Blueswarm: 3D Self-organization in a Fish-inspired Robot Swarm

A DISSERTATION PRESENTED
BY
FLORIAN BERLINGER
TO
THE DEPARTMENT OF COMPUTER SCIENCE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
COMPUTER SCIENCE

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
APRIL 2021

Advisor: Professor Radhika Nagpal Florian Berlinger

Blueswarm: 3D Self-organization in a Fish-inspired Robot Swarm

Abstract

Animals team up to collectively address challenges they could not overcome individually. Several species self-organize into large groups to leverage vital behaviors such as foraging, construction, or predator evasion. Ants, for instance, find shortest paths to food resources by depositing pheromones, bees indicate direction and distance to flower meadows through waggle dances in the hive, and fish display evasive maneuvers to escape predators. These three examples illustrate a collective problem-solving ability that leverages the cognition and actions of individually limited organisms.

With the advancement of robotics and automation, engineered multi-agent systems have been inspired to achieve similarly high degrees of scalable, robust, and adaptable autonomy through decentralized and dynamic coordination. Scientists have demonstrated ground-based collective transport, construction, and self-assembly, in some cases with several hundred robots. Multiple aerial swarms fly complex maneuvers, some of them even with little external assistance. Small robot teams, although with limited autonomy, have been engaged to assist in search and rescue missions at sea, sample oceanic data, and find unknown deep-sea species.

Overall however, robot swarms have been most successfully demonstrated in two-dimensional (2D) space or with partial assistance from central controllers and external tracking. In addition, many more demonstrations of self-organized collectives exist above-ground as opposed to the less explored underwater domain, which is particularly challenging because it often precludes traditional communication methods such as radio and GPS signals. Few underwater swarms exist and achieve limited coordination complexity and scale because they rely on explicit message passing.

In this dissertation, I introduce a novel underwater robot collective, the Blueswarm, which realizes full 3D spatiotemporal coordination without any external assistance. Each Bluebot is equipped with four independently controllable fins and two wide-angle lens cameras for 3D locomotion and perception. The vision system is complemented by three LEDs, which encode information about direction, distance and heading, and facilitate implicit coordination among robots. In the bioinspired design process, I pursued simplicity in both hardware and software to enable real-time onboard multi-robot tracking for local decision making followed by swift action. Blueswarm is the first 3D underwater collective that uses only local implicit vision-based coordination to self-organize.

Inspired by the dynamic and agile coordination of fish, I show that complex and dynamic 3D collective behaviors — synchrony, aggregation-dispersion, dynamic circle formation, search-capture, and escape — can be achieved by sensing minimal, noisy impressions of neighbors without any centralized intervention. To the best of my knowledge, this is the first significant demonstration of unsupervised and autonomous 3D collective coordination underwater.

Accompanied by a custom simulator, the Blueswarm platform gives researchers a much-needed tool to systematically develop and test algorithms for self-organzied 3D collective behaviors in the laboratory. The results of this dissertation provide insights into the power of implicit coordination and advance the potential for future underwater robots that display collective capabilities on par with fish schools for applications such as environmental monitoring and search in coral reefs and coastal environments. In addition, the Bluebots are also well suited as an experimental testbed for investigating natural collective behaviors and biomimicry, for example, studying the energy savings for different formations in schooling fish or the performance landscape of aquatic propulsion with a diverse set of caudal fins.

Contents

List of figures

List of publications

Parts of this dissertation are based on and reproduced from manuscripts which have been published. The Science Magazine composed an excellent Blueswarm video that makes the concepts explained here more tangible. A playlist with additional supplementary videos can be found on the YouTube channel of the Self-organizing Systems Research Group.

Chapters 2, 3, 5

F. Berlinger, M. Gauci, R. Nagpal, Implicit coordination for 3D underwater collective behaviors in a fish-inspired robot swarm. *Sci Robot.* **6**, eabd8668 (2021).

Chapter 3

F. Berlinger, J. Dusek, M. Gauci, R. Nagpal, Robust maneuverability of a miniature, low-cost underwater robot using multiple fin actuation. *IEEE Robotics and Automation Letters 3*, 140-147 (2017).

K. Soltan, J. O'Brien, **F. Berlinger**, R. Nagpal, J. Dusek, Biomimetic actuation method for a miniature, low-cost multi-jointed robotic fish. *MTS/IEEE OCEANS Charleston*, pp. 1-9 (2018).

Chapter 3, 4, 6

F. Berlinger, P. Wulkop, R. Nagpal, Radhika, Self-Organized Evasive Fountain Maneuvers with a Bioinspired Underwater Robot Collective. *IEEE International Conference on Robotics and Automation (ICRA)*, (2021).

Appendix B

F. Berlinger, M. Saadat, H. Haj-Hariri, G.V. Lauder, R. Nagpal, Fish-like three-dimensional swimming with an autonomous, multi-fin, and biomimetic robot. *Bioinspiration & Biomimetics*, 16(2):026018 (2021).

M. Saadat, **F. Berlinger**, A. Sheshmani, R. Nagpal, G.V. Lauder, H. Haj-Hariri, Hydrodynamic advantages of in-line schooling. *Bioinspiration & Biomimetics*, 16(4):046002 (2021).

Appendix C

F. Berlinger, M. Duduta, H. Gloria, D. Clarke, R. Nagpal, R. Wood, A Modular Dielectric Elastomer Actuator to Drive Miniature Autonomous Underwater Vehicles. *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3429-3435 (2018).

M. Duduta, **F.C.J. Berlinger**, R. Nagpal, D.R. Clarke, R.J. Wood, F.Z. Temel, Electrically-latched compliant jumping mechanism based on a dielectric elastomer actuator. *Smart Materials and Structures*, 28(9), p.09LT01 (2019).

M. Duduta, **F. Berlinger**, R. Nagpal, D. Clarke, R. Wood, F.Z. Temel, Tunable Multi-Modal Locomotion in Soft Dielectric Elastomer Robots. *IEEE Robotics and Automation Letters*, (2020).

To K.

Acknowledgments

It all started underwater. While diving in the pristine Galápagos Islands in 2014, I encountered fish of all appearances. Moving fast and slow, striking out on their own or grouping up in innumerable schools, they always blended in gracefully with the stunning underwater world. Back then, I had no idea that I would learn so much more about the extravaganza I just witnessed.

I spent the last five years here at Harvard University to study collective behaviors and fish swimming with a swarm of underwater robots. I was surrounded by tremendous colleagues, who guided my way, taught me their crafts, and brought countless smiles to my face.

Radhika (Prof. Radhika Nagpal), thank you for having been my advisor, for having helped me follow my interests and realize my ideas, and for having showed me how to become a successful researcher. Thank you for having placed your trust in me.

Melvin (Dr. Melvin Gauci), building Blueswarm together with you was as awesome as all the moments we shared outside of the laboratory. You were so welcoming and I learnt in many ways from you. Thank you.

Mishu (Dr. Mihai Duduta), you introduced me to the field of materials science. We had great times together building Fishu and what's more, we became such good friends. I hold that very dear.

Mehdi (Dr. Mehdi Saadat), together with you I brought my robots to life, making them look and swim like real fish. Thanks.

Rob and George (Prof. Robert Wood and Prof. George Lauder), you welcomed me to your laboratories to collaborate and widen my horizon. Thanks for all your advice and mentoring.

Daniel Vogt, Fritz Lekschas, Moritz Graule, Paula Wulkop, David Clarke, Lily Xu, Zeynep Temel, Jeff Dusek, Hossein Haj-Hariri, Lucas Guzman, Jim MacArthur, and James Weaver, you were invaluable collaborators, experts, or classmates. Daniel Calovi, Nicole Carey, Melinda Malley, Justin Werfel, Nathan Melenbrink, Julia Ebert, Helen McCreery, Jordan Kennedy, and Bahar Haghighat, you were the labmates providing me with a stimulating and fun environment on a daily basis. Thank you all so much!

It gives me special delight to thank my family. I love you all dearly. If I ever looked good, it was because of you. Jessica gets extra credit for summarizing my work in artistic form on the back cover of this dissertation.

Lastly, I want to acknowledge the funding sources who in part enabled my research. The Wyss Institute for Biologically Inspired Engineering, the Office of Naval Research, Amazon Web Services, and the David B. Heller Innovation Fund contributed significantly toward research expenses and my stipend; the Swiss Study Foundation supported me through their network of experts and complementary extracurricular activities including weekend seminars and summer schools.

My fascination for fish and their artistic choreography-like behaviors started underwater. A desire to understand how they swim, interact, and organize arose. The rest is written in this dissertation.

1
Introduction

The natural world abounds with self-organizing collectives, where large numbers of relatively simple agents use local interactions to produce impressive global behaviors, such that the system as a whole is greater than the sum of its parts[1]. Ants, for instance, find shortest paths to food resources by depositing pheromones on trails (a form of stigmergy)[2]; bees indicate direction and distance to flower meadows through waggle dances in the hive[3]; and fish display evasive maneuvers to escape preda-

Figure 1.1: Blueswarm roaming a coral reef. The seven underwater robots coordinate their actions to explore an unknown environment. (Credit: used under CC0; Bluebots inserted)

tors[4,5]. These three examples illustrate a collective problem-solving ability that leverages the cognition and actions of individually limited organisms. Self-organization is observed across scales, from cells to insects, to birds, fish, and mammals[1]. What makes these systems so fascinating to scientists and engineers alike, is that even though each individual has limited ability, as a collective they achieve tremendous complexity.

1.1 Motivation and Bioinspiration

Picture the swarm of underwater robots from Figure 1.1 investigating a coral reef, searching for people in distress, or patrolling underwater infrastructure such as ports and offshore wind parks. After having been thrown into the water, the robots are on their own. They start out with controlled dispersion to explore their surroundings, aggregate again and align their headings to migrate collectively to the next waypoint, show elegant evasive behaviors to circumnavigate oncoming traffic, and complete their task in record time by exploiting parallel action. The data collected by the robot swarm allows scientists, rescue forces, and infrastructure operators to monitor their sites and plan their actions.

While such scenarios are a rather far cry from the dynamic coordination underwater robots achieve today, the underwater world is filled with organisms which display stunning group behaviors. Fish schools are particularly impressive — collectives of thousands migrate long distances in the ocean, efficiently search for resources, swim up rivers to reproduce, and even form dynamic shapes like flash expansions or bait balls to evade predators and capture prey[4–7] (Fig. 1.2 A). Even more inspiring are the fish schools that move within coral reefs, coordinating in complex cluttered environments.

In general, biological collectives exhibit properties of autonomy that are highly desirable from an engineering perspective: i) complex and coherent behavior over large physical scales, from individuals whose perception is severely limited; ii) decentralized coordination and high levels of robustness to the failure of individuals; iii) efficiency and adaptability by exploiting parallelism in actions and sensing. Through self-organization, natural systems achieve a high degree of scalable, robust, and adaptable autonomy[1,6].

The research field of collective robotics draws inspiration from these natural systems and aims to engineer their attractive properties into de novo artificial systems[8]. Scientists have demonstrated ground-based collective transport[8], construction[9], and self-assembly[10] (Fig. 1.2 B). Multiple aerial swarms fly complex maneuvers (Fig. 1.2 C), some of them even with limited external assistance[11–15]. As impressive are the several hundred thousand robots operating in Amazon warehouses (Fig. 1.2 D), the small team consisting of the Perseverance rover and Ingenuity helicopter that is currently deployed on Mars, or the rapid progress on self-driving cars (Fig. 1.2 E).

Robots have also been engaged to assist in search and rescue missions at sea[16], sample oceanic data[17–20], and find unknown deep sea species[21]. However, compared with above-ground collectives, underwater robotic systems have not yet been able to achieve similar levels of self-organization. Aquatic environments impose substantial challenges on perception and locomotion and especially

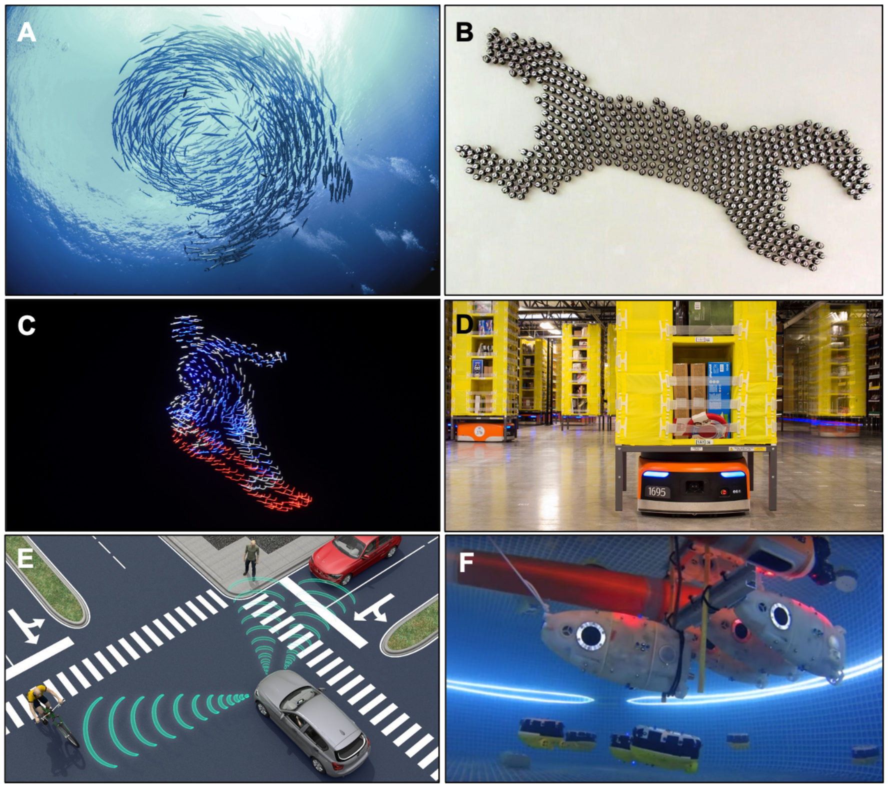


Figure 1.2: Collectives — from nature to the real world. **(A)** A school of barracudas milling. (Source: iStock) **(B)** Shape formation with $1,000$ Kilobots[10]. **(C)** Intel's Shooting Stars during the 2018 Winter Olympics. (Credit: Intel Corporation) **(D)** Robotic fulfillment in an Amazon warehouse. (Credit: Reuters/Noah Berger) **(E)** A self-driving car interacting with other road users. (Credit: ID 87117864 © Thelightwriter | Dreamstime.com) **(F)** The heterogeneous CoCoRo underwater robot collective[23].

limit communication and sensing[22]; traditional above-ground communication methods such as wireless radio perform poorly underwater, and position localization methods such as the Global Positioning System (GPS) are unavailable. As a result, most underwater swarms coordinate only at the surface or have no coordination whatsoever. Limited experimental studies of submerged collective behavior were published[23] (Fig. 1.2 F). A platform for the systematic study of collective behaviors in the laboratory is missing.

In my dissertation, I designed Blueswarm, a swarm of miniature underwater robots, and demonstrated multiple complex three-dimensional (3D) underwater collective behaviors. Inspired by schooling fish, the Bluebots have four independently controllable fins and two wide-angle lens cameras for 3D locomotion and perception. The swarm uses mostly implicit vision-based coordination to self-organize without any externalized assistance in position sensing or control. Such implicit coordination enables scalable and robust swarming because it not only is naturally decentralized and robust to individual failures but also reduces communication complexity in environments where direct explicit message passing is not possible or not desired. Blueswarm represents an important advance in the experimental investigation of underwater 3D self-organized collective behaviors. The Bluebots are also well suited as an experimental testbed for investigating natural collective behaviors and biomimicry, for example, studying dynamic evasive maneuvers or energy savings for different formations in schooling fish.

1.2 Objectives

The overarching objective of this dissertation was to demonstrate three-dimensional and fully decentralized collective behaviors with a novel robot swarm, Blueswarm. To pursue this objective, efforts in hardware and software were necessary. The following list summarizes relevant design goals:

- **A platform to investigate collective behaviors:** I prioritized user-friendliness, ease of manufacture, small size, and low cost in order to create a scalable multi-robot platform geared toward the study of collective behaviors in the laboratory not only by myself, but also by other researchers. I aimed to complement the physical robots by a simulator for the rapid development and extended analysis of algorithms.
- **A robot with 3D locomotion and perception:** The majority of self-organized and autonomous swarms is ground-based. To create robots capable of 3D collective behaviors, I focused on high degrees of maneuverability and comprehensive perception.

- **Unsupervised collective behaviors:** Robotic systems are most adaptive, scalable, and robust if they do not rely on external assistance during operation. In addition, above-ground assistive technologies such as wireless communication to base stations or position localization methods like GPS perform poorly or are unavailable underwater. My goal for Blueswarm was complete autonomy and full decentralization to realize unsupervised and unguided collective behaviors.

- **Bioinspiration:** Because fish swimming and schooling remain unparalleled when compared to robotic systems, I aimed to identify and mimic helpful aspects of the aquatic locomotion and group behavior seen in fish.

1.3 Contributions by a Novel 3D Robot Swarm

My work with Blueswarm makes three major contributions to the field of collective robotics: first, a novel 3D robot swarm; second, a variety of collective behaviors demonstrated underwater; and third, the thorough analysis with insights into the robustness and scalability of those behaviors. A preview of contributions with Blueswarm is given in Figure 1.3.

1. **Design of an underwater robot swarm with 3D sensing and motion**

   Blueswarm is the first 3D underwater collective that can self-organize with local implicit vision-based coordination only. For the systematic study of 3D coordination underwater, I designed miniature, autonomous, fish-inspired underwater robots called Bluebots. Two individual-level capabilities stand out: 3D awareness of neighbors and swift response to neighbors' actions. I realize these capabilities using a suite of sensors and actuators that enables visual perception and multi-fin locomotion along all three dimensions in space.

   Bluebot's streamlined body and multi-fin propulsion was inspired by fish such as the blue tang. The pursuit of bioinspired design solutions inspired two collaborations on biomimicry of fish swimming and schooling, and smart materials for soft robotics.

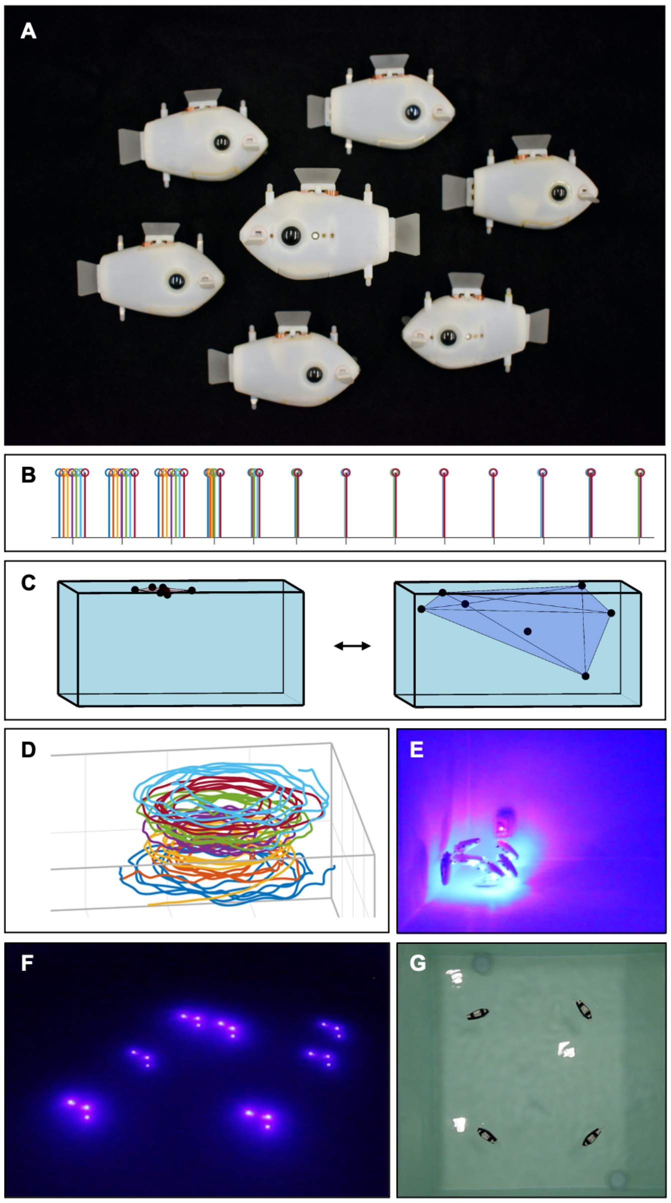


Figure 1.3: Preview of Blueswarm behaviors. (A) The seven Bluebots designed for collective behaviors. **(B)** Synchronization of LED flashes, represented by colored peaks that coincide over the course of an experiment. **(C)** 80-fold changes in occupied volume during aggregation and dispersion, illustrated as convex hulls. **(D)** Robot trajectories tracked from a milling experiment. **(E)** Robots gathering around a red light source after a successful search operation. **(F)** Aligned robots before predator evasion. **(G)** Colored robots milling without LEDs.

