Gazebo Fluids: SPH-based simulation of fluid interaction with articulated rigid body dynamics

Emmanouil Angelidis, Jan Bender, Jonathan Arreguit, Lars Gleim, Wei Wang, Cristian Axenie, Alois Knoll, Auke Ijspeert

***Abstract*—Physical simulation is an indispensable component of robotics simulation platforms that serves as the basis for a plethora of research directions. Looking strictly at robotics, the common characteristic of the most popular physics engines, such as ODE, DART, MuJoCo, bullet, SimBody, PhysX or RaiSim, is that they focus on the solution of articulated rigid bodies with collisions and contacts problems, while paying less attention to other physical phenomena. This restriction limits the range of addressable simulation problems, rendering applications such as soft robotics, cloth simulation, simulation of viscoelastic materials, and fluid dynamics, especially surface swimming, infeasible. In this work, we present Gazebo Fluids, an open-source extension of the popular Gazebo robotics simulator that enables the interaction of articulated rigid body dynamics with particle-based fluid and deformable solid simulation. We implement fluid dynamics and highly viscous and elastic material simulation capabilities based on the Smoothed Particle Hydrodynamics method. We demonstrate the practical impact of this extension for previously infeasible application scenarios in a series of experiments, showcasing one of the first self-propelled robot swimming simulations with SPH in a robotics simulator.**

I. Introduction

Physics-based simulation lies at the core of many different technologies and research directions, covering a wide range of applications from video games over Reinforcement Learning (RL) to robotics. Currently no single does-it-all solution that encompasses all the different requirements for different workflows exists. The vast landscape of general purpose robotics simulation solutions, such as MuJoCo [1], bullet, RaiSim [2], PhysX, and Webots [3], offers a wide range of physics models capturing realistic dynamics. A somewhat crude categorization of physical simulators could differentiate between those that focus on rendering and physical appearance, e.g., physics for game engines prioritizing visual fidelity, and on the other side simulators for which physical accuracy is crucial. The latter is particularly important for robotics workflows, typically found in simulators such as Gazebo [4], CoppeliaSim / V-Rep [5]. A physics engine that behaves as close to reality as possible, is crucial to reduce the sim2real gap, turning such simulators into valuable tools in the hands of the robotics community.

Historically many different approaches to the problem of physical simulation have been taken. Approaches stemming from continuum mechanics, e.g., Finite Element Method (FEM), Finite Volume Method (FVM), and Finite Difference Method (FDM), follow an energy-based derivation of the equations of solid and fluid dynamics. In contrast, robotics simulators and their underlying physics engines typically rely on the simplified treatment of moving bodies as fully rigid, i.e. there is no deformation of the materials, and rely on the solution of the equations of motion starting from a constraint-based formulation. Due to the distinct nature of the two approaches, it is often not feasible to couple them into a unified solution. This is in fact an active field of research [6]–[8]

The limiting factor of most continuum mechanics-based approaches that keeps them from wide adoption in the robotics community is that they are generally more computationally intensive compared to rigid body dynamics solvers. Most modern physics engines used in robotic simulation, such as ODE, DART [9], MuJoCo, bullet, SimBody [10], PhysX or RaiSim, instead formulate the problem of articulated rigid body dynamics via a constraint-based formulation, and use simple geometrical shapes for the computation of forces (contacts, collisions etc.). This usually limits the range of problems that they solve to a category of problems that ignores other physical phenomena. This makes them unsuitable for applications such as soft robotics, cloth simulation, simulation of viscoelastic materials, or fluid dynamics. For such use-cases the norm is to augment an existing simulator with custom-built physics, albeit with the loss of general applicability. One problem, difficult to tackle with current solvers, is open-world simulation, where freely moving bodies interact with their environment but are not spatially limited to a predefined computational domain. Such cases can exhibit complex dynamics with a wide range of temporal and spatial features e.g. a robot transitioning between different media, from ground to water. The current approaches assume the analysis of these dynamics for the extraction of physically important features or modes [11]. Yet, machine learning and data-driven approaches have lately offered new methods to tackle such problems in a flexible way and at scale [12], [13].

In this work, we open up a new world of simulating rigid body with SPH-based fluids and solids interactions in robotics. Starting with the community standard Gazebo simulator, by integrating into it a Smoothed Particle Hydrodynamics (SPH)-based framework that enables the interaction of articulated rigid body dynamics with particle-based fluid and deformable solid dynamics[1]. This tool supports simulating many different categories of robotics problems, such as robots that manipulate fluids with their end effectors, swimming robots, robots that

E. Angelidis was with foritss GmbH, Munich, Germany. He is now with the Huawei Technologies Munich Research Center, Munich Germany, corresponding author (e-mail: manosagelidis@gmail.com)

J. Bender is with RWTH Aachen University, Aachen, Germany (e-mail: bender@cs.rwth-aachen.de)

J. Arreguit and A. Ijspeert are with the École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

L. Gleim, C. Axenie and W.Wang are with Huawei Technologies Munich Research Center, Munich, Germany

A. Knoll is with the Technical University of Munich, Munich, Germany

[1]https://bitbucket.org/hbpneurorobotics/splishsplash/src/master/GazeboFluidSimulator/

