Design and Calibration of a Lightweight Physics-Based Model for Fluid-Mediated Self-Assembly of Robotic Modules

Bahar Haghighat, Hala Khodr, and Alcherio Martinoli

École Polytechnique Fédérale de Lausanne (EPFL), Distributed Intelligent Systems and Algorithms Laboratory (DISAL), School of Architecture, Civil and Environmental Engineering, Lausanne, Switzerland, `firstname.lastname@epfl.ch`

Abstract. In this paper, we consider a system consisting of multiple floating robotic modules performing self-assembly. Faithfully modeling such a system and its inter-module interactions typically involves capturing the hydrodynamic forces acting on the modules using computationally expensive fluid dynamic modeling tools. This poses restrictions on the usability of the resulting models. Here, we present a new approach towards modeling such systems. First, we show how the hardware and firmware of the robotic modules can be faithfully modeled in a high-fidelity robotic simulator. Second, we develop a physics plugin to recreate the hydrodynamic forces acting on the modules and propose a trajectory-based method for calibrating the plugin model parameters. Our calibration method employs a Particle Swarm Optimization (PSO) algorithm, and consists of minimizing the difference between Mean Squared Displacement (MSD) data extracted from real and simulated trajectories of multiple robotic modules.

1 Introduction

Self-assembly (SA) is defined as the reversible and spontaneous phenomenon of an ordered spatial structure emerging from the aggregate behavior of simpler pre-existing entities, through inherently local and random interactions in the system. Self-assembling robotic systems have garnered significant interest for their robust performances in forming structures of varied complexities and scales as well as their minimal design of constituting modules [3], [6], [13]. Among these systems, fluid-mediated self-assembling systems are of particular interest due to their use of fluid flow for providing interactions among the modules. It can be shown that using fluids is a very efficient way for moving particles at sub-millimeter scale [7].

A key component in studying programmable stochastic self-assembling systems is developing models that accurately describe the assembly process dynamics. Such models would help in: (1) accurately predicting the performances (assembly rate and yield) of the distributed system, and (2) evaluating and optimizing control strategies, whether distributed (e.g., ruleset controllers programmed on the modules) or centralized (e.g., modulating environmental features such as mixing forces deriving random interactions among modules), based on model predictions [9, 11]. Yet, relatively little effort has been devoted to modeling fluid-mediated self-assembling robotic systems. Models with high level of abstraction are usually non-spatial and assume well-mixed systems. In fact, the implications of these

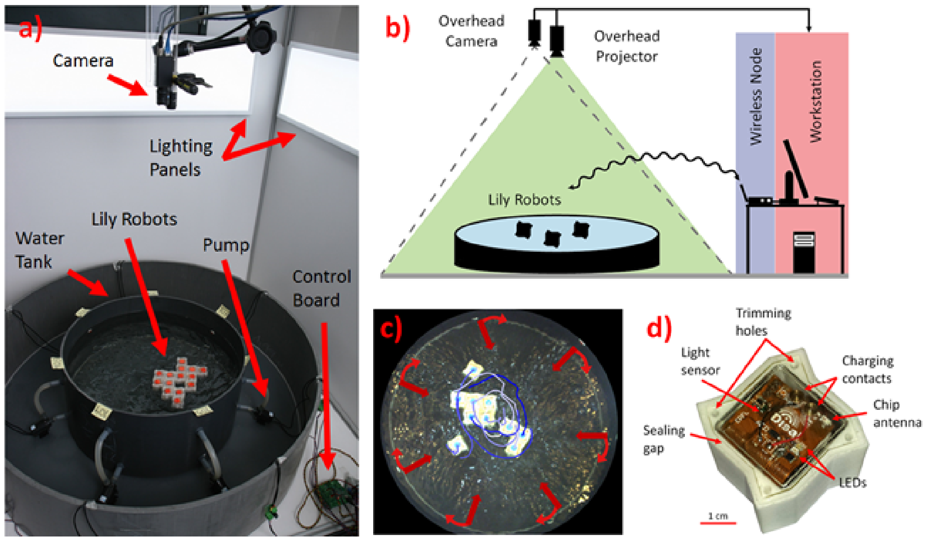


Fig. 1: a) Image and (b) sketch of the experimental system. c) Visual tracking of Lily robotic modules. The red arrows indicate the pump flow. The blue lines indicate a history of the robots trajectories. d) A Lily robotic module.