2. **Design of robust algorithms to demonstrate a variety of collective behaviors**
   I demonstrated multiple complex 3D collective behaviors underwater, several of them inspired by schooling fish. I showed that a variety of collective behaviors — coordinating time, space, dynamics, and task sequencing — can be achieved using very simple modes of communication, and without any externalized assistance in sensing or control. This work highlights the power of decentralized autonomy, providing experimental evidence and new theoretical results for 3D collectives. To the best of my knowledge, this is the first significant demonstration of unsupervised 3D collective coordination underwater.

3. **Simulation-based analysis into the robustness and scalability of collective behaviors**
   To extend the study with the physical robots, I developed a custom simulation environment that is carefully tuned to replicate their dynamics and perception. This simulator enables investigations of collective behaviors along with statistical analysis across a variety of parameters (e.g., larger test environments and robot swarms, or computationally faster robots) in virtual robot experiments. Informed by simulation results, I provided several robustness and scalability insights on flocking, intuition for circle formation, and a theoretical probabilistic analysis of multi-color heading alignment. These results together with physical robot demonstrations provide for the first time strong experimental evidence for multiple complex and robust 3D underwater collective behaviors, achievable entirely through self-organized coordination.

The Blueswarm project was conceptualized together with my advisor Prof. Radhika Nagpal and postdoctoral researcher Dr. Melvin Gauci. As the lead researcher of this project, I spearheaded the overall design of the robots, algorithms, and experiments. Dr. Melvin Gauci and I collaborated closely and developed many aspects of the software control system jointly. Dr. Melvin Gauci led the development of the custom electronics board and the circle formation and milling experiments. He further wrote software to analyze individual robot trajectories from recorded video.

Dr. Jeff Dusek joined the Blueswarm project during its early stages and made valuable contributions to the evaluation of sensing options and the chosen locomotive system as well as the design of a laboratory testbed. In addition, he designed a multi-jointed robotic fish inspired by Bluebot together with his student Katerina Soltan.

Paula Wulkop spent six months with me as visiting master's student from ETH Zurich. She led experimental work on self-organized evasive fountain maneuvers. Lucas Guzman, a Harvard University undergraduate, conducted a semester-long research project on collective mapping and visual odometry with miniature underwater robots.

In addition to my research on Blueswarm, I collaborated with Prof. George Lauder, Prof. Hossein Haj-Hariri and Dr. Mehdi Saadat on fish swimming and schooling, as well as Dr. Mihai Duduta, Prof. Robert Wood, Prof. David Clarke and Dr. Zeynep Temel on smart materials for soft robots.

1.4 Outline

The dissertation has eight chapters and three appendices:

Chapter 2 — Related Work: Discusses previous work in two- and three-dimensional collective robotics with a focus on underwater robots, and also introduces findings on collective behavior of fish schools.

Chapter 3 — A Self-organized and Fish-inspired Robot Swarm: Explains the fish-inspired design of the Bluebot underwater robots.

Chapter 4 — A Realistic Three-dimensional Simulation Environment: Introduces the custom simulation environment and models for Bluebot dynamics and perception.

Chapter 5 — Implicit Coordination for Three-dimensional Collective Behaviors: Presents the design and experimental validation of algorithms for fundamental collective behaviors including synchronization, dispersion, milling, and collective search.

Chapter 6 — Implicit Coordination with Headings: Bioinspired Evasion: Discusses heading-based evasive fountain maneuvers.

Chapter 7 — Toward Open Water Swarming: Outlines a pathway to open water swarming based on colored robots and discusses experiments on dispersion, milling, and alignment without LEDs.

Chapter 8 — Conclusion: Concludes and suggests future research, including on open water navigation, explicit communication, and design improvements.

Appendix A — Main Supplement: Contains additional technical information.

Appendix B — On Robotics and Fish Swimming: Contains a brief introduction and the abstracts of two publications on fish swimming and schooling.

Appendix C — On Smart Materials for Soft Robots: Contains a brief introduction and the abstracts of three publications on smart materials for soft robots.

2 Related Work

The power of collectives becomes particularly apparent when they achieve high degrees of autonomy, decentralization, and parallelism. While we can witness large-scale collective behaviors in nature, it is often unclear how they arise from the many locally interacting agents. Mathematicians and engineers have striven to understand the mapping from local interactions to global behaviors and vice versa in a quest to decode natural collective intelligence and engineer artificial collectives[24–27].

2.1 Collective Robotics in Two- and Three-dimensional Space

Recent advances have shown successful implementations of self-organized robot swarms as large as 1,000 units inspired by cells and social insects, albeit limited to two-dimensional local interactions[8–10,28–30]. For example, the SWARM-BOTS project (Fig. 2.1 A) demonstrated ant-inspired collective transport and chain formation[8], the Kilobot project demonstrated large-scale shape self-assembly[10,29], and the particle robotics project (Fig. 2.1 B) demonstrated emergent complex motion[28]. In the three-dimensional aerial domain, large drone swarms have displayed complex maneuvers, though mainly relying on centralized base stations or external positional feedback, rather than local self-organized interactions[11–15,31,32]. For instance, Intel's Shooting Stars used at the 2018 Winter Olympics or the Crazyswarm (Fig. 2.1 C) are centrally controlled by a single computer and depend heavily on the Global Positioning System (GPS) and motion capture, respectively[31,32]. VIO-Swarm (Fig. 2.1 D) copes without motion capture and GPS by use of Visual Inertial Odometry (VIO), but requires a ground station for coordination since the twelve quadrotors cannot self-organize[11]. Other aerial swarms cope without ground stations and are based on local self-organization, but require the exchange of GPS locations among robots to infer relative positions[12–14]. Most recently, autonomous exploration was demonstrated with up to six quadrocopters, which solely used gradient search toward a signaling home beacon — a minimal form of external positional feedback — to come back to the departure point (Fig. 2.1 E)[15].

The only platform to achieve a fully decentralized and unsupervised behavior in 3D (search and retrieval) to date is Swarmanoid (Fig. 2.1 F), a heterogeneous swarm composed of wheeled, climbing, and flying robots[33]. Swarmanoid is a compelling implementation of practical swarm activities for man-made real-world environments. Robots are devoted to particular tasks, including being a stationary beacon. The flying robots, capable of maneuvering in 3D, attach to the ceiling for increased situational awareness, and temporarily fulfill the task of a semi-central navigation aid to other robots.

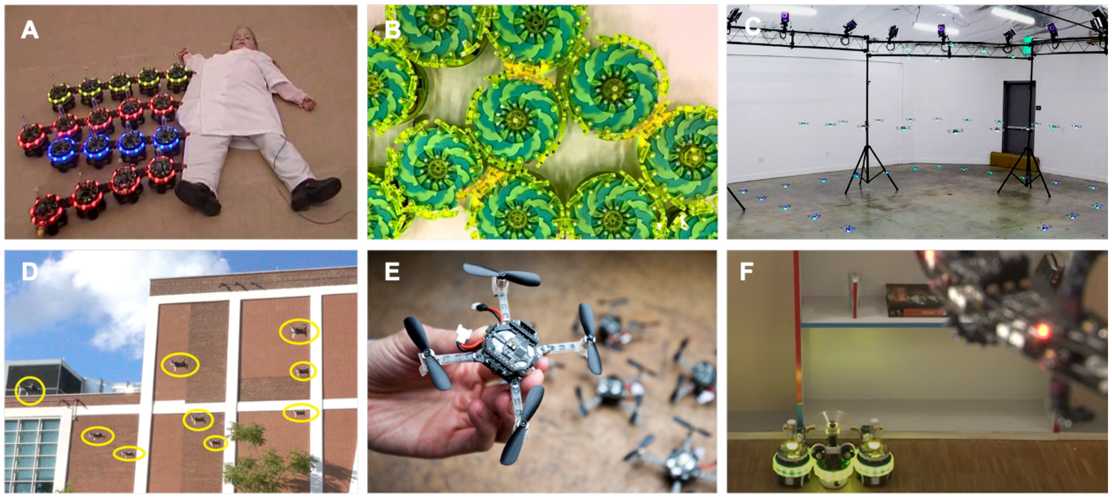


Figure 2.1: Selected above-ground robot collectives. **(A)** SWARM-BOTS chain up to pull children across laboratory floors. **(B)** Robot "particles" form clusters to move together. **(C)** Crazyswarm works only with the assistance of motion capture. **(D)** VIO-Swarm relies on a centralized ground station. **(E)** Tiny swarm uses gradient search toward a signaling home beacon. **(F)** Swarmanoid features three kinds of robots for autonomous collaboration.

While heterogeneity potentially enables more sophisticated behaviors through task specialization, it comes at the cost of increased complexity, and the risk of compromised robustness if one robot type malfunctions.

2.2 Underwater Collective Robotics

My work focused on a radically minimal design approach for robot collectives, suitable for the less explored and particularly challenging underwater domain, where many above-ground sensing and communication modalities are unavailable. This approach is geared toward the systematic experimental study of fundamental algorithms for elementary 3D collective behaviors in the laboratory. Blueswarm's features along with selected related work are summarized in Table 2.1.

Several previous projects have also envisioned 3D underwater robot collectives for applications from environmental monitoring in sites of high ecological sensitivity such as coral reefs, inspections of underwater infrastructure, and search-and-rescue operations[18,23,34–39]. The National Aeronau-

	Domain			Assistance			Coordination				Structure	
	underwater (3D)	*air (3D)*	*land (2D)*	*none*	*positional*	*computational*	*implicit*	*explicit*	*indirect*	*none*	*homogeneous*	*heterogeneous*
Blueswarm	x			x			x				x	
CoCoRo (*23*)	x			x				x				x
Glider fleet (*38*)	x				x	x			x		x	
M-AUE (*18*)	x			x						x	x	
Crazyswarm (*31*)		x			x	x			x		x	
VIO-Swarm (*11*)		x				x			x		x	
SGBA-Swarm (*15*)		x			x			x			x	
Swarmanoid (*33*)		x	x	x				x				x
Kilobot (*10,29*)			x	x				x			x	
SWARM-BOTS (*8*)			x	x				x			x	
Particle robotics (*28*)			x	x			x				x	

Table 2.1: Blueswarm's distinctive features in comparison to other robot collectives. Blueswarm is the first 3D underwater collective that uses only local implicit vision-based coordination to self-organize. Not depending on any assistance, Blueswarm is more autonomous than most aerial swarms. It advances fundamental research on decentralized and self-organized robot collectives from 2D to 3D space.

tics and Space Administration (NASA) wishes to build swarms of autonomous micro-swimmers to lead the search for life in liquid oceans on distant planets (Ethan Schaler, SWIM – Sensing with Independent Micro-Swimmers). However, aquatic environments come with a unique set of challenges, which make above-ground algorithmic approaches difficult to generalize[22]. There are inherent sensory and communication deficits, and several modalities that are available for above-ground systems (e.g., long-range wireless radio and GPS), are not available underwater. These communication deficits also make it more difficult to monitor systems in real time, or manually intervene for malfunctions, thus requiring a higher degree of autonomy and robustness.

In the last couple of decades, theoretical studies have shown that simple implicit coordination rules are sufficient to recreate the complex 3D behaviors observed naturally, in some cases producing algorithms with provable guarantees[40]. However, 3D implementations of such underwater robot swarms still do not exist that can validate these algorithms. There are several engineering challenges in designing underwater robots with 3D sensing and 3D motion. Furthermore, theoretical models make many hidden and unattainable assumptions about perfect robot perception of neighbors that even real fish cannot achieve. As a result, current implementations have been mostly limited to 2D coordination at the water surface using traditional wireless communication and GPS localization, or involved central underwater base stations[18,23,36,41,42]. For example, the commercially-developed Data Divers by Apium Swarm Robotics communicate and spread at the surface before diving for samples, the M-AUE robots drift uncoordinatedly with the ambient flow to sample the ocean for offline data reconstruction[18], and larger autonomous underwater vehicles (AUVs) form a communication chain to maintain connection with a floating base station[17] (Fig. 2.2 A–C).

Some research groups have attempted more complex underwater coordination by designing new explicit communication and localization methods (e.g., optical/acoustic modems[36], centralized or networked underwater base stations[23,37], and bioinspired electroception[43]), demonstrating limited coordination usually with two robots. A recent project, CoCoRo, built a heterogeneous swarm combining multiple modes of mobility and communication: surface robots, underwater robots, and floating base stations, using radio-frequency communication above-water and modulated blue-light and acoustic communication underwater[23]. CoCoRo provided a compelling vision for heterogenous collaboration as well as novel engineering design insights; however, limited experimental studies of 3D submerged collective behavior were published[35]. In a follow up project, subCULTron, the same lead researchers envision an artificial robot society to monitor the canals and lagoons of Venice (Fig. 2.2 D)[44].

Overall, the focus of underwater multi-robot systems has been on coordination through explicit and semi-centralized communication, rather than the implicit and minimalistic visual coordination that fish use. This approach has had limited success — compared to the incredible 3D maneuverability of schooling fish, current artificial systems demonstrate a large gap in achievable collective complexity. By contrast, implicit decentralized coordination can achieve scalability by spreading and reducing computational complexity, and robustness by introducing redundancy and avoiding dependence on

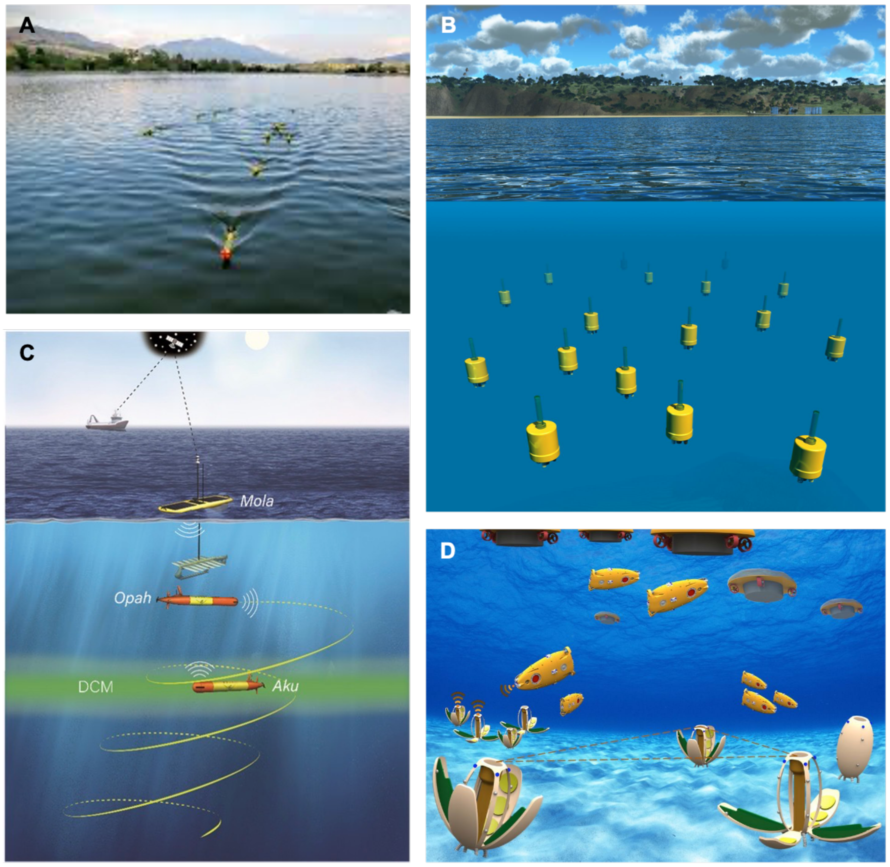


Figure 2.2: Selected underwater robot collectives. **(A)** Data Divers only coordinate at the surface before individually diving for samples and, therefore, elegantly bypass inherent challenges of underwater communication. **(B)** The M-AUE robots do not coordinate while drifting with the ambient flow. **(C)** Mola, Opah, and Aku form a communication chain to explore plankton populations in the deep chlorophyll maximum (DCM) layer. **(D)** The subCULTron project envisions a robotic society including artificial mussles, lilypads, and fish.

a central control hierarchy.

2.3 Collective Behavior of Fish Schools

Nature provides an existence proof on how to deal with the challenges of underwater communication and sensing. In these exact same domains, massive fish schools achieve remarkably complex and dynamic behaviors, by primarily relying on implicit coordination — many fish species base schooling decisions on visual observations of nearby neighbors, and several species use their lateral lines to perceive neighbors in low-visibility conditions[45–49]. By making decisions based on local perception of neighbors, these fish schools elegantly bypass the inherent challenges of underwater communication, achieving enormous scalability and robustness through decentralization[4,6,7,50–52].

A quarter of fish species school for their entire life, and about half the species school as juveniles[5]. Schooling fish for which vision is their dominant sensory modality[53] have excellent spherical 3D perception with a small blind spot at the posterior[6]. Specialized visual patterns, such as "schooling marks" and prominent stripes, have evolved to facilitate the rapid detection of members within a school, and bioluminescence is used by nighttime schooling fish, such as the *Anomalops katoptron*[50,54]. Among the several advantages of schooling described by biologists are, for instance, the reduced risk of being eaten[4,55], higher success in foraging, or energy savings while swimming in groups[56,57].

A wealth of experimental and theoretical work exists on flocking and alignment as it is observed in fish schools[24,58–61]. Researchers have also conducted studies with live fish to investigate collective decision making[62,63], group size choice dependent on ecological factors such as food availability and predation risk[64], or costs of swimming in isolation and various positions relative to other fish[56]. Robot collectives could provide a synthetic means to further understand how fish school complexity arises from the decisions of individual fish. However, to date, robotic examples cannot recreate the many intricate behaviors fish schools display.

3 A Self-organized and Fish-inspired Robot Swarm

At the outset of this dissertation was the search for a robot design that can advance demonstrations of self-organized collective behaviors from two- to three-dimensional space. To navigate a 3D underwater environment, such a robot has to show comprehensive situational awareness, combined with a high degree of maneuverability in all directions. Furthermore, to enable autonomous and real-time decision-making, the robot has to be capable of parsing information and extracting key parameters rapidly onboard. At the same time, small size, ease of fabrication, and low cost are desirable to afford and deploy a collective of several robots in a laboratory testing environment, where data acquisition for the analysis of collective behaviors is most straightforward, reliable, and efficient.

The functional design of the Bluebot underwater robot presented in this chapter was informed by above criteria, and consists of three major modules: i) two cameras allow for 3D perception of surroundings; ii) three LEDs serve as active beacons for neighbor recognition; and iii) four independently controllable fins provide a high degree of maneuverability in 3D space.

3.1 3D Visual Sensing

To enable underwater coordination from onboard perception, we wanted to achieve the ability for a Bluebot to detect fellow robots within a radius of at least 20 body lengths in all directions and infer their direction, distance, and heading. Of all feasible sensing modalities given the robot target size ($10 - 20$ cm), vision was the best option because small cell phone type cameras with wide-angle lenses allow us to have high definition sensing arrays at low cost. In addition, cameras are passive sensors, which do not cause signal interference, and allow for multi-robot detection in a single sample (i.e., image). Finally, fish often use vision as their dominant sensing modality for schooling, leading us to the conclusion that underwater visual coordination can be effective for Bluebots in those same domains.

