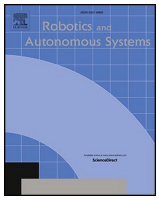
Robotics and Autonomous Systems 185 (2025) 104885

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



A smoothed particle hydrodynamics framework for fluid simulation in robotics

Emmanouil Angelidis [a,d,1,*], Jonathan Arreguit [b,1,*], Jan Bender [c], Patrick Berggold [a], Ziyuan Liu [d], Alois Knoll [a], Alessandro Crespi [b], Auke J. Ijspeert [b]

[a] Technical University of Munich, Munich, Germany
[b] École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
[c] RWTH Aachen University, Aachen, Germany
[d] Munich Research Center, Huawei Technologies GmbH, Germany

ARTICLE INFO

Keywords:
Robotics simulation
Self-propelled swimming
Locomotion
Free surface flows
Smoothed particle hydrodynamics
Multi-physics

ABSTRACT

Simulation is a core component of robotics workflows that can shed light on the complex interplay between a physical body, the environment and sensory feedback mechanisms in silico. To this goal several simulation methods, originating in rigid body dynamics and in continuum mechanics have been employed, enabling the simulation of a plethora of phenomena such as rigid/soft body dynamics, fluid dynamics, muscle simulation as well as sensor and actuator dynamics. The physics engines commonly employed in robotics simulation focus on rigid body dynamics, whereas continuum mechanics methods excel on the simulation of phenomena where deformation plays a crucial role, keeping the two fields relatively separate. Here, we propose a shift of paradigm that allows for the accurate simulation of fluids in interaction with rigid bodies within the same robotics simulation framework, based on the continuum mechanics-based Smoothed Particle Hydrodynamics method. The proposed framework is useful for simulations such as swimming robots with complex geometries, robots manipulating fluids and even robots emitting highly viscous materials such as the ones used for 3D printing. Scenarios like swimming on the surface, air-water transitions, locomotion on granular media can be natively simulated within the proposed framework. Firstly, we present the overall architecture of our framework and give examples of a concrete software implementation. We then verify our approach by presenting one of the first of its kind simulation of self-propelled swimming robots with a smooth particle hydrodynamics method and compare our simulations with real experiments. Finally, we propose a new category of simulations that would benefit from this approach and discuss ways that the sim-to-real gap could be further reduced.

One Sentence Summary:

We present a framework for the interaction of rigid body dynamics with SPH-based fluid simulation in robotics, showcase its application on self-propelled swimming robots and validate the method by comparing simulations with physical experiments.

1. Introduction

Accurate fluid simulation in robotics is relatively uncommon, as fast and freely moving bodies are difficult to simulate with classical computational fluid dynamics (CFD) simulation methods that rely on computational grids. Many simulated robots that interact with fluids, especially mobile ones, need to do so in computational domains that extend to spaces that are not predefined, and as such the a priori computation of a computational grid can be cumbersome. On the contrary, particle-based fluid dynamics simulation relies on freely moving particles that can travel to any location within a simulated domain. This allows for more flexible workflows and enables the simulation of robots swimming on the free surfaces of fluids, robots manipulating fluids or even robots emitting highly viscous materials.

In this work we show, by validating them with real world experiments, that Smoothed Particle Hydrodynamics (SPH) particle-based

* Corresponding author.
E-mail address: manosagelidis@gmail.com (E. Angelidis).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.robot.2024.104885
Received 13 June 2024; Received in revised form 29 October 2024; Accepted 3 December 2024
Available online 8 December 2024

methods present a viable tool for robotics research. The choice of SPH as a simulation method comes from the relative advantages that such methods present compared to Finite Volume (FV) / Finite Differences (FD) methods. In FV and FD as well as in the closely related Finite Elements Method (FEM) the first step is the discretization of a domain with a computational grid. As we discuss in the next section this can be cumbersome in cases where fast swimming bodies that swim on the surface and have complex geometries are involved. In SPH approaches, based on the Lagrangian formulation, the description of the flow relies on discrete particles that carry physical quantities i.e., pressure, density and kinematic quantities i.e., positions, velocities. In this approach the particles are tracked in time and space as they move, which can be conceptualized as marking a piece of a fluid with ink and then following its trajectory to observe the evolution of its movement. In contrast FD/FV approaches, based on the Eulerian formulation, rely on fixed observation points in space through which the flow passes, conceptualized as choosing a point in space and measuring the change of fluid properties as the flow evolves.

From a computational perspective, Eulerian methods rely on a computational grid, which is usually fixed in space, whereas SPH methods discretize space into particles that move freely. This is of particular importance when it comes to modelling freely moving bodies that cross or stay on the free surface of fluids, flows with different fluid phases or material discontinuities. Moreover, problems where the fluid flow is not a priori known e.g., a rush of water down a valley, waves, or splashes, are difficult if not impossible to model with other forms of fluid simulation. Such requirements are typical in robotic applications, specifically in applications where a body swims on the water surface or switches between media e.g., water and land. Another category of problems that can naturally be solved with particle-based methods is robots that manipulate fluids in various ways (transport, firefighting, handling of floods etc.).

One major limitation of FD/FV methods is that they rely on finely generated meshes, which can be computationally expensive and cumbersome to generate, especially on arbitrary, complex shapes. It is frequently necessary to update the mesh or even remesh the computational domain, resulting in algorithmic and computational complexity. Whereas the aforementioned use cases are particularly difficult to handle with FD/FV methods, particle-based methods excel at them, exactly because they do not rely on a fixed computational grid and deal well with free fluid surfaces.

From a software perspective, compared to standalone fluid dynamics solvers, SPH methods are easy to couple with other types of physical simulation, which in our case consists of articulated rigid body dynamics solvers. In terms of computational workload required the cost of SPH can be higher than the FV corresponding code as shown in [1] where they compare the performance of the two approaches for various problems. However utilizing the GPU to accelerate particle-based simulations can lead to significantly lower computational times. We refer the reader to the benchmarks conducted by the DualSPHysics that show simulations of 70 million particles on a Tesla V100 GPU in 40 h [2]. It should be noted that combinations of the two main Eulerian/Lagrangian formulations for the description of the fluid flow exist (i.e. Coupled Eulerian Lagrangian [3], Arbitrary Eulerian Lagrangian [4] and Immersed Boundary Methods [5]), but due to the complexity of coupling them with existing physics simulators, we chose to follow the SPH approach. We show a comparative summary of the characteristics of SPH simulation compared to grid-based methods in Table 1.

To address these limitations related to the applicability of fluid simulation methods in robotics, we present a unified framework of robotics and fluid simulation that enables capturing part of the complexities described above by simulating the fluid flow with the SPH [6] Lagrangian-based formulation of the 3D Navier-Stokes equations. We present how SPH methods can be integrated into existing articulated rigid body dynamics solvers that are commonly used to simulate robotics problems (in our case Gazebo). The architecture scheme of the components that enable the incorporation of SPH methods into robotics' frameworks can be seen in Fig. 1. In the core of the framework lies the exchange of physical information between the rigid body dynamics solver and the fluid simulator. At any simulation step both solvers maintain a representation of the physical world in their respective internal data format, and the framework of interaction manages the synchronization between the two representations.

In more detail, the robotics simulator updates the kinematics and dynamics state of its simulated world and provides this information to the fluid dynamics solver. The fluid dynamics solver in turn updates its own internal information on the rigid bodies' positions, velocities and orientations with the latest information from the robotics simulation and solves for a timestep of the fluid world. It subsequently integrates the fluid forces that are applied on the rigid bodies into forces and torques which are passed back to the robotics simulation. The robotics simulation receives the information on the forces and torques and applies them onto the rigid bodies, solving for the updated accelerations, velocities and positions. At every simulation step sensors and actuator dynamics can be natively simulated and used as input for adaptive control methods. Since for the robotics simulation the only necessary information from the fluid simulation is the applied force/torque pair, and from the perspective of the fluid simulation the positions and velocities of rigid bodies, the exchange of information within the framework is kept to a minimum. There is of course the question of the visualization of the flow, so depending on the choice of rendering methodology, the streaming of the particles' positions to the rendering engine should be considered.