assumptions are difficult to gauge due to the lack of appropriate modeling tools for such robotic platforms [1]. Furthermore, state-of-the-art high-fidelity robotic simulators such as Webots, Gazebo, or V-REP do not support fluid dynamics natively and need therefore to be either coupled with appropriate fluid dynamics simulation tools or to be augmented with appropriate plugins in order to capture faithfully the dynamics of the overall system. While coupling these robotic simulators with fluid dynamic simulation tools allows for capturing the hydrodynamic forces accurately, such accuracy comes at the cost of having a typically very computationally heavy model. Our goal here is thus developing a physics-based model which is computationally lightweight. Such model could then be extensively exploited for predicting and evaluating the system trajectories, in particular by creating large datasets of sample system trajectories which are faithful to reality.

The paper is organized as follows: Section 2 describes the fluid-mediated self-assembling robotic system used to collect the trajectories; Section 3 introduces the physics-based model of the system accounting for simple hydrodynamic effects such as drag, buoyancy, and, to some extent, fluctuations due to stirring; Section 4 describes our method to calibrate the parameters of such models based on the comparison between the simulated and experimental trajectories of the robotic modules using a PSO algorithm. Finally, Section 5 discusses the experimental results of our calibrated physics-based model; while Section 6 concludes the paper.

2 Experimental System

The experimental setup consists of a circular water-filled tank equipped with peripheral pumps, an overhead camera, an overhead projector, a wireless node communicating with the robots, and a workstation (see Figure 1) [5]. The Lily robots are not self-locomoted, they are instead stirred by the flow field produced by the pumps. Each Lily robotic module is endowed with four Electro-Permanent Magnetic (EPM) latches, one at each side, used for cennecting and communicating to neighboring Lily modules [4]. The tank is approximately 0.6 m in diameter and 0.3 m in depth, and has seven inlets perpendicular to the wall which are endowed with a small insert piece to deviate the flow by about 15 degrees, creating a flow field with both radial and circular components. While the perpendicular flow components instigate irregular trajectories and induce collisions in the middle of the tank, they exhibit dead spots around the wall. The tangential components, however, generate a circular field, giving rise to regular closed trajectories which do not favor collisions but eliminate dead spots. To minimize any interference with the surface flow, the outlets are all placed at the bottom of the tank. Each pump's flow rate can be continuously controlled up to 9 l/min, allowing for a variety of flow fields and corresponding induced trajectories.
To monitor the evolution of the system, we use an overhead camera to track a passive marker located at the top of each robot using SwisTrack [8]. The positions of the markers are logged at a rate of approximately 30 Hz. Complementary to the visual tracking data, is the data logged by the wireless node communicating with the robots over radio. These data contain the evolution of the robots' internal states. The wireless node is also used to program the robots.

3 Modeling the Experimental System

In order to realistically recreate our self-assembling robotic system in simulation, we use Webots [10], a physics-based robotics simulator which uses the Open Dynamics Engine (ODE) for simulating rigid body dynamics. Additionally, in order to simulate specific not natively supported physics such as complex fluid dynamics, it is possible to employ custom-designed physics plugins. Building our physics-based model within the Webots simulation framework comprised two main aspects. First, faithful recreation of the Lily robotic module's hardware and firmware features, and second, faithful recreation of the hydrodynamic forces acting on the robotic modules floating in the tank filled with water. The latest version of Webots supports a basic fluid node which allows for a simple uniform stream velocity, but is not capable of simulating the complex fluidic field in our experimental arena.