3.1.1 Hardware Implementation

Bluebot achieves 3D vision and neighborhood sensing using a combination of cameras and blue-light LEDs. Two cameras with 195 $^\circ$ wide-angle lenses offer a quasi-omnidirectional field of view (FOV)

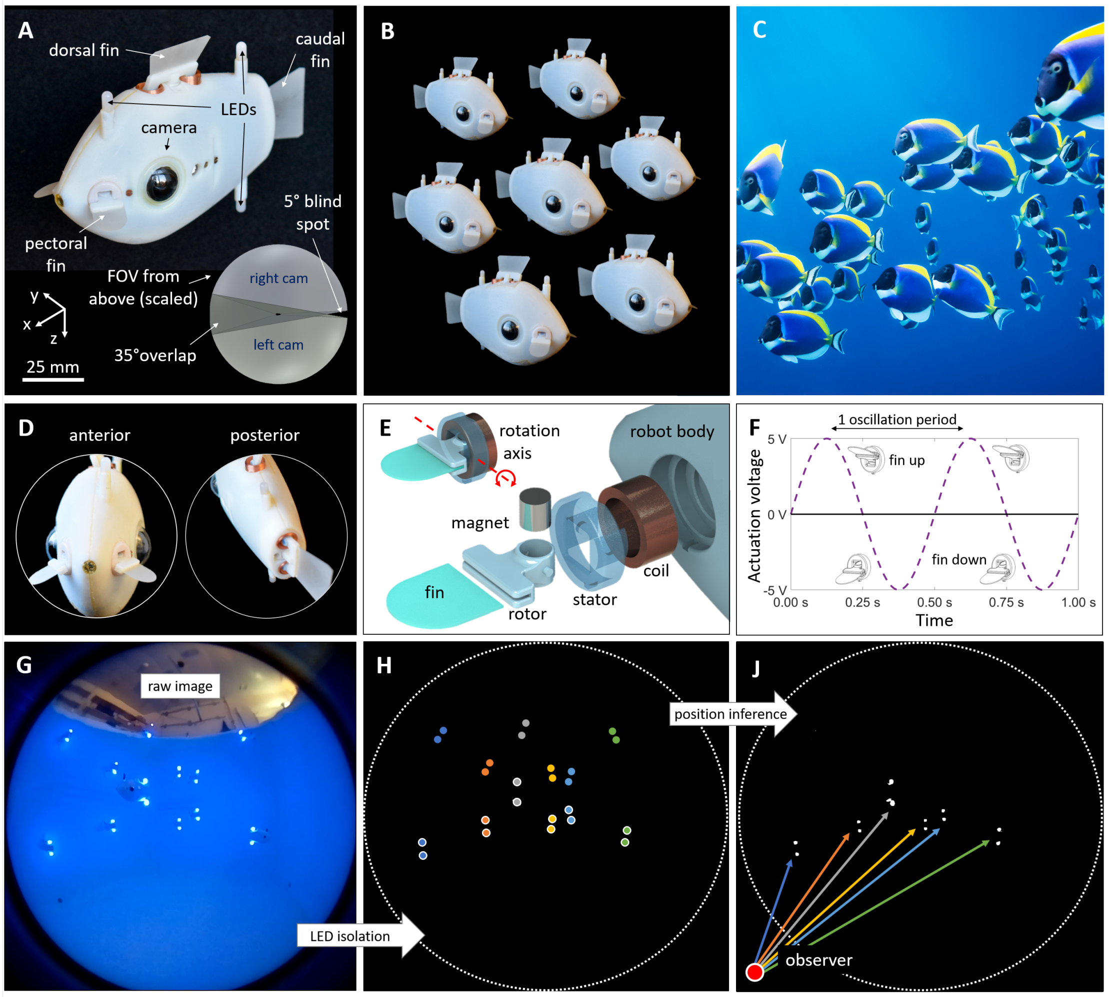


Figure 3.1: Bluebot overview. Bluebot combines autonomous 3D multi-fin locomotion with 3D visual perception. **(A)** Two cameras cover a near-omnidirectional field of view (FOV). One caudal and two pectoral fins enable nearly independent forward and turning motions; a dorsal fin effects vertical diving for depth control. **(B-D)** Seven Bluebots with streamlined, fish-inspired bodies are used in Blueswarm experiments. (Source of panel C: iStock) **(E-F)** The fins are powered by a custom electromagnetic actuator. **(G)** Information on neighboring robots extracted from images enables local decision making. **(H)** Fast onboard image processing is achieved by setting the cameras such that only the LEDs (and potential surface reflections) of neighboring robots appear in images. For illustration purposes, pairs of posterior LEDs belonging to the same robot are color-coded and have a white outline; a pair of the same color without the outline marks the respective surface reflection. The third LED is omitted here. **(J)** Bluebots' relative positions, headings, and distances are derived from triplets of LEDs assigned to individual robots (color-coded vectors), facilitating self-organized behaviors such as visual synchronization, potential-based aggregation-dispersion, dynamic circle formation, and collective search.

with an emphasis on the critical anterior direction (35 ° overlap), and limited only by a narrow 5 ° blind spot at the posterior of the robot (Fig. 3.1 A). Schooling fish, too, have spherical 3D vision with a small blind spot (~40 °) at the posterior[6], and vision is their dominant sensory modality[53]. In order to rapidly detect members of their school, many fish species have evolved specialized visual patterns (e.g., "schooling marks" and prominent stripes), and nighttime schooling fish, such as the *Anomalops katoptron*, exploit bioluminescence[50,54]. Inspired by these natural visual features, the Bluebots incorporate three blue-light LEDs that allow neighbors to quickly identify direction, distance, and heading of each other via projective geometry (Fig. 3.1 G–J). Direction is the easiest to infer based on where a robot appears in the visual field. Distance and heading are more complex and require solving for scaling factors to compute the mapping from pixel to real world positions. The camera-LED system is conceptually similar to a previous demonstration on aerial drones[65], but requires only three instead of five active markers and does not need to distinguish between fellow robots. The visibility of the LEDs (~5 m) exceeds the current testing environment in its longest dimension (~2.78 m). Image acquisition and processing is computationally expensive, limiting the sensing iteration frequency to 2 Hz. However, since the Bluebots move at speeds close to one body length per second (BL/s), they are able to achieve sensing-to-motion response times that are similar to schooling fish such as jack mackerels[66]. Using this vision system, Bluebot is able to approximate aspects of fish vision such as constant awareness and swift response to surroundings. However, as with real fish[51], the Bluebot vision system has natural limitations, such as noisy observations when many neighbors are present, and occlusions from nearby neighbors, yielding incomplete and imperfect representations during dynamic swarming activities.

The cameras (Raspberry Pi Camera Module v2) penetrate the body on either lateral side and are angled 10 ° forward against the y-axis. Custom thermoformed hemispheres made of clear plastic (Curbell Plastics PETG, 0.5 mm in thickness) cover the lenses for waterproofing (Fig. 3.1 A). On the inside, the camera cables are routed to a duplexer board (Arducam), which is then connected to the onboard computer (Raspberry Pi Zero W). One camera can be used at a time, and the duplexer board allows for superfast switching (~20 μs).

Still images can be captured at resolutions of up to 1944×2592 pixels. In most cases, however, the images are downscaled to a resolution of 192×256 pixels to allow for faster image processing and shorter control cycles. Captured images are in RGB color, but since the points of interest are Bluebot LEDs, which are blue, only the blue channel gets used in image processing for collective behaviors. The camera settings (brightness, contrast, white balance gains) are tuned such that under experimental lighting conditions, only the bright Bluebot LEDs register significantly in the images and appear as quasi-circular blobs; everything else appears mostly black (Fig. 3.1 J). Vision-based algorithms for neighbor detection that deal with the spherical distortion of the camera lens, LED reflections at the water surface, and the assignment of LED triplets to individual robots are presented in the following section.

3.1.2 Algorithmic Implementation

Bluebot's underwater navigation is vision-based, and local decision-making for collective behaviors relies on information on LED triplets that belong to individual robots. Upon taking an image with each of the two cameras, a Bluebot executes the following steps to infer relative positions, distances, and headings of other robots: i) identification of individual LED blobs; ii) undistortion of LED centroids; iii) inference of relative positions; iv) assignment of three LEDs per robot; v) inference of distances; vi) inference of headings (Fig. 3.2). The extracted information on robots' positions and headings enables a suite of collective behaviors, which are discussed in the next chapter.

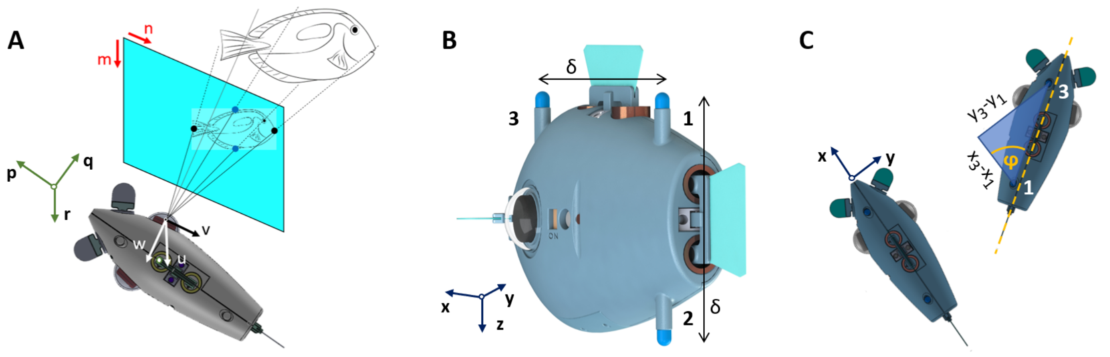


Figure 3.2: Image processing and position inference. (A) Bluebot identifies individual LED blobs in the mn image plane, and uses a predetermined calibration function to undistort their centroids. Undistorted centroids are first projected onto a camera-based uvw unit sphere, and then aligned with the pqr coordinate frame attached to the robot's center of mass. The direction in which an object is observed is known from its normalized pqr coordinates. **(B)** The distance in scaled xyz coordinates to such object can be found from two object points, whose real-world distance is known, e.g., the vertical posterior LEDs 1 and 2 of a Bluebot, which are $\partial = 86$ mm apart. **(B,C)** The heading φ of such object can be found from a non-aligned third object point at a known real-world distance to the first two, e.g., the anterior LED 3 of a Bluebot, which is in the same horizontal plane as and at a distance of $\partial = 86$ mm from LED 1.

Identification of Individual LED Blobs

LEDs appear as quasi-circular blobs in the images. These blobs are identified using a custom-designed algorithm, which is described here briefly and in more detail in Appendix A.1. First, each image undergoes thresholding to convert it to a binary image. Then, blob detection proceeds by searching for continuity in white pixels in the vertical m and horizontal n directions (Fig. 3.2 A). Groups of white pixels that are continuous in both directions become designated as blobs. In experiments with seven robots and image resolutions of 192×256 pixels, this algorithm typically requires about 2.9 times fewer computational steps compared to conventional algorithms (e.g., depth- or breadth-first search), and runs an order of magnitude faster on the Bluebot's Raspberry Pi Zero W computer. The significant speedup comes at the cost of susceptibility to some pathological cases, where discontinuous groups of white pixels get lumped together as one blob; however, these cases appear very rarely in practice.

Undistortion of LED Centroids

The wide field of view provided by the lenses results in very significant spherical distortion in the images. Thus, an undistortion function (obtained with the `OCamCalib` toolbox for `MATLAB`) is essential to convert *mn* LED locations in images to *uvw* directions in the real world (Fig. 3.2 A). For Bluebot's particular camera and lens setup, a third order polynomial yields the best results. In order to retain fast processing times, this polynomial function is used only on feature points of interest (e.g., LED centroids).

Inference of Relative Positions

Given any one LED belonging to a Bluebot, that Bluebot's relative position is known up to a scaling factor from the normalized *pqr* coordinates of the LED. The *pqr* coordinates in the robot frame are obtained after rotation and coordinate transformation of the camera-attached *uvw* coordinates (Fig 3.2 A). For the right camera, a clockwise rotation of 10 ° around the vertically downward u_R axis precedes the following coordinate transformation:

$$p = -v_R \qquad q = -w_R \qquad r = u_R$$

For the left camera, a counterclockwise rotation of 10 ° around the vertically downward u_L axis precedes the following coordinate transformation:

$$p = v_L \qquad q = w_L \qquad r = u_L$$

Assignment of Three LEDs per Robot

A typical image from a Bluebot's camera contains several LEDs because multiple other Bluebots are within the field of view. Moreover, reflections may appear if these Bluebots are close to the water surface. For obtaining information about the number of Bluebots or their positions, it is necessary to eliminate reflections and to identify triplets of LEDs originating from the same Bluebot. Reflections are identifiable since they appear mirrored and vertically above a true triplet. The process of bundling LEDs relies on the fact that Bluebots are passively stable in roll and pitch, and therefore their two posterior LEDs are always vertically aligned, while the two dorsal LEDs are in the same horizontal plane. These two heuristics allow us to find triplet candidates. For some experiments, we additionally used a Kalman filter with a simple constant velocity model to predict where LEDs should appear based on historical observations. The predicted LEDs get matched greedily with the heuristic candidates, unless the two triplets are more than 300 mm apart in the real world. The Kalman filter is effective in achieving higher bundling accuracies and avoiding false positives.

Inference of Distances

Without loss of generality, the posterior dorsal and ventral LEDs are labelled 1 and 2, and the anterior LED 3 (Fig. 3.2 B). When a pair of posterior LEDs 1 and 2 originating from the same Bluebot is available, it is possible to estimate the distance to that Bluebot via projective geometry. The calculation finds scaling factors α and β to go from the normalized *pqr* directions to the scaled *xyz* real-world coordinates of both LEDs:

$$\alpha(p_1, q_1, r_1) = (x_1, y_1, z_1)$$

$$\beta(p_2, q_2, r_2) = (x_2, y_2, z_2).$$

It makes use of the known vertical real-world distance ($\delta = 86$ mm) between the LEDs:

$$z_1 + \delta = z_2 \iff \alpha r_1 + \delta = \beta r_2,$$

and the fact that both LEDs are equally far away in the x direction:

$$x_1 = x_2 \iff \alpha p_1 = \beta p_2.$$

Combining above equations yields the two scaling factors α and β, from which real-world distances are recovered (Eq. 3.1):

$$\begin{aligned} \alpha &= \frac{\delta p_2}{p_1 r_2 - p_2 r_1} \\ \beta &= \alpha \frac{p_1}{p_2} \end{aligned} \tag{3.1}$$

Inference of headings

Given all three LEDs of a single Bluebot, the anterior LED 3 can be used to find its heading (Fig. 3.2 B–C). Analogous to distances, the calculation finds a scaling factor γ to go from the normalized *pqr* directions to the scaled *xyz* real-world coordinates of the anterior LED 3 of that robot:

$$\gamma(p_3, q_3, r_3) = (x_3, y_3, z_3).$$

It makes use of the fact that LED 3 is on a horizontal circle of radius δ around LED 1 (Eq. 3.2):

$$\begin{gathered}(\gamma p_3 - x_1)^2 + (\gamma q_3 - y_1)^2 = \delta^2 \\ \gamma^2 \underbrace{(p_3^2 + q_3^2)}_{\text{a}} + \gamma \underbrace{(-2p_3x_1 - 2q_3y_1)}_{\text{b}} + \underbrace{(x_1^2 + y_1^2 - \delta^2)}_{\text{c}} = 0 \\ \gamma_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a},\end{gathered} \tag{3.2}$$

where the initial quadratic equation (first line) was rearranged in standard form (second line), and the quadratic formula was used to solve it (third line). Now, the γ, for which $|z_1 - z_3|$ is smaller, is selected, since LEDs 1 and 3 are in the same horizontal plane. Given the xyz coordinates of LEDs 1 and 3, the relative heading φ of the neighboring robot follows from Figure 3.2 C (Eq. 3.3):

$$\varphi = \text{atan2}\left(\frac{y_3 - y_1}{x_3 - x_1}\right) \tag{3.3}$$

3.1.3 Accuracy of Visual Detections

The estimation error for a single robot position depends on the angle more than on the distance, as became apparent in a series of tests performed to evaluate the accuracy of the Bluebot's vision system. One Bluebot (A) was placed in a fixed position and programmed to infer the position of another Bluebot (B). Bluebot B was placed in various positions relative to A, namely in all possible combinations of:

i) Distance: 0.5 m, 1.0 m, 1.5 m, 2.0 m, 2.5 m, 3.0 m

ii) Planar bearing: 0 °, 45 °, 90 °, 135 °

iii) Height: 0.0 m, 0.5 m, 1.0 m

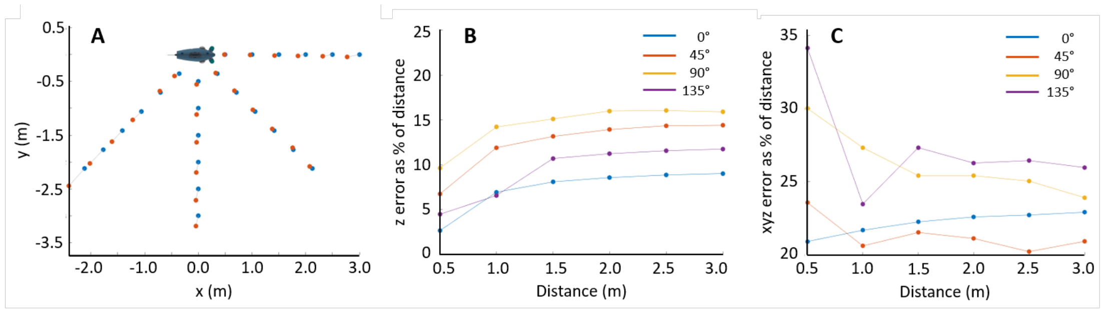


Figure 3.3: Position estimation accuracy. **(A)** xy planar error in Bluebot's position estimation for 24 cases. Blue dots represent the ground truth, red dots the estimated position. **(B)** z error as a percentage of distance. Colors represent different angles. **(C)** Overall xyz error as a function of distance. Colors represent different angles.

This gives a total of $6 \times 4 \times 3 = 72$ measurements. Here I only present the results for the case where the height is 0.0 m as the other heights gave similar results.

Figure 3.3 A shows the position estimation error along the x and y directions. Blue dots represent the ground truth (i.e., the actual position of Bluebot B), while red dots represent Bluebot A's estimate of Bluebot B's position. The error grows slightly with distance for all angles, but is within 20 % even at a distance of 3.0 m.

Figure 3.3 B shows the error along the z direction as a percentage of the distance between the Bluebots. The z error depends on the angle, but levels off at a maximum of 15 %.

Finally, Figure 3.3 C shows the overall error as a percentage of distance d. The overall error is defined as:

$$e = 100\sqrt{\left(\frac{\Delta x}{d}\right)^2 + \left(\frac{\Delta y}{d}\right)^2 + \left(\frac{\Delta z}{d}\right)^2}$$

The results show that the error for a single robot position depends on the angle more than it does on the distance, and levels off at around 20 % in the best-case scenario and 25 % in the worst-case scenario.

Overall, estimation of positions is more accurate than headings. Small errors in LED locations have a minor effect on the relatively large magnitude of distance vectors. However, the same LED errors can drastically affect the heading estimate, which is based on short inter-LED distances (Eq. 3.3). In addition, selecting the correct solution for γ in Equation 3.2 becomes unreliable if two robots are close to levelled; for robots that are exactly levelled, the heading remains ambiguous.

The accuracy of LED locations could be improved in two ways. First, images could be taken at higher resolutions. This approach was tried and showed negligible benefits, while taking significantly longer to compute. Second, every camera could be calibrated individually. However, this would add at least 30 minutes to the assembly process. Instead, the same calibration function is used for all cameras. Generally, other sources of error and noise, such as reflections and occlusions, have a more severe impact on robot detection.

3.1.4 Discussion of Alternative 3D Perception

Our visual sensing approach is geared toward minimalistic neighbor detection that runs onboard and on the fly. Limited computational power forced us to compromise between the amount as well as complexity of information we can process, and the time required to do so. For instance, LEDs of different colors or more than three LEDs could have simplified their allocation to individual robots; however, their extraction from raw images would have been more time consuming. In general, the three major limitations of our current camera-LED system as discussed below are: first, LEDs are active beacons which may not work in all lighting conditions; second, a vision-based approach to underwater coordination may not work in turbid waters; third, the current system cannot detect obstacles or boundaries in the environment.

1. The introduction of progressively more powerful and smaller microcomputers will enable more sophisticated patterns and recognition to infer distance and heading of neighbors without LEDs. For instance, deep learning-generated visual patterns (i.e., artificial schooling marks)

may allow future Bluebots to recognize neighbor pose[67]. By contrast, we can already find the direction of fellow robots without LEDs by simply coloring them differently than the surrounding environment. This is shown in Chapter 7.

2. We evaluated several alternatives to vision-based coordination that would work in turbid waters. Bioinspired artificial lateral lines[68,69] and electroception[43] could theoretically provide 360 ° pressure and electrical maps of objects in the vicinity. However, their fabrication is involved and off-the-shelf packages ready to use on small robots do not exist. Acoustic modems are used for communication among larger autonomous underwater vehicles[36] and sound navigation and ranging (sonar) on manned ships and submarines. Aside from being bulky, their concurrent use on multiple swarming robots would lead to signal interference.

3. Bluebots can currently only detect and coordinate with fellow robots. The perception of obstacles or boundaries in the environment is still open and was outside the scope of this thesis. While cameras are well suited to deliver complex images, more computational power is required to process them and extract meaningful information on environmental features.

Ultimately, we are still very far from fish — they can detect many aspects of their mates beyond just distance and heading. Cichlids (*Astatotilapia burtoni*), for instance, use visual pattern recognition to infer moods of conspecifics[70]. Fish also navigate a variety of underwater environments, using a suite of sensing modalities to school with mates, evade predators, or find food.

3.2 3D Multi-fin Locomotion

A high degree of maneuverability allows Bluebots to capitalize on 3D visual information by exerting 3D locomotive responses. Our goal was to achieve independent forward motion, turning in place, and depth control at reasonable speeds (~1 BL/s). This required multiple actuators per robot in order to vector thrust forces in different directions. Therefore, actuators themselves had to be small, low cost, and low maintenance. We were able to realize this high maneuverability with a compact, fish-inspired design that uses multiple fins powered by electromagnetic actuators.