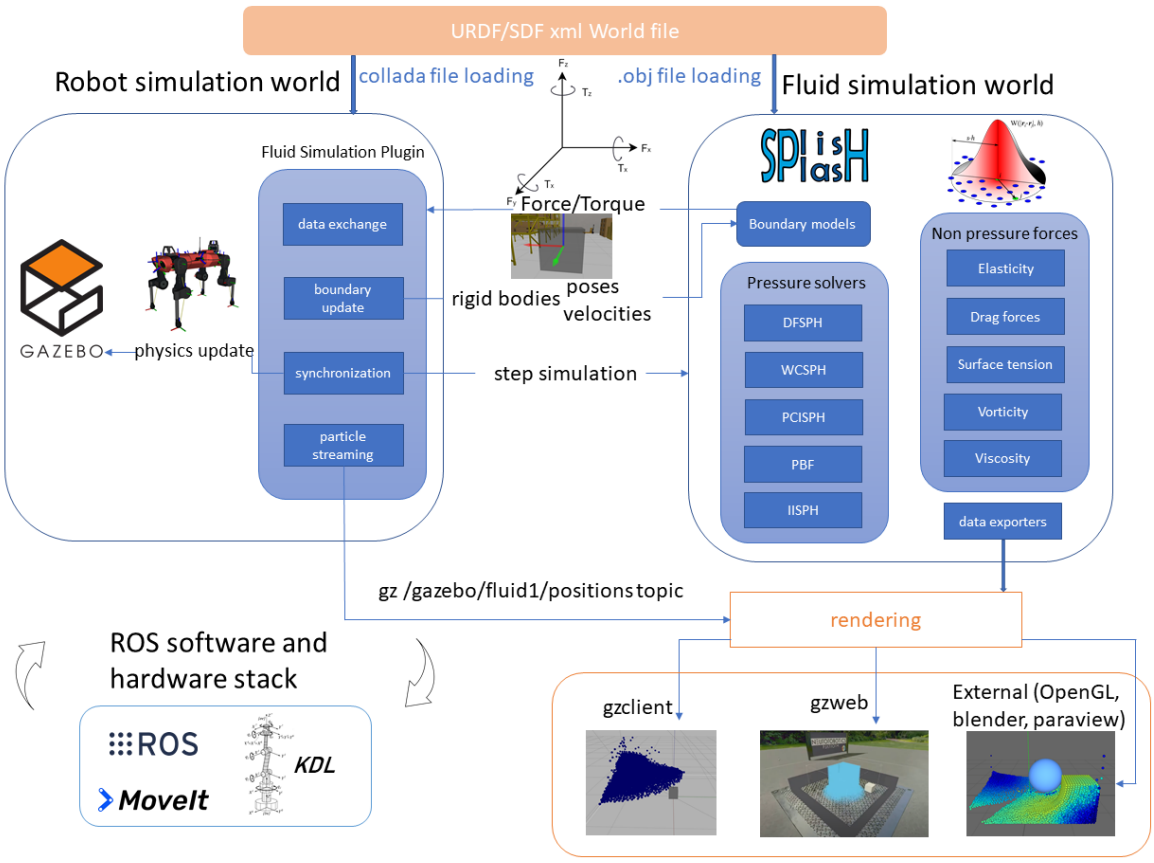

Figure 1: The Gazebo / SPlisHSPlasH plugin architecture. In the beginning of the simulation, the world file (SDF/URDF supported) is parsed, and each simulator creates its own representation of the world in their respective format. In each physical simulation step the Fluid Simulation Plugin triggers a data exchange consisting of a Force/Torque pair that expresses the fluid forces applied on every rigid body, and the poses and velocities of the rigid bodies so that the fluid simulation can localize them it its own world. The Fluid Simulation Plugin also manages the update of the boundaries representation, most commonly by updating the positions of boundary particles attached on the rigid bodies. Each simulator is performing its own computations without knowledge of the existence of the other, leading to a loose coupling between the two. This architecture means that the ROS ecosystem around Gazebo / other potential simulators can be reused without the need for further modifications. Optionally, the particles' positions can be streamed via Gazebo topics

come in contact with fluids and need to learn a locomotor policy through RL in simulation, robots used in 3D printing that emit highly viscous materials [14], medical robots, search and rescue operation robots, cloth simulation etc.

The advantage of building such an open-source tool on top of Gazebo is that it leverages its rich ecosystem (i.e. ROS support, sensors, and multiple physics engines backends). Furthermore, this initial implementation can serve as the template for the development of similar tools based on other simulators. In our implementation and evaluation, we used the open-source SPlisHSPlasH [15] solver for the fluid dynamics simulation. This framework supports multiple SPH algorithms, different boundary handling methods, as well as highly viscous and elastic materials.

The core of the tool is an efficient data exchange engine that ensures a minimal set of quantities transfer between the rigid body and the fluid simulation. Additionally, it enforces the synchronization between the two simulators. More precisely, only the positions, orientations and velocities of the rigid bodies are required at the fluid simulation level, and only the forces and torques that are applied on the rigid bodies as a result of the fluid pressure on the rigid body dynamics world are considered **(Fig. 1)**.

It is worth mentioning here that there exists an initial implementation of SPH in Gazebo through the Fluidix library which was not completed and remained closed source. In the Fluidix implementation a single SPH method and a single box collision shape were provided, as well as a collision-based force and torque boundary handling method. This work has two main contributions:

- An open-source SPH-based Gazebo plugin that enables the simulation of rigid bodies with fluids
- One of the first of its kind simulation of self-propelled robot swimming with SPH methods, capturing complex behaviors such as forward swimming and turning

II. Related Works

Recently, an overview of the features of the most widely used robotics simulators, where the lack of accurate fluids dynamics simulation is made evident was published [16]. According to their finds most of the robotics simulators either completely lack or minimally support fluid simulation. What is specifically missing is the simulation of fluids based on the Navier-Stokes equations. Popular methods for the simulation of solid bodies with fluids can be roughly divided into FEM/FVM/FDM grid-based and SPH particle-based methods. Existing efforts to couple FEM methods with