3.1 Recreating the Robotic Module

We recreate the Lily robotic module within the simulated world of Webots in several steps (see Figure 2). In the first step, we defined the physical entity of the module. A CAD design of the external shell of the module was designed in SolidWorks and directly exported to Webots in the VRML V2.0 format. This defines the bounding object (i.e. bounding volume) associated with a Lily and is the one referred to by the ODE engine for simulating the collisions among

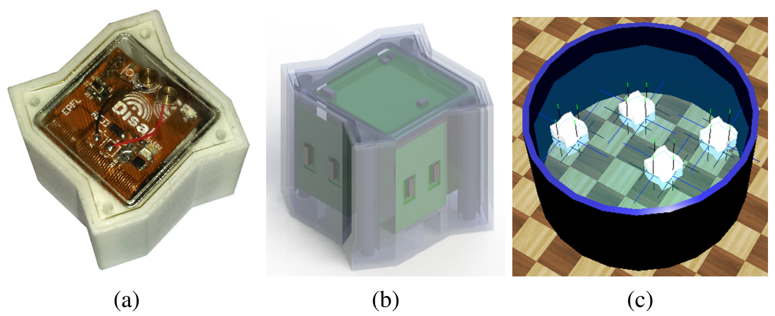


Fig. 2: (a) A real Lily robotic module. (b) CAD design of the Lily robotic module exported from SolidWorks to Webots. (c) A sample world of simulated Lily robotic modules in Webots. The lines on the modules indicate the EPM connector axes.

modules. In the second step, a Lily robotic module PROTO was created. Within Webots, a PROTO allows for capturing all the features of a certain object within one PROTO container. The Lily PROTO was then augmented with the physical features of the Lily robotic modules. In particular, its bounding object as exported from SolidWorks, mass, and center of mass. A physical object in Webots has its associated linear and angular damping coefficients which are used to slow down an object. The rotational and linear speed of each object is reduced by the specified percentage (between 0.0 and 1.0) every second allowing for coping with simulation instability, all initially set to a default value of 0.5 each.
The Lily PROTO was then augmented with several *functionality nodes*, that is four connector nodes located on the sides to replicate the EPM latching mechanism, four emitter as well as four receiver nodes located on the sides with a range of 0.5 mm replicating the EPM inductive channel function, one light sensor node on the top, and an emitter node as well as a receiver node located on the top with an infinite range to replicate the radio channel communication with the base station. The next step was then importing the Lily robotic modules' embedded controller software into the Webots simulated world. The low-level functionalities such as EPM communication through sending current pulses needed to be abstracted away and replaced by similar functions from Webots. However, the adapted controller maintains the same structure as the original one programmed on the real Lilies. The specific rulesets employed on the simulated and real modules are identical.

3.2 Recreating the Flow Field

In order to reproduce the complex flow field and the hydrodynamic forces acting on the Lilies in the real world, we use an approach inspired by the one in [1]. Our approach distinguishes from the one of [1] in two ways: (i) we capture trajectories of multiple floating blobs rather than a single one, with the aim of capturing the effects of interactions of the floating objects which disturb the flow field, (ii) rather than brute force search, we employ a PSO algorithm to optimize the model parameters.

A spherical object has an isotropic drag coefficient, i.e. a constant value in all directions, while submerged in a fluidic flow. We record the trajectory of floating spherical blocks (diameter of 3 cm), roughly the same size of a Lily robotic module, for three experiments with random starting positions and duration of 10 minutes each. For this, we use ping pong balls whose weight is tuned such that the submersion level is similar to that of a Lily robotic module (25 mm below water level). The captured velocity fields acquired from different experiments are then augmented and discretized on a regular grid of 50 cells on each side, for our water tank of 60 cm in diameter. For each cell of the grid, the average and standard deviation of the observed velocity vectors are computed and assigned to that cell as expressed in Eq. 4. The fluid velocity field can be computed considering the drag force. The value of the Reynolds number R_e determines the flow regime and the form of the drag force. The Reynolds number is a dimensionless value that measures the ratio of inertial forces to viscous forces and describes the degree of laminar or turbulent flow. The Reynolds number is calculated as below for the parameters of our system:

$$Re = \frac{\rho V L}{\mu} \simeq 6700 \tag{1}$$

This value of Reynolds number indicates a quadratic drag force:

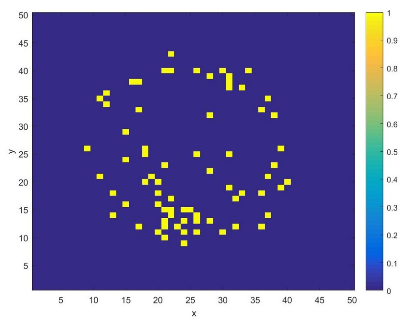
$$|\mathbf{F}_{drag}| = \frac{1}{2}\rho A C |\mathbf{v}_{block} - \mathbf{v}_{flow}|^2 \tag{2}$$

where $\rho = 10^3\ kg/m^3$ is the density of water, $V \simeq 20\ cm/s$ the experimentally-measured mean velocity of a ball, $L = 3$ cm the characteristic dimension, and $\mu = 8.90\ .\ 10^{-4}$ Pa.s the dynamic viscosity of water. The submerged area of the globe is, $A = 7\ cm^2$ and the drag coefficient constant in all directions $C = 0.47$. The velocity and acceleration of the ping pong balls are computed using the captured trajectory data. Considering the mass of a ball m, the flow velocity is then computed from Eq. 2 as below, considering $\mathbf{F}_{drag} = m\mathbf{a}_{drag} = m\mathbf{a}_{drag}$:

$$\mathbf{v}_{flow} = \mathbf{v}_{block} + \frac{m\mathbf{a}}{\sqrt{\frac{1}{2}\rho A C m \sqrt{a_x^2 + a_y^2}}} \tag{3}$$

A customized physics plugin was then designed for Webots so that an appropriate drag force is applied to a simulated Lily module based on the velocity of the module and the flow velocity at its location at each time instant. In order to account for rotational effects, the drag force is integrated over each face of the module. Each face is divided into $N = 10$ sections, and the drag force is computed for each section using Eq. 2 with C being the estimated Lily robotic module's drag coefficient C_{Lily}.

For each cell j in the grid of a total of 2500 cells, we record the average μ_j and the standard deviation σ_j of the computed flow velocity vectors. We also test the normality of the distribution in each cell using the KS test. Results, shown in Figure 3, demonstrate that for the majority of the grid cells, the KS test failed to reject the hypothesis that the samples do not belong to a normal distribution with a confidence level of 95%. We can thus assume that the velocity at the location of each grid cell can be drawn from a normal distribution. Therefore, when a block falls in a given cell j, the physics plugin applies the corresponding flow velocity as below, where K_v is a free model parameter to be optimized.


(a)

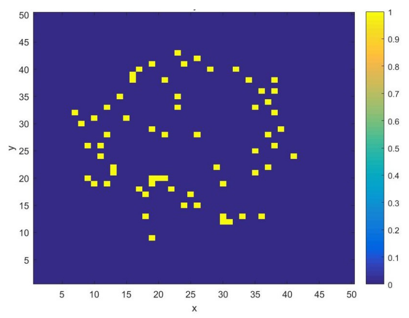

(b)

Fig. 3: Visualization of the KS test results on the captured flow velocity field. The points of higher temperature (yellow color) indicate the grid cells for which the hypothesis that the data points within the corresponding cell have a Gaussian distribution $\mathcal{N}(0,\sigma^2)$ is rejected at a significance level of 5%.

$$v_{flow,j} = K_v \mathcal{N}(\mu_j, \sigma_j) \tag{4}$$

We consider the drag force as below, where K_F is a free optimization parameter.

$$|\mathbf{F}_{drag}| = K_F \frac{1}{2} \rho A C |\mathbf{v}_{block} - \mathbf{v}_{flow}|^2 \tag{5}$$

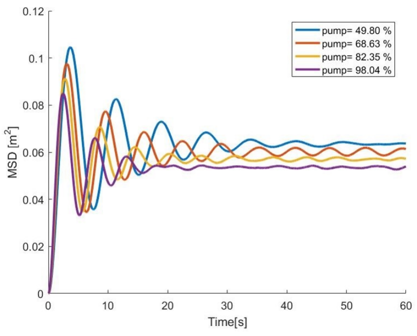
The physics plugin also adds a stochastic force $F_s \sim \mathcal{N}(0, \sigma^2_{stoch})$ to the center of mass of each block in order to take into account the stochasticity and non-modeled effects as in [1]. The standard deviation σ_{stoch} defines our third optimization parameter. Moreover, we have two additional free parameters in Webots which are the linear and angular damping of the block: $D_{linear}, D_{angular}$. In summary, we will have a total of five free optimization parameters to be calibrated: the constants K_v, K_F, σ_{stoch}, D_{linear}, and $D_{angular}$.