3.2.1 Body and Actuator Design

The streamlined body of Bluebot, measuring $130\,\text{mm}(=1\,\text{BL})$ in the longest dimension at a volume of $235\ \text{cm}^3$, was modeled after typical surgeonfish from the *Acanthuridae* family (Fig. 3.1 B–C). Surgeonfish are ubiquitous members of coral reef communities and capable of forming agile coordinated groupings across a wide range of complex topographies[71,72]. To achieve high maneuverability with Bluebot, four independently controlled fins provide precise locomotion in 3D space. Turning in place, forward motion, and stopping in the horizontal *xy*-plane are achieved with two pectoral fins (Fig. 3.1 D, left) and a caudal fin (Fig. 3.1 D, right), respectively, and diving along the vertical *z*-axis is controlled with a single dorsal fin and slight positive buoyancy[73]. Bluebot is passively stable in roll and pitch. Operating at fixed amplitudes, the actuation frequencies of the caudal and dorsal fin can be modulated to reach cruise speeds of up to $150\ \text{mm/s}(=1.15\ \text{BL/s})$ and dive speeds of up to $75\ \text{mm/s}$. The pectoral fins allow for near on-the-spot turning at radii as small as $65\ \text{mm}(=0.5\,\text{BL})$, and $180\,^\circ$ changes of direction can be achieved in less than five seconds.

All fins are powered by our custom electromagnetic actuators consisting of a coil inside which a permanent magnet is hinged (Fig. 3.1 E)[73,74]. Oscillating the direction of an electric current flowing through the coil induces an oscillating magnetic field, with which the magnet tries to stay aligned. As a result, the fins oscillate around a single axis in a sinusoidal pitching motion (Fig. 3.1 F). The power of the fins can be controlled by changing the voltage across the coil with pulse width modulation (PWM). The actuators are submersible and only two wires from each coil penetrate the Bluebot's body, avoiding any need to seal off moving parts. The housings are 3D-printed in assembled state, i.e., including the pivoted hinge to which fins, laser-cut from flexible plastic shims (ARTUS), are attached. The caudal and the dorsal actuators are equipped with two coils each for enhanced thrust and connect magnetically to the robot body. The magnetic connection allows for fast switching between different fins, which was used in a collaboration on fish swimming and the propulsive efficiency of fin designs (Appendix B). For the investigation of collective behaviors, the optimization of fin shape, size, and material was not a priority. I chose fins in accordance with the robot geometry and actuator strength. However, I used the collaboration on robotic fish swimming to investigate the performance landscape of different fin types. The results allow to optimize the swimming performance and mimic key characteristics of fish swimming such as U-shaped cost of transport and reverse Kármán wakes[75].

3.2.2 Navigation Results

We validated our multi-fin locomotion approach in an early robot design with an inertial measurement unit (IMU) and pressure sensor but no vision system[73].

Depth Control with a Pressure Sensor

In two initial experiments on vertical motions, we investigated long-duration depth maintenance (i.e., hovering), and long-range repeatable diving. In the first experiment, the robot hovered at a set depth of 437 mm. The results indicate that depth maintenance is possible within half of a body height (Fig. 3.4 A). In the second experiment, the robot dove periodically between two thresholds set at 248 mm and 748 mm. The results demonstrate excellent accuracy and precision (Fig. 3.4 B). The overshoots observed in this experiment can be explained by the robot's inertia, and the time delay introduced by smoothing the pressure sensor readings with an exponential moving average filter. Summing the time when the dorsal fin was active results in approximately 20 % on-time for hovering and 60 % on-time for diving, although both conserve identical net potential energies. The discrepancy is mostly due to acceleration forces that build up while rising, and to a lesser extent due to increasing

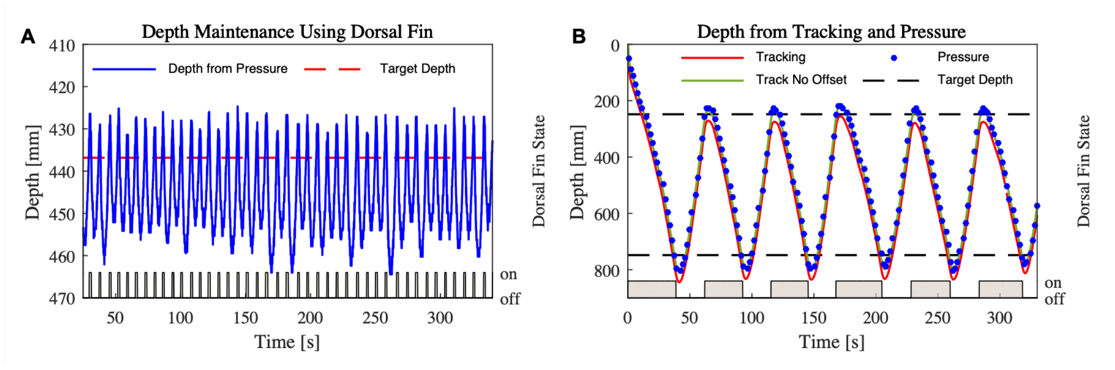


Figure 3.4: Submerged vertical locomotion. **(A)** Active maintenance of a prescribed target depth (red dashed line) is achieved by using feedback from a pressure sensor and the dorsal fin to oppose positive buoyancy. **(B)** The robot can also precisely control different target depths, in this case 248 mm and 748 mm. The red line denotes the robot depth from visual tracking, and the blue circles denote depth found from pressure sensor measurements (samples shown every 2 s for clarity). The green line denotes the depth from visual tracking with a constant offset of 30 mm (1/2 body height) removed.

drag forces at higher rates of ascent or descent.

3D Swimming with an Inertial and Pressure Sensor

To evaluate planar trajectory control, we conducted a series of experiments including repeated square trajectories at a given fixed depth (Fig. 3.5 A), and at incrementally decreasing depths (Fig. 3.5 B). The three basic behaviors were open-loop forward swimming, gyroscope-based turning, and pressure-based diving. A switch statement switched between forward, turning, and diving states whenever the respective thresholds corresponding to distance, angle, and depth were reached. The sensor readings were filtered with an exponential moving average filter (smoothing factor $= 0.1$).

We used visual tracking in 3D and readings from the pressure sensor (see Appendix A.2) to characterize repeatability and drift over several repetitions of a prescribed trajectory. This allowed us to know with high confidence where the robot was at all times in 3D space. The trajectories found using the visual tracking system were partitioned into straight swimming, turning, and rising segments based on the controller state (Fig. 3.5 C). The tracking videos were synchronized with the robot's onboard

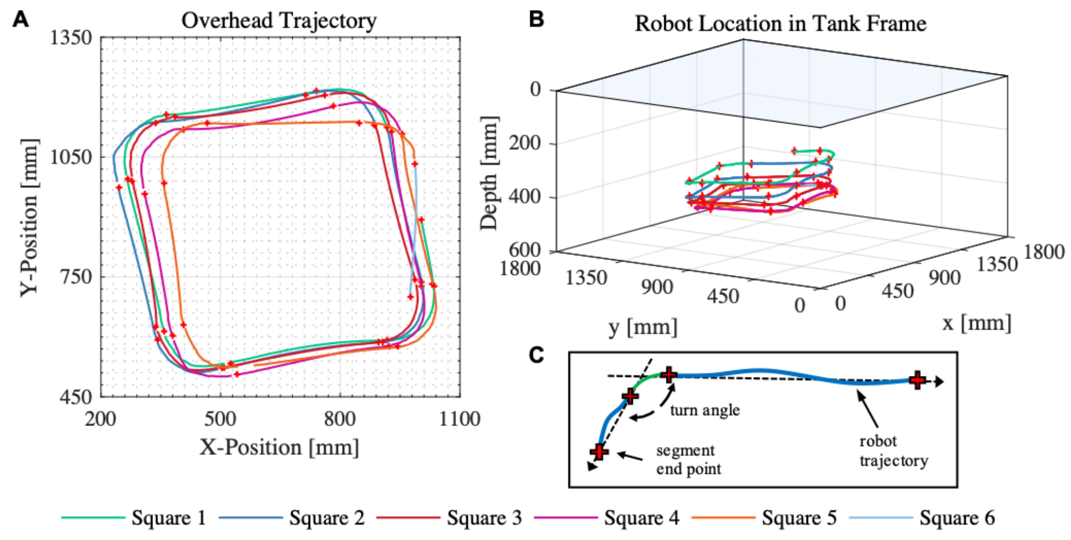


Figure 3.5: Submerged planar locomotion. (A) A robot trajectory for five complete squares in the *xy*-plane demonstrates precise locomotion. **(B)** The 3D trajectory was obtained by combining visual tracking from the overhead camera with pressure readings from the robot's onboard sensor. **(C)** Trajectories were broken into straight (blue) and turn segments (green) based on the recorded state of the robot (lower right). The turn angle is calculated using the vectors between segment end points (red crosses).

data file using a sequence of LED flashes. By recording the LED and controller states throughout the experiment, the location in world coordinates where the robot changed state could be identified. Using the 3D coordinates and time stamps recorded along each segment of the trajectory, the segment lengths, swimming velocities, and turning angles could be calculated.

Complexities of moving in 3D space included coupled dynamics, such as the dorsal fin inducing undesired turning as a side effect of diving, imperfect straight forward motion, and drifting gyroscope readings. To mitigate the latter, an in situ drift model was obtained at the beginning of each experiment while the robot was at rest. The observed static drift in yaw was fitted with a linear regression to model its time-dependent progression, and then subtracted from the raw values. The final estimates used for turning during the experiments translated linearly to degrees, allowing us to set thresholds for any turning angle.

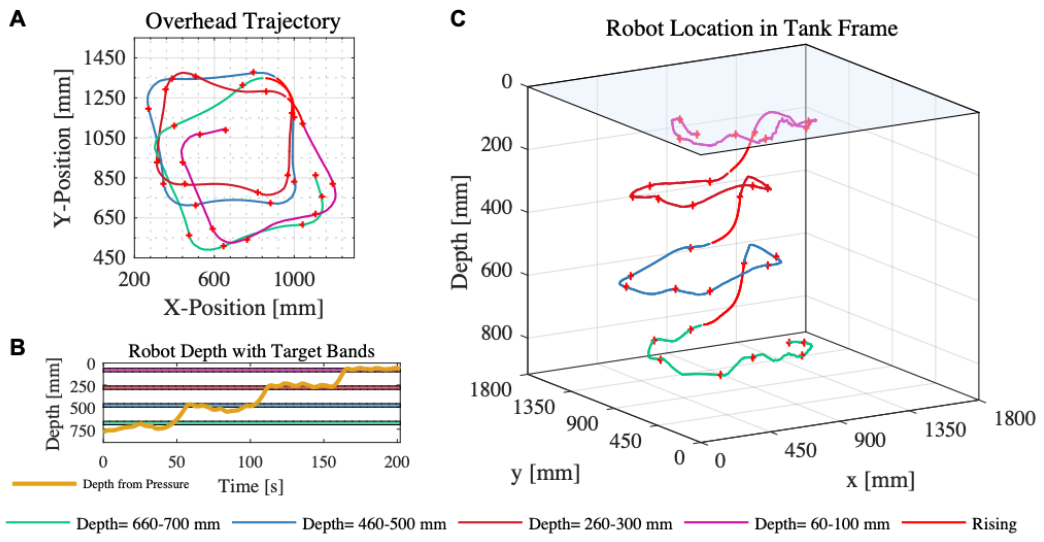


Figure 3.6: Submerged 3D locomotion. (A) A robot trajectory for squares at incremental depths shows drift induced by vertical motion. **(B)** The robot depth (orange line) is controlled with an onboard pressure sensor and set to remain within target depth bands for each square. Depth band colors correspond to square colors above and right. **(C)** The 3D trajectory validates prescribed 3D locomotion.

Swimming repeated squares at a fixed depth resulted in high consistency between trajectories (Fig. 3.5). During the experiment the robot executed five complete squares, with 20 turns of nominally 90 °. All turns were between 96 ° and 82 ° with an average of 90 ° and a standard deviation of 4 °. The average turning radius was 110 mm, or just over 1 BL. During the straight line segments of the squares, the robot swam without control for a period of 7.5 seconds at an average speed of 53 mm/s (0.53 BL/s). The robot speed was consistent across the 21 straight segments with a standard deviation of approximately 2 mm/s (0.02 BL/s), resulting in an average segment length of 388 mm (~4 BL) with a standard deviation of 14 mm (~0.1 BL). Because of the low deviation in turn angle and segment length, the squares were found to be largely repeatable with minimal rotation or drift in the prescribed shape.

Swimming squares at incremental depths, the robot started at the bottom of the tank and decreased its depth by 200 mm after each completion of a square (Fig. 3.6). The heave and yaw motions of the robot during the depth transitions appeared to be coupled. While the average turning angle was found to be 90 °, equal to the fixed depth case, the standard deviation of the turning angle was 14 ° as opposed to 4 ° shown previously. This increased variation in turning angle was particularly prevalent in turning maneuvers immediately following depth transitions (transitions shown in red in Figure 3.6 C), and introduced drift on the congruence of the squares. The average turning radius during the experiment was found to be 97 mm, or approximately 1 BL. While completing a square at each prescribed depth, the robot performed as expected, with an average swimming speed of 55 mm/s (0.55 BL/s) and an average segment length of 403 mm (~4 BL), both consistent with previous experiments.

Additional experiments on swimming straight-line trajectories based on IMU readings can be found in my related work on fish swimming and schooling[75].

Homing to a Light Source with a Photodiode

As a precursor to multi-robot navigation and coordination based on exteroceptive sensing, we also tested homing to a light source. In general, homing behaviors are a valuable primitive for enabling autonomous behaviors based on a global stimulus, such as autonomous charging or data transfer. To achieve homing, our robot has a front facing photodiode with a narrow viewing angle. The photodiode was used to perceive a light signal, which has the shape of a bell curve, peaking whenever the light is directly faced. The threshold for detection of the light source was either sampled during an initial 360 ° turn, or preset from a separate light characterization experiment. As a source, we used a 4-LED rod emitting blue light, which propagates best in water.

The robot control was programmed to turn left until the light intensity crosses the detection threshold, then swim towards the source. It starts turning again to realign with the source whenever the intensity value falls below the threshold for reasons such as drift. The implemented homing behavior allowed the robot to recover from any position, therefore guaranteeing robustness.

To test homing, we conducted the following experiment. We placed the LED rod at the south end

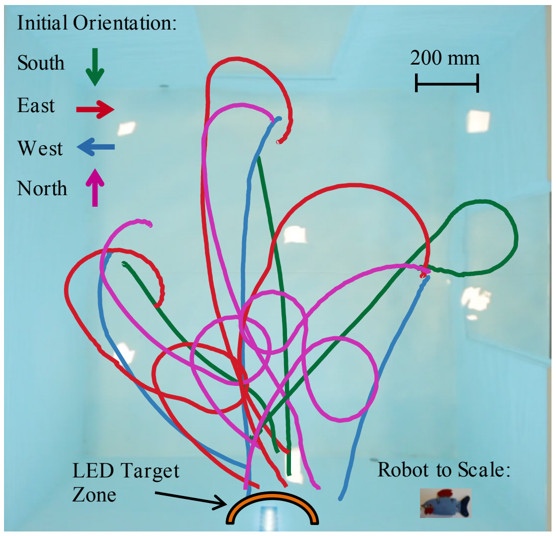


Figure 3.7: Homing to a light source — trajectories from twelve experiments. Experiments were conducted using four initial orientations (denoted by trajectory color), and from initial positions near the center of each of the three unlit tank walls. Experiments were ended when the robot reached a hemispherical zone surrounding the LED rod denoted in orange.

and started the robot in the remaining three extremes of the tank (N,E,W), oriented in four directions (N,E,S,W). In total we did 48 trials. We measured success by seeing if the robot always reached an approximately two body length hemispherical zone surrounding the LED rod. Inside this zone, tracking was found to be unreliable as the blue light from the LED rod significantly altered the color of the robot in both the overhead and side videos.

Figure 3.7 shows a subset of 12 trajectories. We observed that the robot occasionally lost the LED rod, resulting in a full loop until it was reacquired. We consider the trials very successful because in each case, losing the LED rod was not fatal to the experiment's objective, and the robot successfully tracked to the hemispherical target zone, validating the robustness of the homing behavior.

The experiment was extended to moving back and forth between two sources, which were switched on and off alternately. This extension hints toward the robot's potential to recover from a signal loss, realign with a source that changed position, or follow a moving source. Moreover, the robot's behavior may be interrupted and altered by a signal that is suddenly switched on, for instance to initiate homing.

3.2.3 Discussion of Alternative 3D Locomotion

Our locomotion approach is geared toward high robot maneuverability at low actuator maintenance and space requirements. Several other groups have designed miniature robots for laboratory environments. Most examples involve a single actuator system to achieve low cost and small size. Kopman et al. presented a robot (117 mm in length, $ 100) with a fin actuated by a waterproof servomotor[76]. In Clark et al.'s robot (58 mm) a fin is driven by an electromagnetic actuator[77]. Yet other designs have fins powered by piezo-electric actuators, such as the one by Aureli et al. (100 mm)[78]. In these single-fin robots, forward motion is achieved by symmetric flapping of the fin, and large-radius turning (i.e., not on-the-spot) is achieved by biasing the flapping. Takada et al. also presented a robot with a fin driven by an electromagnetic actuator[79]; in addition to planar motion, this robot is also able to pitch using an internal servomotor that shifts the robot's center of mass. All of these examples rely on flapping actuation, in part because propellers are hard to miniaturize while still sealing the moving mechanism in a way that is reliable. Toy submarines for example can be based on small propellers; however, their reliability is extremely limited.

At slightly larger sizes (~250 mm), several groups have designed 3D maneuverable robots using multiple actuators including propellers, servomotors, and pumps[80]. For example, the Jeff robot (250 mm, $ 1300) was designed as part of the CoCoRo project, and is intended for operation in collectives[81]. It uses two vertically-stacked propellers at the back for forward motion and pitch variation, another propeller for turning on the spot, and piston-based buoyancy modulation for in-place depth control. The propellers use custom-designed magnetic shaft couplings, which are relatively large and expensive but eliminate the waterproofing issues associated with a body-penetrating shaft.

Other robots in this range use flapping-fin based designs, with both rear/caudal and side/pectoral fins controlled by multiple servomotors. Examples include the Jessiko robot[82] (220 mm) and the boxfish-inspired robot by Hu et al.[83] (350 mm). The use of multiple flapping fins allows these robots to achieve forward motion, turning, pitch control, and in some cases rapid braking. A third propulsion mechanism for 3D maneuverability is the use of multiple internal pumps. For example the robot by Bhattacharyya et al.[84] (203 mm) uses four internal pumps to suck water from inlets and distribute it to outlets, creating jets to control five degrees of freedom. The robots described in this paragraph, while having high degrees of maneuverability and reliability, use sophisticated means of actuation that are not easily scaled down in size and cost.

In our design, we developed a robot with multiple propulsors to control 3D motion using an actuation strategy that is easy to manufacture (no complex sealing hardware), low-cost, and small profile. To achieve this, we took inspiration from one of the actuators used in miniature robots, namely the electromagnetic actuator[77,79], and modified it for a multiple fin design. Our electromagnetic actuator is ideal for miniature robots and enables increased maneuverability for considerably more complex motion and autonomous control compared to previous robots at the same scale. A limitation of our propulsor is, however, that it cannot be scaled up without becoming severely inefficient as we discuss in the supplement of our publication on robot locomotion and maneuverability[73].

3.3 Scaling up to a Swarm

Designing a robot that lends itself to be manufactured efficiently and operated in a swarm requires careful thought about components, fabrication, handling, and testing[10,85]. Our goals were to build tens of robots with minimal need for maintenance that can be run by a single researcher without having to program each robot individually.