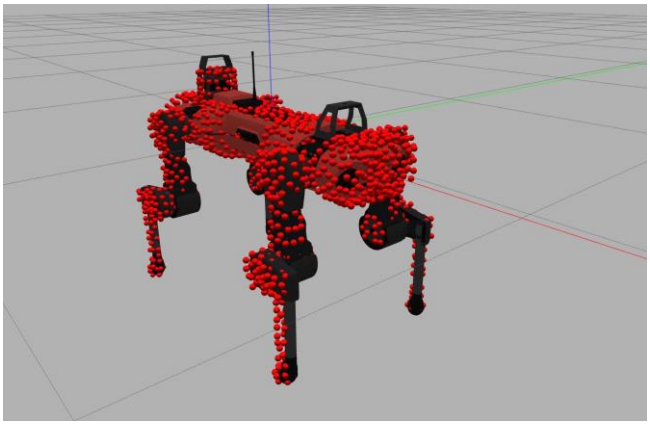

Figure 2: Example surface particle discretization of an Anymal C robot [42]. These particles are moving with the rigid body, while pressure and viscous forces are computed on them. This results to a force/torque pair that is applied on the rigid body. It is worth noting, that as is the norm in robotics simulation, the particles are sampled on top of the collision shapes' surfaces, not on the visual shapes. The framework of interaction ensures the synchronization between the two simulators, by constantly updating the locations and velocities of the boundary particles and applying the resulting fluids forces on the rigid bodies. The particles have been intentionally undersampled in order to better visualize the robot. In reality, the whole robot surface has been sampled with particles. Visualization in gzclient.

particle-based fluid dynamics can be found in the SOFA framework [17]. Simulation of rigid bodies with FLIP-based [18] fluids has been implemented in Blender with Mantaflow [19], though it should be noted that FLIP is a method that requires both an Eulerian grid as well as particles. Fluid simulation with the Position Based Fluids (PBF) [20] method has been implemented in the PhysX framework through the Flex library. As it is shown in [21], PBF comes with its own limitations, and is generally slower compared to other methods such as Divergence Free SPH (DFSPH) [21]. Perhaps the most closely related prior work to our approach is found in Project Chrono, where they couple rigid bodies with the DualSPHysics [22] Weakly Compressible SPH (WCSPH) [23] solver. One limitation of this approach is that WCSPH is only conditionally stable and depends strongly on the correct choice of the stiffness parameter of the solver that is difficult to tune. The lack of ROS support and sensor support in Project Chrono make the support of generic robotic workflows more complex than in Gazebo.

Some of the widely used robotics simulators, i.e. Webots, Gazebo, CopelliaSim, support basic fluids simulation albeit based on simple hydrodynamics. In Gazebo a plugin has been implemented that simulates fluid forces based on the relative velocity between the fluid and the robot bodies[2], as well as a plugin that simulates buoyancy and lift/drag forces[3]. The same principle of fluid forces proportional to the relative velocity between the fluid and the rigid bodies is also applied in Webots[4]. That can indeed be useful for applications where a crude approximation of the fluid forces is enough.

SPH simulation is particularly useful for the simulation of bodies swimming on the surface of fluids, a use-case that grid-based methods cannot easily cope with. Self-propelled swimming with SPH has been showcased before, first by Kajtar and Monaghan where they show three linked ellipses moving within a fluid surface [24]. In 2008, Hieber et al. [25], presented an Immersed Boundary Method within SPH and showcased self-propelled fish-like anguilliform swimming with it. More recently, Sun et al [26] presented a self-propelled fish-like swimming simulation with the δ^+-SPH variant. To our knowledge, there have not been publications showcasing self-propelled robots with SPH.

With this work a rich ecosystem of robot models, sensors, actuators is made available to the robotics community. We believe that this will open the way for new categories of simulations of interest to different robotics communities, e.g. for the simulation of amphibious robots.

III. Method

A. Summary of the interaction framework

The implementation of the Fluid Simulation Plugin is based on the generic world plugin mechanism of Gazebo, that allows the dynamically linking of C++ libraries. During the initialization phase, all the physical properties of the models, i.e. viscosity parameters, density, inertial tensors, collision shapes are loaded, and the particles' positions are initialized (**Alg. 1).** In every physical simulation step, the updated particles positions are computed, as well as the forces and torques that are applied on each rigid body, leading to a loose coupling between the two simulators. As part of the synchronization method, each simulator can be run freely with its own timestep, preventing unnecessary waiting times, exchanging information at prescribed intervals. It is important to note that in the fluid simulation world the rigid body surface is sampled with particles that serve as boundary particles (**Fig. 2**), according to the method of Akinci et al. [27]. For every rigid body the torques and forces are computed in the appropriate frame and applied along with the rest of the simulated forces (e.g. contact, collision, gravity forces). The simulation of different phenomena, i.e. highly viscous and elastic materials is feasible through the definition of different material properties. The option of streaming the particles' positions through Gazebo topics and visualizing them in

Algorithm 1: Algorithm of the interaction between fluids and rigid bodies

Initialization:

```
LoadFluidProperties()
SetupParticlePositions()
for all rigidsInScene do
  generateBoundaryParticles()
end for
```

Simulation:

```
for all physicalTimesteps do
  updateRigidBodies()
  updateFluids()
  for all boundaryModels do
    extractFluidToRigidTorquesAndForces()
  end for
  for all rigidsInScene do
    applyFluidForceAndTorque()
    updateBoundaryParticles()
  end for
end for
```

[2] Link to open-source repository. The README contains installation instructions

[3] https://gazebosim.org/tutorials?tut=hydrodynamics&cat=physics

[4] https://cyberbotics.com/doc/reference/fluid

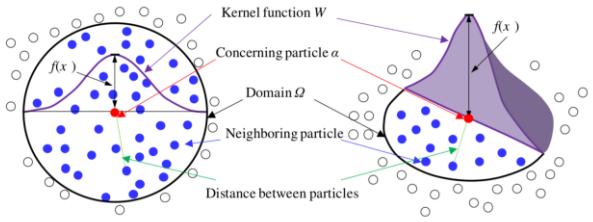

Figure 3: The smoothing kernel of SPH visualized. Here a Gaussian kernel with a support radius is shown. The value of the function f(x) at the location of the concerning particle α is computed as the sum of the values of the neighboring particles, weighted by the Gaussian kernel. The further away a particle is, the smaller the value of the Gaussian, hence the less the particle contributes to the computation. Adapted from [41].

gzclient or **gzweb** is available for prototyping. For offline rendering, the particles' positions are stored into log files at each simulation step for later visualization. The file formats that are supported are .vtk files and .partio files that can then be visualized in OpenGL, Paraview, or Blender.

