Supplementary Information for "Self-Reconfiguring Modular Robotic Boats"

Wei Wang[1,†,*], Niklas Hagemann[2,*], Alejandro Gonzalez-Garcia[3,*], Carlo Ratti[4,5], Daniela Rus[2,†]

[1]Marine Robotics Lab, Department of Mechanical Engineering, College of Engineering, University of Wisconsin-Madison, Madison, USA

[2]Computer Science and Artificial Intelligence Lab (CSAIL), Massachusetts Institute of Technology, Cambridge, USA

[3]MECO Research Team, Department of Mechanical Engineering, KU Leuven, Belgium

[4]Senseable City Laboratory, Massachusetts Institute of Technology, Cambridge, USA

[5]ABC Department, Politecnico di Milano, Milano, Italia

[*]These authors contributed equally to this work.

Supplementary Note 1

Mini Thruster Design

Due to the lack of commercially available brushless thrusters of suitable size for the miniature robotic boats, we design custom mini thrusters, with mechanical details provided in (see Supplementary Fig. 3b). To facilitate integration and maintenance, we further develop detachable holders for the thrusters (see Supplementary Fig. 3a). Additionally, to enhance the rotational inertia of the robot and improve yaw control, we implement a detachable fin, as shown in see Supplementary Fig. 3a.

Electronics and Sensors

Each *FloatForm* module relies on an onboard embedded computer (Raspberry Pi 4), running Ubuntu 20.04 operating system, and Robot Operating System (ROS). The Raspberry Pi controls the robot's basic behaviors, communication with other agents, and interactions with a base station. Each robot also includes an STM32 microcontroller for lower-level control of peripheral hardware. The microcontroller receives force and latching commands from the Raspberry Pi. It translates these commands into signals for the Electronic Speed Controllers (ESCs), which drive the thrusters and signals to the micro-servo motor to actuate the latching mechanism. On the sensing side, a low-cost Adafruit BNO055 IMU provides angular velocity data, and two Marvelmind acoustic beacons provide position and heading data. Experimental data is logged onto an onboard microSD card. All electronics are connected to a custom printed circuit board (PCB). An 11.1-V 2650-mAH Li-Po battery provides power for up to 3 hours of runtime and is placed in the base of the hull for stability. Assembling a module takes around 2 hours: starting with the assembly of the thrusters, placement of the battery and electronics, wiring, fixation with screws, placement of the latching system, and fixation of the localization beacons on the acrylic cover.

Localization

Localization of all robots is facilitated using the Marvelmind Navigation System: an off-the-shelf indoor navigation system, designed to provide precise (± 2 cm) location data to autonomous robots via mobile ultrasonic beacons. The navigation system consists of a network of stationary beacons interconnected via a radio interface in a license-free band; two mobile beacons installed on each module to be tracked; and a modem providing a gateway to the system from a base-station computer. The location of each mobile beacon can be inferred from the propagation delay of ultrasonic pulses (Time-Of-Flight or TOF) between stationary and mobile beacons using a trilateration algorithm. An Extended Kalman Filter is then used to provide a more accurate and higher frequency estimation of the robot state (pose and velocity) by fusing the data from the beacons with values from the onboard IMU. Nonetheless, the localization data from the robots is naturally noisy due to two main factors. First, the confined water surface causes complex multi-path effects that affect trilateration. Second, the presence of numerous neighbors causes consistent disturbances that further impact trilateration. As a result, incomplete and imperfect representations may occur during self-reconfiguration behaviors.

Each *FloatForm* robot can obtain its neighbor's position at a sampling rate of 50 Hz using a multi-master communication framework implemented in ROS via Wi-Fi. Although each robot can theoretically know the position of all its neighbors within our testing area, the communication range during experiments was artificially limited by actively discarding messages originating outside the desired communication range (0.5 m in our case). This limited range reduces the communication load between neighbors and allows the coordination algorithm to be compatible with different sensing or communication strategies, such as infrared or cameras.