Numerous technical challenges needed to be addressed in order to establish the framework of interaction between the two simulation engines. Firstly, and most importantly, the handling of boundaries with complex geometries at relatively large e.g., 1 ms timesteps with fast moving rigid bodies. Fast moving SPH particles can penetrate through solid boundaries unless treated explicitly through the sampling of solid surfaces with multiple layers of boundary particles. Furthermore, SPH simulations are computationally intensive and can be parallelized on the GPU or by employing single instruction, multiple data (SIMD) [7] techniques that parallelize computations on the CPU. In our implementation we used both GPU acceleration for the neighborhood search problem of finding the particles in the vicinity of a particle as well as SIMD instructions using the Advanced Vector Extensions (AVX) framework. Finally, achieving numerical stability can prove to be a complex task for robots interacting with fluids due to the instabilities that the

Table 1
Qualitative comparison of SPH and grid-based methods. SPH has found limited applications in engineering, as grid-based methods have historically prevailed in fluid simulation. Even though more mature in terms of mathematical accuracy proof, grid-based methods cannot easily handle scenarios such as fast moving boundaries, material discontinuities, free surfaces and multiphase.

	Moving boundaries	Free surface flows	Multiphase flows	Mathematical accuracy proof	Computational cost
Grid-based methods (FEM, FVM, FDM)	With remeshing	difficult due to flow discontinuity	needs special treatment	well-established	High
SPH	✓	✓	built-into the method	limited	Higher than grid-based[15]

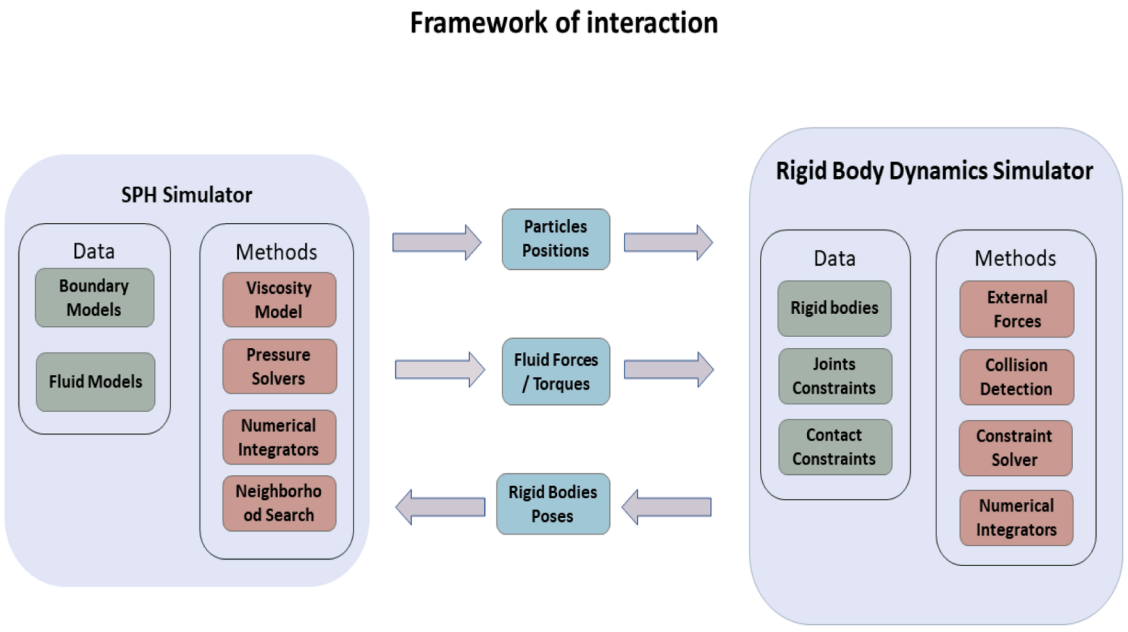


Fig. 1. The abstract framework of interaction between the rigid body dynamics and the fluid solver. In this framework each simulator aggregates the information about the physical world in its own internal representation. Typical requirements for an SPH simulator are the support for different boundary models, a pressure solver, viscosity models, numerical integration, surface tension and vorticity models as well as neighborhood search. The rigid body dynamics simulator supports collision detection, computation of external forces (i.e., gravity), the definition of joint and friction constraints and the solver that enforces them, as well as numerical integration. The exchange of information is kept to a minimum of forces and torques applied by the fluid to the rigid bodies and of the poses of rigid bodies from the rigid body simulator to the fluid simulator. Optionally the particles positions can be communicated for rendering purposes. The framework of interaction enforces synchronization and executes a numerical integration step for each simulator. The data is exchanged during explicit synchronization steps. This means that practically the simulator with the largest timestep (which in our case is the fluid simulator) has to wait for multiple simulation steps of the slower simulator. The particles positions vector occupies a significant amount of memory, especially for large geometrical objects. The data that describe the fluid forces and torques as well as the rigid bodies positions are small and in practice add little memory overhead.

fast-moving links of the robots can induce to the fluid simulation when large timesteps are used. To address the latter, we employed a combination of adaptive time-stepping, explicit treatment of viscosity through the use of dedicated viscosity models and an implicit time integration scheme. However, this often comes at the expense of increased computational costs. One parameter which defines the accuracy of the simulation from the theoretical perspective is the particle size. We investigated thoroughly which particle sizes can better capture the simulation's complexity and provide accuracy without hindering performance. To better capture this dependency, we present simulations with very few particles and show that although they are stable they cannot fully capture the behavior of the robot.

In the next section we analyze previous efforts for the simulation of fluid dynamics problems in self-propelled swimming and robotics.

1.1. Previous works

Multiple CFD methods have been employed in the past for the simulation of swimming bodies such as the Finite Differences (FD), the Finite Volumes (FV) [8] methods, the Arbitrary Eulerian-Lagrangian (ALE) [9,10] method, the Immersed Boundary method (IB) [11,12] and particle based methods like Vortex Methods [13] among others. Some combinations of the Smoothed Particle Hydrodynamics (SPH) method with IB have been shown in the past [14] for the simulation of self-propelled swimmers, but employ remeshing techniques that induce computational cost and programming complexity.

Historically the Finite Element (FEM), FV and FD methods for the solution of fluid dynamics problems have been prevalent, since their development starts already in the 1940s and whose algorithmic maturity exceeds that of SPH, complimented by the lack of a complete mathematical treatise on the numerical accuracy of SPH methods [15]. While FD and FV methods offer discretization schemes with varying degrees of accuracy even up to 8th order [16], practically 1st and 2nd order schemes are employed in most widely used commercial and research software. It should be noted that accuracy increases as the cell size decreases so the tradeoff between cell size and discretization scheme order should be considered relative to the computational workload.

On the other hand, SPH based methods have been widely adopted by the computer graphics community [17], and more recently in the engineering community, within various fields of applications [18–20]. The current understanding of the SPH community is that the discretization methods commonly employed in SPH lead to a 2nd order theoretical accuracy and practically converge to an accuracy between 1st and 2nd order [15,21], with some efforts focusing on higher order approximations [22]. Intense research effort goes into improving the convergence properties and the numerical accuracy of SPH methods [15].

Even with the latest developments, particle-based methods have found practically limited to non-existent applications in robotics apart from the work of Lopez-Guevara et al.[23], where they employ the Position Based Fluids method [24], implemented into the Nvidia FleX solver. Some reasons are the fact that they have higher computational complexity, boundary handling can be cumbersome for established boundary conditions like pressure and velocity inlet/outlet conditions, and the fact that there are no de-facto standard tools and frameworks widely adopted by the community. Moreover the technological challenges of ensuring stability between the robotics and fluid simulation, handling fast moving rigid bodies and complex geometries render the development and use of such tools cumbersome. A recent application of our proposed framework was shown in [25], where the authors simulate 3D material deposition with moving robots and in [26], where the authors describe the software implementation of SPH and rigid body dynamics coupling.

For applications where accuracy of the fluid forces is not essential, researchers have employed simpler fluid dynamics models [27], that can

estimate the drag forces applied on a rigid body based on a drag coefficient [28]. Such models have proven very useful for the testing of robot control strategies, but their intrinsic lack of accuracy can often fail to capture the complex phenomena attributed to the fluid flow, such as vorticity, surface tension and turbulence. Dedicated fluid dynamics solvers based on Lighthill's theory have been developed for anguiliform swimmers [29]. They present very good accuracy but their intrinsic lack of applicability to other types of swimmers limits their adoption.

2. Methods

2.1. SPH method

SPH research has provided many variations of the original Gringold and Monaghan's 1977 method, many of which originate in computer graphics [17,21]. In our work we used the weakly compressible SPH (WCSPH) [30] and divergence-free SPH (DFSPH) [31] formulations, both of which are well-established methods in the field. For the modelling of viscosity, we used the XSPH viscosity formulation. Various methods exist in SPH for the handling of boundaries between solids and fluids. One approach is to sample the surface of solids with dummy particles that can be used for the computation of the fluid's properties, but which are updated with the positions and velocities of the rigid bodies, thus remaining attached to the rigid body. One of the most commonly used techniques was first presented by Akinci et al. [32]., where they solve the problem of non-uniform particle sampling. The problem can arise when sampling complex surfaces with particles, and in their method the sampling is performed by weighting the contributions of the boundary particles depending on their sampling density. An example of this sampling of robot surfaces with particles can be seen in Fig. 2. Another approach uses density [33] or volume maps [34], which are implicit structures that enable a smooth boundary representation. We implemented a version of Akinci's model due to the ease of implementation and robustness. The timestep used was 1 ms.