B. The Smoothed Particle Hydrodynamics method

Even though a full treatise on the SPH method goes beyond the scope of this article, we summarize the main points and the simulation methods that we used in our examples. A minimal theoretical treatment is addressed and supported through our framework. For a more detailed explanation we refer the reader to [28]–[30]. It is worth noting that SPH was first formulated for the simulation of astrophysics phenomena, but soon became a general simulation method for the solution of fluid dynamics problems, later expanding into the simulation of solid dynamics [31]. It is a mesh-free Lagrangian method, which is an excellent candidate for the solution of problems with moving boundaries, such as the simulation of robots or other rigid bodies swimming on the surface of fluids, as well as deformable solids.

In SPH fluids are sampled with freely moving particles, that carry with them physical quantities, e.g. density, pressure etc. The interaction between particles relies on the mathematical construct of a weighted kernel (**Fig. 3**). The kernel weights the contribution of each particle to the computation of a physical quantity based on its distance to a particle under consideration. In order to compute a physical quantity of a particle, the weighted contributions of the quantities of the neighboring particles are summed. Even though it is useful to conceptualize particles as physical particles, it should be noted that in strict mathematical terms the particles are function sampling points.

More formally, the SPH discretization starts from the Dirac δ-identity, that states that the convolution of a continuous compactly supported function $A(x)$ with the Dirac δ-distribution is identical to $A(x)$ itself [15]

$$A(\boldsymbol{x}) = (A * \delta) = \int_{\Omega} A(\boldsymbol{x}')\delta(\boldsymbol{x} - \boldsymbol{x}')d\boldsymbol{x}' \quad (1)$$

where dx' denotes the volume integration variable over domain $\boldsymbol{\Omega}$ and $\boldsymbol{x}$ is a vector in 3D space. We can substitute the Dirac function with an approximation kernel $W(\boldsymbol{r}, h)$, e.g. Gaussian or Cubic Spline that depends on the distance $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{x}'$ and the variable h that denotes the smoothing length. The smoothing length determines how much the value of A is affected by the function values in its proximity. This leads to the approximation of a field quantity with the smoothing kernel

$$A(\boldsymbol{x}) \approx (A * \mathrm{W}) = \int A(\boldsymbol{x}')\mathrm{W}(\boldsymbol{x} - \boldsymbol{x}', h)d\boldsymbol{x}'. \quad (2)$$

The next step in the SPH formalization is to discretize this integral from analytical to its approximation by summing over discrete sampling points

$$\mathrm{A}(\mathbf{x}) \approx \int \frac{A(\boldsymbol{x}')}{\rho(\boldsymbol{x}')} W(\boldsymbol{x} - \boldsymbol{x}', h) \underbrace{\rho(\boldsymbol{x}')d\boldsymbol{v}'}_{d\boldsymbol{m}'} \quad (3)$$

$$\mathrm{A}(\mathbf{x}) \approx \sum_j A_j \frac{m_j}{\rho_j} \mathrm{W}(\boldsymbol{x} - \boldsymbol{x}', h), \quad (4)$$

where j denotes the j_{th} sampling point/particle. The main advantage of SPH starts being obvious upon discretization of differential operators that reduce to differentiations of the – by definition differentiable – kernel functions. E.g., the gradient of a function can be approximated as the sum

$$\nabla A_i \approx \sum_j A_j \frac{m_j}{\rho_j} \nabla \mathrm{W}(\boldsymbol{x_i} - \boldsymbol{x_j}, h), \quad (5)$$

where i denotes the sampling point in consideration and j the neighboring sampling points. However, this simple approximation can lead to unstable simulations [32]. Therefore, in practice improved formulations are preferred to get a more accurate approximation or to conserve linear and angular momentum, e.g., the difference formula or the symmetric formula [15]. Finally, without going into detail, the incompressible Navier-Stokes equations

$$\rho \frac{Dv}{Dt} = -\nabla \mathrm{p} + \mu \nabla^2 v + f_{ext}, \quad \nabla v = 0 \quad (6)$$

for 3D fluid flow can be solved using the SPH approximations, and a time integration scheme, e.g. semi-implicit Euler. The original SPH formulation of Gingold and Monaghan [29] has been further enhanced, with most formulations focusing on the solution of pressure.

C. Current advances in SPH

In the last two decades SPH has become a popular approach for the simulation of fluids and deformable solids and an important topic of ongoing research. One major challenge is the development of efficient and stable pressure solvers for incompressible fluids. In this field first explicit solvers for almost incompressible fluids were investigated, e.g., the Weakly-Compressible SPH solver [23]. However, these solvers are only conditionally stable and require parameter tuning. To solve these problems more stable implicit pressure solvers were developed which enforce a constant density in the fluid. Recent research in this field further improved the stability and efficiency of the pressure solvers and methods were introduced that enforce both a constant density and a divergence-free velocity field [21].

In recent years also SPH formulations for other complex materials were investigated. For example implicit solvers for

the simulation of highly viscous fluids and deformable solids were developed [33], [34]. Moreover, a wide range of physical effects were implemented like surface tension, vorticity, or air drag.