Supplementary Note 2

<u>Thrust allocation</u>

Given that the thrusters used in the robot are incapable of bi-directional motion (they only provide forward thrust), have a limited rotational speed, and the vehicle is over-actuated, the following procedure is required to allocate the appropriate command into each thruster. First, consider the forces and moments vector $\tau = [\tau_u, \tau_v, \tau_r]^T$ and the actuators vector $f = [f_1, f_2, f_3, f_4]^T$, where its relationship is:

$$f = B^{+}\tau \tag{1}$$

Notice that the Moore-Penrose pseudo-inverse is used since B is not a square matrix. Nevertheless, this allows negative values to be sent to each thruster, which would mean reversing the thruster blade's rotation. Then, each desired thruster signal needs to be mapped to ensure a positive command. First, the minimum value $f_l = \min(f)$ is computed. If $f_l < 0$, then $f_i = f_i - f_l$, with $i = 1,2,3,4$.

However, the thrust could still be larger than the maximum value $f_{\max}$. Likewise, if each thruster is limited to $f_{\max}$, the thrust ratio or direction could be lost, resulting in a different movement than the desired one, i.e., the motion vector would be redirected. Thus, the maximum command is then identified as $f_h = \max(f)$. If the maximum command is larger than the maximum allowed thrust, $f_h > f_{\max}$, then $f_i = \frac{f_i f_{\max}}{f_h}$, which maintains the original motion direction. Nevertheless, this means the rotational velocity may be diminished given its different performance range compared to linear velocities. Hence, the procedure is followed again using a new vector $\hat{\tau} = Bf$, where $\hat{\tau}_r$ is replaced by the original heading controller τ_r.

Supplementary Note 3

<u>Centralized Task assignment algorithm to solve local imperfections</u>

For the centralized part of the proposed system, each module is assigned a position from a shape matrix G. The position assignment is introduced to ensure that a perfect structure is achieved, as the potential field algorithm can approximate a shape without guarantees to avoid imperfect square lattices. First, the shape matrix $G = [g_1, g_2, \dots, g_M]^T$ is composed of goal positions $g_m \in R^2$, for $m = 1,2, \dots M$, where M is the total number of goals/robots (assuming the shape is designed with the same number of available goal positions as there are robots). Next, a distance matrix D is computed using the distance between each *FloatForm* module's position p_k and each goal g_m as

$$D_{k,m} = ||p_k - g_m||^2 \tag{2}$$

Then, an assignment matrix Φ is considered, where each matrix position is set to 1 if the robot is assigned to a goal m, or to 0 if the robot is not assigned to said goal m. Thus, it results in the following linear assignment problem:

$$\sum_{k=1}^{M} \sum_{m=1}^{M} \Phi_{k,m} D_{k,m} \tag{3}$$

which is solved via the Hungarian algorithm[1]. After each module has received a fixed position, the decentralized position-reference potential field algorithm drives each module to its respective goal.

Distributed position-reference algorithm towards perfect assembly

At the final stage of the shape formation process, once the task assignment algorithm has assigned each robot a desired position, a position-reference formation algorithm is employed to bring the swarm gradually into the correct configuration. Specifically, each robot receives a position reference, p_d, that indicates its designated location within the target shape G. This algorithm is based on the Artificial Potential Field method and will compute the desired velocities $[u_d, v_d]^T$ based on the potential force F_p (Equation 9 in the Methods section). The repulsive force F_r, responsible for preventing collisions and maintaining safe spacing between neighboring robots, remains unchanged from Equation 11 in the Methods section. However, the attractive force, which pulls each *FloatForm* module toward its assigned goal p_d, is defined by a different equation. The attractive force uses the error vector $e_p = p_d - p$, *i.e.*, the difference between the desired position and the robot's current position. Mathematically, the attractive force is expressed as:

$$F_{a,p} = k_{a,3}\exp(\| e_p \|)\, e_p \tag{4}$$

where $k_{a,3}$ is a gain parameter, $\exp(\| e_p \|)$ is an exponential term that increases with the distance to the goal p_d, and e_p is the current positional error. Again, the potential field force F_p is initially calculated in the inertial reference frame and then transformed into u_d, v_d with equation 14 in the Methods section. Thus, the surge and sway speed controllers can drive the module to its assigned position p_d.