2.2. Software implementation

We provide an open-source software implementation of the framework of interaction described in the introduction (see also Fig. 1), using Gazebo [35] with ODE as its physics engine (the user can also change the simulator to bullet), a simulator which is widely adopted by the robotics research community. Gazebo comes with a rich ecosystem built around the Robot Operating System (ROS) [36], and support for multiple physics engines (ODE, bullet, DART [37], simbody [38]). For the fluid dynamics simulation, we coupled the open-source SPH-based SPlisHSPlasH[2] library with Gazebo. The coupling is achieved through a C++ plugin[3] written as an interface between the robot and the fluid simulation. Within every simulation timestep Gazebo executes a physics simulation step and computes the poses of all the rigid bodies in its scene, by solving the equations of motion along with constraint equations, e.g., joint and non-penetration constraints. It then calls the fluid simulation plugin to perform an update of its own world, using the updated rigid bodies poses as input. The fluid simulator solves the SPH equations of flow and computes the forces applied on the rigid bodies by the particles. Each particle contributes to the force and torque applied on the rigid bodies which are summed and sent back to the robot simulator. It is worth noting that the two simulation engines can run with independent timesteps, as often the control loops required in robotics are in the range of ms, whereas the fluid simulation can stably run with higher timesteps in the order of 10 s of ms. As such the robot simulation can be run for some steps before a step of the fluid simulation is executed. However, when larger timesteps are taken, the SPH solver requires more iterations to converge so in practice very large timesteps should be avoided. In order to ensure stability, we implemented a synchronization strategy based on the Courant-Friedrichs-Lewy (CFL) condition that dictates that the maximum distance a particle travels within a timestep, should be smaller than two times the radius of the particle to prevent high repulsive forces between particles. Typically, the robot simulation stays close to real time when run independently whereas the fluid simulation takes some seconds per ms of simulation time as shown in the results section on the benchmarks that we conducted. The specifications of the computers that we used are on the middle of the spectrum of computers for personal use i.e., a laptop with an intel i9–11900H CPU and an NVIDIA GeForce RTX 3080 GPU. We expect the simulation to scale excellently on CPUs with multiple cores -such as the ones used on High-Performance Computing clusters- as the code is parallelized with OpenMP. The simulation is directly interfaced with ROS and other plugins can be used simultaneously, providing control and sensor loops along with the robot and fluid dynamics simulation.

2.3. The simulation model

As the robot of choice, we used the Amphibot robot developed by the Biorobotics Lab at EPFL. The Amphibot robot has been employed in the past by multiple researchers as a means to examine various hypotheses around the organization of locomotor neural networks [39] and to test robot control algorithms [40]. Notably, the robot is slightly buoyant and swims therefore just below the water surface, which makes it difficult to simulate with other fluid simulation methods that are typically designed for deeply immersed swimming. We employed a detailed 3D model of the physical robot (Fig. 2), along with measurements of its dynamics data. We conducted sets of experiments with the purpose of testing locomotion controllers in silico compared to their physical counterparts and to examine the applicability of our simulation architecture under different conditions. We present two sets of simulations, one used for the software validation of the simulations based on a simple-sine-based controller, and a second one based on replaying the kinematics of physical experiments with the real robot in order to assess the accuracy of the simulation. The usefulness of our proposed simulation paradigm becomes evident as a tool for the validation of the control algorithms as well as the estimation of the accurate fluid flow. With this type of simulation, we can explain the complex fluid phenomena that arise, which simpler models are unable to capture.

The first set of simulations, with the purpose of validating the software implementation, involves two different scenarios:

- Forward swimming
- Right turning

In both cases the control method that we used is a simple sine-based controller, that generates target angles for the simulated robot joints and applies them with a PID controller. We used the travelling wave function:

$$x_i = A_i \sin(\omega t + 2\pi\Delta\varphi(i-1)) + b_i,$$

where x_i is the target angle for the i th joint, A_i is the i th joint's amplitude, ω is the angular frequency, $\Delta\varphi$ is the phase difference between controlled joints and b_i is a term that adds bias to the wave and induces turning. This system of equations generates the travelling wave pattern. It should be noted that this simplified control method resembles a steady-state Central Pattern Generator (CPG), a control method that has previously been used to control the amphibot robot [39] and is used as an initial validation step for the more complex CPG-controlled

[2] https://github.com/InteractiveComputerGraphics/SPlisHSPlasH
[3] https://bitbucket.org/hbpneurorobotics/splishsplash/src/master/

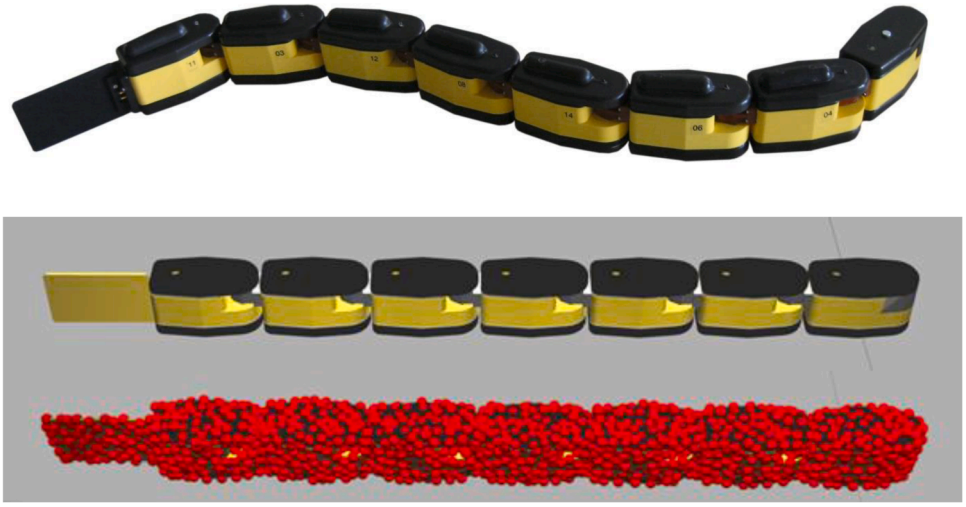


Fig. 2. Sampling of the simulated amphibot robot's surface with particles[26]**.** In the top figure the physical amphibot robot model is shown. In the middle figure, a side view of the simulated robot is shown. In the bottom figure the robot's sampling with boundary particles is shown. With this technique proposed by Akinci et al.[32], the surface of a robot's mesh is sampled with particles that contribute to the solution of the SPH equations. This way, complex geometries with irregular sampling can be accurately simulated. Here we show a discretization with particles of 0.008 radius, with a total number of particles at around 2.5 K. Simulations where the dimensions of the robot's links are comparable to the particle size can lead to the inaccurate estimation of the fluid forces applied on the boundary particles. We chose a maximum particle size of 0.008 m which leads to an average of 300 particles on top of the robot's links surfaces. As we show in Table 3 the particle size plays an important role in the estimation of the fluid forces and subsequently of the swimming velocity.

locomotion patterns we used in the physical experiments.

The outcome of these simulations can be seen in **Videos 1**,[4]**2** ,[5]**3** ,[6]**4** ,[7] where the forward swimming (**Video 1**) and a comparison of the forward swimming with different number of particles (**Video 2)** can be seen. Similarly, we show the robot turning to the right (**Video 3**) and compare the turning gait with different number of particles (**Video 4)**. One important observation from this set of simulations is that the trajectory and velocity of the robot is practically identical and unaffected by the number of particles, as long as the particle size is small enough to capture the complexity and to ensure the stability of the simulation. In order to properly test this hypothesis we performed a set of experiments of the amphibot robot swimming forward starting from a mere 1 K particles up to 447 K particles. The outcome of this experiment can be seen in Video 5.[8] We can observe that when the number of particles is $<$20 K the simulation explodes. When the number of particles is higher we get more stable simulations, however the robot changes direction instead of following the expected forward trajectory that we observe in the experiments with more particles. For our experiments around 300 K particles in a simulated pool of dimensions of 3×6 m with particle size of 0.008 m were enough to get stable simulations. However, the quality and details that we can capture with the fluid simulation vary significantly, as can be shown in Figs. 3 and 4. Details such as the formation of vortices and wave propagation are minimally – if at all- captured when a low resolution of particles is used, as opposed to the finest resolution where such phenomena are successfully simulated. As such, depending on the application at hand, a finer resolution might or may not be essential, as the robot's behavior is well captured in both cases. We note that it is not the number of particles per se that affects the simulation rather than the resolution i.e., the particle size. Indeed, when very few particles are employed, the particle radius is large. When the radius is comparable to the dimensions of the robot's links the numerical accuracy that is necessary to capture the fluid forces is limited, thus the swimming could be hindered. We empirically found that when the particle radius compared to the robot's length is at around 1/10 the simulations were stable and the robot would follow the desired trajectory.