4 Calibrating the Model

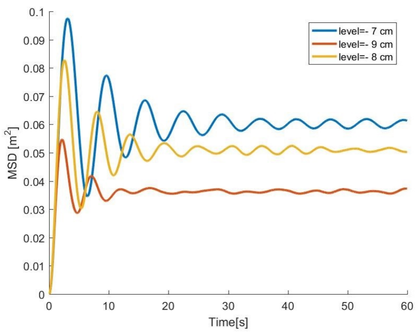
By definition, model calibration is the process of adjustment of the model parameters to obtain a model representation of the processes of interest that satisfies pre-agreed criteria, typically expressed in the form of faithfulness metrics. We use the trajectories of the blocks floating on the simulated and real flow fields and refer to the MSD extracted from each data set. We then define our faithfulness metric to be optimized as the error between the real and simulated MSD functions. We run a PSO algorithm in order to optimize the free model parameters.

4.1 MSD Metric

Diffusion drives mixing in our system. In statistical mechanics, the MSD is a measure of the deviation of the position of a particle with respect to a reference



(a)



(b)

Fig. 4: (a) Effect of changing pump power on MSD. All experiments were done at the same water level, i.e. -7 cm. The pump power is fixed at 68.63%.

position over time. It is the most common measure of the spatial extent of random motion, and can be thought of as measuring the portion of the space "explored" by the random walker. In the realm of biophysics and environmental engineering, the MSD is measured over time to determine if a particle is spreading solely due to diffusion, or if an advective force is also contributing. For instance, MSD analysis is a technique commonly used in colloidal studies and biophysics to determine the dynamics of displacement of particles over time. The MSD is expressed as below.

$$< \Delta r^2(t) > = \sum_{k=0}^{n} < [R_k(t) - R_k(t_0)]^2 > \tag{6}$$

Where R is the position vector, and n is the total number of particles. MSD is usually used to calculate the diffusion coefficient of a given system [2]. In order to verify that the MSD captures the change in the system dynamics, experiments of 10 minute length per water level and pump power configuration were conducted using 24 ping-pong balls. As indicated before, we use ping-pong balls for simplicity as they are symmetric and have an isotropic drag coefficient. We used 3 water levels in the tank, measured from the upper border as -7, -8, and -9 cm, respectively.

Figure 4 illustrates the MSD curves for different pump power and water level settings. Each pump has a maximum flow of 9 *l*/min at 100% power. First, we can notice that the MSD has an oscillating pattern and a convergence plateau. The oscillations are related to a situation where the blocks are affected by the stirring flow generated by the pumps and the frequency of the oscillations is related to the speed of circulation. On the other hand, the plateau indicates a maximum effective displacement explored by the blocks. As we can see in Figure 4(a), the higher the pump power, the higher the frequency of the oscillations and the lower is the plateau. This can be explained considering that when the pumps power is increased, the force pushing the blocks towards the center is higher and therefore the effective radius of the area explored by the blocks is smaller. Furthermore, by keeping the same pump power, but reducing the water level, we can see a similar

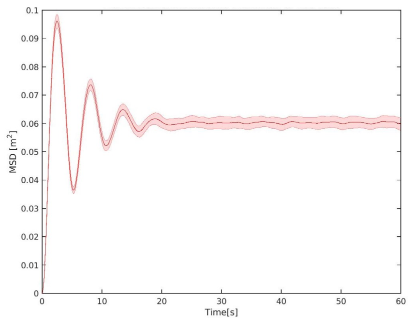
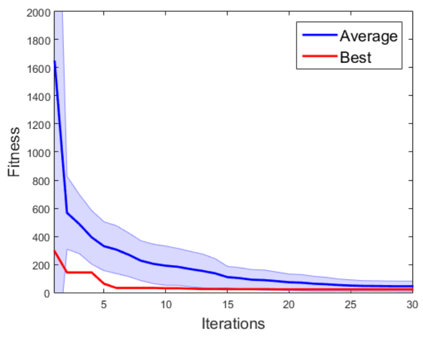


Fig. 5: Mean and standard deviation of MSD of simulated trajectories in Webots. The data is extracted from 100 runs.