<u>Collective transport algorithm</u>

When the swarm has latched together into a floating structure, the system can perform collective motion. In this sense, the assembled structure can hold its position or navigate as a single unit. The collective transport algorithm uses a shape matrix $G_0 = [g_{01}, g_{02}, \dots, g_{0M}]^T$, where $g_{0\mathrm{m}} = [\Delta_{xm}, \Delta_{ym}]^T$ are the relative positions with respect to the shape center, where $m = 1, 2, \dots, M$. When the collective structure is tasked to move as a unit, a target shape $G_T(g_{cT}) = [g_{T1} + g_{cT}, g_{T2} + g_{cT}, \dots, g_{TM} + g_{cT}]^T$ is computed based on the desired target shape center $g_{cT} \in R^2$ and the structure desired orientation θ_d angle:

$$g_{Tm} = R(\theta_d) g_{0m} \tag{5}$$

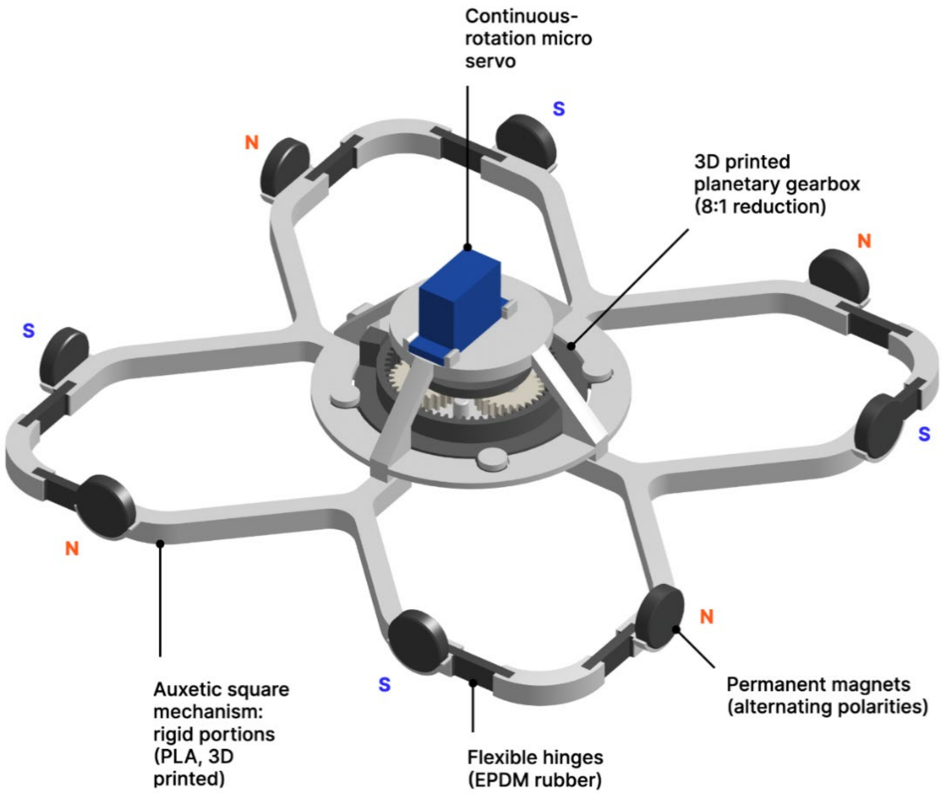
where $R(\theta_d)$ is a rotation matrix dependent on θ_d:

$$R(\theta_d) = \begin{bmatrix} \cos\theta_d & -\sin\theta_d \\ \sin\theta_d & \cos\theta_d \end{bmatrix} \tag{6}$$