One interesting confirmation of our simulations by real experiments is that the formation of vortices behind the swimmer's tail has been confirmed both in other simulations and with physical experiments from Particle Image Velocimetry (PIV) data [41] recorded in experiments of coral catfish swimming. The hydrodynamics of undulatory swimming have been thoroughly analyzed by Lauder and Tytell in [42] and we refer the reader to their publication for further details. It should be noted that numerical instability has been shown to affect the formation of vortices in SPH [43], which can be mitigated by employing higher order kernels, using an appropriate viscosity model, and appropriately small timesteps, which are techniques we used in our work.

After validating the software implementation and performing the first set of simulations, we compared them to physical experiments. This is a crucial step towards understanding the applicability of particle-based simulation methods in robotics as well as a means to quantify the sim-to-real gap. As such, four different physical experiments were performed using the Amphibot robot. The robot is equipped with LED lights on each element that are tracked from above with a camera for extracting swimming kinematics. This provides joint angles, as well as the robot position and orientation over time, at a rate of 143 Hz. The extracted joint angles are provided as reference angles for the simulated robot PID controllers. We can then compare the resulting swimming trajectories between the simulation and real experiments to assess how closely the simulation replicates reality. In these experiments **(Videos 6,7,8,9)**, the robot swims under different physical experiment scenarios:

1. Forward swimming – Dataset 1
2. Forward swimming and fast turning – Dataset 2
3. Slow turning – Dataset 3
4. In-place fast turning – Dataset 4

The robot's control method is a CPG model [39], that is applied through PID controllers. An analysis of the different gaits along with their control signals is shown in Fig. 5. In the Figure multiple timesteps have been chosen that correspond to a dynamic change of the robot's

[4] https://youtu.be/IbyMPYz9kLc
[5] https://youtube.com/shorts/6oeqGLC14v8
[6] https://youtu.be/k5d1IuYph7s
[7] https://youtube.com/shorts/DTxOKq5XWOo
[8] https://youtu.be/qLFwJTDszo0

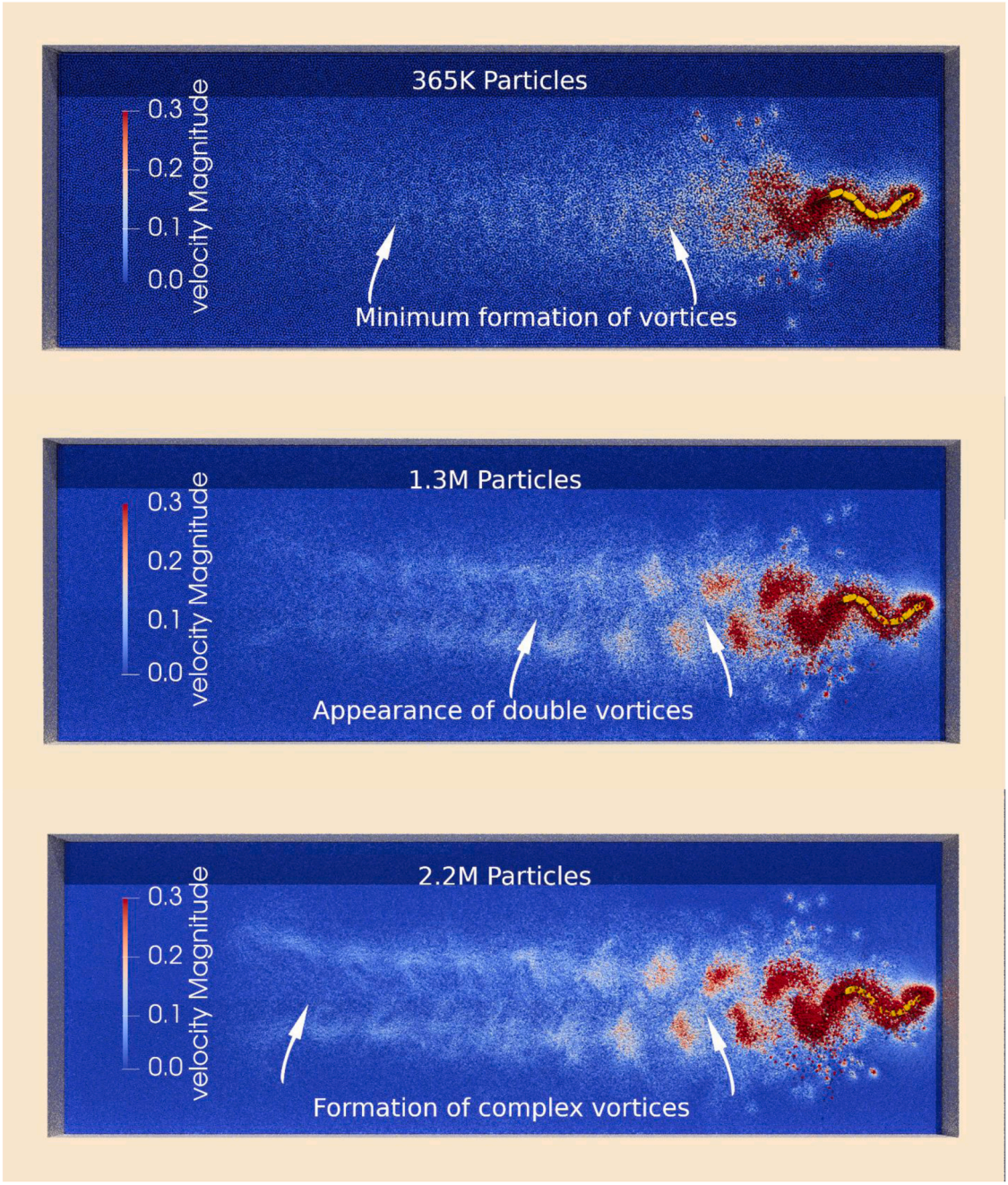


Fig. 3. Effect of the number of particles and radius on the accuracy of the simulation for the forward swimming scenario. Comparison of the forward swimming simulated with 365 K particles / 0.008 m radius (top), 1.3 M particles / 0.007 m radius (middle) and 2.2 M particles / 0.006 m radius (bottom). It can be observed that the robot 's pose is practically identical and invariant to the number of particles in the forward swimming gait. On the contrary, the level of details that can be accurately captured by the fluid simulation varies significantly. The formation of vortices as well as other aspects such as the propagation of waves depends heavily on the particle radius used to represent the fluid flow. With a particle radius smaller than 0.008 m the simulation breaks down due to numerical instabilities as the number of particles used to represent the complex fluid flow is not enough.

gait. These complex gaits capture a wide range of the physical robot's behaviors, from forward swimming under a steady-state signal to abrupt changes in the robot's direction and swimming velocity. They also show that the recorded joint positions closely follow the target signals patterns. This is however not the case for the simulated robot, whose target signals are not precisely followed by the simulated PID controllers (see also Fig. 6). A direct comparison of the simulated and the real robot swimming can be seen in Videos 6,[9]7 ,[10]8 ,[11]9 .[12] In these videos the physical robot's LEDs are overlayed with yellow markers on top of the simulated robot. This visualization is the most straightforward way to visually inspect the accuracy of the simulation compared to the real experiments. The simulation parameters are provided in Table 2.

