that must have been filled, for this example, with ${}^i\boldsymbol{f}_{\text{ext}}$ and ${}^i\boldsymbol{n}_{\text{ext}}$ computed by (31). Note that, to implement (31), it is necessary to compute the approximated acceleration $\dot{\tilde{\boldsymbol{\nu}}}_i$, which is the approximate acceleration of the ith link at the current time step; Section III-D discusses the implications of this approximation.

The V-REP plug-in that acquires the state of each single body composing the ROV and its manipulators, and computes and applies ${}^i\boldsymbol{f}_{\text{ext}}$ and ${}^i\boldsymbol{n}_{\text{ext}}$ could be developed directly in C++; another option is to use MATLAB and Simulink, to implement (31) and use the Simulink automatic code generation tool to produce the plug-in C++ code automatically. This latter option is what we followed in this article: The code necessary to acquire body state and to apply forces and moments was embedded into Simulink blocks, and a Simulink scheme was built for implementing all the elements of (31). Then, the Simulink scheme was used, together with Mathworks embedded coder to generate directly the plug-in DLL that is loaded by the Lua child script. This approach has the tremendous advantage of allowing the use of MATLAB and Simulink for implementing those parts of the simulator (hydrodynamic maps, thruster dynamic models, etc.) where these tools are a de facto standard. In addition, sensor models, actuator models, and GNC systems can be included in this part of the simulator allowing the simulation of a complete ROV mission including its interaction with other objects of the environment, such as loads to be lifted, valve panels, etc., and also to simulate collisions.

Fig. 5 shows a conceptual block diagram of the hybrid simulator where the blocks indicated with R and S represent the "read body state vector" and the "set body forces and moments" functions, respectively. These constitute the actual interface between V-Rep and Simulink, and thus the entire Simulink scheme implements the function $g(\cdot)$ introduced above. The approximate acceleration $\dot{\tilde{\boldsymbol{\nu}}}$, needed for the added mass term, is computed inside the block "Hydrodynamic model." The block diagram also contains the sensor and actuator dynamics blocks, and the GNC systems for the ROV and the manipulator. Everything inside the Simulink contour is implemented inside the Simulink scheme; the block "ROV+Manipulator Articulated Body Dynamics" is implemented by V-REP and the selected PE.

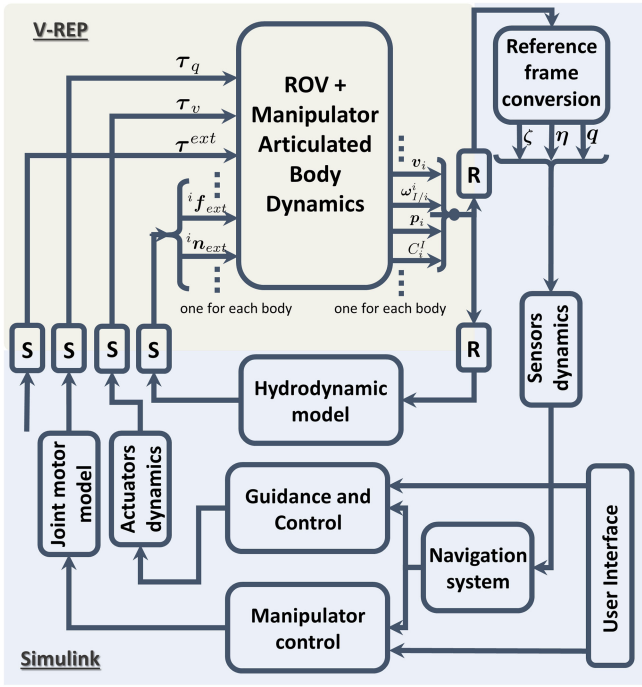

Fig. 5. Conceptual scheme of the elements composing the hybrid simulation.

D. Simulation Accuracy and Approximations

To understand the effects of using the approximate acceleration $\dot{\tilde{\boldsymbol{\nu}}}$ in (31) for the simulation of the ROV dynamics, let us first consider (14) as follows:

$$M_v\dot{\boldsymbol{\nu}} + C_v(\boldsymbol{\nu})\boldsymbol{\nu} + D_v(\boldsymbol{\nu})\boldsymbol{\nu} + g_v(\boldsymbol{\eta}) = \boldsymbol{\tau}_v + \boldsymbol{\tau}_v^{\text{ext}}. \quad (32)$$

To integrate it numerically, as in a simulator, let us bring all terms without $\dot{\boldsymbol{\nu}}$ to the right-hand side

$$\begin{aligned}(M_{\text{RB}} + M_A)\dot{\boldsymbol{\nu}} = M_v\dot{\boldsymbol{\nu}} &= -C_v(\boldsymbol{\nu})\boldsymbol{\nu} - D_v(\boldsymbol{\nu})\boldsymbol{\nu} - g_v(\boldsymbol{\eta}) \\ &\quad + \boldsymbol{\tau}_v + \boldsymbol{\tau}_v^{\text{ext}} \\ &= \boldsymbol{\tau}'\left(\boldsymbol{\nu}, \boldsymbol{\eta}, \boldsymbol{\tau}_v, \boldsymbol{\tau}_v^{\text{ext}}\right) \quad (33)\end{aligned}$$

that can be put into the canonical form as

$$\dot{\boldsymbol{\nu}} = (M_{\text{RB}} + M_A)^{-1}\boldsymbol{\tau}'. \quad (34)$$

Note that, since all terms in $\boldsymbol{\tau}'(\cdot)$ do not depend on $\dot{\boldsymbol{\nu}}$, no approximation is introduced.

All the PEs considered for this article adopt the Euler integration method, thus (34) is integrated numerically yielding

$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k + T(M_{\text{RB}} + M_A)^{-1}\boldsymbol{\tau}' + \epsilon \quad (35)$$

where T is the integration time step and ϵ represents the integration approximation error. Let us call (35), with a little abuse since it is affected by the numerical integration error ϵ, exact integration to symbolize what should be achieved in a simulation where simulation of rigid body dynamics and hydrodynamics is not performed separately.

It is well known that the numerical integration error ϵ can be made small by selecting an appropriately small integration time step T; nonetheless, an even small ϵ is unavoidable. For this

reason, it is not considered in the following analysis that focuses instead in the possibly larger simulation error due to simulation of added masses as external forces as in our hybrid approach.

It could be speculated that the masses and inertia of each single body in the articulated body simulator could be set so that the term $(M_{\text{RB}} + M_A)$ could be actually simulated inside it. Unfortunately, the structure of the M_{RB} matrix and that of M_A, which is usually obtained by identification, are not compatible (see for instance [19]), unless extreme simplifications are adopted. Thus, there is no possibility to correctly simulate the added mass effects by simply "modifying" the mass of the body and its inertia tensor.