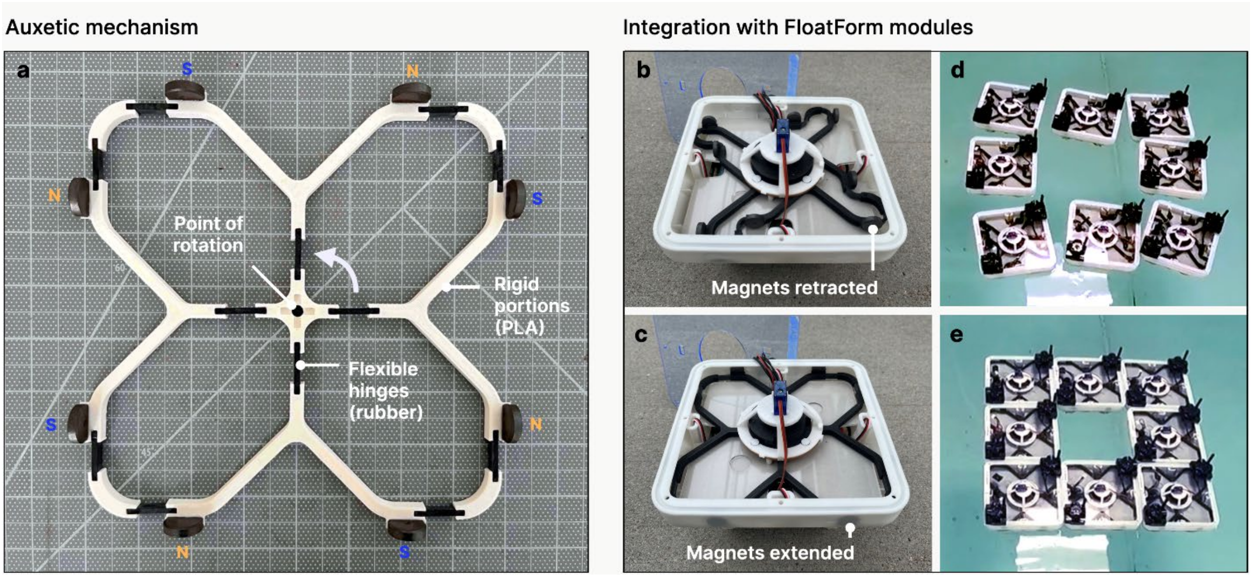
This computation describes the translation and rotation of the shape matrix G_0 to the desired position and orientation of the floating structure. Following the task assignment solution, each robot can receive its updated desired position p_d, and navigate there using the attractive force from Equation 4.

References

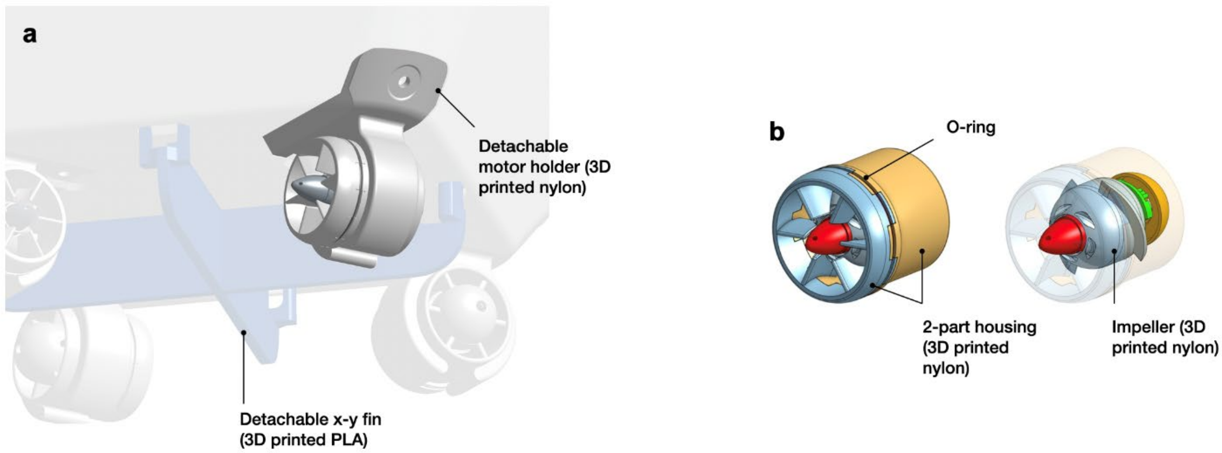
1 Kuhn, H. W. The Hungarian Method for the assignment problem. *Nav Res Log* **52**, 7-21 (2005). https://doi.org/DOI 10.1002/nav.20053