[9] https://youtu.be/O645VGloQ9g
[10] https://youtu.be/cQZdVhbEekM
[11] https://youtu.be/TDyQKz3ZIsU
[12] https://www.youtube.com/watch?v=YDcLCObAWhI

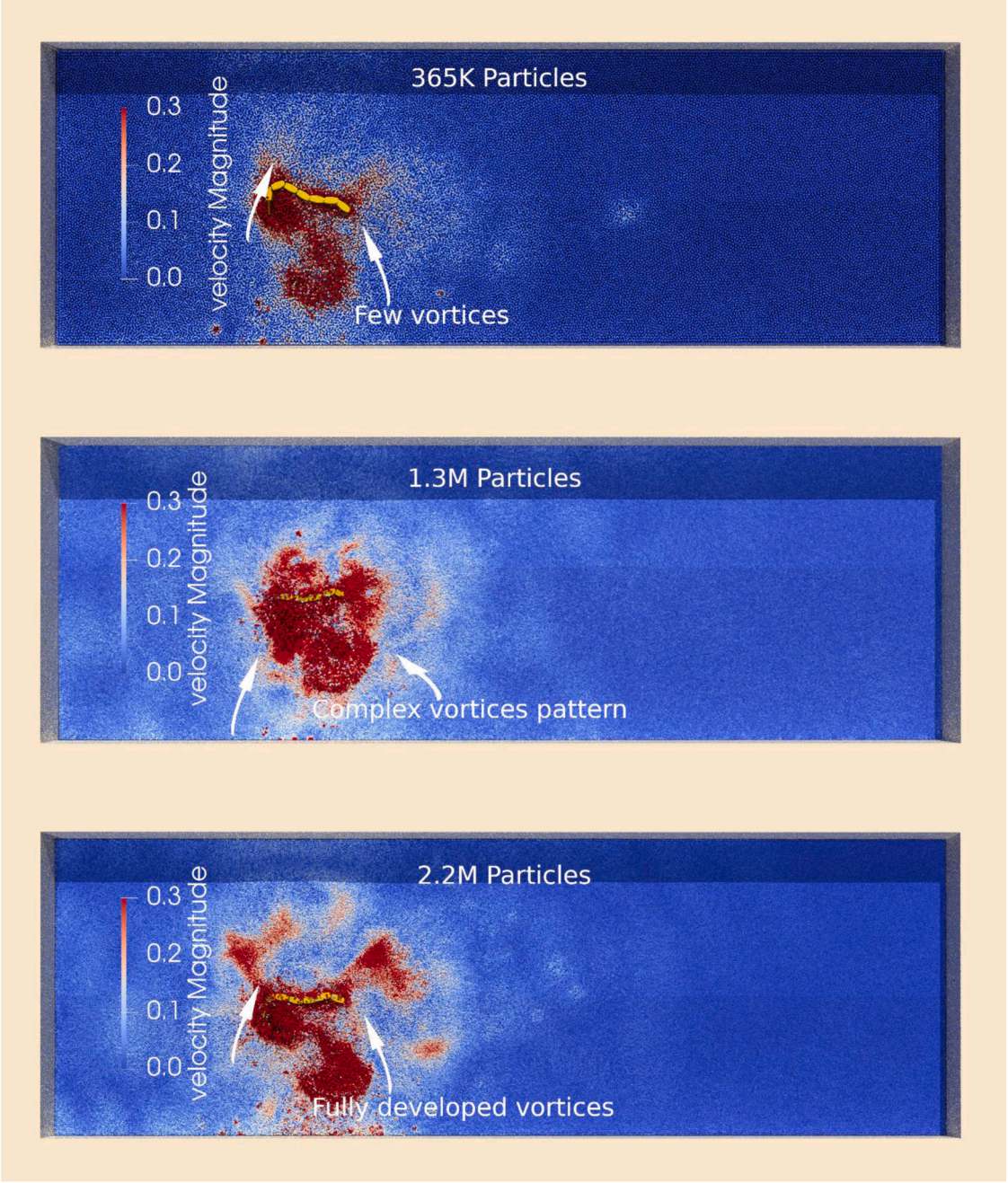


Fig. 4. Effect of the number and radius of particles on the accuracy of the simulation for the robot turning scenario. Comparison of the turning gait simulated with 365 K particles / 0.008 m radius(top), 1.3 M particles / 0.007 m radius (middle) and 2.2 M particles / 0.006 m radius (bottom). As in the case of the forward swimming, the particle radius largely defines the details captured by the fluid simulation as it is the parameter that defines the spatial discretization. Here we can observe that the waving patterns and formation of complex vortices is captured progressively more accurately as the number of particles increases. In the 2.2 M particles simulation, the formation of waves is modelled whereas in the simulations with fewer particles the waves are dampened.

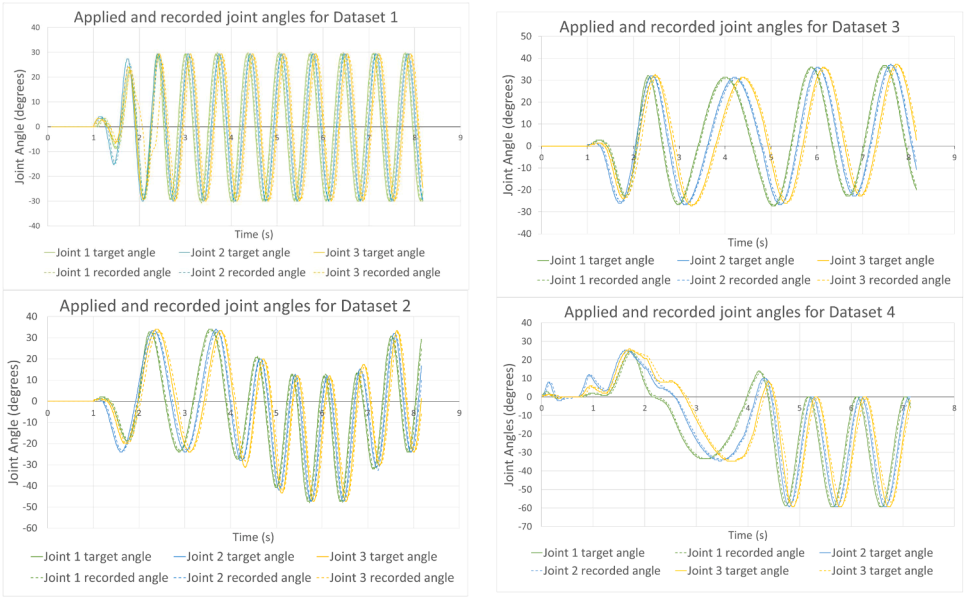


Fig. 5. The datasets input data. Depicted are the target joint angles vs the recorded angles. The locomotor gait is generated by a CPG model. The datasets correspond to forward swimming (Dataset 1), forward swimming with fast turning (Dataset 2), slow turning (Dataset 3) and in-place fast turning (Dataset 4). To further analyze the gait we choose time points of interest for each Dataset. For Dataset 1 the timestep T1 corresponds to the first peak of the travelling wave where the robot deforms towards the right side. The timestep T2 is the corresponding peak towards the left side. At timestep T3 the gait has fully developed, and the robot is fully deformed towards the right side. Timestep T4 corresponds to one of the peaks to the left side. For Dataset 2 timestep 1 corresponds to the initial right peak, T2 to the initial left peak. At timestep T3 the gait has shifted towards the right. At timestep T4 the wave has fully developed towards the right. By comparing Datasets 1 and 2 we can already observe the turning behavior, which is confirmed by the simulations. For Dataset 3 timestep T1 corresponds to a minor shift of the gait towards the right at T1 and towards the left at T2 and T3. These timesteps can be used to determine whether the effect of small shifts of the gait towards the side leads to turning behavior in the simulation. Finally, Dataset 4 is the most complicated pattern, presenting a slow left turn, followed by an abrupt change of direction towards the right. This very fast transition is one of the many complex gaits that can be encoded through a CPG. Timestep T1 shows the beginning of the abrupt turning. At timesteps T2 and T3 the gait has fully developed and shifted towards the right, with peak amplitude and frequency, causing the robot to turn towards the right. All these expected behaviors are well captured by our simulations, proving their value as a validation tool for robot control algorithms.