When, as described in Section II-C, the added mass term is brought to the right-hand side and the approximate acceleration $\dot{\tilde{\boldsymbol{\nu}}}$ is used, we obtain

$$M_{\text{RB}}\dot{\boldsymbol{\nu}} = -M_A\dot{\tilde{\boldsymbol{\nu}}} + \boldsymbol{\tau}' \tag{36}$$

that is, in canonical form

$$\dot{\boldsymbol{\nu}} = M_{\text{RB}}^{-1}\boldsymbol{\tau}' - M_{\text{RB}}^{-1}M_A\dot{\tilde{\boldsymbol{\nu}}} \tag{37}$$

which can be discretized as

$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k + T\left(M_{\text{RB}}^{-1}\boldsymbol{\tau}' - M_{\text{RB}}^{-1}M_A\dot{\tilde{\boldsymbol{\nu}}}_k\right) + \epsilon \tag{38}$$

where $\dot{\tilde{\boldsymbol{\nu}}}_k$ is the approximate acceleration at time k.

Equation (38) represents the state update map of the proposed hybrid simulator. Lemma 1 below proves that if it would be possible to approximate $\dot{\tilde{\boldsymbol{\nu}}}$ using noncausal differentiation, then the proposed hybrid simulation approach represented by (38) would be equivalent to the exact integration represented by (35), that is, no error, except clearly for the numerical integration error ϵ, would be actually introduced.

Lemma 1 (Noncausal Derivative): The proposed hybrid simulation approach, with the added masses introduced as external forces as in (38)

$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k + T\left(M_{\text{RB}}^{-1}\boldsymbol{\tau}' - M_{\text{RB}}^{-1}M_A\dot{\tilde{\boldsymbol{\nu}}}_k\right) + \epsilon$$

is equivalent to the "exact" added masses simulation as in (35)

$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k + T\left(M_{\text{RB}} + M_A\right)^{-1}\boldsymbol{\tau}' + \epsilon$$

if $\dot{\tilde{\boldsymbol{\nu}}}$ is obtained by noncausal numerical derivative.

Proof: Let us define the noncausal derivative $\dot{\tilde{\boldsymbol{\nu}}}$ as follows:

$$\dot{\tilde{\boldsymbol{\nu}}}_k \triangleq \frac{\boldsymbol{\nu}_{k+1} - \boldsymbol{\nu}_k}{T}. \tag{39}$$

Then, by substituting (39) into (38), and omitting the numerical integration error ϵ from the comparison for simplicity, we obtain

$$\begin{aligned}\boldsymbol{\nu}_{k+1} &= \boldsymbol{\nu}_k + T\left(M_{\text{RB}}^{-1}\boldsymbol{\tau}' - M_{\text{RB}}^{-1}M_A\frac{\boldsymbol{\nu}_{k+1} - \boldsymbol{\nu}_k}{T}\right)\\ &= \boldsymbol{\nu}_k + T\,M_{\text{RB}}^{-1}\boldsymbol{\tau}' - M_{\text{RB}}^{-1}M_A\left(\boldsymbol{\nu}_{k+1} - \boldsymbol{\nu}_k\right).\end{aligned} \tag{40}$$

Let us bring all terms with $\boldsymbol{\nu}_{k+1}$ to the left-hand side

$$\begin{aligned}\boldsymbol{\nu}_{k+1} + M_{\text{RB}}^{-1}M_A\boldsymbol{\nu}_{k+1} &= M_{\text{RB}}^{-1}\left(M_{\text{RB}} + M_A\right)\boldsymbol{\nu}_{k+1}\\ &= \boldsymbol{\nu}_k + T\,M_{\text{RB}}^{-1}\boldsymbol{\tau}' + M_{\text{RB}}^{-1}M_A\boldsymbol{\nu}_k\end{aligned} \tag{41}$$

and multiply all terms by $\left(M_{\text{RB}} + M_A\right)^{-1}$ to obtain

$$\begin{aligned}\boldsymbol{\nu}_{k+1} = \left(M_{\text{RB}} + M_A\right)^{-1}M_{\text{RB}}\boldsymbol{\nu}_k + T\,\left(M_{\text{RB}} + M_A\right)^{-1}\\ \times\, M_{\text{RB}}M_{\text{RB}}^{-1}\boldsymbol{\tau}' + \left(M_{\text{RB}} + M_A\right)^{-1}M_{\text{RB}}M_{\text{RB}}^{-1}M_A\boldsymbol{\nu}_k.\end{aligned} \tag{42}$$

By collecting the terms with $\boldsymbol{\nu}_k$, we finally obtain

$$\begin{aligned}\boldsymbol{\nu}_{k+1} &= \left(M_{\text{RB}} + M_A\right)^{-1}\left(M_{\text{RB}} + M_A\right)\boldsymbol{\nu}_k\\ &\quad + T\,\left(M_{\text{RB}} + M_A\right)^{-1}\boldsymbol{\tau}'\\ &= \boldsymbol{\nu}_k + T\,\left(M_{\text{RB}} + M_A\right)^{-1}\boldsymbol{\tau}'.\end{aligned} \tag{43}$$

■

Lemma 2 proves, instead, that, if $\dot{\tilde{\boldsymbol{\nu}}}$ is approximated by causal differentiation, a simulation error is introduced by the proposed hybrid approach. A discussion on the practical effects of this approximation follows.

Lemma 2 (Causal Derivative): The proposed hybrid simulation approach, with the added masses introduced as external forces as in (38)

$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k + T\left(M_{\text{RB}}^{-1}\boldsymbol{\tau}' - M_{\text{RB}}^{-1}M_A\dot{\tilde{\boldsymbol{\nu}}}_k\right) + \epsilon$$

if $\dot{\tilde{\boldsymbol{\nu}}}$ is obtained by causal numerical derivative, is equivalent to the "exact" added masses simulation as in (35) with the addition of the disturbance force $\boldsymbol{\tau}_{\text{dist}}$ that is linearly proportional to the added mass matrix, the integration step, and the trajectory Jerk

$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k + T\left(M_{\text{RB}} + M_A\right)^{-1}\left(\boldsymbol{\tau}' + \boldsymbol{\tau}_{\text{dist}}\right).$$

Proof: Let us define the causal derivative as follows:

$$\dot{\tilde{\boldsymbol{\nu}}}_k \triangleq \frac{\boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}}{T}. \tag{44}$$