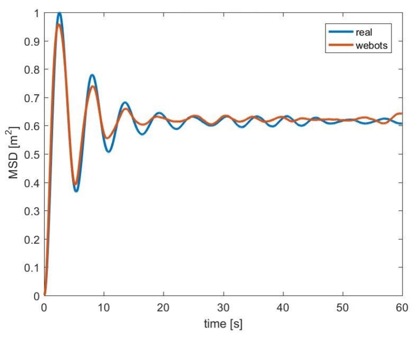
effect. Lowering the water level will change the alignment between the pumps' nozzles and the surface of the water, therefore increasing the fluidic force acting on the blocks' which in turn will decrease the effective exploration radius.

4.2 Parameter Optimization

As discussed earlier, the basic principles of the calibration is to match the simulated and real MSD curves in an attempt to faithfully reproduce the dynamics of the real system in simulation. There are five model parameters to be tuned in the calibration process; K_v defining a scaling factor for the randomness in the velocity field as described in Eq. 4, K_F defining a scaling factor in the drag force as described in Eq. 5, σ_{stoch} defining the standard deviation of the stochastic force field, and the two linear and angular damping coefficients used by Webots for any body mass expressed as D_{linear} and $D_{angular}$, respectively. As mentioned before, we use a PSO algorithm to fine tune these parameters so that real and simulated MSD curves are as aligned as possible. The optimization takes place only within a specific range for each of these parameters in order to ensure the stability of the simulation. The PSO parameters of inertia, personal best coefficient, and global best coefficient are set to -0.1832, 0.5287, and 3.1913, respectively, according to [12]. No particular attempt to optimize the PSO parameterization was carried out. To verify the necessity to use a noise-resistant version of PSO, we have to verify the amount of noise characterizing the chosen fitness function described in Eq. 7. To do that, we carried out 100 runs of the same simulation setup with random initialization of the five parameters, and we calculate the mean and standard deviation. The result is illustrated in Figure 5. We note that the standard deviation is small enough to assume that a noise-resistant PSO will not be necessary. The fitness function is therefore the difference between the resulting MSD from simulation and the mean MSD measured on our real set-up. When computing the MSD value the trajectories of each of the floating blocks are aggregated, rather than being considered separately, and the resulting MSD curves are averaged. Mathematically, we can formulate as below, where N_s is the number of time steps in one sample. Where $N_s = 2400$ in our case corresponds to 60 s emulated wall-clock time with a simulation time step of 25 ms.



(a)



(b)

Fig. 6: Ping-pong balls experiment: (a) Learning of the fitness function throughout the PSO algorithm iterations. (b) Comparison of simulated and real MSD data.

$$Fitness = \sum_{i=1}^{N_s} |MSD_{webots} - MSD_{real}| \tag{7}$$

5 Experiments and Results

In what follows, we will consider two cases: we will first apply the optimization to the case of trajectories obtained from experiments with floating ping-pong balls for a specific pair of water level and pump power. The optimized model is then used as an starting point for the PSO optimization using Lily robotic modules. We then apply the optimization to match the MSD curves in the case of trajectories obtained from experiments with Lily robotic modules. In a first step, we consider experimenting with ping-pong balls. We use 24 ping-pong balls, the water level is equal to -8 cm from the edge of the tank, the pump power is set to 68.6 %. The PSO results as well as the matching MSD are shown in Figure 6. The simulation as well as the optimized model parameters are listed in Table 1.
As a validation step, we compare the normalized number of samples in both Webots and real experiment, and we can see a matching between the two as illustrated in Figure 7. The number of samples in each cell gives an idea about the spatial distribution of the floating objects in the arena, i.e the cells with highest number of samples will indicates the most visited cells. Consequently, a visual

Table 1: PSO algorithm parameters and optimized physics-based model parameters for simulated and real experiments with ping-pong balls and Lily robotic modules.