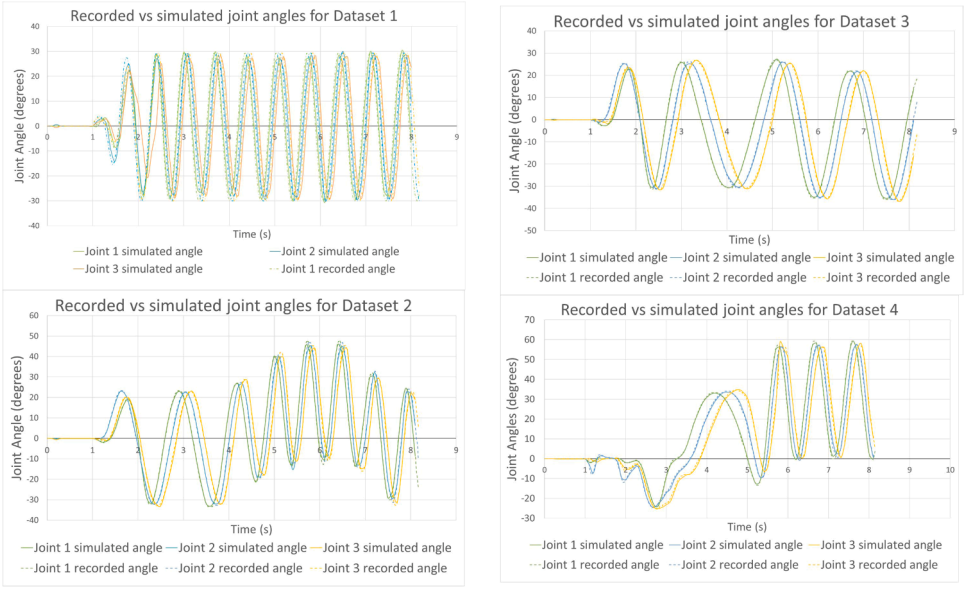


Fig. 6. The recorded vs simulated joints. A good evaluation metric of the divergence between simulation and reality is to examine the robot's experimentally recorded joints angles in comparison to the robot's simulated angles. The simulated PID controller tries to apply the angles, but due to the rigid body dynamics solver's inability to exactly enforce the joint angles, there is small discrepancy between the simulated and the experimentally recorded joint angles. This is another source of sim-to-real gap, coming from one component of the physics simulation, the rigid body dynamics solver.

Moreover, to better compare the outcome of the simulation with the real experiments we performed an analysis of the swimming trajectory of the robot shown in Fig. 7. The data that we used for the simulation and for the experiment are not geometrically identical as the experimental data is acquired from the LEDs on the robot whereas on the simulated robot we used the frames attached to the robot's links, but they present a good basis for quantitative comparison. The main observation of the trajectories is that the overall pattern is well matched, but there is some

Table 2
The parameters used for the simulations. The timestep size is the most important parameter when it comes to the stability of the simulation. It can vary between the fluid and the robotics simulators, with the cost of adding synchronization complexity. In most cases the control loops of the robotic simulation require timesteps in the order of 1 ms, whereas the fluid simulation for slow moving particles can be simulated with timesteps in the order of 10 s of ms. However the larger the timestep, the more iterations it takes for the SPH solver to converge. This tradeoff should be taken into account when designing simulations such as the ones we show. The viscosity model that we used for Gazebo/SPlisHSPlasH is XSPH with a model-specific parameter of 0.001. This model-specific value was chosen as a compromise between stability and excessive damping of the robot's swimming velocity. Smaller values lead to chaotic particle movement whereas higher values overdamp the robot's velocity. The adaptive timestep size for the fluid simulation has an upper limit that is given by the Courant–Friedrichs–Lewy (CFL) condition[47]. In the computation of the particles velocity needed by the CFL condition, the boundary particles' velocities are also taken into account. The accuracy of the simulation as well as the computational time are directly influenced by the particle size which also determines the number of particles. The SPH method for Gazebo/SPlisHSPlasH is DFSPH. We use a semi-implicit Euler integration scheme for the fluid simulation. The time integration scheme used in the Open Dynamics Engine that powers the Gazebo simulation is referred to as a first order Euler integration scheme, but no more details about the implementation are given.[v]

SPH method	DFSPH
Boundary handling method	Akinci et al.
Timestep Size	1ms
Viscosity Model	XSPH
Viscosity Parameter (model specific)	0.001
Particle Radius	0.006 m - 0.008 m
Particle Number	~365 K - ~2.2M
SPH Kernel	Cubic Spline

[v] http://www.ode.org/ode-latest-userguide.html#sec_3_3_0

damping of the robot's velocity in the simulation as we analyze in the Discussion section. We observe that the simulated robot's PID controller cannot follow the recorded angles of the physical robot precisely due to the numerical solver's inability to satisfy the joint constraints (Figs. 5 and 6).

It is well established that a plethora of factors contribute to the sim-to-real gap, namely sensors' noise, PID controller imperfections, differences between the dynamical model of the simulated robot and the dynamics of the real robot, the infeasibility of replicating the initial conditions of the experiment and the difficulty of estimating the fluid dynamics simulation parameters, such as viscosity coefficients. Despite all these sources of discrepancy, the simulation captures the overall behavior of the robot well.

To further quantify the sim-to-real gap, we present an analysis of the orientation vector of the simulated and the real robot. The vector's origin is at the tail link and its end at the head link, thus giving a good estimate of the robot's pose. The animations for the 4 Datasets can be seen in **Animations 1,**[13] **2,**[14] **3,**[15] **4.**[16] The robot's orientation is close to the experiments, given the sources of sim-to-real gap. The robot rolls minimally as is observed in the real robot but does not fall to the sides as would be the case if there was an inaccurate estimation of the forces that are applied on the robot link's side surfaces. Buoyancy is heavily dependent on the parametrization of the simulation and the mass and volume data, and as we can see in the Videos 1 and 3, the robot swims close to the surface, as is the case with the real robot. An inaccurate estimation of the mass can quickly lead to numerical instabilities as the robot might be swimming closer to the surface and thus fewer particles would be in contact with the robot's links. In this case the number of particles would not be enough to accurately compute the forces applied from the fluid to the robot's links.

Another important metric for the estimation of the effect of the particle simulation is shown in Table 3. On the table an estimation of the forward swimming velocity for a varying number of particles is given. The analysis shows that the finer particle resolution gives a more accurate estimation of the swimming velocity, which is higher as the particle size decreases. Nevertheless, the effect of the particle size is not so significant, as with the finest particle resolution the velocity changes from 0.627 m/s to 0.645 m/s which is a 2.7% underestimation. Thus, simulations with fewer particles which are computationally less costly can be used to give an estimation of the overall swimming behavior. Finer resolutions can be used to estimate the velocity more accurately as well as to add more details to the fluid simulation, such as the formation of vortices, waves, splashes etc. Finally we attempt to quantify the difference between the trajectories for the experiments and the simulations by employing similarity metrics as per [44] as shown in Table 4. For the different datasets we can observe that the similarity measures remain relatively small, meaning that the errors between the curves are not high. Furthermore the values remain similar between the different datasets, allowing us to conclude that the gap between simulation and reality remains relatively consistent throughout the experiments.

3. Discussion

To our knowledge this is one of the first applications of SPH on robot swimming, and the first one that compares simulations with real experiments in order to estimate the sim-to-real gap. We found that the agreement between the experimental and simulated data is good, given all the factors that contribute to the sim-to-real gap. This line of works paves the way for future works that would further focus on closing the sim-to-real gap.

It has been shown in the literature that SPH simulations tend to overpredict dissipation at finite viscosities, whereas in the absence of viscosity, the particle movement is chaotic [45]. This leads to a damping effect due to dissipation, which can be observed in our simulations. It is particularly difficult to find parameters for viscosity that are at the same time physically meaningful and lead to stable simulations. We note that in our simulations there is energy loss due to the XSPH formulation of viscosity as analyzed by Monaghan [18]. This means that it is impossible to accurately predict the behavior of the robot when it is not actively swimming, as is the case when it is pushed to drift on the swimming pool.

The simulation results show that SPH-based robotics simulations can capture the intricacies of self-propelled swimming, even when complex interactions take place, such as abrupt changes in the velocity and direction of the robot. The simulations are nevertheless very close to the real experiments, given the various factors that are difficult to model, including the initial position of the robot, the noise in the sensory data, the estimation of the dynamics parameters of the simulated models, the parametrization of viscosity and numerical errors among others. These factors are worth exploring in future work as they would help reduce the sim-to-real gap. We found in practice that the parametrization of viscosity is the parameter that is the most important for the fluid simulation, as too much viscosity dampens the robot's movement. On the contrary low viscosity leads to chaotic movement of the particles and instability of the fluid simulation as is mentioned previously. The rest of the factors that are coming from the differences between the experimental setup and the simulation setup can be alleviated by establishing

[13] https://youtu.be/J8zbCbF3GP0
[14] https://youtu.be/gUhWXMozPVs
[15] https://youtu.be/R3hh6iYN7HQ
[16] https://youtu.be/rFAxx5mPr50