Then, by substituting (44) into (38), we obtain

$$\begin{aligned}\boldsymbol{\nu}_{k+1} &= \boldsymbol{\nu}_k + T\left(M_{\text{RB}}^{-1}\boldsymbol{\tau}' - M_{\text{RB}}^{-1}M_A\frac{\boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}}{T}\right)\\ &= \boldsymbol{\nu}_k + T\,M_{\text{RB}}^{-1}\boldsymbol{\tau}' - M_{\text{RB}}^{-1}M_A\left(\boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}\right).\end{aligned} \tag{45}$$

It can be shown, by adding and removing the term $M_{\text{RB}}^{-1}M_A\left(\boldsymbol{\nu}_{k+1} - \boldsymbol{\nu}_k\right)$ on the right-hand side of (45) and using a little algebra, that

$$\begin{aligned}\boldsymbol{\nu}_{k+1} &= \boldsymbol{\nu}_k + T\left(M_{\text{RB}} + M_A\right)^{-1}\boldsymbol{\tau}' - \left(M_{\text{RB}} + M_A\right)^{-1}\\ &\quad \times\, M_A\left(-\boldsymbol{\nu}_{k+1} + 2\boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}\right)\\ &= \boldsymbol{\nu}_k + T\left(M_{\text{RB}} + M_A\right)^{-1}\boldsymbol{\tau}' - T\left(M_{\text{RB}} + M_A\right)^{-1}\\ &\quad \cdot 1/T \cdot M_A\left(-\boldsymbol{\nu}_{k+1} + 2\boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}\right)\\ &= \boldsymbol{\nu}_k + T\left(M_{\text{RB}} + M_A\right)^{-1}\left[\boldsymbol{\tau}' + \boldsymbol{\tau}_{\text{dist}}\right]\end{aligned} \tag{46}$$

where $\boldsymbol{\tau}_{\text{dist}} = -\frac{1}{T}M_A\left(-\boldsymbol{\nu}_{k+1} + 2\boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}\right)$.

By expanding the term $\boldsymbol{\tau}_{\text{dist}}$, we obtain

$$\begin{aligned}\boldsymbol{\tau}_{\text{dist}} &= -\frac{1}{T}M_A\left(-\boldsymbol{\nu}_{k+1} + \boldsymbol{\nu}_k + \boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}\right)\\ &= M_A\left(\frac{\boldsymbol{\nu}_{k+1} - \boldsymbol{\nu}_k}{T} - \frac{\boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}}{T}\right)\end{aligned} \tag{47}$$

where the term in parenthesis can be recognized as the difference between accelerations at the kth and $(k-1)$th instants

$$\boldsymbol{\tau}_{\text{dist}} = M_A \left(a_k - a_{k-1}\right). \tag{48}$$

Then, by multiplying the above equation by $\frac{T}{T}$, we obtain

$$\boldsymbol{\tau}_{\text{dist}} = M_A T \left(\frac{a_k - a_{k-1}}{T}\right) = M_A T \cdot \text{Jerk}_k \tag{49}$$

where

$$\text{Jerk}_k \triangleq \frac{a_k - a_{k-1}}{T}. \tag{50}$$

■

Lemma 2 states that the net effect of the hybrid simulation is equivalent to that of the disturbance force $\boldsymbol{\tau}_{\text{dist}} = M_A T\,\text{Jerk}_k$. This result indicates that the hybrid simulator should be able to produce accurate results when the vehicle is near a steady condition, moving at constant velocity or even posses a constant acceleration. Smoothness of motion is a usual assumption on underwater vehicles, thus the value of Jerk should always be small, yielding accurate simulation results, at least comparable with those achievable by a "standard" approach. Furthermore, the presence of the term T in the disturbance force expression indicates that its effects may be mitigated by choosing smaller integration steps, and selecting small integration steps is usually the case when using the Euler integration method to reduce the integration error ϵ.

Note also that the disturbance force is independent from $\boldsymbol{\tau}'(\cdot)$. This means that the disturbance force is independent from any force, drag, and propulsion (or other modeled forces) acting on the vehicle.

To further investigate the implications of Lemma 2, let us consider the vehicle in a forced oscillatory motion since, even if this condition is of little interest in ROV simulation, the added mass term plays a fundamental role in simulation of a vehicle in oscillatory motion due to the presence of waves. Let us assume that, limiting for simplicity this analysis to motion along a single axis, the position of the vehicle is: $p_k = W \sin(\omega\, t)$, where ω is the wave frequency and W is the amplitude of the oscillatory motion, and the corresponding Jerk is clearly: $\text{Jerk}_k = -\omega^3 W \cos(\omega\, t)$. It is not difficult to show, using Lemma 2, that, in a forced oscillatory motion, the ratio of the equivalent disturbance force $\boldsymbol{\tau}_{\text{dist}}$ and the sum of all forces acting on the vehicle to sustain the oscillatory motion $\bar{\boldsymbol{\tau}}$ is

$$\frac{\boldsymbol{\tau}_{\text{dist}}}{\bar{\boldsymbol{\tau}}} = \frac{M_A}{M_{\text{RB}} + M_A} T\omega \tag{51}$$

where the ratio $M_A/(M_{\text{RB}} + M_A)$ is always less than 1 on all degrees of freedom, usually $T << 1$, and realistic wave frequencies ω of relevant power are always less than 1 (see, for instance, [19] and [47]–[49]), making the equivalent disturbance force due to inclusion of the added mass term as an external force only a small fraction of the total force involved.

IV. Simulation Results

To asses qualitatively the proposed simulator architecture, a complete ROV simulator including a user station with a display and a joystick for pilot commands was created. The ROV mass, moments of inertia, and hydrodynamic model were derived from a work-class ROV model (weight around 3800 kg) available from a previous research activity. ROVs are usually connected with the support ship and the pilot using a power and data cable called umbilical cable. The umbilical cable may generate strong disturbance forces on the vehicle, especially in small-sized ones (see [50] for instance). Accurate umbilical cable simulation was not in the scope of this article, and the results of this section have been achieved without simulating the effects of the umbilical cable and with $\tau^{\text{ext}} = 0$. However, thanks to the proposed hybrid simulation architecture, it could be possible, for instance, to model the umbilical cable in V-REP with a finite-element-like approach and to add hydrodynamic forces in Simulink for it, exactly as it is done for the ROV, otherwise cable dynamics could be simulated completely in Simulink creating an appropriate τ^{ext} and including it as an external force. It should be noted, finally, that the proposed hybrid simulation approach can be used to simulate not only ROVs but also other types of underwater vehicles or objects; as a matter of fact, the same approach was used to simulate the hydrodynamics forces acting on the object manipulated by the ROV in Scenarios 1 and 2.