The open-source framework SPlisHSPlasH, which is used in our Gazebo plugin, implements most recent solvers for incompressible and compressible fluids, highly viscous materials, deformable solids, as well as several physical effects. Therefore, our plugin enables the simulation of a robot interacting with all these material types while considering complex physical effects.

D. Boundary handling

At the core of the interaction framework lies the application of the fluid forces/torques on the rigid bodies. In SPH the treatment of boundaries is most commonly handled with boundary particles, sampled on the surfaces of rigid bodies that move along with the bodies. Akinci et al. [27] proposed a method that supports irregular sampling of the boundary surfaces. Other researchers treat boundaries with density [35] or volume maps [36] that are computed in the beginning of the simulation. These implicit boundary methods allow a better representation of smooth surfaces. Another approach is the direct numerical simulation via the immersed boundary projection method [11] or machine learning [37].

From the coupling perspective, it is important to maintain a data structure that maps these rigid bodies / boundary models in the fluid world to rigid bodies in the robotic simulation world and update them accordingly (**Alg. 1**). In each simulation step the poses of the rigid bodies are extracted from the robot simulation and used to recompute the positions and velocities of the boundary particles **(Fig. 2)**. This ensures synchronization between the two simulators and an efficient exchange of data. At the same time, the pressure and viscous forces that are applied on the boundary particles are integrated and passed as a force/torque pair to each rigid body and applied as external forces.

E. Rendering

Depending on the number of particles simulated, it might be feasible to visualize them while the simulation runs or offline. For prototype applications with a small number of particles, a visualizer plugin for **gzclient** and an enhancement of **gzweb** for web-based visualization is provided. For larger-scale applications that support more sophisticated shaders and other 3D rendering functionality, .vtk and .partio files containing the particles' positions can be saved through the plugin and used for offline rendering.

F. Parallelization and performance considerations

SPH simulations are very good candidates for parallelization on the CPU/GPU [38], as the computational complexity of SPH increases with the number of particles. This is because in principle the interactions between all particles must be computed. In practice, neighborhood search techniques that maintain a list of the particles that are in proximity to the particle under consideration are used. This speeds up the computation significantly and facilitates the GPU and CPU parallelization. Our code specifically executes the neighborhood search both on the CPU and on the GPU.

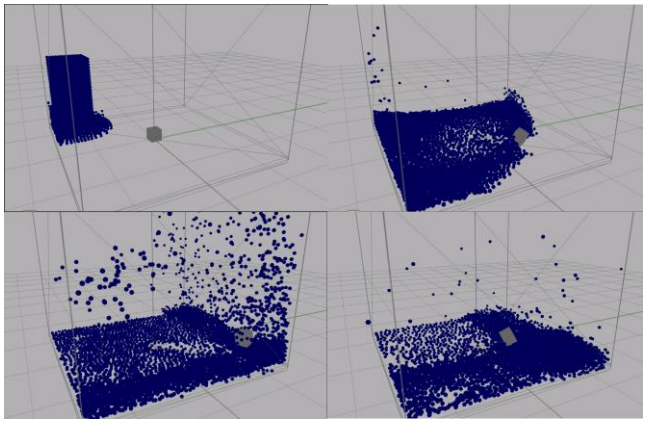

Figure 4: Consecutive timesteps of a moving box under the effect of fluid forces. In the beginning (top left) of the simulation a column of water collapses under the effect of gravity forces. Upon contact (top right), the water particles apply viscous and pressure forces on the rigid box, and it starts to move. The box subsequently collides with the rigid wall and collision forces are applied (bottom left). The effect of the collision force along with the fluid forces leads to a reverse of the box velocity towards the center of the environment. Visualization in gzclient.

IV. Experiments

We setup two experiments, one to verify the simulation, and one to showcase complex robot behaviors such as forward swimming and turning. In our setup both simulators are running on multiple CPU cores. The fluid simulation is parallelized with OpenMP and vectorized using AVX. However, the order of execution of each simulator must be kept sequential. Depending on the desirable accuracy, each simulator step can be run with multiple iterations, offering a tradeoff between accuracy and execution time. It is common in practice to let the fluid simulator run with larger timesteps and perform multiple iterations of the robot simulator, e.g. with a 1/10 ratio. One consideration that must be taken into account is the Courant-Friedrichs-Lewy (CFL) condition. It states that the fluid simulation timestep must be small enough to ensure that the distance that a particle with the max velocity can travel, is smaller than the particle size. Intuitively this means that particle should not travel more than the particle diameter per timestep.

For small prototype applications, as long as the particles number is kept small, e.g. 10.000, the simulation can be run close to real-time, e.g. with a 0.5-0.8 simulation to real-time ratio. For larger applications where mathematical accuracy is important, a higher number of particles in the order of 100.000-1.000.000 particles might be necessary to achieve the desirable accuracy. In such scenarios real-time visualization adds significant overhead and leads to a drop to 1-2 FPS, so offline visualization is preferred. These simulations take about 1-2 hours for 10 seconds of simulation time. All the simulations were performed on a laptop with i7-11800H CPU, 32 GB RAM and GeForce RTX 3060 graphics card.