3.3.1 Operating a Swarm

Bluebot is designed to be easy to manufacture, recharge, and program in order to facilitate multi-robot operations (Fig. 3.8). Up to ten Bluebots can be put on a custom charging rig simultaneously. The

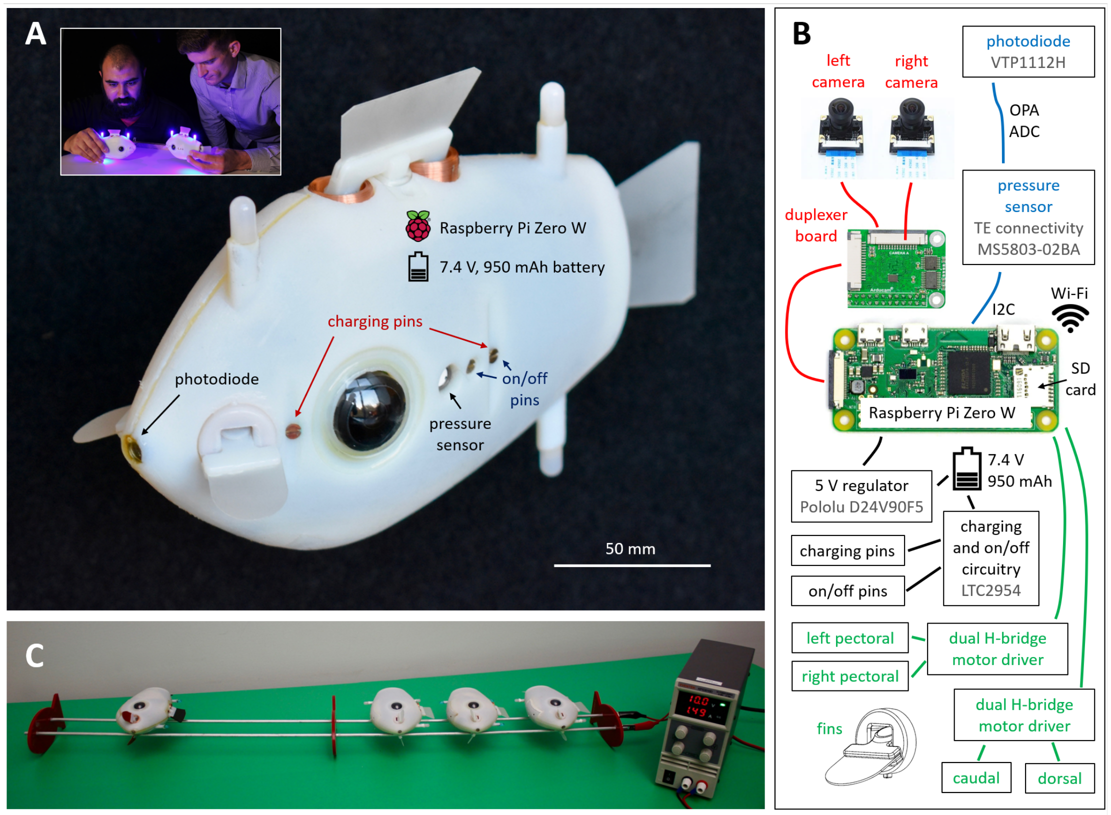


Figure 3.8: Bluebot electronics. (A) A photodiode, a pressure sensor, and pins for charging and switching the robot on/off penetrate the body. **(B)** Sensors (blue) and actuators (green) are internally routed to a custom PCB and connected to a Raspberry Pi Zero W onboard computer. The cameras (red) are operated via a duplexer board that enables fast switching (~20 μs) between them. **(C)** Several Bluebots can be charged simultaneously on a custom charging rail.

robots have an onboard charging circuitry (LTC2954), and the rig is powered from a power supply at 10 V. Similarly, programming multiple robots with a single command is possible using the Wi-Fi module of the onboard computer (Raspberry Pi Zero W). Programs (implemented in `Python3`) can be started simultaneously on multiple robots: a light pulse perceived by the robots' forward-facing photodiodes (VTP1112H) causes switching from an idle loop to the main program. Experimental data from the perspective of Bluebots can be logged onboard on a microSD card. For instance, data from a pressure sensor (TE connectivity MS5803-02BA) was used to reconstruct diving depths. All electronics are connected to a custom printed circuit board (PCB, OSH Park). A 7.4 V 950 mAh battery (Turnigy) provides power for run times of up to 2 h, whereby the onboard voltage is reduced to 5 V by a step-down voltage regulator (Pololu D24V90F5). Bluebot is switched on and off with a custom ignition key that applies 3.7 V to two external pins, which are connected to an on/off controller (LTC2954). The controller also automatically shuts down the robot if battery is low.

3.3.2 Building a Swarm

Assembling a Bluebot takes roughly six hours, and starts with the installation of all actuators, cameras, and electronics inside the two 3D-printed plastic halves (Stratasys PolyJet Objet500), continues with soldering all electronic components to the PCB and sealing those components that penetrate the body from the inside, and concludes with fusing the two halves into a single robot using plastic bonding epoxy (Loctite). Passive stability in roll and pitch as well as near-neutral buoyancy are achieved by careful placing of components such that the center of mass is directly below the center of buoyancy. A small compartment on the ventral side of Bluebot, which is sealed from the rest of the body and opened with a single bolt, allows for fine tuning of buoyancy with additional mass blocks.

3.3.3 Testing Arena

All experiments were conducted in a 1.78 m $\times$ 1.78 m $\times$ 1.17 m fresh water tank (Fig. 3.9)[73], which had a depth of 0.91 m (effectively 13.7 BL $\times$ 13.7 BL $\times$ 7.0 BL). Most experiments are under ten minutes in length; the robots have a top speed of 1.15 BL/s and are able to cross the surface of the tank in about 11.9 seconds. A DSLR camera was mounted above the tank to film experiments and allow for planar tracking of individual robots. Experimental data including diving depth values along the vertical dimension were acquired onboard the robots. The reconstruction of 3D robot trajectories was possible from video materials and depth values and validated in previous research[73]. A custom-built software automates large parts of this reconstruction, only asking for user intervention when there is potential

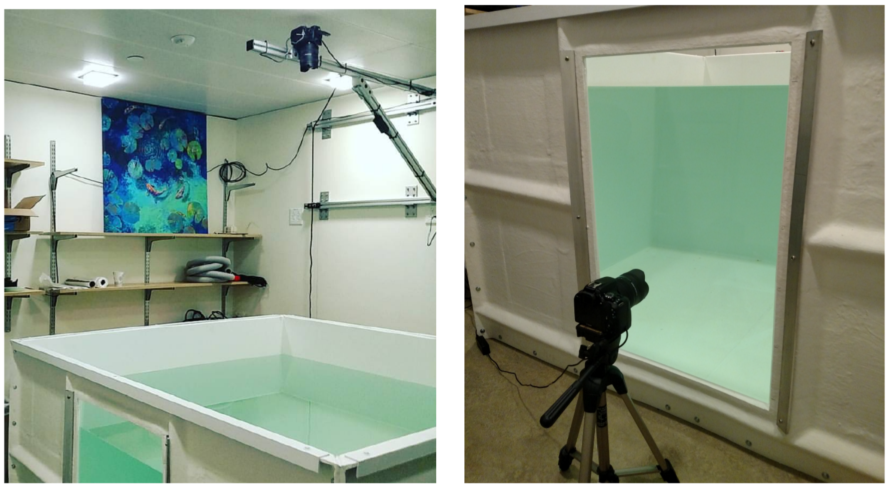


Figure 3.9: Tank setup. Experiments were conducted in a $1.78\,\mathrm{m} \times 1.78\,\mathrm{m} \times 1.17\,\mathrm{m}$ tank. The main camera used to track horizontal movements is mounted above the tank (left). Additional cameras were placed outside and inside the tank to produce the supplemental videos (right).

for ambiguity. The software works by first tracking the video data for Bluebot positions using image processing techniques, then isolating the trajectories of individual robots, and finally matching these trajectories spatially and temporally with depth values from the Bluebots' pressure sensors and fusing all the data to recreate individual 3D trajectories. Our experimental protocol and procedures for multi-robot tracking are described in Appendix A.2

4
A Realistic Three-dimensional Simulation Environment

Only experiments with physical robots can expose behavioral algorithms to the full complexity of real-world physics and test their robustness against hidden failure modes. However, such physical robot experiments are complicated and time consuming compared to the many parameter sweeps that can easily be done in simulation. A simulator that captures essential parts of the robot dynamics and visual interactions is still extremely valuable in accelerating the design of new collective behaviors because it allows to easily experiment with a handful or several dozens of robots, ideal versus noisy locomotion and perception, and environments much larger than our physical tank.

In this chapter, I explain how I modelled Bluebot's perception and dynamics to build a realistic and physically-validated simulator named Bluesim. My two main goals were that: i) robot behaviors implemented in simulation can be easily transferred to the physical Bluebots; and ii) reasonably similar observations can be made by analysing the results of simulations and corresponding physical experiments.

4.1 Simulator Architecture

The Bluesim simulator has a central database that keeps track of positions, velocities, relative positions, and distances of all simulated robots. The robots are ordered by a heap data structure, and simulated asynchronously and one at a time. Each robot has access to a local view of its environment; all robots have the same dynamics. Robot variables such as cognition speed or visual range can be changed, as can the perception complexity by introduction of noise, occlusions, and LED parsing. The decision making algorithms use the same logic in simulation and on the physical robots. Their syntax looks alike with `Python3` being used everywhere to facilitate simulator-to-robot transitions. I explain the architecture of Bluesim in further detail by going through one simulation iteration for one robot in Algorithm 1 and Figure 4.1 A. The simulation results are stored in data files from which figures and animations can be generated (Fig. 4.1 B–C).

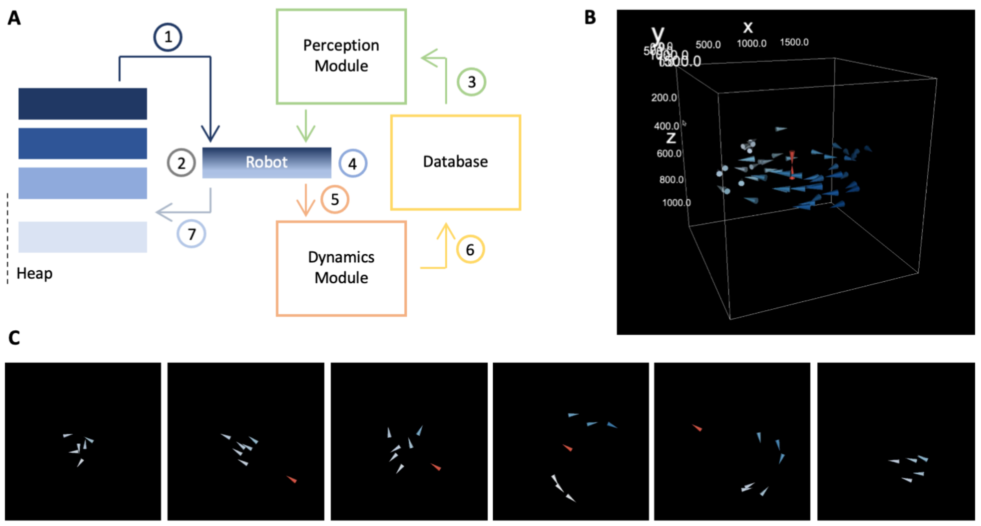


Figure 4.1: Bluesim architecture. **(A)** Robots are sorted in a heap (1,2,7), pull information from a database that keeps track of all robot positions (3,6), and execute behavioral code locally (4). Separate modules simulate perception (3) and dynamics (5). See Algorithm 1 for details. **(B)** Simulation results can be animated and analyzed from different viewing angles. **(C)** Top view on six instances of the evasive fountain maneuver (predator in red).

4.2 Bluebot Perception Model

When deciding where to move next, Bluebots rely on their local perception of neighbors, and decisions can be made based on visible neighbors only. Therefore, the perception of Bluebot was replicated by simulating its visual range (3 m) and narrow posterior blind spot, as well as neighbor occlusions. In addition, Bluebots use imperfect algorithms to parse the multitude of LEDs, remove reflections from the water surface, and assign triplets to individual neighbors. The Bluebot perception model contains reverse geometry to calculate individual LED positions from an idealized robot location in order to test such algorithms.

Bluebot's blind spot is modelled as a cuboid corridor. A neighbor within the blind spot has a relative angle α greater than $\pi/2$ and a lateral distance d smaller than half the cuboid width $w = 50$ mm (Fig. 4.2 A). Bluebots can also be hidden behind each other (Fig. 4.2 B). For such occlusions, each

Algorithm 1: One iteration for one robot in Bluesim.

```
1 A robot got selected for a simulation step because it had the lowest time of all robots
    in the heap.
2 The duration of the simulation step is drawn as a normal deviate with a mean
    equivalent to the expected duration of a single perception-cognition-action cycle
    (0.5 s), and a standard deviation of 10 % (0.05 s).
3 The robot gets its current local view from the central database. This includes either
    the set of visible LEDs after occlusions, or the relative positions and distances to
    visible robots if parsing is not simulated.
4 Based on this local view, the robot decides on where to move next according to the
    pre-programmed behavior and respective algorithms.
5 The dynamics of the robot are simulated for the drawn duration according to where
    the robot decided to move.
6 The attained new position and velocity is entered in the central database. The
    respective relative positions and distances to neighbors are recalculated.
7 The robot re-enters the heap with updated time = time + duration (and not
    necessarily at the end of the heap, allowing to alter the robot order).
```

robot was geometrically simplified as a visually blocking sphere of radius 50 mm that forms the upper base of a blocking conical frustum. Occlusions can be detected by checking whether the relative angle β between two neighbors is smaller than the opening angle δ of the cone between the closer neighbor and the observing Bluebot.

Robots outside the visual range, inside the blind spot, or hidden behind another robot get removed from the set of neighbors taken into account for local decision making. For the remaining robots, positional information can be returned in two ways: i) at the robot level with perfect or noisy relative distance and heading; ii) at the LED level including reflections at the water surface, which allows for the testing of LED parsing and tracking algorithms.

The perception of individual LEDs including reflections at the water surface can be simulated since the central database keeps track of the *xyz* (surge, sway, heave) positions and heading orientation φ (yaw) of all robots. For simplicity, the robot's position coincides with the upper posterior LED 1. By

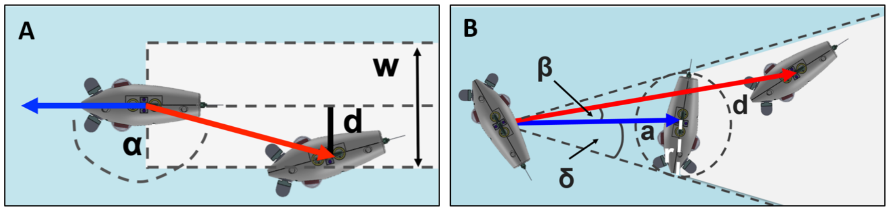


Figure 4.2: Bluebots have near-omnidirectional perception. (A) Robots can be hidden in the blind spot if $\alpha > \pi/2$ and $d < w/2$. **(B)** Robots can be occluded by other robots if $\beta < \delta$.

the known distance $\delta = 86$ mm to the lower posterior LED 2 and upper anterior LED 3 (Fig. 3.2), we can calculate the three LED positions (Eq. 4.1):

$$\begin{aligned} x_{LED1} &= x_{robot} & x_{LED2} &= x_{LED1} & x_{LED3} &= x_{LED1} + \cos\varphi \cdot \delta \\ y_{LED1} &= y_{robot} & y_{LED2} &= y_{LED1} & y_{LED3} &= y_{LED1} + \sin\varphi \cdot \delta \\ z_{LED1} &= z_{robot} & z_{LED2} &= z_{LED1} + \delta & z_{LED3} &= z_{LED1} \end{aligned} \tag{4.1}$$

Surface reflections are added as duplicate LEDs with identical *xy*-positions but a new *z*-position, mirrored at the water surface (Eq. 4.2):

$$x_{reflected} = x_{LED} \qquad y_{reflected} = y_{LED} \qquad z_{reflected} = -z_{LED} \tag{4.2}$$

4.3 Bluebot Dynamics Model

I modelled the dynamics of Bluebot as a non-linear time-invariant system, and expressed the equations of motion as a set of four second order differential equations (Fig. 4.3, Table 4.1, Eq. 4.3–4.6). The forward motion along the *x*-axis (Eq. 4.3), for instance, depends on the thrust forces generated by all fins acting in that direction, and is opposed by the parasitic drag on Bluebot's body as it moves through the water. Other motions along and around the *y*- and *z*-axes follow accordingly. Note that Bluebot

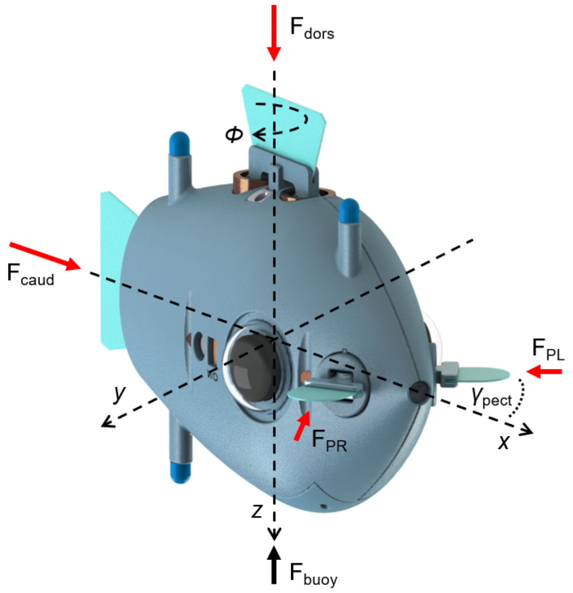


Figure 4.3: Bluebot's dynamics are modelled as a non-linear time-invariant system. Propulsive forces are in red, buoyant force in black, and the coordinate system is indicated by dashed lines. Abbreviations: caud — caudal, dors — dorsal, PL/PR — pectoral left/right, buoy — buoyant.

has four degrees of freedom in total because I made it passively stable against rotations around the *x*- and *y*-axes (roll and pitch) by placing its center of gravity below the center of buoyancy. Consequently, Bluebot's planar motions in the *xy*-plane are decoupled from vertical motions along the *z*-axis; two separate proportional controllers use feedback from the vision system and a pressure sensor for 3D underwater navigation.

Translational motion along the x-axis:

$$\ddot{x} = \frac{1}{m}\left(\underbrace{F_{caud} - \sin\gamma_{pect}(F_{PL} + F_{PR})}_{\text{thrust}} - \underbrace{\frac{1}{2}\rho c_{dx} A_x \operatorname{sgn}(\dot{x})\dot{x}^2}_{\text{drag}}\right) \tag{4.3}$$

Symbol	Unit	Value	Description
F	mN	0-20	empirical thrust forces
c_d	-	0.5-2.1	estimated drag coefficients
A	m^2	0.004-0.011	reference areas
m	kg	0.500	robot inertial mass
I	kgm^2	7.23×10^{-4}	robot moment of inertia
l	m	0.150	robot length (incl. fin)
γ_{pect}	rad	$\pi/6$	pectoral angle
d_{pect}	m	0.055	pectoral distance
ρ	kg/m^3	998	water density

Table 4.1: Parameters for the robot model.

Translational motion along the y-axis:

$$\ddot{y} = \frac{1}{m}\left(\cos\gamma_{pect}(F_{PL} - F_{PR}) - \frac{1}{2}\rho c_{dy} A_y \operatorname{sgn}(\dot{y})\dot{y}^2\right) \tag{4.4}$$

Translational motion along the z-axis:

$$\ddot{z} = \frac{1}{m}\left(F_{dors} - F_{buoy} - \frac{1}{2}\rho c_{dz} A_z \operatorname{sgn}(\dot{z})\dot{z}^2\right) \tag{4.5}$$

Rotational motion around the z-axis:

$$\ddot{\varphi} = \frac{1}{I}\left(d_{pect}\cos\gamma_{pect}(F_{PL} - F_{PR}) - \frac{1}{2}\rho c_{d\varphi} A_\varphi \operatorname{sgn}(\dot{\varphi})\frac{l}{6}\dot{\varphi}^2\right) \tag{4.6}$$

I simulated Bluebot's motion by solving these equations continuously using Euler integration. The dynamics were based on measured parameters from experimentation (thrust forces of fins), the computer design (mass, characteristic dimensions, cross-sectional areas), and comparisons to similar structures (drag coefficients) to closely match the physical motion of Bluebot. The simulator was further physically validated by running identical experiments in simulation and with a single Bluebot, and fine-tuning parameters until the results matched well. Because Bluebots are strong enough to overcome turbulence induced by fellow robots and collisions occur rarely, I decided not to model them.

Bluesim was used to develop and test the behaviors outlined in the upcoming chapters. It further enabled experiments with idealized perception and larger swarms that provided insights into the robustness and scalability of our algorithmic implementations.

5

Implicit Coordination for Three-dimensional Collective Behaviors

Implicit coordination allows a group of fish or robots to coordinate without having to communicate directly among one another. Each member of the group continuously expresses its status to all others through action or mere presence. To coordinate, all members periodically observe and interpret the available information in their local neighborhood. Such implicit coordination constitutes a compelling approach to scalable, robust and effective swarming because it is naturally decentralized, resilient to individual failures and parallel. It further reduces communication complexity in environments where direct explicit message passing is not possible or not desired. The underwater world is one such environment where robot collectives have struggled to achieve group coordination because traditional above-ground approaches based on radio communication and supported by GPS data do not work.

Blueswarm is the first 3D underwater collective that can self-organize with local implicit vision-based coordination only. Bluebots have three LEDs for fellow robots to detect their position and orientation. They achieve multiple 3D collective behaviors by exploiting biologically-inspired coordination techniques that are inherently robust to imperfect knowledge, and that enable the emergence of complex and dynamic global behaviors from seemingly simple interactions.

For this work, we demonstrated well-studied classic collective behaviors like synchronization and dispersion, introduced novel dynamic behaviors like circle formation and milling, and programmed complex tasks such as search operations by connecting multiple behaviors. By focusing on a minimalist form of visual coordination, we were able to achieve versatility and demonstrate programmability for an underwater robot swarm.

The chosen behaviors rely on different levels of implicit coordination. During synchronization across time, Bluebots simply observe when the LEDs of neighbors are on but do not care about positional information. While milling, robots check whether any other robot is present in the front but do not require their distance. Dispersion across space relies on the detection of direction and distance to fellow robots. The search operation builds on dispersion and adds active signaling to trigger a behavioral change from dispersion to aggregation and homing. These examples show that implicit coordination lends itself to the programming and composition of a wide variety of meaningful behaviors. We further demonstrated the ability to replicate existing theory as well as study the theory-experiment gap, and tested the robustness of behaviors to perception errors.