Floating blocks	# Dimensions	# Iterations	Swarm size	K_v	K_F	σ_{stoch}	D_{linear}	$D_{angular}$
Ping-pong balls	5	30	47	1.24	2.266	195.25	0.3483	0.2476
Lilies	5	100	47	1.365	0.442	126.82	0.332	0.12

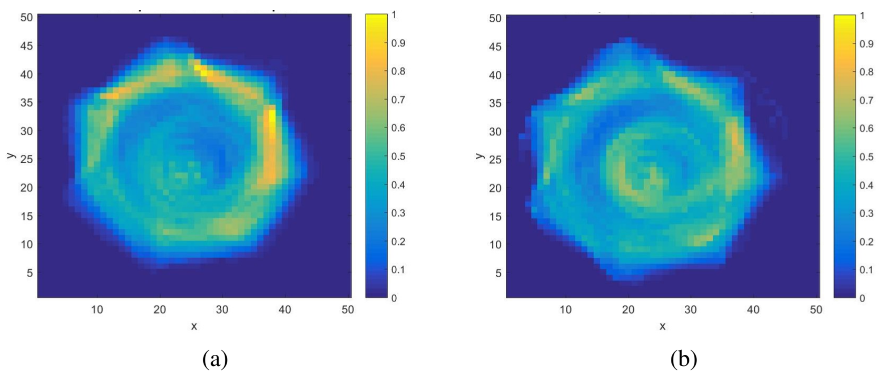


Fig. 7: Comparison of (a) real, and (b) simulated number of samples. The data is extracted from 10 minutes of experiment. The points of higher temperature (yellow color) indicate the grid cells for which the normalized presence of the floating blocks, i.e. the ping-pong balls, were more frequent.

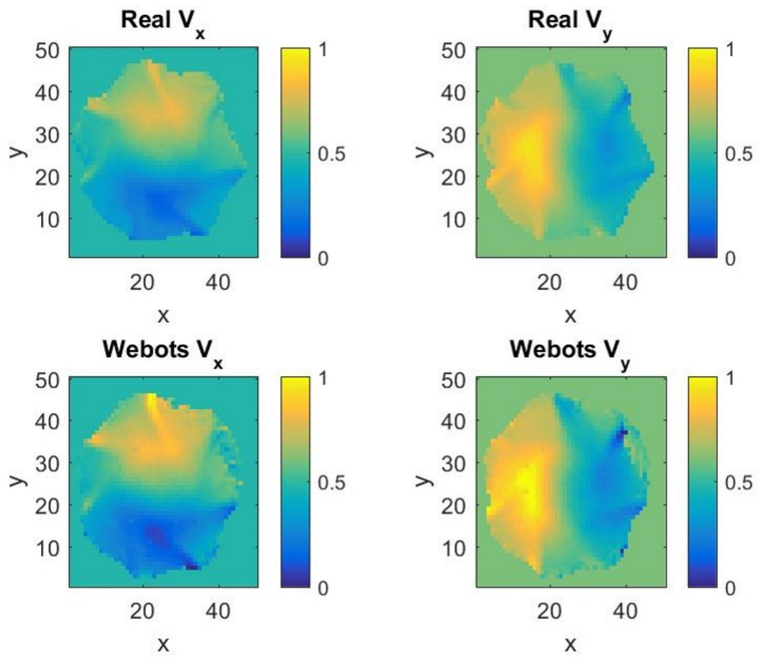
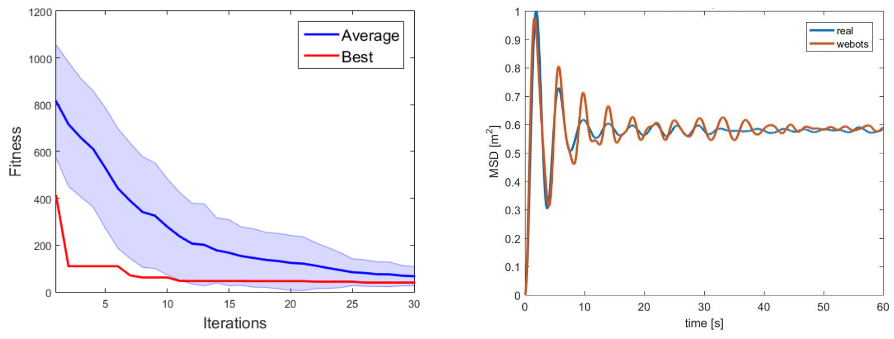


Fig. 8: Comparison of real (top row) and simulated (bottom row) mean velocity.

inspection of the matching between the left and right plots of Figure 7 will indicate similar dynamics. This result is confirmed by comparing the mean velocities



(a) (b)

Fig. 9: Lily modules experiment:(a) Learning of the fitness function throughout the PSO algorithm iterations. (b) Comparison of simulated and real MSD data.

in x and y directions as depicted in Figure 8, where the data is extracted from 10 minutes of experiment.