A. Passive rigid body with dam break

To verify the correctness of boundary handling, the update of the rigid bodies' position and the handling of force/torque exchange, we conduct an experiment simulating a column of

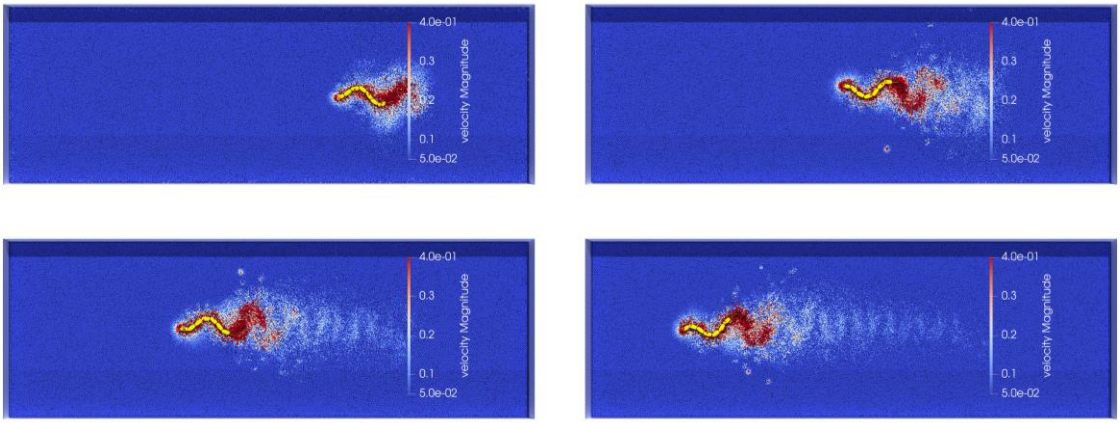

Figure 5: Simulation of an amphibot robot [40] swimming forward with a travelling wave pattern. Ordering is from left to right and from top to bottom. It is well-established that this robot locomotor policy leads to self-propelled swimming, similar to lamprey fish in the animal kingdom. The fluid simulation can capture complex phenomena, such as the formation of vortices in the posterior side of the robot. Such simulations can shed light on complex phenomena that were previously neglected in robot simulations and lead to more robust control methods. Visualization in Paraview.

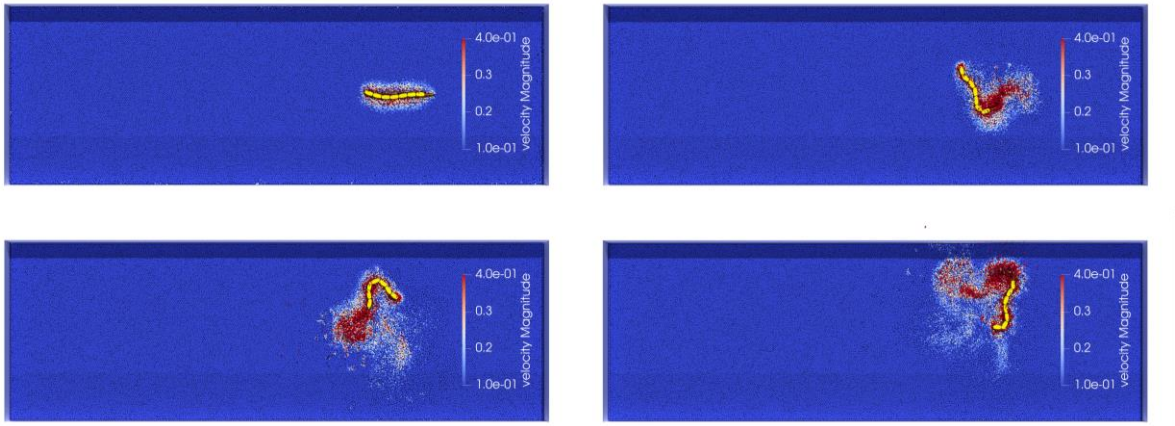

Figure 6: The interaction framework can capture more complex locomotor patterns, such as robot turning. This is achieved with a slight modification of the control method, that adds some bias into the sinusoidal generator. By modifying the control parameters, many different patterns, such as fast and abrupt change of direction, reverse swimming, swimming at different speeds can be simulated to validate and improve robot control methods

water collapsing under the effect of gravity and its dynamic interaction with a floating box (**Fig. 4**). This test verifies that the boundary handling, the update of the rigid bodies' position and the force/torque exchange is handled properly. From the point of view of the robotics simulation, the fluid forces are just another type of external force applied to the rigid body, meaning that they are seamlessly applied along with collision and gravity forces.

While no ground truth is available to verify the correctness of the simulation results -- a well-known problem of fluid-simulation -- the simulation yields believable results, attesting to the correct implementation of above-mentioned conditions.

B. Self-propelled swimming simulation

A more complex scenario, showcasing the interaction between an actuated robot and the fluid dynamics simulation can be seen in **Fig. 5-6.** In these scenarios an amphibot robot actuated with a sinusoid travelling wave controller swims forward (**Fig. 5**) and turns right (**Fig. 6**) under the effect of the fluid forces. Such simulations are usually performed with simplified drag models that fail to capture the complexity of the interaction between the rigid body and fluid dynamics. The force/torque pair applied on the rigid bodies can be extracted and used as the substrate for RL / optimization-based control methods.

This simple sinusoidal pattern, inspired by the travelling waves produced by the spinal cord of lamprey fish, can be replaced with more complex control models such as Central Pattern Generators (CPG), such as the one already implemented on the amphibot robot [39]. These types of SPH-based simulations can shed light to the complex locomotor mechanisms behind animal swimming, as already shown in [24]–[26] but also serve as a testbed for the development of robotics controllers.

One challenge that we plan to address in the future is the comparison of the simulation results with experimental data. Even though this is a general challenge in robotics simulation, as the sim2real gap is almost always present, it would strengthen the argumentation in favor of using SPH as a simulation method for robotics. It should be noted here that the maturity of SPH when it comes to engineering applications is already very high [28]. Nevertheless, which physical phenomena can be accurately captured, and how much physical accuracy is necessary for robotics applications is still an open question.