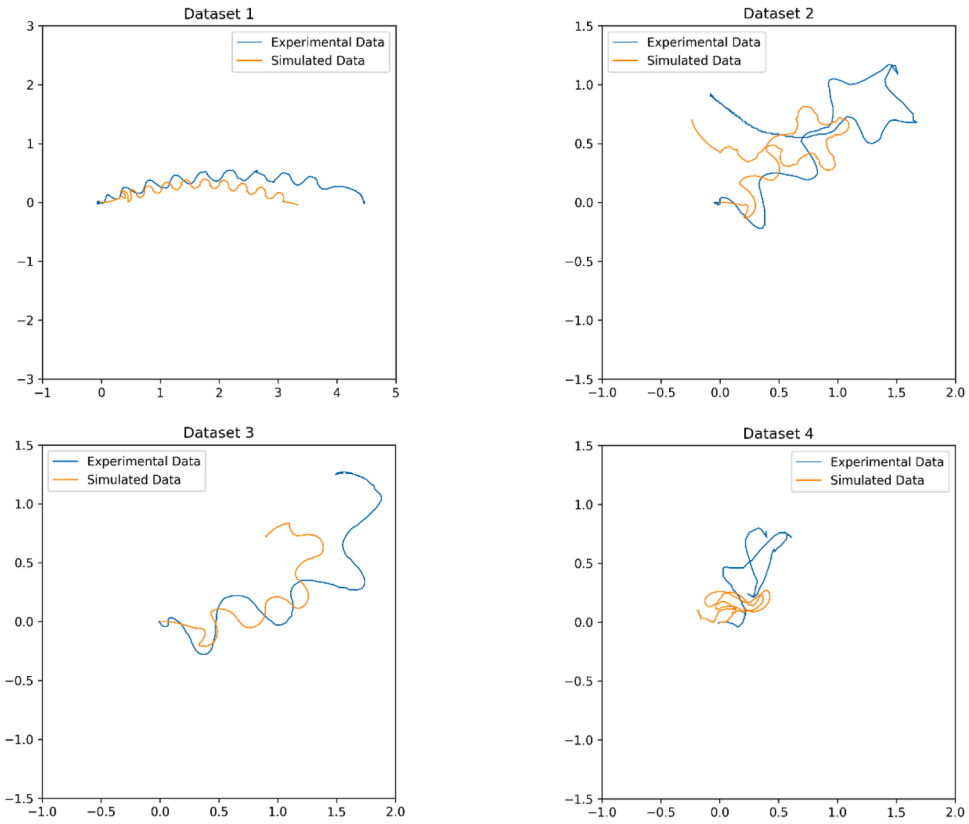


Fig. 7. Simulated and recorded trajectories for the experiments. The overall behavior of the robot is well captured by the simulation and is close to the real experiments. Differences between the simulation and the experiment can be attributed to the noisy data, the dynamics model, the imperfect PID controller, the difficulty of tuning the simulation parameters and the infeasibility of precisely reproducing the initial conditions of the experiment. Especially for Dataset 4, the recording of the robot's trajectory is interrupted due to the sensor's inability to accurately record the link frames, as can also be seen in Video 8.

Table 3
Forward swimming velocity relative to the number of particles. The robot's forward swimming velocity is influenced by the particle size, nevertheless its effect does not significantly increase as the resolution increases. This means that simulations with lower resolutions that are computationally less costly can be used to estimate the overall behavior of a robot. A more costly simulation with a higher number of particles can provide a more accurate estimation of the swimming velocity as well as the overall behavior of the fluid as shown in the videos and figures.

Number of particles	Particle size	Forward swimming velocity
365K	0.008 m	0.627 m/s
1.3M	0.007 m	0.640 m/s
2.2M	0.006 m	0.645 m/s

protocols for the successful reproduction of experiments in silico. Such examples could be the accurate recording of the robot's pose at the beginning of every experiment, the experimental determination of the robot's links inertial tensors and the recording of multiple camera views for better visual inspection.

The simulations that we show are just a small subset of the possible robotics simulations that would benefit from our framework. We envision scenarios such as simulating amphibious field robots, biorobots that receive sensory feedback and adapt their control loops, robots manipulating fluids, as well as highly viscous and elastic materials simulations such as the ones shown in [25]. Furthermore, multi-physics simulations combining material models for soft body simulation (FEM) and advanced contact resolution methods such as Incremental Potential Contact [46] (IPC) would give the possibility to model different types of media in the same simulation e.g., granular media and environments that mix water, sand, mud, dry ground etc.

From the fluid simulation point of view, various solutions that increase the SPH accuracy have been proposed, including the adaptation of the particles size depending on the proximity to the areas in the environment where motion occurs. In the context of robotics simulations, the fluid flow is more turbulent in the areas proximal to the robot, where the fluid forces need to be estimated accurately and where a finer particle granularity could provide such accuracy. On the contrary areas where the fluid is resting could be modelled with fewer particles.

Various methods like FVM, FDM, IBM have been proposed for the simulation of self-propelled swimming robots as we discuss in the Introduction part. The difficulty of boundary handling for fast moving and shape changing bodies, mesh generation and updates for complex geometries, the treatment of open simulation worlds limit their applicability to pre-determined computational domains. Particle-based methods on the other hand give a very useful tool for use cases such as swimming on the surface, abrupt changes in the robot velocity, fast changing robot morphology.

Table 4
Quantitative comparison of the trajectories' curves. In order to quantify the gap between the simulations and the experimental data we use two metrics that can be used to assess the difference between two curves. The two metrics are the Discrete Frechet distance and the Curve Length similarity measure. The Discrete Frechet distance computes the shortest leash distance that is required to connect the two curves, whereas the Curve Length distance estimates the differences in arc-length distance between the two curves[44]. For both methods a smaller value represents a higher similarity between the two curves.

Similarity Measure	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Discrete Frechet distance	1.21	0.62	0.59	0.58
Curve Length	7.26	8.36	6.35	10.42

As a general conclusion we anticipate that the adoption of SPH methods and the usage of multi-physics simulation in robotics will bring us one step closer to bridging the gap between simulation and reality. We believe that our work is a useful step towards this direction.

Data and materials availability:

All data and materials will be made publicly available upon publication acceptance. The software implementation is already open-sourced and can be found in[17]

CRediT authorship contribution statement

Emmanouil Angelidis: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Jonathan Arreguit:** Software, Methodology, Investigation, Conceptualization. **Jan Bender:** Writing – review & editing, Supervision, Software, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Patrick Berggold:** Software, Conceptualization. **Ziyuan Liu:** Resources, Project administration. **Alois Knoll:** Supervision, Resources, Project administration. **Alessandro Crespi:** Visualization, Validation, Software, Investigation, Data curation, Conceptualization. **Auke J. Ijspeert:** Writing – review & editing, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Funding: J.A. and A.J.I are/were funded by the ERC Synergy grant number 951477 and the HFSP grant number RGP0027/2017.

Data availability

Data will be made available on request.