The ROV guidance system was modeled as a high-end integrated DVL–USBL–inertial navigation system. A control system for regulation of speed references, generated by the pilot using the joystick, was designed as a Integral Linear Quadratic Regulator (or Servo LQR) similarly to what is described in [51]. A three-DOFs manipulator was simulated, and a simple control system was designed using inverse kinematics and closed-loop joint control allowing the pilot to move its end effector in Cartesian coordinates [23], [52]. In addition, a very simple controller for holding position and attitude, which the pilot can activate with the push of a button, was designed as an outer loop controller. Everything described above, with the exclusion of rigid body dynamics, was implemented inside a Simulink scheme.

Rigid body dynamics is simulated instead by the Bullet PE. Bullet has an internal fixed sample time of $(1/60\ \text{s})$. This means that, even if we try to increase the frequency at which the simulator works, this only generates a fictitious upsampling. Thus, 60 Hz represents, at the current moment, until V-Rep would allow to change it, a constant of the simulator.

Fig. 6 shows how the ROV environment appears from an external viewpoint; the ROV operator station shows instead the view from a camera mounted onboard the ROV.

Three different simulated missions were run with the aim of highlighting the simulator features and put it to stress: pick and release of a heavy object, fast manipulation, and sudden release of a heavy object, crash of the ROV against a fixed structure. The following describes in more detail the three missions that were conducted and shows relevant signals that highlight how the simulator "handles" these situations.

Since the purpose of this article is to show a simulation approach rather than to present and validate the simulator of a specific vehicle, a comparison between simulation and experimental data is not presented in this article. In other words, assuming valid the underwater vehicle dynamics represented by

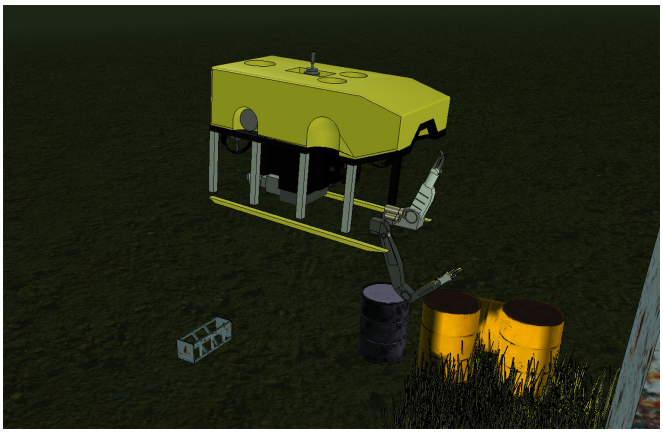

Fig. 6. ROV simulation environment as seen in V-Rep.

(32), we may state that comparing simulated data with real-world data will "only" validate the numerical model of the specific underwater vehicle. As a matter of fact, virtually the entire scientific literature on underwater vehicle modeling assumes these equations valid, including many different research papers where a comparison between real and simulated underwater vehicles, in scale or not, is performed (see, for instance, [53]–[55]). This article is intended to validate what we called a "hybrid" simulator, also by means of the accuracy analysis of Lemma 2, to show that the reliability of contact and physical interaction simulation of existing PEs can be joined, without loss of simulation accuracy, with the flexibility of Simulink for simulation of many different vehicle features, such as specific underwater drag models, sensors, actuators, and any kind of controllers.

A. Scenario 1: Pick and Release

The ROV is started a few meters away from the object to be picked (a metallic tabular frame; weight: 400 kg; weight in water around 100 kg). The pilot has to move and orient the ROV to get the object in reach of the manipulator. Then, the user has to put the ROV in position hold mode and use the manipulator to grab it. Finally, the pilot has to lift it by commanding a vertical velocity for the ROV body.

Fig. 7 shows the time histories of relevant variables for the sequence described above: depth, vertical force commanded by the control system, pitch angle, and pitch rate (roll and yaw angles are less affected than pitch given the position of the manipulator). Also indicated are the instants at which the operator closes the manipulator gripper (around time = 12 s) and when the gripper is opened again letting the object fall (around time = 22.9 s). Since the grasp of the object is not firm, it oscillates lightly when the ROV moves and lifts it. These oscillations can be easily seen in the pitch rate and the vertical force graphs. Also the increase of the vertical force needed to regulate the depth can be seen during the lift transient (approximately in the window t = 12–17 s) and when the depth is kept constant (the window time = 17–22.9 s). Moreover, the transient in all variables can be noticed when the object is released (around $t = 23$ s).

B. Scenario 2: Fast Manipulation

In the second scenario, the pilot has to grab the same object as in Scenario 1, then he/she has to activate a predefined sinusoidal trajectory of the end effector through the push of a button: The ROV will hold its position, and the manipulator will start a sinusoidal motion of the end effector of amplitude 40 cm and with maximum velocity of about 10 cm/s for a total of four periods. Then, the object is released, and the oscillatory motion is activated again.

Fig. 8 shows the portion of the mission where the sinusoidal trajectories of the manipulator are activated. The figure starts (around time = 40 s) with the object grasped by the manipulator. Note that, even in the first 10 s, the pitch rate, although small, is not perfectly constant due to the small oscillations of the object hanging from the gripper and to the simulation of contact force (between the object and the gripper). Also the pitch moment M reflects that: In addition, the pitch moment is sensibly high due to the need to keep the pitch angle constant while holding the object; and the pitch moment value is, as expected, approximately equal to the weight in water of the proposed object times the distance between the gripper and the ROV center of mass (around 2.5 m in this simulation). It can be noticed that, due to the presence of the attitude control system, the oscillatory manipulator motion (from time = 50 s to time = 80 s) has little effect on the ROV body variables. The torque τ_1 generated by the joint motor at the manipulator shoulder is also shown. When at time = 88 s, the gripper is opened and the object grasp is lost, there is a large transient on all variables: Thanks to the action of the attitude control system, the pitch moment falls back rapidly to the value it had before grasping the object, the pitch rate shows a corresponding transient, and also the end-effector position shows a little transient due to the sudden change in the joint moments needed to hold the manipulator in place.

When from time = 91 s to time = 121 s, the sinusoidal motion is activated again, the joint torque of the manipulator appears much lower than before since the inertia of its load is greatly decreased.

C. Scenario 3: Crash Against a Fixed Structure

During ROV operations, collision avoidance is a critical task, but, from the simulator standpoint, simulating reliably a collision between the ROV, the manipulator, and fixed structures or the sea bottom is even a greater challenge. This simulation presents the ROV simulator behavior when a collision happens against a fixed unmovable structure.

The ROV is initially moving at 1 m/s along its X body axis until, around time = 7.5 s, it hits a pillar. The hit point is on the front left of the ROV body. The ROV control system is automatically powered off at the hit time to observe the transient behavior of the ROV after the crash. Fig. 9 shows the ROV surge velocity, yaw rate, and roll, pitch, and yaw angles.