Conclusion

This work introduces the first comprehensive SPH-based open-source fluid dynamics simulator integrated with the Gazebo robotics simulator via SPlisHSPlasH. We compare our work with prior approaches, focusing particularly on the ecosystem support of Gazebo and on the richness of the SPH methods that can be explored with our implementation. We discuss important practical considerations, e.g. boundary handling, synchronization, rendering and the computational workload of the simulation. We demonstrate the performance of our extension in two separate experiments. First, we highlight the correct handling of boundary conditions, fluid/solid interaction and demonstrate the performance characteristics of the combined simulation. Second, we showcase self-propelled forward swimming and turning on the water surface, a category of robotics simulation previously infeasible to simulate within the Open Robotics ecosystem.

By implementing this open-source tool that couples Gazebo with particle-based SPH simulation, we simultaneously propose a new category of robotics simulations. Some robotics work flows that we envision include the simulation of soft robots, robots interacting with deformable solids or highly viscous and elastic materials, freely swimming robots or robots manipulating fluids with their end effectors. Large-scale simulations such as flood scenarios where search and rescue robots are simulated are a particular area of interest.

We believe that the next evolution step of the tool is to incorporate learning capabilities to support the reconstruction of fluid flow dynamics and interactions. This will enable a more realistic reconstruction due to the fact that data-driven approaches already outperform traditional approximation techniques commonly used for flow reconstruction.

Acknowledgments

This research is supported by European Union's Horizon 2020 research and innovation programme under grant agreement No. 945539 (SGA3) Human Brain Project.

References

[1] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 5026–5033. doi: 10.1109/IROS.2012.6386109.

[2] J. Hwangbo, J. Lee, and M. Hutter, "Per-Contact Iteration Method for Solving Contact Dynamics," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 895–902, Apr. 2018, doi: 10.1109/LRA.2018.2792536.

[3] O. Michel, "WebotsTM: Professional Mobile Robot Simulation," *Int. J. Adv. Robot. Syst.*, vol. 1, Mar. 2004, doi: 10.5772/5618.

[4] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Sep. 2004, vol. 3, pp. 2149–2154 vol.3. doi: 10.1109/IROS.2004.1389727.

[5] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 1321–1326. doi: 10.1109/IROS.2013.6696520.

[6] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim, "Unified particle physics for real-time applications," *ACM Trans. Graph. TOG*, vol. 33, no. 4, pp. 1–12, 2014.

[7] M. Müller, M. Macklin, N. Chentanez, S. Jeschke, and T.-Y. Kim, "Detailed Rigid Body Simulation with Extended Position Based Dynamics," *Comput. Graph. Forum*, vol. 39, no. 8, pp. 101–112, 2020, doi: 10.1111/cgf.14105.

[8] C. Gissler, A. Peer, S. Band, J. Bender, and M. Teschner, "Interlinked SPH Pressure Solvers for Strong Fluid-Rigid Coupling," *ACM Trans. Graph.*, vol. 38, no. 1, p. 5:1-5:13, Jan. 2019, doi: 10.1145/3284980.

[9] J. Lee *et al.*, "DART: Dynamic Animation and Robotics Toolkit," *J. Open Source Softw.*, vol. 3, p. 500, Feb. 2018, doi: 10.21105/joss.00500.

[10] M. Sherman, A. Seth, and S. Delp, "Simbody: Multibody dynamics for biomedical research," *Procedia IUTAM*, vol. 2, pp. 241–261, Dec. 2011, doi: 10.1016/j.piutam.2011.04.023.

[11] T. Colonius and K. Taira, "A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions," *Comput. Methods Appl. Mech. Eng.*, vol. 197, pp. 2131–2146, Apr. 2008, doi: 10.1016/j.cma.2007.08.014.

[12] M. P. Brenner, J. D. Eldredge, and J. B. Freund, "Perspective on machine learning for advancing fluid mechanics," *Phys. Rev. Fluids*, vol. 4, no. 10, p. 100501, Oct. 2019, doi: 10.1103/PhysRevFluids.4.100501.

[13] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine Learning for Fluid Mechanics," *Annu. Rev. Fluid Mech.*, vol. 52, no. 1, pp. 477–508, 2020, doi: 10.1146/annurev-fluid-010719-060214.

[14] U. Berdica, Y. Fu, Y. Liu, E. Angelidis, and C. Feng, "Mobile 3D Printing Robot Simulation with Viscoelastic Fluids," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 7557–7563. doi: 10.1109/IROS51168.2021.9636114.

[15] D. Koschier, J. Bender, B. Solenthaler, and M. Teschner, "Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids," presented at the Eurographics 2019 Tutorials, 2019. doi: 10.2312/egt.20191035.

[16] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A Review of Physics Simulators for Robotic Applications," *IEEE Access*, vol. 9, pp. 51416–51431, 2021, doi: 10.1109/ACCESS.2021.3068769.

[17] J. Allard, H. Courtecuisse, and F. Faure, "Implicit FEM and fluid coupling on GPU for interactive multiphysics simulation," in *ACM SIGGRAPH 2011 Talks*, New York, NY, USA, Aug. 2011, p. 1. doi: 10.1145/2037826.2037895.

[18] J. U. Brackbill and H. M. Ruppel, “FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions,” *J. Comput. Phys.*, vol. 65, no. 2, pp. 314–343, Aug. 1986, doi: 10.1016/0021-9991(86)90211-1.

[19] Nils Thuerey and Tobias Pfaff, *Mantaflow*. 2018. [Online]. Available: http://mantaflow.com

[20] M. Macklin and M. Müller, “Position Based Fluids,” *ACM Trans. Graph.*, vol. 32, p. 104:1, Jul. 2013, doi: 10.1145/2461912.2461984.

[21] J. Bender and D. Koschier, “Divergence-free smoothed particle hydrodynamics,” in *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, New York, NY, USA, Aug. 2015, pp. 147–155. doi: 10.1145/2786784.2786796.