All experiments were conducted with seven robots swimming in a confined freshwater tank of size $1.78 \times 1.78 \times 1.17$ meters (or $13.7 \times 13.7 \times 9.0$ body-lengths). Bluebots solely relied on local visual information, which was acquired and processed onboard in real time, and imperfect motion due to underwater inertia. No external global position information or centralized control was used, and all 3D trajectories were tracked for post-experiment analysis (see Appendix A.2).

5.1 Self-organization across Time through Visual Phase Matching of LED Flashings

Spontaneous synchrony is a classic example of self-organized coordination in time. Millions of fireflies (*Photuris lucicrescens*) synchronize and flash in unison to attract mates every year (Fig. 5.1 A); studies have shown that this global behavior emerges from individual fireflies visually detecting the flashes of neighbors and adjusting to match their phase[86]. For Bluebots, the ability to synchronize can enable time-coordinated actions like sampling of an environment. The approach presented here exploits flashing as a tacit mechanism to achieve synchrony and is based on the well-known Mirollo-Strogatz model[87] (Fig. 5.1 B).

Bluebots were initialized with different start times and programmed to periodically flash with a nominal time interval of $t_f = 15$ s. The program running on each Bluebot (see Alg. 2) proceeded in discrete time steps toward the next flash at t_f, a 2-second-long light up of LEDs, by updating a counter variable n. Whenever a Bluebot i flashed, all observing neighbors j jumped ahead by $m = f(n)$ steps (Eq. 5.1). Mirollo-Strogatz proved that synchrony is guaranteed under any monotonically increasing and concave down function for $f(n)$, for instance $f(n) = \sqrt{n}$:

$$n_i = t_f \quad \rightarrow \quad n_j = \min(t_f, n_j + \sqrt{n_j}) \qquad \forall j \neq i \tag{5.1}$$

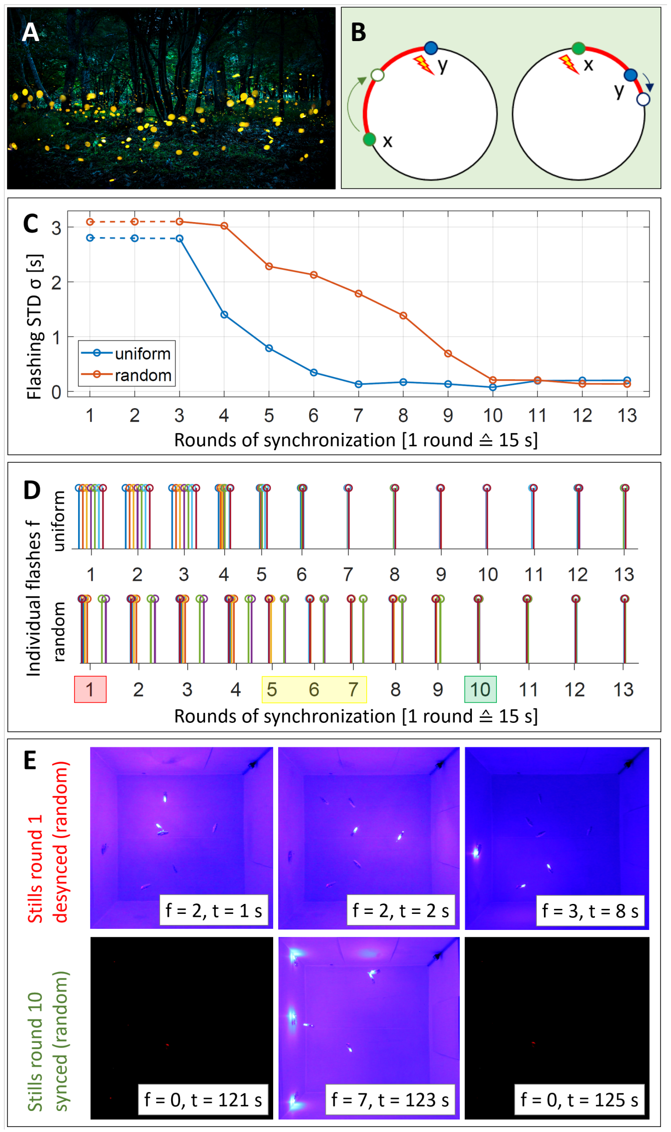


Figure 5.1: Self-organization across time. **(A)** Fireflies flashing in unison. (Source: iStock) **(B)** Mirollo-Strogatz synchronization model: Firing agents pull up observers closer to their firing times, and the pull-up magnitude increases monotonically with time an observer spent already on a given firing cycle. Left: y fires, x is pulled up; right: x fires, y is pulled up; result: their phase difference (red) was reduced. After multiple such rounds, x and y will fire in unison. **(C)** Seven Bluebots observed LED flashes of neighboring robots and adjusted their flash cycles to achieve synchrony (solid) after three initial rounds of desynchronized flashing (dashed). The standard deviation σ (STD) in flash times among robots disappeared after four and seven rounds of synchronization for uniformly (blue) and randomly (red) distributed initializations, respectively. **(D)** Robots with randomly initialized flash times synchronized slower because they partitioned into two competing subgroups (rounds $5 - 7$). **(E)** Stills from the randomly initialized experiment show uncoordinated (top, round 1) and synchronized (bottom, round 10) flashing.

Algorithm 2: Coordination in Time Based on LED Flashings

```
1  INPUTS:
2  $t_f = 15$ s # flash time interval
3  $d_f = 2$ s # flash duration
4  $f_{thresh} = 0.01$ # brightness threshold for flash (empirically tuned)
5  INITIALIZE:
6  $n = 0$ # counter variable
7  $t_{now} =$ time.time() # current time
8  $t_{lastflash} = t_{now}$ # time of last flash
9  $t_{lastupdate} = t_{now}$ # time of last counter variable update
10 MAIN:
11 while time coordination in progress do
12     Take 2 binary images (left and right) in $m \times n$ matrix format with only black and
         white pixels.
13     Update time: $t_{now} =$ time.time()
14     if mean brightness in either image $> f_{thresh}$ then
15         Flash detected: $\Delta t = \sqrt{t_{now} - t_{lastflash}}$
16     else
17         No flash detected: $\Delta t = t_{now} - t_{lastupdate}$
18     Update counter variable: $n = n + \Delta t$
19     Update time of last counter variable update: $t_{lastupdate} = t_{now}$
20     if $n \geq t_f$ then
21         Flash LEDs for $d_f$ seconds
22         Reset counter variable: $n = 0$
23         Update time of last flash: $t_{lastflash} = t_{now}$
```

Bluebots checked for flashes by capturing images with both cameras and searching them for LED blobs. This took up to 1 s, and the exact time of each capture is not controlled during that interval. Therefore, a flash duration of 2 s turned out to be at the lower end in order to achieve reliable detection without missing flashes. The 15 s long nominal time interval was chosen accordingly with the goal to avoid substantial overlap of flashes among the robots (i.e., quasi-synchronization) at initialization.

Two types of experiments with seven Bluebots moving randomly underwater were run: one with all the robots initialized uniformly over the 15 s time interval, and one with the robots initialized randomly, which often led to opposing sub-groups (a sort of worst case scenario). In both scenarios, the discrepancy between flashing times quickly decayed as Bluebots achieved synchrony within 105 seconds (Fig. 5.1 C–E). Final standard deviations between flash times were as low as 0.20 s for the uniform and 0.14 s for the random initialization. The total durations of synchronized flashes in the final 13th rounds were 2.55 s for the uniform and 2.36 s for the random initialization, which are close to the theoretical 2 s best case.

One of the key features of this algorithm is the simplicity of interactions: an individual Bluebot does not need to distinguish between neighbors. Because of the importance of time synchronization to many applications, several implementations of firefly inspired synchrony exist in sensor networks and robots[88–91]. The results with Blueswarm show that this same approach also works well underwater, where access to global clocks is much more challenging than above ground.

5.2 Self-organization across Space through Attractive and Repulsive Virtual Forces

Biological collectives also self-organize spatially, for example fish shoals disperse over a region to feed or defend, but stay connected as a group[7] (Fig. 5.2 A). Control over the spread of a robotic collective is important, for instance, to disperse robots for better coverage during environmental sampling or search, or to aggregate robots for recovery[8]. Fish shoaling and dispersion have been extensively modelled[7,24,92,93]. Most models assume that an individual fish experiences virtual forces from nearby neighbors based on distance, with neighbors that are too close repelling and those further away attracting, although the exact form of the virtual forces is unknown[7]. Controlled dispersion has also been extensively implemented in 2D ground robots, ocean surface robots, and some 3D aerial robots[12,14,33,41,42,94]; typically, robots detect relative positions of neighbors by using an infrared communication ring or by explicitly exchanging GPS positions wirelessly. In contrast, fish use vision to determine relative positions of neighbors, and implicitly react without any direct communication. Regardless of implementation, the emergent result of the virtual force model is the same: the fish school or robot swarm tends to disperse over an area, and the balance of repulsive versus attractive forces determines the density and spread of the group[7].

To demonstrate coordination in space, I implemented fish-inspired dispersion using implicit interactions: each Bluebot attempts to visually determine the relative distance and bearing of all visible neighbors in real-time, compute their forces, and then move in the direction of the resulting 3D motion vector (Fig. 5.2 B, Alg. 3). I picked a commonly-used artificial potential, the Lennard-Jones potential[8,93,95,96], to model the non-linear interaction between robots based on relative positions extracted in real time from onboard vision. A single adjustable parameter, namely a target neighbor distance d_t, controls the spacing of the collective. Neighbors j closer than d_t exert a repulsive force on a robot i which approaches infinity as robots collide, neighbors farther away an attractive force which decreases to zero for far away neighbors (Fig. 5.2 C). The force contributions F_{ij} of all N visible neighbors are obtained by taking the first derivative of the Lennard-Jones potential V_{ij} with respect to their distances $|r_{ij}|$. The average of all individual forces multiplied with the respective relative positions r_{ij} determines the next move vector $\mathbf{p}_i$ of a robot i:

$$\begin{aligned}\mathbf{p}_i &= \frac{1}{N}\sum_{j=1}^{N} F_{ij} r_{ij} = \frac{1}{N}\sum_{j=1}^{N} \frac{\partial V_{ij}}{\partial |r_{ij}|} r_{ij} \\ &= \frac{1}{N}\sum_{j=1}^{N} -\frac{1}{|r_{ij}|}\left[a\left(\frac{d_t}{|r_{ij}|}\right)^{a} - 2b\left(\frac{d_t}{|r_{ij}|}\right)^{b}\right] r_{ij} \qquad \forall j \neq i\end{aligned} \tag{5.2}$$

Individual forces are repulsive ($<< 0$) if robots are closer than a target distance d_t and attractive (> 0) otherwise (Fig. 5.2 C). The individual force magnitudes scale non-linearly with distance be-

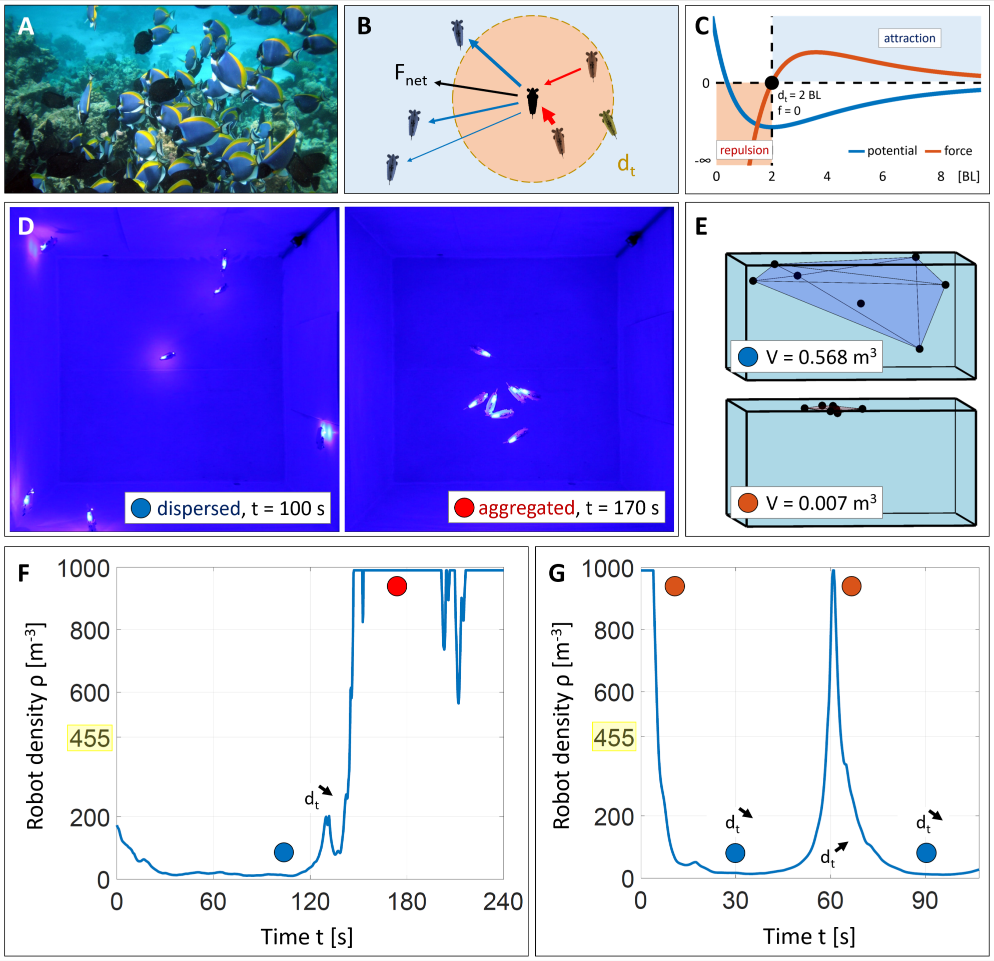


Figure 5.2: Self-organization across space. (A) A shoal of surgeonfish foraging in a reef. **(B)** During aggregation-dispersion, a Bluebot (black) calculates its next move (black vector) as the weighted average of all attractive (blue) and repulsive (red) forces from neighboring robots. **(C)** Inter-robot forces (red) are calculated as the first derivative of the corresponding Lennard-Jones potential (blue) with standard parameters $a = 12$ and $b = 6$, and a tunable target distance d_t ($= 2\,\mathrm{BL}$). The forces f are dependent on the distance d between robots: $f = 0$ for $d = d_t$, $f << 0$ for $d < d_t$ (repulsive), $f > 0$ for $d > d_t$ (attractive). The target distance d_t defines the robot density of the collective. **(D-F)** 3D dispersion (blue markers, $d_t = 2\,\mathrm{BL}$) and aggregation (red markers, $d_t = 0.75\,\mathrm{BL}$) with seven Bluebots. Robot density ρ changed eightyfold; $\rho = 455\,\mathrm{m}^{-3}$ equates to one individual per cube of BL, a density commonly observed in fish schools[97]. **(G)** Dynamically repeated aggregation and dispersion by change of d_t between $0.75\,\mathrm{BL}$ and $2\,\mathrm{BL}$.

Algorithm 3: Coordination in Space Based on Virtual Forces

```
1  INPUT:
2  $d_t$ # target distance (set to 0.75 BL for aggregation and 2 BL for dispersion)
3  MAIN:
4  while space coordination in progress do
5      Take 2 binary images (left and right) in $m \times n$ matrix format with only black and
         white pixels.
6      Obtain a list of blob centroids by running rapid LED blob detection (Alg. 6) on
         each image.
7      Parse the LEDs and isolate individual neighbors (Section 3.1.2).
8      Derive the relative positions $r$ and distances $|r|$ to all neighbors (Section 3.1.2).
9      Calculate the next move vector $\mathbf{p}$ based on Equation 5.2.
10     Use the fins to move along $\mathbf{p}$. Set fin power based on $|\mathbf{p}|$ and momentary error,
         using proportional control.
```

tween robots with an emphasis on nearby neighbors that exert extreme repulsive forces to avoid collisions, or significant attractive forces to maintain cohesion. The stronger the final averaged force, the higher the oscillation frequencies of the actuated fins to swim toward the corresponding direction. Bluebots move continuously to minimize the average of all forces, thereby achieving dispersion with controllable density. The robots are not, however, moving into particular and stable formations.

5.2.1 Experimental Results

We tracked 3D trajectories for all experiments (see Appendix A.2), and report on several commonly used metrics[7], such as density, volume, and average nearest neighbor distance (NND). In the following, we provide detailed analyses of the dispersion behavior, including an investigation of inter-robot distances and the number of visible neighbors over the course of an experiment.

Inter-robot Distances

We ran two types of experiments on dispersion with the same inter-robot target distances of 2 body lengths for the dispersed and 0.75 body lengths for the aggregated state, respectively. In the first experiment, seven Bluebots were centrally deployed and I set $d_t = 2\,\mathrm{BL}$ during the first 120 s ("dispersed state"), and $d_t = 0.75\,\mathrm{BL}$ during the second 120 s ("aggregated state") (Fig. 5.2 D). I measured robot density ρ as the number of robots N divided by the volume V of their convex hull. Results show that the robot density quickly plateaus within 30 s after d_t is set (Fig. 5.2 F), and that a large density and volume change can be achieved ($\rho = 12\,\mathrm{m}^{-3}$ and $V = 0.568\,\mathrm{m}^3$ in dispersed, and $\rho = 990\,\mathrm{m}^{-3}$ and $V = 0.007\,\mathrm{m}^3$ in aggregated state). In the dispersed state, the convex hull of the Bluebots is able to cover a large fraction of the tank (Fig. 5.2 E), amenable for coverage or search. In the aggregated state, the robots group tightly together, although this creates collisions that temporarily break the group. As an additional metric, I measured average nearest-neighbor distances $\mathrm{NND} = 0.8\,\mathrm{m}$ (~6 BL) in dispersed and $\mathrm{NND} = 0.2m$ (~1.5 BL) in aggregated state. The parameter d_t acts as a conservative lower bound for NND since a single too-near neighbor can trigger additional dispersion due to the heavily non-linear Lennard-Jones potential. Experiments with $d_t > 2\,\mathrm{BL}$ did not increase dispersion because Bluebots started to collide with the tank boundary frequently. When fish congregate in schools, typical densities are on the order of one fish per cubic body length[97] with distances between nearest neighbors ranging from 0.5 BL to 4 BL[66], which is similar to distances achieved by Bluebots during density control experiments.