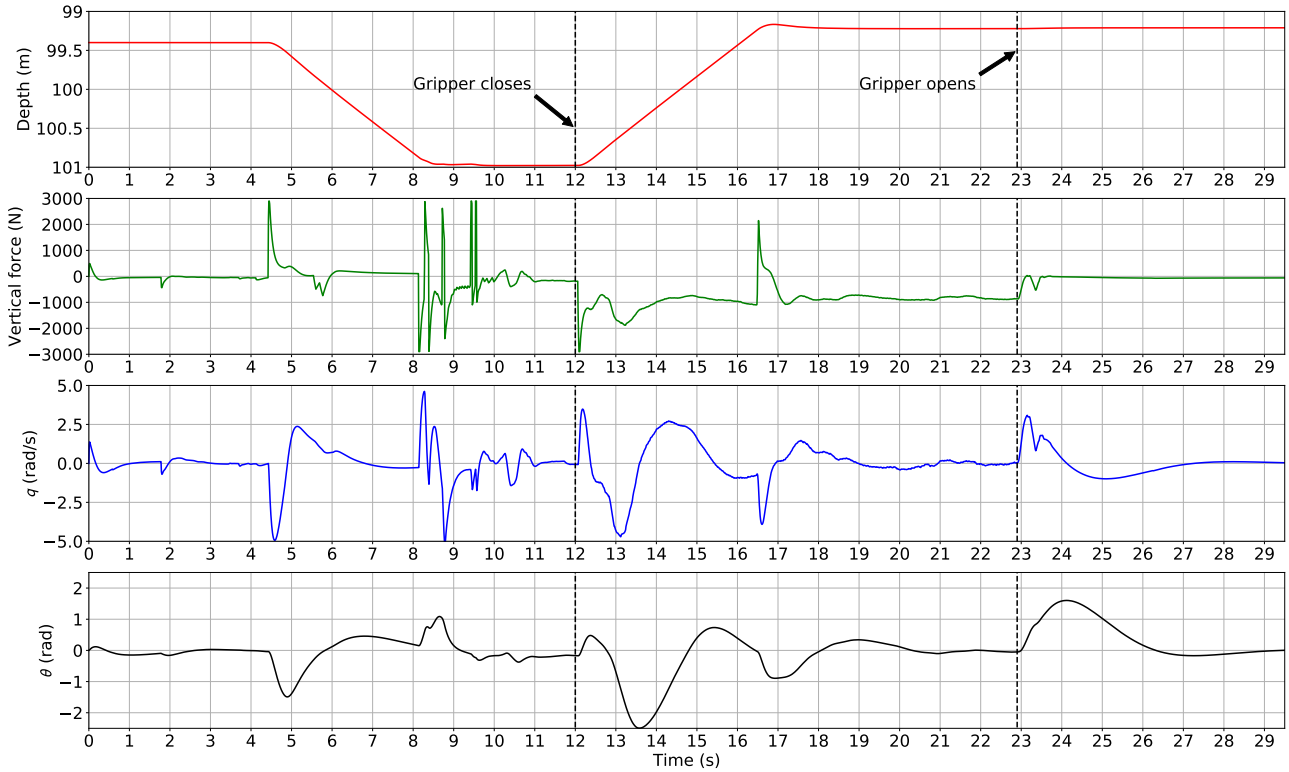

Fig. 7. Sample mission: pick and release of a heavy object.

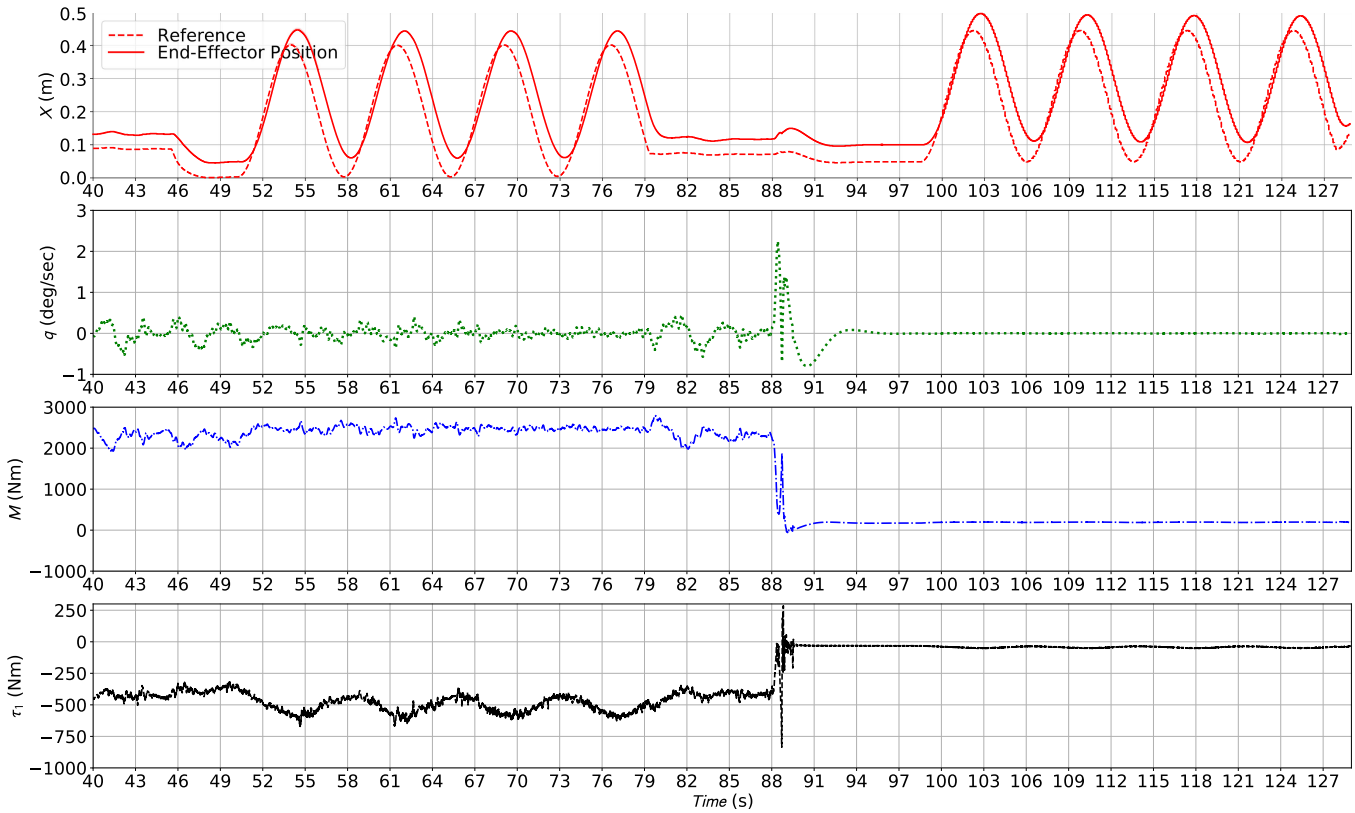

Fig. 8. Sample mission: fast object manipulation.