[22] J. M. Domínguez *et al.*, “State-of-the-art SPH solver DualSPHysics: from fluid dynamics to multiphysics problems,” *Comput. Part. Mech.*, Mar. 2021, doi: 10.1007/s40571-021-00404-2.

[23] M. Becker and M. Teschner, “Weakly Compressible SPH for Free Surface Flows,” in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Jan. 2007, vol. 9, p. 217. doi: 10.1145/1272690.1272719.

[24] J. Kajtar and J. J. Monaghan, “SPH simulations of swimming linked bodies,” *J. Comput. Phys.*, vol. 227, no. 19, pp. 8568–8587, Oct. 2008, doi: 10.1016/j.jcp.2008.06.004.

[25] S. E. Hieber and P. Koumoutsakos, “An immersed boundary method for smoothed particle hydrodynamics of self-propelled swimmers,” *J. Comput. Phys.*, vol. 227, no. 19, pp. 8636–8654, Oct. 2008, doi: 10.1016/j.jcp.2008.06.017.

[26] P.-N. Sun, A. Colagrossi, and A.-M. Zhang, “Numerical simulation of the self-propulsive motion of a fishlike swimming foil using the δ+-SPH model,” *Theor. Appl. Mech. Lett.*, vol. 8, no. 2, pp. 115–125, Mar. 2018, doi: 10.1016/j.taml.2018.02.007.

[27] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, and M. Teschner, “Versatile rigid-fluid coupling for incompressible SPH,” *ACM Trans. Graph.*, vol. 31, no. 4, p. 62:1-62:8, Jul. 2012, doi: 10.1145/2185520.2185558.

[28] S. J. Lind, B. D. Rogers, and P. K. Stansby, “Review of smoothed particle hydrodynamics: towards converged Lagrangian flow modelling,” *Proc. R. Soc. Math. Phys. Eng. Sci.*, vol. 476, no. 2241, p. 20190801, Sep. 2020, doi: 10.1098/rspa.2019.0801.

[29] R. A. Gingold and J. J. Monaghan, “Smoothed particle hydrodynamics: theory and application to non-spherical stars,” *Mon. Not. R. Astron. Soc.*, vol. 181, no. 3, pp. 375–389, Dec. 1977, doi: 10.1093/mnras/181.3.375.

[30] M. Ihmsen, J. Orthmann, B. Solenthaler, A. Kolb, and M. Teschner, “SPH Fluids in Computer Graphics,” 2014.

[31] J. P. Gray, J. J. Monaghan, and R. P. Swift, “SPH elastic dynamics,” *Comput. Methods Appl. Mech. Eng.*, vol. 190, no. 49, pp. 6641–6662, Oct. 2001, doi: 10.1016/S0045-7825(01)00254-7.

[32] D. J. Price, “Smoothed Particle Hydrodynamics and Magnetohydrodynamics,” *J. Comput. Phys.*, vol. 231, no. 3, pp. 759–794, Feb. 2012, doi: 10.1016/j.jcp.2010.12.011.

[33] M. Weiler, D. Koschier, M. Brand, and J. Bender, “A Physically Consistent Implicit Viscosity Solver for SPH Fluids,” *Comput Graph Forum*, 2018, doi: 10.1111/cgf.13349.

[34] A. Peer, C. Gissler, S. Band, and M. Teschner, “An Implicit SPH Formulation for Incompressible Linearly Elastic Solids: Implicit Elastic SPH Solids,” *Comput. Graph. Forum*, vol. 37, Dec. 2017, doi: 10.1111/cgf.13317.

[35] D. Koschier and J. Bender, “Density maps for improved SPH boundary handling,” in *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, New York, NY, USA, Jul. 2017, pp. 1–10. doi: 10.1145/3099564.3099565.

[36] J. Bender, T. Kugelstadt, M. Weiler, and D. Koschier, “Volume Maps: An Implicit Boundary Representation for SPH,” in *Motion, Interaction and Games on - MIG ’19*, Newcastle upon Tyne, United Kingdom, 2019, pp. 1–10. doi: 10.1145/3359566.3360077.

[37] N. Erichson, L. Mathelin, Z. Yao, S. Brunton, M. Mahoney, and J. Kutz, “Shallow neural networks for fluid flow reconstruction with limited sensors,” *Proc. R. Soc. Math. Phys. Eng. Sci.*, vol. 476, p. 20200097, Jun. 2020, doi: 10.1098/rspa.2020.0097.

[38] M. Ihmsen, N. Akinci, M. Becker, and M. Teschner, “A parallel SPH implementation on multi-core CPUs,” *Comput Graph Forum*, vol. 30, pp. 99–112, Mar. 2011, doi: 10.1111/j.1467-8659.2010.01832.x.

[39] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen, “From Swimming to Walking with a Salamander Robot Driven by a Spinal Cord Model,” *Science*, vol. 315, no. 5817, pp. 1416–1420, Mar. 2007, doi: 10.1126/science.1138353.

[40] A. Crespi and A. J. Ijspeert, “AmphiBot II: An Amphibious Snake Robot that Crawls and Swims using a Central Pattern Generator,” *Proc. 9th Int. Conf. Climbing Walk. Robots CLAWAR 2006*, Jan. 2006.

[41] Z. Dai, F. Wang, Y. Huang, K. Song, and A. Iio, “SPH-based numerical modeling for the post-failure behavior of the landslides triggered by the 2016 Kumamoto earthquake,” *Geoenvironmental Disasters*, 2016, doi: 10.1186/s40677-016-0058-5.

[42] M. Hutter *et al.*, “ANYmal - a highly mobile and dynamic quadrupedal robot,” *2016 IEEERSJ Int. Conf. Intell. Robots Syst. IROS*, 2016, doi: 10.1109/IROS.2016.7758092.