References

[1] D. De Padova, L. Calvo, P.M. Carbone, D. Maraglino, M. Mossa, Comparison between the Lagrangian and Eulerian Approach for Simulating Regular and Solitary Waves Propagation, Breaking and Run-Up, Appl. Sci 11 (2021) 9421.
[2] J.M. Domínguez, et al., DualSPHysics: from fluid dynamics to multiphysics problems, Comput. Part. Mech. 9 (2022) 867–895.
[3] G. Qiu, S. Henke, J. Grabe, Application of a Coupled Eulerian–Lagrangian approach on geomechanical problems involving large deformations, Comput. Geotech. 38 (2011) 30–39.
[4] A.J. Barlow, P.-H. Maire, W.J. Rider, R.N. Rieben, M.J. Shashkov, Arbitrary Lagrangian–Eulerian methods for modeling high-speed compressible multimaterial flows, J. Comput. Phys. 322 (2016) 603–665.
[5] T. Kempe, J. Fröhlich, An improved immersed boundary method with direct forcing for the simulation of particle laden flows, J. Comput. Phys. 231 (2012) 3663–3684.
[6] R.A. Gingold, J.J. Monaghan, Smoothed particle hydrodynamics: theory and application to non-spherical stars, Mon. Not. R. Astron. Soc. 181 (1977) 375–389.
[7] C.J. Hughes, Single-Instruction Multiple-Data Execution, Springer Nature, 2022.
[8] S. Pacholak, S. Hochstein, A. Rudert, C. Brücker, Unsteady flow phenomena in human undulatory swimming: a numerical approach, Sports Biomech 13 (2014) 176–194.
[9] S. Kern, P. Koumoutsakos, Simulations of optimized anguilliform swimming, J. Exp. Biol. 209 (2006) 4841–4857.
[10] L. Yan, et al., A numerical simulation method for bionic fish self-propelled swimming under control based on deep reinforcement learning, Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci. 234 (2020) 3397–3415.
[11] J. Zhang, et al., Numerical Study of the Fish-like Robot Swimming in Fluid with High Reynolds Number: immersed Boundary Method, Actuators 11 (2022) 158.
[12] E.D. Tytell, C.-Y. Hsu, T.L. Williams, A.H. Cohen, L.J. Fauci, Interactions between internal forces, body stiffness, and fluid environment in a neuromechanical model of lamprey swimming, Proc. Natl. Acad. Sci 107 (2010) 19832–19837.
[13] M. Gazzola, P. Chatelain, W.M. van Rees, P. Koumoutsakos, Simulations of single and multiple swimmers with non-divergence free deforming geometries, J. Comput. Phys. 230 (2011) 7093–7114.
[14] S.E. Hieber, P. Koumoutsakos, An immersed boundary method for smoothed particle hydrodynamics of self-propelled swimmers, J. Comput. Phys. 227 (2008) 8636–8654.
[15] S.J. Lind, B.D. Rogers, P.K. Stansby, Review of smoothed particle hydrodynamics: towards converged Lagrangian flow modelling, Proc. R. Soc. Math. Phys. Eng. Sci. 476 (2020) 20190801.
[16] Moukalled, F., Mangani, L. & Darwish, M. The Finite Volume Method. in *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab* (eds. Moukalled, F., Mangani, L. & Darwish, M.) 103–135 (Springer International Publishing, Cham, 2016). doi:10.1007/978-3-319-16874-6_5.
[17] Ihmsen, M., Orthmann, J., Solenthaler, B., Kolb, A. & Teschner, M. SPH Fluids in Computer Graphics - Eurographics State-of-the-art report. (2014).
[18] J.J. Monaghan, Smoothed Particle Hydrodynamics and Its Diverse Applications, Annu. Rev. Fluid Mech 44 (2012) 323–346.
[19] M.S. Shadloo, G. Oger, D. Le Touzé, Smoothed particle hydrodynamics method for fluid flows, towards industrial applications: motivations, current state, and challenges, Comput. Fluids 136 (2016) 11–34.
[20] (叶挺)(潘定一)(黄灿) T Ye, D. Pan, C. Huang, M. Liu, Smoothed particle hydrodynamics (SPH) for complex fluid flows: recent developments in methodology and applications, Phys. Fluids 31 (2019) 011301.
[21] Koschier, D., Bender, J., Solenthaler, B. & Teschner, M. Smoothed Particle Hydrodynamics Techniques for the Physics Based Simulation of Fluids and Solids. (2019) 10.2312/egt.20191035.
[22] H. Gotoh, A. Khayyer, H. Ikari, T. Arikawa, K. Shimosako, On enhancement of Incompressible SPH method for simulation of violent sloshing flows, Appl. Ocean Res. 46 (2014) 104–115.
[23] T. Lopez-Guevara, N.K. Taylor, M.U. Gutmann, S. Ramamoorthy, K. Subr, Adaptable Pouring: teaching Robots Not to Spill using Fast but Approximate Fluid Simulation, in: Proceedings of the 1st Annual Conference on Robot Learning, PMLR, 2017, pp. 77–86.
[24] M. Macklin, M. Müller, Position based fluids, ACM Trans. Graph. 32 (2013) 104, 1-104:12.
[25] U. Berdica, Y. Fu, Y. Liu, E. Angelidis, C. Feng, Mobile 3D Printing Robot Simulation with Viscoelastic Fluids, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021, pp. 7557–7563, https://doi.org/10.1109/IROS51168.2021.9636114.
[26] E. Angelidis, et al., Gazebo Fluids: sPH-based simulation of fluid interaction with articulated rigid body dynamics, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 11238–11245, https://doi.org/10.1109/IROS47612.2022.9982036.
[27] Ö. Ekeberg, A combined neuronal and mechanical model of fish swimming, Biol. Cybern. 69 (1993) 363–374.
[28] R. Godoy-Diana, B. Thiria, On the diverse roles of fluid dynamic drag in animal swimming and flying, J. R. Soc. Interface 15 (2018) 20170715.
[29] M. Porez, F. Boyer, A.J. Ijspeert, Improved Lighthill fish swimming model for bio-inspired robots: modeling, computational aspects and experimental comparisons, Int. J. Robot. Res. 33 (2014) 1322–1341.
[30] J.P. Morris, P.J. Fox, Y. Zhu, Modeling Low Reynolds Number Incompressible Flows Using SPH, J. Comput. Phys. 136 (1997) 214–226.
[31] J. Bender, D. Koschier, Divergence-free smoothed particle hydrodynamics, in: Proceedings of the 14th ACM SIGGRAPH /Eurographics Symposium on Computer Animation, New York, NY, USA, Association for Computing Machinery, 2015, pp. 147–155, https://doi.org/10.1145/2786784.2786796.
[32] N. Akinci, M. Ihmsen, G. Akinci, B. Solenthaler, M. Teschner, Versatile rigid-fluid coupling for incompressible SPH, ACM Trans. Graph. 31 (62) (2012) 1–62, 8.
[33] D. Koschier, J. Bender, Density maps for improved SPH boundary handling, in: Proceedings of the ACM SIGGRAPH /Eurographics Symposium on Computer Animation 1–10 (Association for Computing Machinery, New York, NY, USA, 2017, https://doi.org/10.1145/3099564.3099565.
[34] Bender, J., Kugelstadt, T., Weiler, M. & Koschier, D. Volume Maps: an Implicit Boundary Representation for SPH. in *Motion, Interaction and Games on - MIG '19* 1–10 (ACM Press, Newcastle upon Tyne, United Kingdom, 2019). doi:10.1145/3359566.3360077.
[35] N. Koenig, 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) 3, 2004, pp. 2149–2154, vol.3.
[36] Quigley, M. et al. ROS: an open-source Robot Operating System. in *ICRA Workshop On Open Source Software* vol. 3 5 (Kobe, Japan, 2009).
[37] J. Lee, et al., DART: dynamic Animation and Robotics Toolkit, J. Open Source Softw. 3 (2018) 500.
[38] M. Sherman, A. Seth, S.Simbody Delp, Multibody dynamics for biomedical research, Procedia IUTAM 2 (2011) 241–261.
[39] A. Crespi, A.J. Ijspeert, AmphiBot II: an Amphibious Snake Robot that Crawls and Swims using a Central Pattern Generator, in: Proc. 9th Int. Conf. Climbing Walk. Robots CLAWAR 2006, 2006.

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

[40] E. Angelidis, et al., A spiking central pattern generator for the control of a simulated lamprey robot running on SpiNNaker and Loihi neuromorphic boards, Neuromorphic Comput. Eng. 1 (2021) 014005.
[41] N.B. Tack, K.T. Du Clos, B.J. Gemmell, Anguilliform Locomotion across a Natural Range of Swimming Speeds, Fluids 6 (2021) 127.
[42] G.V. Lauder, E.D. Tytell, Hydrodynamics of Undulatory Propulsion. in *Fish Physiology*, 23, Academic Press, 2005, pp. 425–468.
[43] W. Dehnen, H. Aly, Improving convergence in smoothed particle hydrodynamics simulations without pairing instability, Mon. Not. R. Astron. Soc. 425 (2012) 1068–1082.
[44] C.F. Jekel, G. Venter, M.P. Venter, N. Stander, R.T. Haftka, Similarity measures for identifying material parameters from hysteresis loops using inverse analysis, Int. J. Mater. Form. 12 (2019) 355–378.
[45] S. Adami, X.Y. Hu, N. Adams, Simulating 3D turbulence with SPH, Undefined (2012).
[46] M. Li, et al., Incremental potential contact: intersection-and inversion-free, large-deformation dynamics, ACM Trans. Graph. 39 (2020), 49:49:1-49:49:20.
[47] Goswami, P. & Pajarola, R. Time adaptive approximate SPH. in *Goswami, Prashant; Pajarola, Renato (2011). Time adaptive Approximate SPH. In: Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS, Lyon, France, 5 December 2011 - 6 December 2011. Eurographics, 19-28.* (eds. Bender, J., Erleben, K. & Galin, E.) 19–28 (Eurographics, VRIPHYS 2011, 2011). doi:10.2312/PE/vriphys/vriphys11/019-028.

Dr. Emmanouil Angelidis holds a PhD from chair of Robotics, Artificial Intelligence and Embedded Systems of the Technical University of Munich (TUM), in collaboration with the Biorobotics Lab of the École Polytechnique Fédérale de Lausanne (EPFL). He is currently a principal robotics and simulation research engineer with Huawei Technologies in Munich, Germany. His research interests include autonomous locomotion, robotics simulation, multi-physics simulation, neurorobotics and artificial intelligence.