In a second step, we consider experimenting with Lily robotic modules. The optimized model parameters obtained from experimenting with ping-pong balls are used as the initialization values for the PSO algorithm run for Lily robotic modules. We use 15 Lilies, the water level is -8 cm and the pump power is set to 68.6 %. The PSO results as well as the resulting matching MSD are shown in Figure 9 and the used simulation parameters as well as the optimized model parameters are listed in Table 1. It can be seen that the MSD curves from the simulated and real Lily robotic modules have a close matching using the optimized model parameters.

6 Conclusion

In this paper, we addressed the problem of designing and calibrating physics-based models of stochastic self-assembly of water-floating robotic modules. In particular, we considered the case of our fluid-mediated self-assembling robotic system and used the Webots robotic simulator to capture a faithful yet computationally low-weight physics-based model of the system. The main motivation for our approach is to develop accurate yet computationally light simulation models in order to allow for investigating different programmable behaviors on the robotic modules without having to compromise on the simulation speed due to computationally heavy simulation of fluid dynamics in the system. To this end, we first recreated our robotic modules' hardware and firmware in the simulated world. We then developed a dedicated physics plugin to apply appropriate hydrodynamic force on the simulated robotic modules. In particular, we introduced a novel method for automatically calibrating the model parameters employing a PSO algorithm.

Acknowledgement

This work has been partially sponsored by the Swiss National Science Foundation under the grant numbers 200021_137838/1 and 200020_157191/1.

References

1. Di Mario, E., Mermoud, G., Mastrangeli, M., Martinoli, A.: A trajectory-based calibration method for stochastic motion models. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 4341–4347 (2011)
2. Frenkel, D., Smit, B.: Understanding molecular simulation: From algorithms to applications (Academic, San Diego, 2002) pp. 63–107 (1997)
3. Ganesan, V., Chitre, M.: On stochastic self-assembly of underwater robots. IEEE Robotics and Automation Letters 1(1), 251–258 (2016)
4. Haghighat, B., Droz, E., Martinoli, A.: Lily: A miniature floating robotic platform for programmable stochastic self-assembly. In: IEEE International Conference on Robotics and Automation. pp. 1941–1948 (2015)
5. Haghighat, B., Martinoli, A.: Characterization and validation of a novel robotic system for fluid-mediated programmable stochastic self-assembly. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2778–2783 (2016)
6. Haghighat, B., Mastrangeli, M., Mermoud, G., Schill, F., Martinoli, A.: Fluid-mediated stochastic self-assembly at centimetric and sub-millimetric scales: Design, modeling, and control. Micromachines 7(8), 138 (2016)
7. Jacot-Descombes, L.: Fluid-mediated Self-assembly of MEMS Micro-capsules for Liquid Encapsulation and Release. Ph.D. thesis (2013)
8. Lochmatter, T., Roduit, P., Cianci, C., Correll, N., Jacot, J., Martinoli, A.: Swistrack-a flexible open source tracking software for multi-agent systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 4004–4010 (2008)
9. Matthey, L., Berman, S., Kumar, V.: Stochastic strategies for a swarm robotic assembly system. In: IEEE International Conference on Robotics and Automation. pp. 1953–1958 (2009)
10. Michel, O.: Webots: Professional mobile robot simulation. Advanced Robotic Systems 1(1), 39–42 (2004)
11. Pavlic, T.P., Wilson, S., Kumar, G.P., Berman, S.: Control of stochastic boundary coverage by multirobot systems. Journal of Dynamic Systems, Measurement, and Control 137(3), 034504 (2015)
12. Pedersen, M.E.H.: Good parameters for particle swarm optimization. Hvass Lab., Copenhagen, Denmark, Tech. Rep. HL1001 (2010)
13. Yim, M., Shen, W., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.: Modular self-reconfigurable robot systems [grand challenges of robotics]. IEEE Robotics & Automation Magazine 14(1), 43–52 (2007)