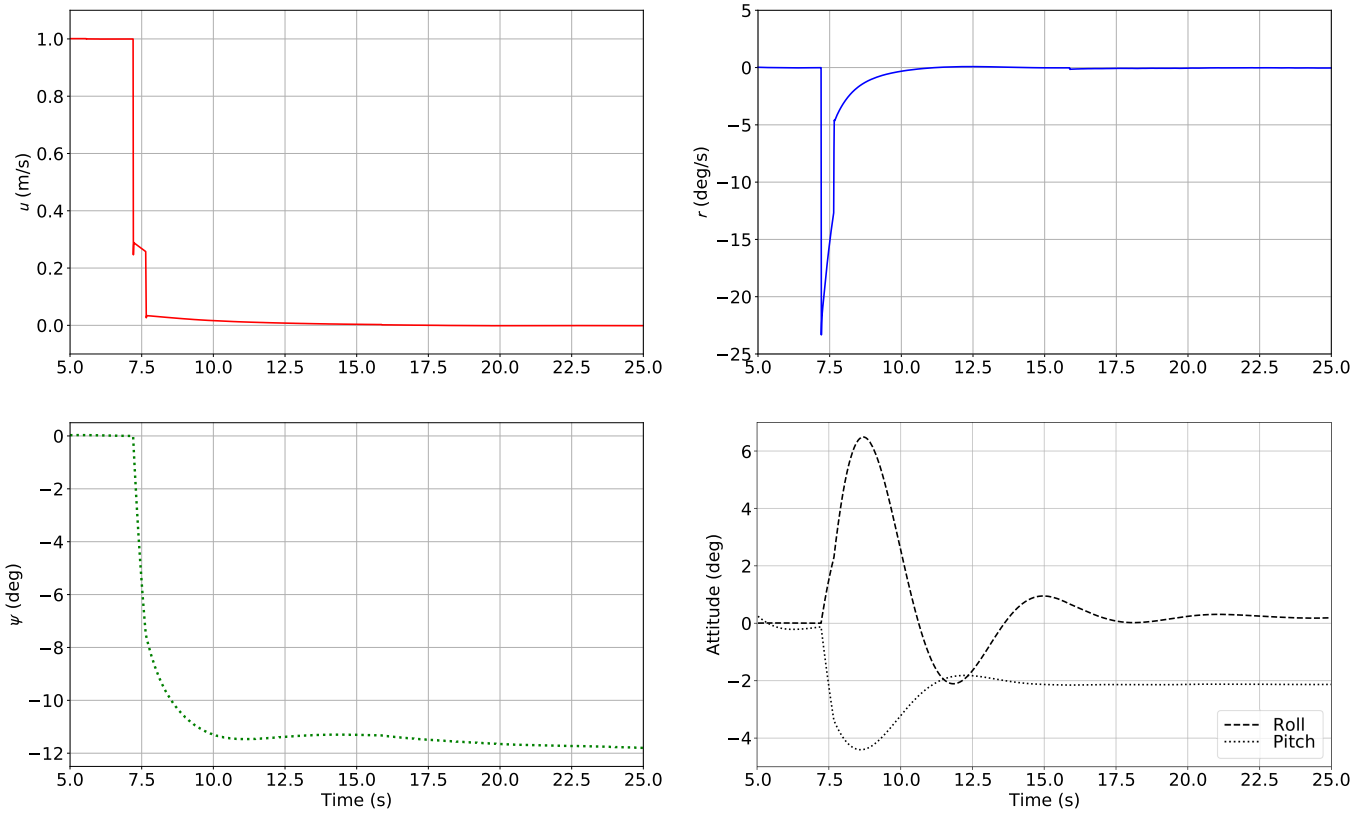

Fig. 9. Sample mission: crash against a fixed structure.

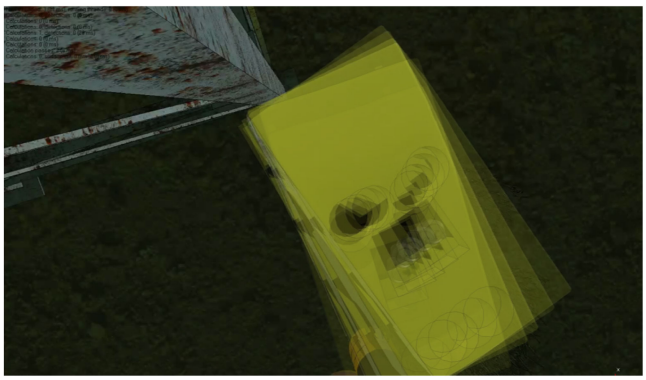

Fig. 10. Crash mission: stroboscopic view of the crash sequence.

The effect of the collision is clearly noticeable as an almost immediate drop of surge velocity toward 0 m/s. Velocity does not null immediately because the hit point is not perfectly centered. As a result, as expected, the yaw rate increases immediately and the vehicle starts turning to the left. The vehicle then settles to a constant attitude and zero velocity due to the hydrodynamic drag. The pitch angle does not settle to zero since the buoyancy center is little behind the center of mass in the simulated ROV. Fig. 10 shows a stroboscopic view of the crash sequence.

V. Conclusion

This article presented a hybrid simulation approach for underwater vehicles with one or more manipulators to be used to simulate robotic intervention mission. Mathematical foundations in support of the proposed simulation architecture were provided together with an analysis of the simulation accuracy connected with the simulation of added masses, proving that this can be very small in typical conditions. A sample implementation was proposed using both free open-source and commercial simulation softwares showing that the amount of C++ coding that is necessary is very limited. Finally, simulation of three operational scenarios was presented to validate qualitatively the simulator performance.

It is important to emphasize that the validity, applicability, and the theoretical results on simulation accuracy of the proposed hybrid simulation approach are independent of the specific software packages that were used for the development and the demonstration of the concept. As a matter of fact, the proposed hybrid simulation approach could have been demonstrated with a different combination of techniques for multibody dynamics simulation and the addition of hydrodynamics effects and GNC components.