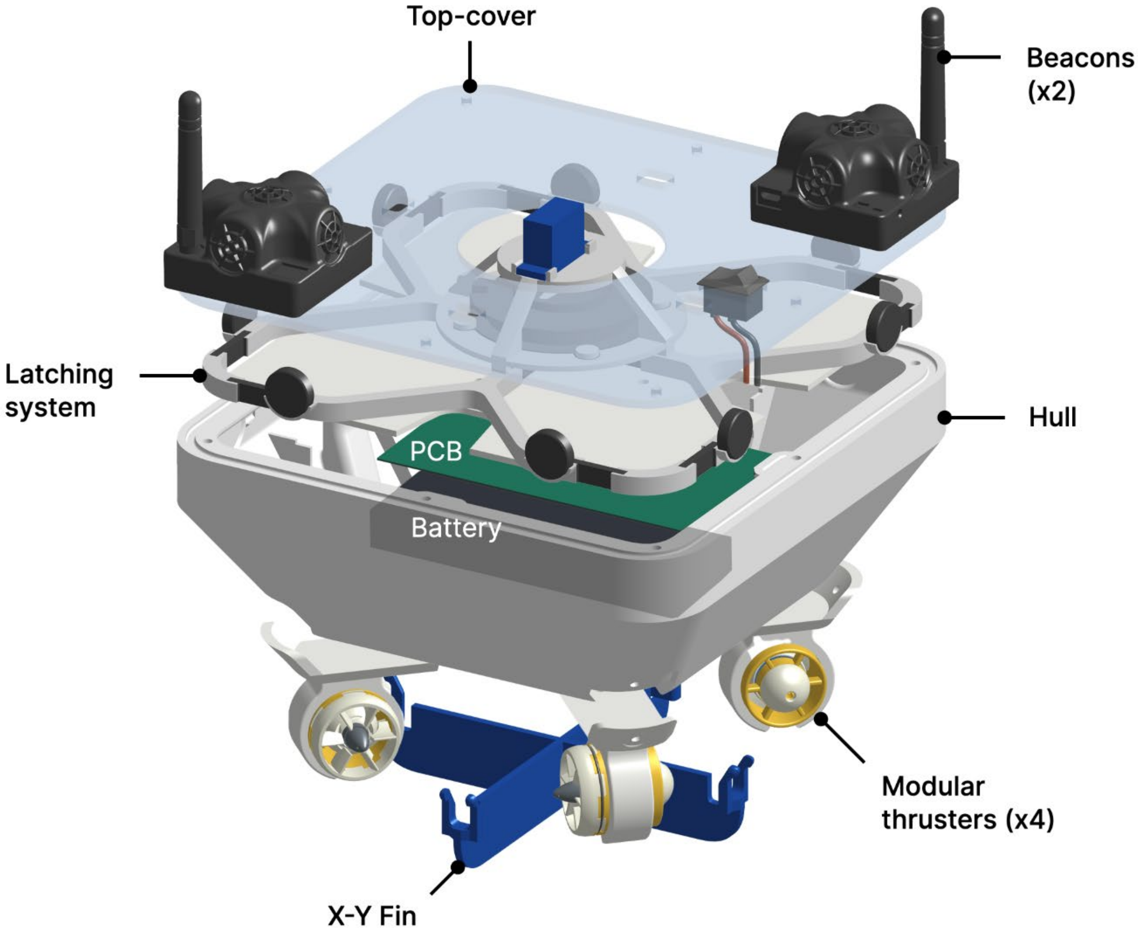
SUPPLEMENTARY FIGURE 1 Mechanical details of the latching mechanism showing the servo motor, origami-inspired auxetic mechanism, and 3D printed gearbox assembly.



SUPPLEMENTARY FIGURE 2. Latching mechanism design. (**a**) The auxetic mechanism before assembly with the motor and gearbox, showing rigid (3D printed PLA) and flexible (EPDM rubber) portions. Magnets are arranged in an alternating fashion; rotation at the center causes the mechanism to contract from all sides, bringing the magnets into their retracted (de-latched) positions (**b** and **d**). **c**, and **e** show the mechanism in its latched state.



SUPPLEMENTARY FIGURE 3 Mechanical details of the robot thrusters: (**a**) showing the detachable fin for altering the rotational inertia of the boat and detachable holders for the thrusters, (**b**) mechanical details of the customized mini thrusters that were developed.



SUPPLEMENTARY FIGURE 4 Exploded view of the robot assembly.

Supplementary Video 1: Motion Demonstration

https://youtu.be/n2NeAlJllRM

Demonstrates individual module maneuverability in water.

Supplementary Video 2: Latching Mechanism

https://youtu.be/KXmRcd14U6k

Illustrates the magnetic latching process between modules during docking and undocking.

Supplementary Video 3: Self-Assembly and Reconfiguration with Four Modules

https://youtu.be/PDxQCSw4xlU

Shows self-assembly and reconfiguration using a small-scale four-module setup.

Supplementary Video 4: Self-Assembly, Reconfiguration, and Collective Transport with Eight Modules

https://youtu.be/DidZ9nz3ax4

Demonstrates self-assembly, reconfiguration and coordinated transport using eight modules.