Overall, Bluebots were able to hold the average distance to neighboring robots relatively constant (Fig. 5.3 A). The differences in measured average distances among the robots are expected, and larger in dispersed (~50 s – 100 s) than aggregated (~170 s – 230 s) state (Fig. 5.3 A): a robot positioned around the center should have a lower average distance to neighbors compared to another robot at the periphery. In contrast, the distance to a single nearest neighboring robot (NND) fluctuated con-

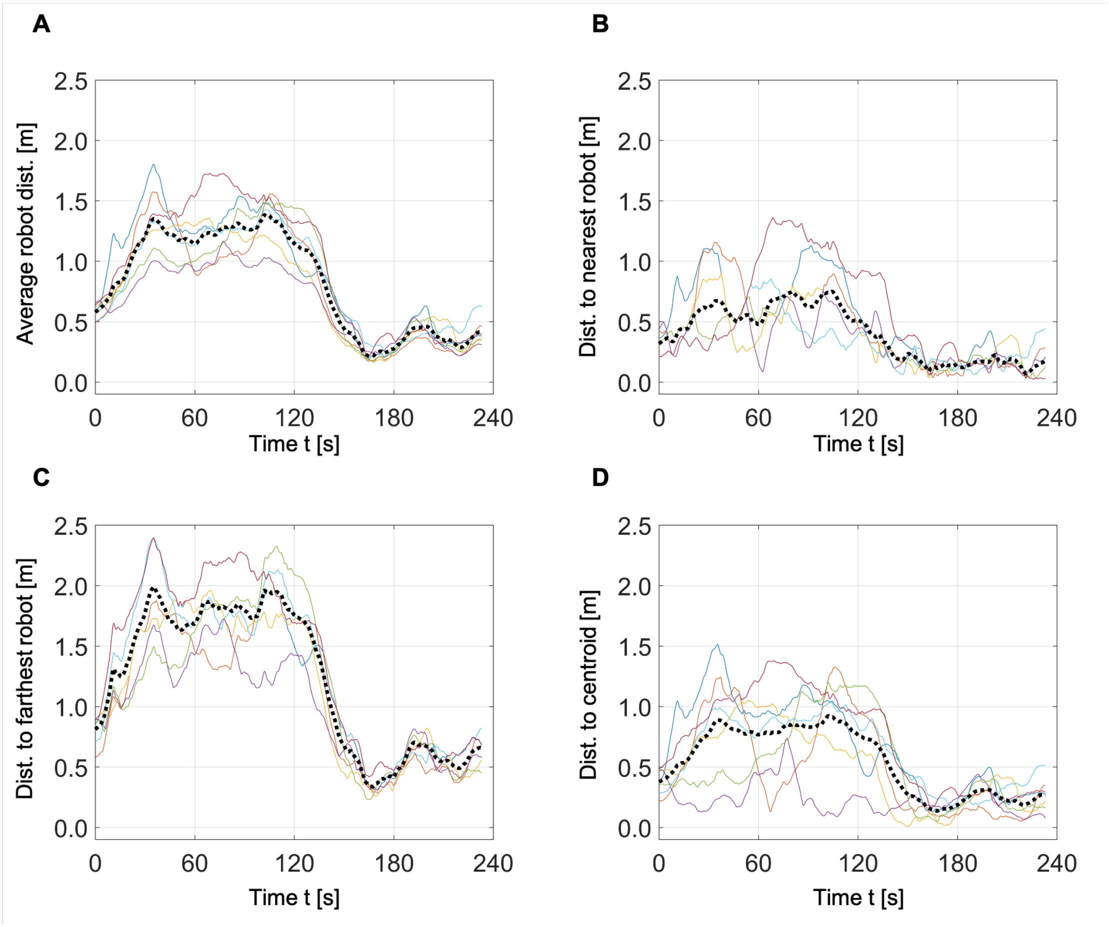


Figure 5.3: Inter-robot distances during single aggregation-dispersion: **(A)** average distance to all neighboring robots; **(B)** distance to the nearest neighboring robot (NND); **(C)** distance to the farthest neighboring robot (NND); **(D)** distance to the momentary centroid of all robots. Colored solid lines represent individual robots; the black dashed line represent the average of the seven-robot collective.

siderably, especially during dispersion (Fig. 5.3 B). This shows that Bluebots continuously adjusted their positions in an effort to minimize the virtual forces acting on them, whereby the nearest neighbor is more likely to exert a strong force than far away neighbors. The distance to a single farthest robot (Fig. 5.3 C) is generally larger than to the nearest robot (as expected), and within the longest dimension (2.97 m) of our test bed. Finally, the distances of all robots to their momentary collective centroid demonstrates that they spread out effectively during dispersion (with some robots close and

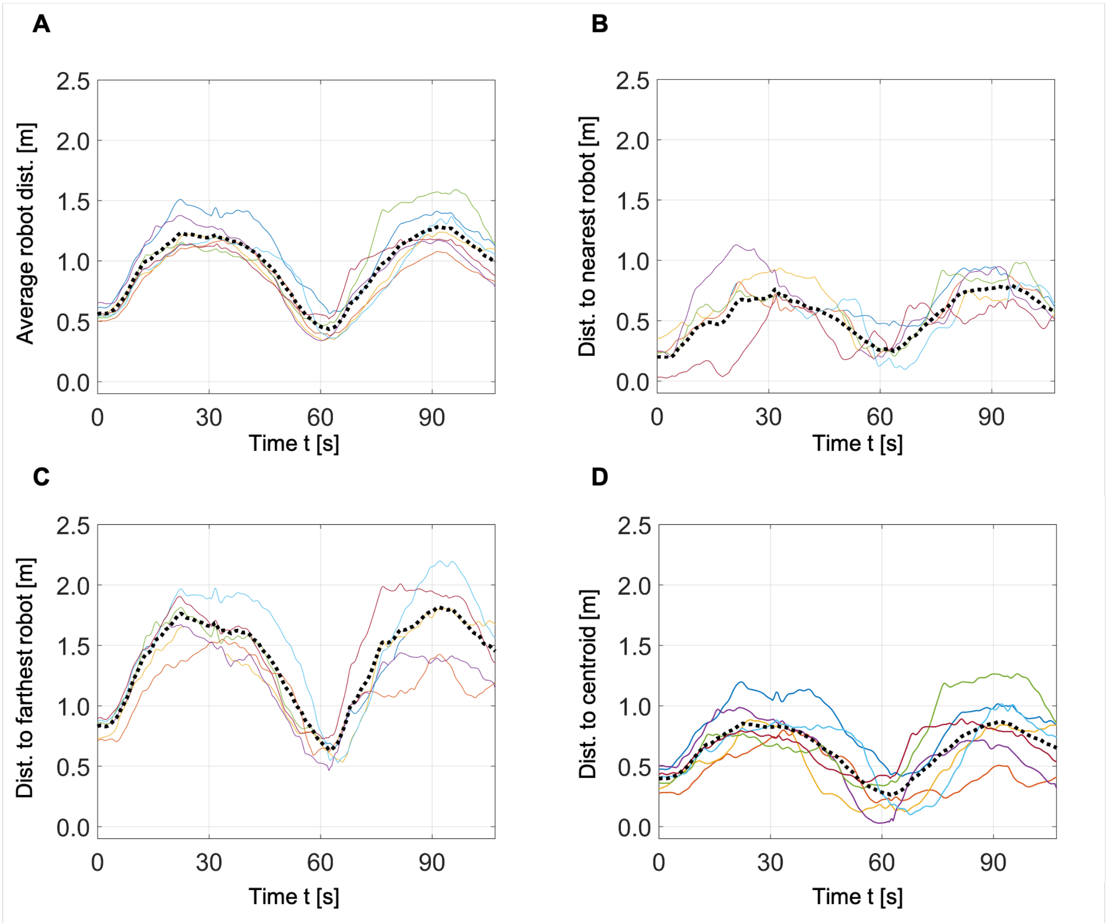


Figure 5.4: Inter-robot distances during repeated aggregation-dispersion: (A) average distance to all neighboring robots; **(B)** distance to the nearest neighboring robot (NND); **(C)** distance to the farthest neighboring robot (NND); **(D)** distance to the momentary centroid of all robots. Colored solid lines represent individual robots; the black dashed line represent the average of the seven-robot collective.

some far from the centroid), and reduced this spread during aggregation (Fig. 5.3 D).

To demonstrate dynamic and repeatable control over robot density, a second experiment was conducted (Fig. 5.2 G), during which d_t was varied four times in the following sequence: (d_t = 2 BL, t = 0 s), (d_t = 0.75 BL, t = 30 s), (d_t = 2 BL, t = 60 s), (d_t = 0.75 BL, t = 90 s). The density results mirror the results from the first experiment, showing that it is possible to quickly and repeatedly switch between dispersed and aggregated states. Interestingly the trajectories in this condition resemble trajectories seen during "flash expansion" [5,7,98], where an aggregated group of fish or insects are startled by an overhead predator and seem to "explode" away from the center of the group but then later re-aggregate. In experiments with Bluebots, when the target distance shifts from aggregation to dispersion, the artificial potential directs the robots away from the center of the swarm, seemingly aligning their heading radially away from the center of the aggregation without any explicit alignment sensing.

The average distances during dispersion and aggregation peaks are comparable between the two experiments: approximately 1.2 m and 0.5 m in Figure 5.4 A versus 1.3 m and 0.4 m in Figure 5.3 A. The dynamic switching between dispersion and aggregation (Fig. 5.4 A–C) appears to yield marginally more uniform inter-robot distances than programming robots to hold dispersion and aggregation for prolonged durations (Fig. 5.3 A–C).

Number of Visible Neighbors

During dispersion, Bluebots estimate the distances to their visible neighbors from pairs of vertically stacked LEDs they believe to belong to individual robots (Section 3.1.2). This estimation is noisy by nature of the visual system, but more drastic errors are introduced when the parsing algorithm mistakenly assumes two LEDs to belong together and represent a fellow robot. Furthermore, neighboring robots may not be seen at all due to occlusions, and occasional "ghost robots" appear when surface reflections of LEDs are not filtered out correctly.

During the first experiment on controlled dispersion (Fig. 5.2 F), Bluebots dispersed for 120 s and aggregated for the next 120 s. From their onboard log files, we know how many neighbors they believed to see in each iteration. 8287 neighbor counts were made by all robots over the time of the entire experiment. On average, individual robots saw between 3.9 and 4.7 neighbors during the experiment (Fig. 5.5 A); in the ideal case each robot would see all 6 neighbors at all times. The collective median was lower during dispersion (4.2) than aggregation (5.6), most likely because more robots were hidden

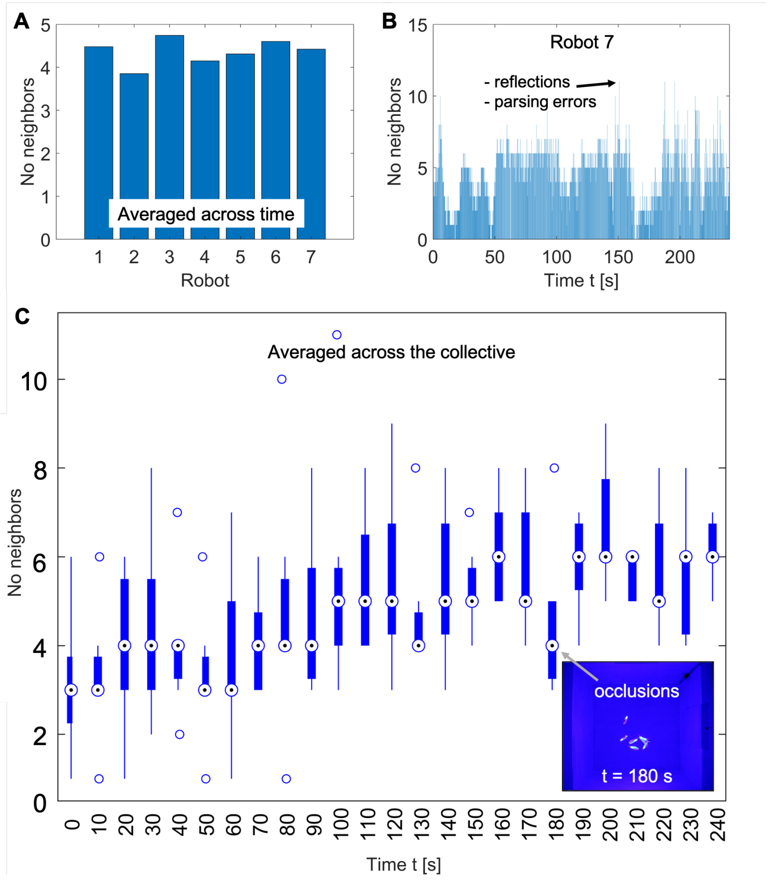


Figure 5.5: Number of visible robots during controlled dispersion: **(A)** for each of $N = 7$ robots, averaged across time; **(B)** for robot 7 at each sampling iteration; **(C)** for each of 25 equally spaced instances in time, averaged across the collective of $N = 7$ robots, showing the median, 25^{th} and 75^{th} percentiles, most extreme data points, and outliers.

in the blind spots (Fig. 5.5 C).

921 neighbor counts (11 % of all counts) were greater than 6; i.e., impossible because there were only 7 robots total (see Fig. 5.5 B for such instances). 11 % is a lower bound for over-counting since other counts might also include mistaken neighbor identifications. The main reason for over-counts is incorrectly handled surface reflections leading to "ghost robots". Such reflections always appear higher up than the actual robots. As a consequence, over-counts introduce two systematic biases:

- Robots aggregate at the surface because over-counts appear high up and exert an additional attractive force toward the surface.

- Over-counts magnify the z-component (depth) of dispersion (but not for robots that remain at the surface and do not see over-counts).

Despite these shortcomings, the overall density of the collective could be controlled effectively, and a large volume change could be achieved by changing the target distance parameter.

5.2.2 Simulation Results

In order to better understand how parameters affect behavior, and understand how well experimental results track theoretical models, I did additional experiments in simulation. Here I present scaling intuition on the collective spread influenced by the inter-robot target distance and the number of participating robots from an idealized 3D point-mass simulation.

The Effect of Target Distance and Number of Robots on Dispersion

I ran idealized 3D point-mass simulations to study the effect of the target distance and the number of robots on dispersion. The simulations assumed holonomic point-mass robots with omnidirectional perfect vision and constant speed locomotion (1 Bluebot body length per iteration). In this idealized simulation, the environment was unbounded, and the number of robots could be varied easily, as opposed to our physical experiments in a water tank.

Overall, the simulation results show that the resultant average distance between robots grows with the prescribed target distance d_t, and also with the number of robots N_R in the collective (Fig. 5.6 A–D). The distance to the single nearest neighboring robot (NND) grows linearly with the target distance as well, but tends to plateau with the number of robots (Fig. 5.6 E–F). Therefore, the target distance is an effective predictor and control parameter for NND, regardless of the number of robots in the collective. From the linear scalings between the average robot distance and target distance, as well

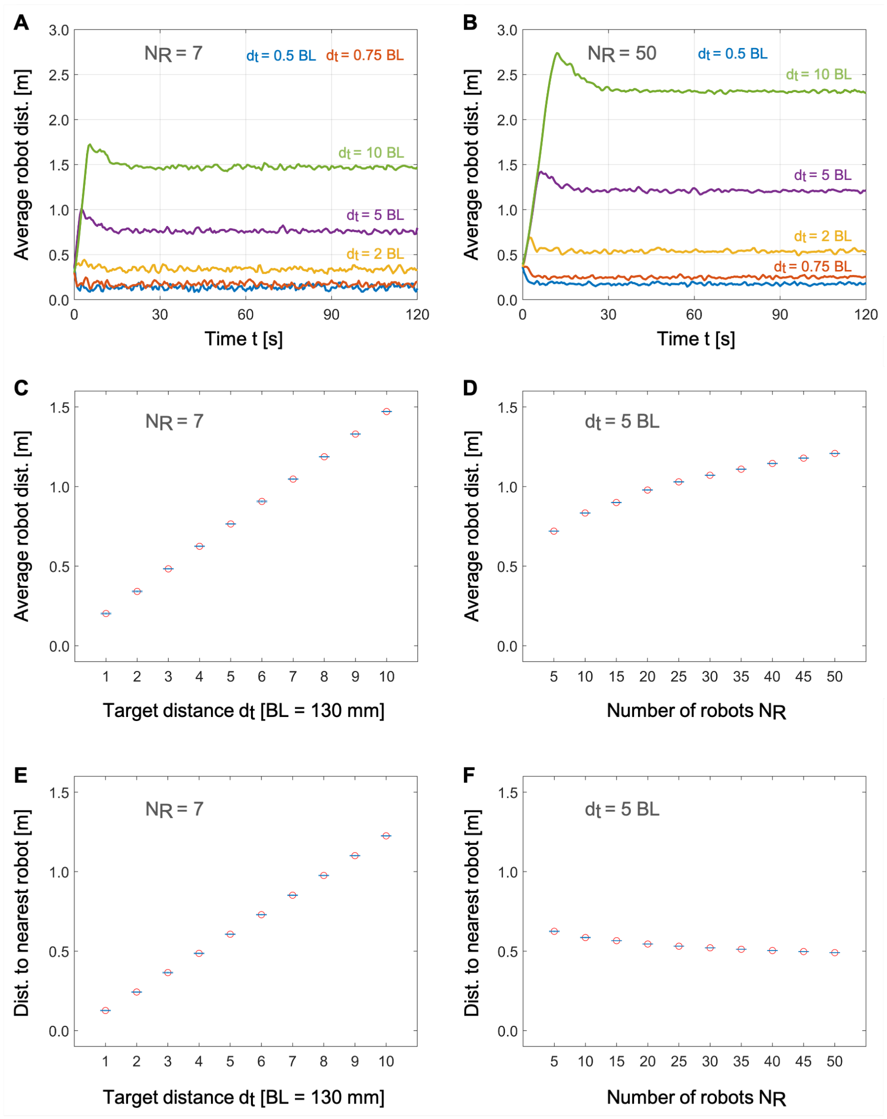


Figure 5.6: Target distance and number of robots influence inter-robot distances during dispersion: (A) the resultant average distance between $N_R = 7$ robots grows with the prescribed target distance d_t; **(B)** the average distance between $N_R = 50$ robots is larger than for $N_R = 7$ robots; **(C)** the simulation runs are repeatable and indicate a linear scaling between the average robot distance and the target distance (BL = body length) for a fixed number of $N_R = 7$ robots; **(D)** the average robot distance also grows with N_R for a fixed $d_t = 5\,\mathrm{BL}$; **(E)** moreover, the distance to the nearest neighboring robot (NND) grows linearly with d_t; **(F)** however, NND does not grow with N_R. Subfigures (A,B) show the time series of a single experiment; (C–F) show the means (red circles) and standard deviations (blue lines) from $N = 10$ simulation runs per data point.

as NND and target distance (Fig. 5.6 C,E), we conclude that it should be theoretically possible to accurately control the spread of a robot collective with the potential-based dispersion protocol used in our real robot experiments.

There is a gap between theory and practice however: in simulation, seven robots spread to average distances of approximately 0.2 m and 0.34 m for $d_t = 0.75$ body lengths (BL) and $d_t = 2$ BL, respectively (Fig. 5.6 C), while in practice 0.4 m and 1.3 m were measured for the same $d_t = 0.75$ BL and $d_t = 2$ BL (Fig. 5.3 A). Two differences between simulations and real robot experiments that could potentially explain this discrepancy are:

- The robots in simulation have perfect vision and see all neighbors at all times, while the real robots do see a limited number of neighbors only (Section 5.2.1). Because of occlusions, it is more likely that the real robots see close than distant neighbors. This may bias the virtual forces toward repulsion and trigger larger than theoretically predicted dispersion.

- The robots in simulation move at constant speed, while the real robots adjust their speeds (or fin frequencies, respectively) according to the magnitude of the net virtual force. Because of the non-linear nature of the potential-based virtual forces, repulsive forces are much stronger than attractive forces. The mapping from attractive net forces to fin frequencies to robot speeds may have been too weak to produce sufficiently strong aggregation in real robot experiments.

The Lennard-Jones potential is one of many possible potential functions for controlled dispersion. The above limitations could be addressed by selecting a different potential function that weighs attractive forces more heavily, and/or enforcing a minimum robot speed even at very low net forces. Although the dispersive effects are greater in the Blueswarm, this does not prevent the collective from aggregating effectively and switching repeatedly between aggregated and dispersed states. Future experimental comparison of multiple dispersion algorithms can lead to a deeper understanding of practical issues and theory-realization gaps.

5.2.3 Conclusions

Bluebots control their collective spread through decentralized attractive and repulsive virtual forces. The collective is capable of changing the volume it covers up to eightyfold in our laboratory environment (Fig. 5.2). Overall, the results show that at the system level, potential-based aggregation-dispersion in 3D can be achieved underwater, using purely local visual interactions without external assistance. Further analysis showed, however, that at the local level, Bluebots see fewer neighbors than theoretically possible — 4.9 on average during the first experiment with expected loss due to occlusion, and occasional misidentified robots due to reflections. In spite of this, the behavior is robust and repeatable and allows for effective changes between low and high densities. In addition, idealized point-mass simulations indicate that the resultant average distance between robots grows linearly with the prescribed target distance d_t and sub-linearly with the number of robots, which shows that fine control over the spread of a robot collective via d_t is theoretically possible.

5.3 Dynamic Circle Formation and Milling based on Binary Sensing of Neighbor Presence

Milling is an impressive dynamic formation commonly observed in fish schools when evading predators[4,7] (Fig. 5.7 A), where the whole school coherently swims in a clockwise or anti-clockwise circular formation, often forming large 3D funnel or ball like shapes. Theoretical models of fish schools suggest that milling may be achieved as a special case of flocking[25–27], through a delicate balance between parameters for alignment and attraction-repulsion, or through radially asymmetric attraction-repulsion. Currently however, such self-organized dynamics formations have not been implemented in physical robots. Experimental studies with flocking in ground and aerial robots[12,94] suggest that detecting the alignment of neighbors is more challenging and noisy than determining position and bearing, and in both cases alignment matching is achieved by robots explicitly exchanging messages with global heading information rather than local perception. Even in the alignment-free form[27], the system parameters need to be carefully tuned, although there is the potential for achieving many more dynamic formations. Currently, it is not fully understood how fish schools actually achieve milling[99], and several biological studies suggest that fish and birds may react to a limited number of neighbors rather than the whole neighborhood[100,101].

Recently a new trend in swarm robotics has been the study of minimalist self-organization, using models inspired by physics and derived by evolutionary algorithms; this work has shown that surprisingly complex behaviors such as aggregation, clustering, and collective transport can be achieved by agents with extremely simple neighborhood sensing (e.g., binary sensing of presence or absence of neighbors, or analog sensing of the amount of neighbors)[102–104]. In one such study, a new behavioral rule for milling-like formation was discovered through evolutionary means by one of my collaborators[30]. Instead of reacting separately to each visible neighbor, this rule relies only on a single binary source of information that indicates whether at least one other robot is within the line of sight. The rule takes the form of a memoryless mapping from each of the two possible cases onto a predefined locomotion pattern: in this case, turning slightly right if no one is visible and turning slightly left if any robot is visible. For many values of turning radii, robots spontaneously aggregate[30]. However, for some parameters, emergent circle formation was observed, where robots spread equidistantly in a circle and rotated indefinitely. This emergent circle formation behavior was demonstrated in a simulation for 2D ground-based robots under the assumption of zero inertia, but so far remains unvalidated on physical robots.