References

[1] W. E. Landay III, M. LeFever, R. Spicer, R. Levitre, and S. Tomaszeski, "The navy unmanned undersea vehicle (UUV) master plan," U.S. Dept. Navy, Washington, DC, USA, US Navy Rep., 2004. [Online]. Available: https://www.navy.mil/navydata/technology/uuvmp.pdf

[2] L. Voss, "Unmanned systems face a changing market," *COTS J.*, vol. 10, no. 10, pp. 20–27, 2008.

[3] D. Lillo *et al.*, "Advanced ROV autonomy for efficient remote control in the DexROV project," *Mar. Technol. Soc. J.*, vol. 50, no. 4, pp. 67–80, 2016.

[4] P. Abreu *et al.*, "Widely scalable mobile underwater sonar technology: An overview of the H2020 WiMUST project," *Mar. Technol. Soc. J.*, vol. 50, no. 4, pp. 42–53, 2016.

[5] L. Brun, "ROV/AUV trends: Market and technology," *Mar. Technol. Rep.*, vol. 5, no. 7, pp. 48–51, 2012.

[6] S. Shaker, "Preparing for the future of unmanned undersea surface vehicles." 2013. [Online]. Available: http://www.slideshare.net/sshaker/future-uuv-usv-aug-2013

[7] IndustryARC, "Unmanned underwater vehicles market." 2015. [Online]. Available: http://www.slideshare.net/IndustryARC/unmanned-underwater-vehicles-mark et-20152020

[8] G. Antonelli, "Workshop on marine robotics: Vehicle-manipulator systems." 2016. [Online]. Available: http://www.slideshare.net/GianlucaAntonelli/workshop-on-marine-robotics-vehiclemanipulator-systems

[9] I. Schjølberg and I. B. Utne, "Towards autonomy in ROV operations," *IFAC-PapersOnLine*, vol. 48, no. 2, pp. 183–188, 2015.

[10] A. L. Alexander, T. Brunyé, J. Sidman, and S. A. Weil, "From gaming to training: A review of studies on fidelity, immersion, presence, and buy-in and their effects on transfer in PC-based simulations and games," DARWARS Training Impact Group, vol. 5, pp. 1–14, 2005.

[11] J. Craighead, R. Murphy, J. Burke, and B. Goldiez, "A survey of commercial & open source unmanned vehicle simulators," in *Proc. IEEE Int. Conf. Robot. Autom*, 2007, pp. 852–857.

[12] D. Cook, A. Vardy, and R. Lewis, "A survey of AUV and robot simulators for multi-vehicle operations," in *Proc. IEEE/OES Auton. Underwater Veh.*, 2014, pp. 1–8.

[13] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, "UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation," in *Proc. OCEANS 2016 MTS/IEEE Monterey*, 2016, doi: 10.1109/OCEANS.2016.7761080.

[14] M. Prats, J. Pérez, J. J. Fernández, and P. J. Sanz, "An open source tool for simulation and supervision of underwater intervention missions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 2577–2582.

[15] P. Xie *et al.*, "Aqua-Sim: An NS-2 based simulator for underwater sensor networks," in *Proc. OCEANS Conf.*, 2009, pp. 1–7.

[16] S. K. Choi, S. Menor, and J. Yuh, "Distributed virtual environment collaborative simulator for underwater robots," in *Proc. IEEE/ RSJ Int. Conf. Intell. Robots Syst.*, 2000, vol. 2, pp. 861–866.

[17] CM Labs, "Vortex FMC schilling robotics ROV pilot simulator." [Online]. Available: http://www.f-e-t.com/products/drilling-and-subsea/subsea-technologies/v max-project-simulator-software, Accessed: 2019.

[18] Forum Energy Technologies, "Vmax simulator." [Online]. Available: http://www.f-e-t.com/products/drilling-and-subsea/subsea-technologies/v max-project-simulator-software, Accessed: 2019.

[19] T. I. Fossen, *Guidance and Control of Ocean Vehicles*. Hoboken, NJ, USA: Wiley, 1994.

[20] The Society of Naval Architects and Marine Engineers, Technical and Research Bulletin No. 1–5, "Nomenclature for treating the motion of a submerged body through a fluid," *Soc. Nav. Archit. Mar. Eng., Tech. Res. Bull.*, pp. 1–5, 1950. [Online]. Available: https://www.itk.ntnu.no/fag/TTK4190/papers/Sname%201950.PDF

[21] O. Faltinsen, *Sea Loads on Ships and Offshore Structures*, vol. 1. Cambridge, U.K.: Cambridge Univ. Press, 1993.

[22] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," *J. Appl. Mech.*, vol. 77, no. 2, pp. 215–221, 1955.

[23] G. Antonelli, *Underwater Robots*, vol. 96. New York, NY, USA: Springer, 2013.

[24] L.-W. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. Hoboken, NJ, USA: Wiley, 1999.

[25] R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," *Int. J. Robot. Res.*, vol. 2, no. 1, pp. 13–30, 1983.

[26] B. Mirtich and J. Canny, "Impulse-based simulation of rigid bodies," in *Proc. Symp. Interact. 3D Graph.*, 1995, pp. 181–188.

[27] G. Gilardi and I. Sharf, "Literature survey of contact dynamics modelling," *Mechanism Mach. Theory*, vol. 37, no. 10, pp. 1213–1239, 2002.

[28] P. Song, "Modeling, analysis and simulation of multibody systems with contact and friction," Ph.D. dissertation, Mech. Eng. Appl. Mech., Univ. Pennsylvania, Philadelphia, PA, USA, 2002.

[29] S. Giovanni and K. Yin, "LocoTest: Deploying and evaluating physics-based locomotion on multiple simulation platforms," in *Proc. Int. Conf. Motion Games*, 2011, pp. 227–241.

[30] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 4397–4404.

[31] I. Metrikin, A. Borzov, R. Lubbad, and S. Løset, "Numerical simulation of a floater in a broken-ice field: Part II—Comparative study of physics engines," in *Proc. ASME 31st Int. Conf. Ocean, Offshore Arctic Eng.*, 2012, pp. 477–486.

[32] S. Ivaldi, J. Peters, V. Padois, and F. Nori, "Tools for simulating humanoid robot dynamics: A survey based on user feedback," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 842–849.

[33] D. W. Marhefka and D. E. Orin, "Simulation of contact using a nonlinear damping model," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1996, vol. 2, pp. 1662–1668.

[34] D. E. Stewart and J. C. Trinkle, "An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction," *Int. J. Numer. Methods Eng.*, vol. 39, no. 15, pp. 2673–2691, 1996.

[35] A. Boeing and T. Bräunl, "Evaluation of real-time physics simulation systems," in *Proc. 5th Int. Conf. Comput. Graph. Interact. Techn. Australia Southeast Asia*, 2007, pp. 281–288.

[36] A. Enzenhöfer, S. Andrews, M. Teichmann, and J. Kövecses, "Comparison of mixed linear complementarity problem solvers for multibody simulations with contact," in *Proc. 14th Workshop Virtual Reality Interact. Phys. Simul.*, 2018, pp. 11–20.

[37] C. Lacoursière, M. Linde, Y. Lu, and J. Trinkle, "A framework for data exchange and benchmarking of frictional contact solvers in multibody dynamics," in *Proc. ECCOMAS Thematic Conf. Multibody Dyn.*, 2015, pp. 2–3.

[38] R. Smith, "Ode: Open dynamics engine." [Online]. Available: http://www.ode.org, Accessed: 2019.

[39] E. Coumans, "Bullet physics engine." [Online]. Available: http://bulletphysics.org, Accessed: 2019.

[40] J. Jarez and A. Suero, "Newton game dynamics," Opensource Phys. Engine, 2008.

[41] CM Labs, "Vortex simulator." [Online]. Available: http://www.cm-labs.com/. Accessed: 2019.

[42] I. Desberg, "Vortex simulator," US Patent App. 29/378 867, 2012.

[43] D. Baraff, "Fast contact force computation for nonpenetrating rigid bodies," in *Proc. SIGGRAPH*, 1994, pp. 23–34.

[44] O. Kermorgant, "A dynamic simulator for underwater vehicle-manipulators," in *Proc. Int. Conf. Simul., Model., Program. Auton. Robots*, 2014, pp. 25–36.

[45] E. H. Henriksen, I. Schjlberg, and T. B. Gjersvik, "UW MORSE: The underwater modular open robot simulation engine," in *Proc. IEEE/OES Auton. Underwater Veh.*, 2016, pp. 261–267.

[46] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular open robots simulation engine: MORSE," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 46–51.

[47] H. Mitsuyasu *et al.*, "Observation of the power spectrum of ocean waves using a cloverleaf buoy," *J. Phys. Oceanogr.*, vol. 10, no. 2, pp. 286–296, 1980.

[48] J. Battjes, T. Zitman, and L. Holthuusen, "A reanalysis of the spectra observed in JONSWAP," *J. Phys. Oceanogr.*, vol. 17, no. 8, pp. 1288–1295, 1987.

[49] S. Beji and J. Battjes, "Experimental investigation of wave propagation over a bar," *Coastal Eng.*, vol. 19, no. 1/2, pp. 151–162, 1993.

[50] M.-C. Fang, C.-S. Hou, and J.-H. Luo, "On the motions of the underwater remotely operated vehicle with the umbilical cable effect," *Ocean Eng.*, vol. 34, no. 8/9, pp. 1275–1289, 2007.

[51] S. Rodríguez, M. Peña, and R. Ramírez, "Dynamic control design LQR PI vectorial of remotely operated underwater vehicle," in *Proc. III Int. Congr. Eng. Mechatronics Autom.*, 2014, pp. 1–4.

[52] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*. New York, NY, USA: Springer, 2012.

[53] N. Q. Hoang and E. Kreuzer, "Adaptive PD-controller for positioning of a remotely operated vehicle close to an underwater structure: Theory and experiments," *Control Eng. Pract.*, vol. 15, no. 4, pp. 411–419, 2007.

[54] H.-H. Chen, "Vision-based tracking with projective mapping for parameter identification of remotely operated vehicles," *Ocean Eng.*, vol. 35, no. 10, pp. 983–994, 2008.

[55] R. B. Ambar, S. Sagara, and K. Imaike, "Experiment on a dual-arm underwater robot using resolved acceleration control method," *Artif. Life Robot.*, vol. 20, no. 1, pp. 34–41, 2015.

Matteo Razzanelli received the B.Sc. degree in computer science engineering, the M.Sc. degree in robotics and automation engineering, and the Ph.D. degree in information technology from the University of Pisa, Pisa, Italy, in 2011, 2015, and 2019, respectively.

He was a Research Fellow in a project funded by SAIPEM SpA in 2014. He was involved in the European H2020 project WiMUST "Widely Scalable Mobile Underwater Sonar Technology." His works include guidance, navigation, and control of aerial, terrestrial, and marine unmanned vehicles, visual navigation systems for multirotor vehicles, distributed control algorithms, and model predictive control.

Simona Casini received the B.Sc. degree in mechanical engineering from the University of Genoa, Genova, Italy, in 2011, and the M.Sc. degree in robotics and automation engineering from the University Pisa, Pisa, Italy, in 2015.

She was a Research Fellow for a project funded by SAIPEM SpA in 2014, and a Robotic Engineer with the Italian Institute of Technology from 2016 to 2019. She is currently a Software Developer Engineer for a CNC machine industrial simulator.

Mario Innocenti received the Laurea degree in aeronautical engineering from the University of Pisa, Pisa, Italy, in 1977, and the Ph.D. degree in aeronautics and astronautics from Purdue University, West Lafayette, IN, USA, in 1982.

From 1982 to 1992, he was a faculty member with the Department of Aerospace Engineering, Auburn University, Auburn, AL, USA. Since 1993, he has been with the University of Pisa, where he is currently a Full Professor in automatic control with the Department of Information Engineering. He was a member of various NATO working groups, a Visiting Scientist at several U.S. laboratories, and has performed sponsored research at the national and international levels. He has published more than 230 scientific publications and tutored 19 Ph.D. students. His research interests are in the areas of control theory, unmanned systems guidance and control, and rendezvous problems in space under a three-body perturbation.

Prof. Innocenti, for 2008–2009, was a recipient of a senior research associate fellowship by the U.S. National Research council to perform research at the Munitions Directorate of Air Force Research Laboratory, Eglin AFB, FL, USA.

Lorenzo Pollini (M'01) received the Laurea degree in computer engineering (*cum laude*) and the Ph.D. degree in robotics and industrial automation from the University of Pisa, Pisa, Italy, in 1997 and 2000, respectively.

He is currently an Associate Professor in automatic control with the University of Pisa, Pisa, Italy. He is the Deputy Director for the Master Degree in Robotics and Automation with the University of Pisa, and teaches the courses of guidance and navigation systems, digital control systems, and control systems theory. He authored more than 100 scientific publications, including three book chapters, and he is also the coauthor of a chapter of the *Encyclopedia of Aerospace Engineering* (Wiley, 2010). His research interests include guidance and navigation systems, vision-based control, haptic support systems, fuzzy, neural, and nonlinear adaptive controls, and real-time dynamic systems simulation with specific application to unmanned systems.

Dr. Pollini is a member of the IEEE Systems Man and Cybernetics Society—Shared Control Technical Committee and the AIAA Guidance Navigation and Control Technical Committee. He is currently an Associate Fellow of AIAA.