Here, we demonstrate self-organized milling, or dynamic circle formation behavior, for 3D underwater robots based on this minimalist formulation of milling. Instead of a line-of-sight sensor, the behavior rule we defined uses a triangular prism with non-zero opening angle 2α (Fig. 5.7 B). We prove

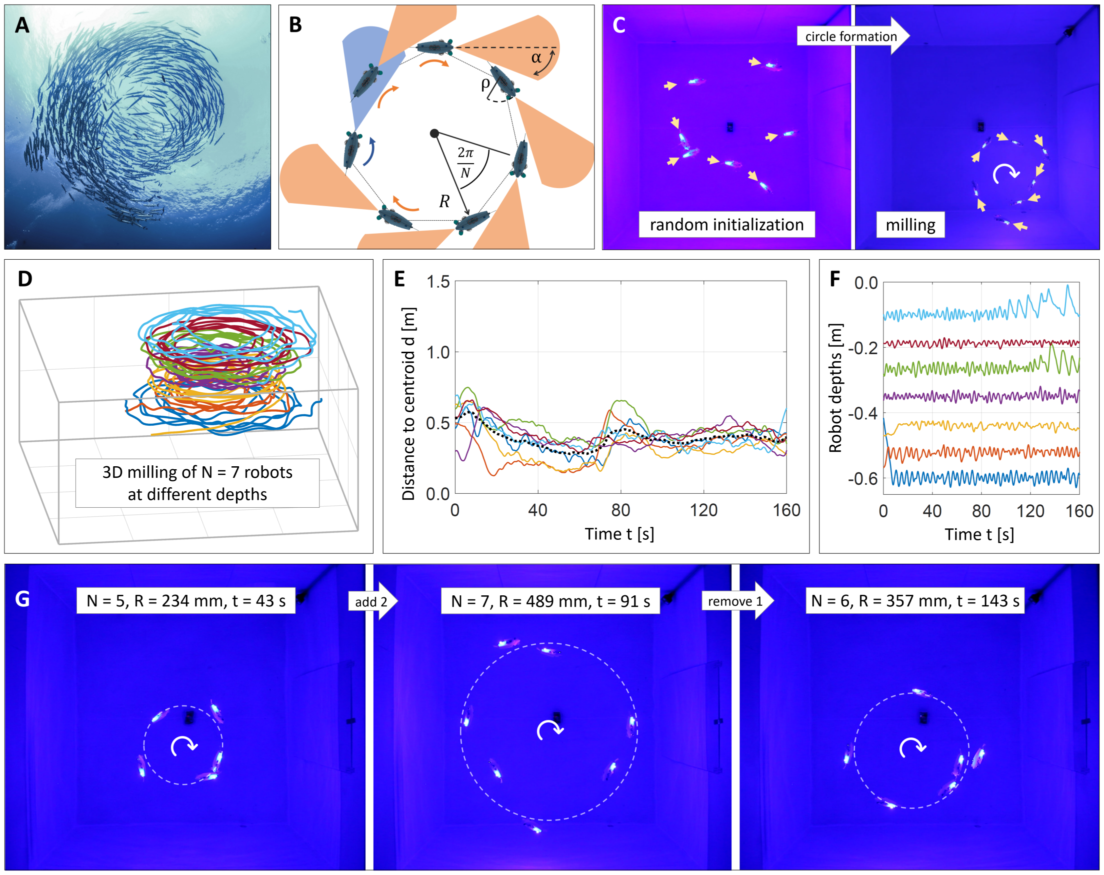


Figure 5.7: Self-organized dynamic circle formation. (A) A school of barracudas milling. (Source: iStock) **(B)** Dynamic circle formation with binary sensors: A robot turns clockwise if no other robots are present within a pre-defined segment view (orange) and counterclockwise if at least one robot is present (blue). The emergent circle size is determined by the angle of view α and the number of robots N (Eq. 5.3). **(C)** Dynamic circle formation on Blueswarm (arrows indicate robot headings). **(D-F)** Experimental data from 3D dynamic circle formation with seven Bluebots, where each Bluebot maintains a preferred depth, resulting in a cylindrical shape: **(D)** Trajectories of all robots; **(E)** Distances to centroid of all robots (colors) and their mean (black, dashed); **(F)** Depths of all robots. **(G)** Addition and removal of robots during a continuous $2D$ experiment demonstrating robustness of the formation process and emergent adjustment of circle radius R to the number of robots N.

that the radius R of the emergent dynamic circle is (see end of this section):

$$R = \rho / \left(\cos\alpha - \cos\left(\frac{2\pi}{N} - \alpha \right) \right), \tag{5.3}$$

where ρ is the characteristic body length and N is the number of participating robots. Note that all robots have the same fixed turning radius, but the actual radius R of the circle emerges as a function of the number of interacting robots and does not depend directly on turning radii; parameter limits are discussed in Appendix A.3. To test dynamic circle formation on the Bluebots, we chose $\alpha = \pi/12$, so that a circle with all seven robots would fit within our tank. Each Bluebot used a pre-computed mask on both cameras that returned a binary value based on whether at least one robot was within the specified field of view. The Bluebot then moved clockwise or counterclockwise depending on its sensor reading, which was achieved by actuating the caudal fin in conjunction with a pectoral fin, such that the ratio of frequencies determines the turning radius. This program is shown in Algorithm 4.

5.3.1 Experimental Results

We tested dynamic circle formation and maintenance with seven Bluebots, all of them spread out on the water surface randomly at initialization (Fig. 5.7 C–F). In our first experiment, we also pre-programmed each robot to dive to a different preferred depth, such that the dynamic formation is 3D, forming a rotating cylinder similar to some natural observations of milling. Tracked 3D trajectories are shown in Figure 5.7 D. The Bluebots were able to form and maintain the dynamic circle formation for several minutes at a time, limited in part by collisions with the tank boundary. While rotating, the Bluebots could maintain a radius accuracy of under 20 % and a depth accuracy of within 5 % (Fig. 5.7 E–F). Unlike simulated or 2D ground robots, the Bluebots are subject to inertia and imperfect motion, and our results suggest that this minimalist rule is robust to the real-world dynamics. We also observed that the collective was often able to recover from collisions with the tank, forming

Algorithm 4: Circling Based on Neighbor Presence

```
1  INPUT:
2  $\alpha = \pi/12$ # defining angle of the prismatic field of view (FOV)
3  $b_{thresh} = 30$ # brightness threshold for neighbor presence (empirically tuned)
4  MAIN:
5  Pre-compute masks that isolate the prismatic FOV in the left and right camera
     images.
6  Run the caudal fin at 3 Hz.
7  while circling in progress do
8      Take 2 binary images (left and right) in $m \times n$ matrix format with only black and
         white pixels.
9      Multiply the images with the respective masks. All pixels outside of the FOV are
         now of value zero, i.e., black.
10     if total brightness in either image > $b_{thresh}$ then
11         Neighbor present: turn counterclockwise by running the right pectoral fin at
             6 Hz.
12     else
13         No neighbor present: turn clockwise by running the left pectoral fin at 6 Hz.
```

a new circle after a short time period. We investigated this robustness further with a second experiment, where we manually removed and added robots at different times to form circles with 5, 6, and 7 robots at the same depth (Fig. 5.7 G). The robots were able to reform the dynamic circle after each perturbation in less than 30 seconds, even in the face of inter-robot collisions. The experiment also confirmed that the radius R of the emergent circle varies with the number of robots N, yielding radii $R = (234, 357, 489)$ [mm] for $N = (5, 6, 7)$ that are close to the predictions from Equation 5.3; i.e., $(190, 309, 496)$ [mm].

Overall, these experiments show that we can achieve milling-like dynamic formations using this simple emergent behavioral rule. The presence of substantial inertia in an underwater setting does not prevent circle formation, however the instantaneous formation at any given time is qualitatively less regular than previous simulations of inertia-free ground robots[30]. To the best of our knowledge this is the first implementation of emergent milling-like formations on physical robots. Our success with milling also illustrates the opportunity for new forms of implicit coordination algorithms, more similar to synchronization than dispersion, in that an individual agent does not need to explicitly detect and react to all neighbors, but rather react anonymously or to some simple summary statistic about the neighborhood (e.g., presence/absence, amount, optic flow, etc.). This may enable more complex self-organization in real robots than previously possible.

5.3.2 Theoretical Results

Here we provide a simplified model overview to derive the radius of the dynamic circle formation; a more detailed model with explicit bounds on all parameter values is presented in Appendix A.3. In addition, we offer some intuition on how circles form.

Derivation of Circle Radius

Assume that we are given a number of robots N, with an approximately circular body of radius ρ, and with a binary sensor whose field of view is defined by the half-angle α. We can compute a formula for the size of a "perfect" circle (Fig. 5.8 A), where each robot is placed equidistant along a circle, oriented in a clockwise direction such that each robot's field of view is empty but just on the edge of detecting the robot in front of it. In this configuration, each robot's binary sensor will detect a zero and the robot will turn clockwise in the next time step. If we consider two adjacent robots and the triangle formed between them (Fig. 5.8 B), we can use trigonometry to derive the formula for the radius of this perfect circle. In the uppermost triangle in Figure 5.8 B, we have:

$$\tan\alpha = \frac{R - R\cos\Theta - \rho\cos\alpha}{R\sin\Theta + \rho\sin\alpha}$$

which can be rearranged to give:

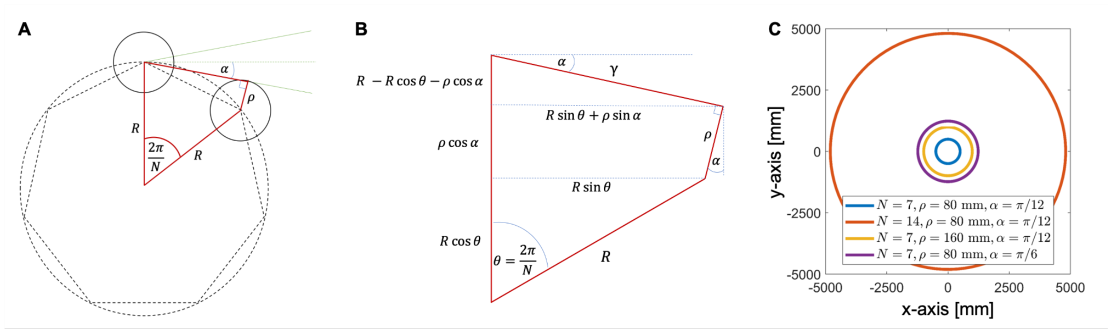

A
α
ρ
R
$\frac{2\pi}{N}$
R
B
$R - R\cos\theta - \rho\cos\alpha$
α
γ
$R\sin\theta + \rho\sin\alpha$
ρ
$\rho\cos\alpha$
α
$R\sin\theta$
$R\cos\theta$
$\theta = \frac{2\pi}{N}$
R
C
y-axis [mm]
x-axis [mm]
$N = 7, \rho = 80$ mm, $\alpha = \pi/12$
$N = 14, \rho = 80$ mm, $\alpha = \pi/12$
$N = 7, \rho = 160$ mm, $\alpha = \pi/12$
$N = 7, \rho = 80$ mm, $\alpha = \pi/6$

Figure 5.8: Dynamic circle formation geometries. (A) A regular polygon configuration with field-of-view sensors. Only two robots are shown for simplicity; additional robots lie on each vertex of the polygon. **(B)** Geometry for calculating the milling radius R with field-of-view sensors (γ, interrobot distance). **(C)** Scaling intuition based on Equation 5.3 for circle radii R with nominal parameters (blue) and doubling of robots N (red), robot size ρ (yellow), and viewing angle α (purple), respectively.

$$R = \frac{\rho}{\cos\alpha - \cos\Theta\cos\alpha - \sin\Theta\sin\alpha}$$

By simplifying the denominator using trigonometric identities and noting that $\Theta = 2\pi/N$ for equidistant robots, we obtain Equation 5.3 and see that larger circles result from more participating robots N as well as larger viewing angles α and body sizes ρ which cause robots to see each other more easily (Fig. 5.8 C):

$$R = \frac{\rho}{\cos\alpha - \cos\left(\frac{2\pi}{N} - \alpha\right)}$$

Given N robots in a perfect circle, we can show that this circle rotates stably under certain assumptions. Full proofs and assumptions are provided in Appendix A.3, here we describe the intuition behind the stability. One key assumption is that the robots' clockwise turning radius when the sensor reports no other robots, r_0, must be smaller than or equal to the milling radius: $r_0 \leq R$. Intuitively we can see that if the turning radius is perfect, i.e., $r_0 = R$, then all the robots will simply rotate in that circle without ever being perturbed; in fact, milling is smoothest if the two values are similar. If

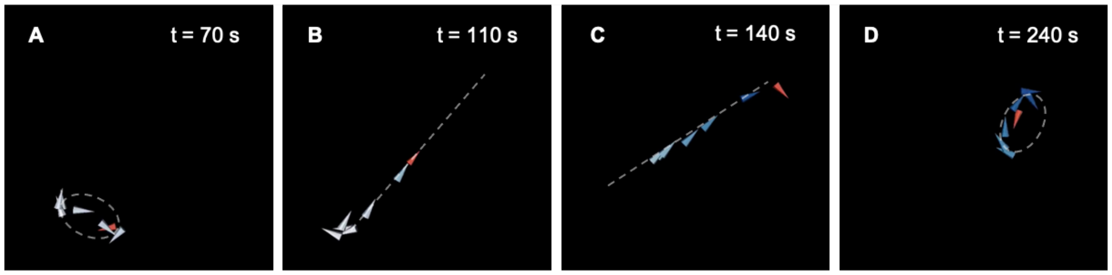


Figure 5.9: Milling is a following behavior. Snapshots of a simulation: **(A,D)** All robots start and finish with the same rule set for circle formation (0 s — 90 s and 130 s — 240 s). **(B,C)** The red robot swims a straight line at intermediate times (90 s — 130 s) without any others changing their behavioral rules.

r_0 is less than R, then in the next step, each robot will rotate slightly into the circle and observe their front neighbor intersect their field of view. This will cause an immediate response to rotate counter-clockwise, once again putting them in the perfect circle. In addition to bounds on turning radii, there are also bounds on α, ρ and response times, which are theoretically derived in Appendix A.3.

In our setting with a maximum of seven robots, we chose alpha to be $\pi/12$, which results in an expected circle radius of 496 mm (tank planar dimension is 1.78 m $\times$ 1.78 m). We tuned pectoral fin actuation frequencies to 6 Hz for clockwise and counterclockwise turning and ran the caudal fin at 3 Hz to swim such circles.

Intuition for Circle Formation

While we were able to show that robots can maintain circles and derived a formula for circle size, the formation of circles from random initialization is much harder to prove. Here I offer some intuition on why circles form and back them with a simple simulation. If we consider just one robot that turns left when seeing anyone and right otherwise, that robot will follow anyone it sees. It namely turns right until it sees anyone at first, then left, because it saw someone, then right because it is no longer seeing this someone, and so on. In brief: dynamic circle formation and milling is a following behavior. If everyone follows everyone but no one is opinionated on where to go, a circle forms.

In simulation, I started off with the circle formation described in Algorithm 4. I then programmed one single robot, the one in red, to abandon the circle rules and instead embark on a straight line. The group of robots transitions to leader following, and this can also transition back to milling whenever the red robot changes back to the same rule set that all the others use (Fig. 5.9).

5.4 Multi-behavior Collective Search with Transitions from Search to Gather and Alert

In fish, robots, or even human collectives, the work of scanning surroundings can be shared among the constituent individuals, potentially reducing the burden on each individual while achieving a higher level of collective vigilance. Schooling fish, for instance, find food faster as group size increases[105], and each fish can devote more time to feeding since all others are also watching for predators[106,107]. Swarms of underwater robots may also exploit this "many eyes" effect for collective sampling of oceanic data, mapping of plumes in coastal waters[18,36], or faster search and rescue missions in collaboration with ocean surface and aerial robots[16,23]. Since a search operation may involve several subtasks, it is also a motivational example for swarm programmability, where multiple collective behaviors must be sequenced together[67,108].

In this search experiment, seven Bluebots were placed at the center of the water surface and tasked to search for a red-light source in a bottom corner of the tank (Fig. 5.10 A–E). To achieve this task, Blueswarm combines three behaviors — *Search*, *Alert*, and *Gather* — using flashing LEDs as a visual signal to initiate behavioral transitions (Fig. 5.10 F). As a first step, the robots used the dispersion behavior described earlier to collectively *Search* by spreading out in the tank (with $d_t = \infty$ for unbounded dispersion). The first Bluebot to detect the red-light source (i.e., within a range of approximately 3 BL) switched to *Alert* behavior, where it holds its position and flashes its LEDs at 15 Hz as a signal to recruit others. As other robots observed the flashing signal, they switched off their LEDs and

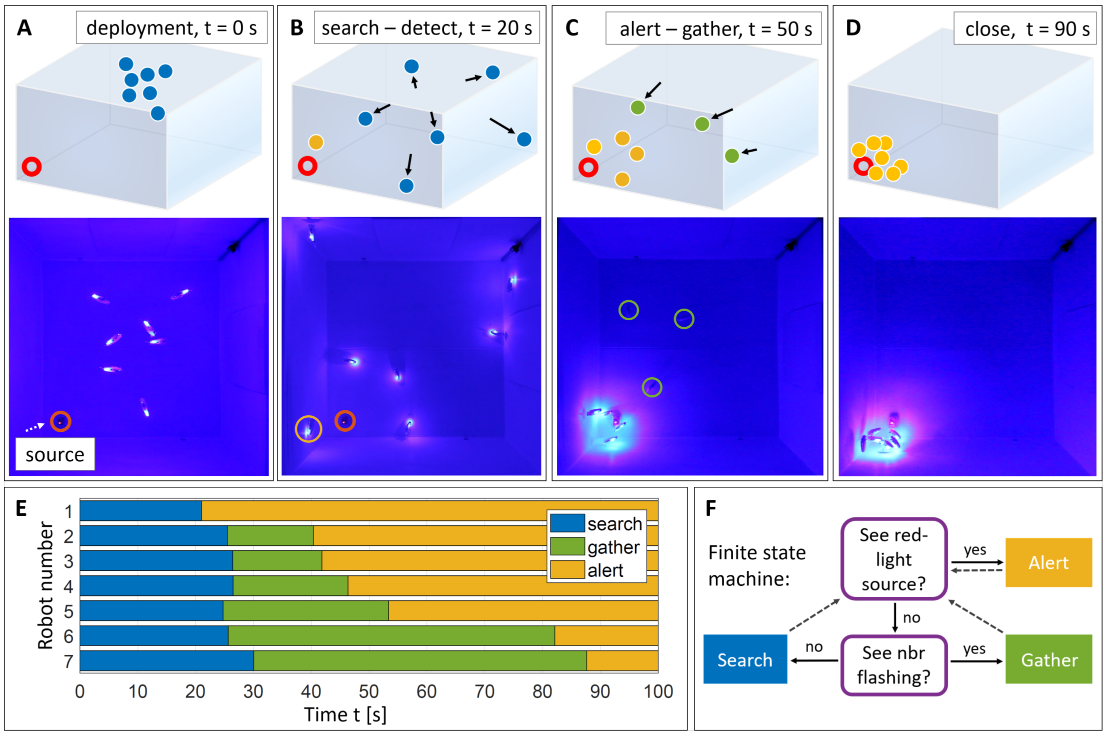


Figure 5.10: Search operation composed from multiple behaviors. **(A-D)** Experimental validation: **(A)** Seven Bluebots were deployed centrally and searched for a red-light source at the left bottom corner of the tank. Robots switch between three behaviors: Search, Gather, and Alert, indicated by blue, green and yellow in figure diagrams. **(B)** Initially, the robots dispersed to cooperatively locate the source. The first robot detecting the source switched to Alert behavior, maintaining position and flashing its LEDs ($15\ \mathrm{Hz}$). **(C)** Other robots close by the source also detected it and switched to Alert. Further away robots that had detected the flashing LEDs switched to Gather, turning off their LEDs and moving towards the flashing robots. **(D)** The experiment concluded when all robots had found the red-light source. **(E)** The timing of events shows the cascade of information spreading. The first robot detected the source after $20\ \mathrm{s}$ of controlled dispersion. Within $10\ \mathrm{s}$, all other robots noticed its alert and started migrating toward the flashing LEDs. Incoming robots catching the light source started flashing as well to reinforce the alert signal. The source was surrounded by all robots after $90\ \mathrm{s}$. **(F)** During search, all robots acted according to the same finite state machine.

Algorithm 5: Multi-behavior Collective Search

```
1  INPUTS: Empirical parameters for sub-behaviors including red-blue threshold for
     source detection, streak length for flash detection, and target distance for dispersion
2  MAIN:
3  status = "search" # initialize robot status to Search
4  while multi-behavior search in progress do
5      Check for red-light source by taking 2 RGB-images (left and right) and searching
         for a blob with a high red-to-blue ratio.
6      if found source then
7          Update status to Alert: status = "alert"
8          Flash LEDs at 15 Hz
9      else
10         Check for flashing neighbors in a sequence of 30 binary images (see Alg. 7).
11         if found flash then
12             Update status to Gather: status = "gather"
13             Switch LEDs off
14             Swim toward flashing neighbor
15         else
16             Keep searching new areas by dispersing from neighbors (see Alg. 3).
```

moved toward the signal (*Gather* behavior). Once they were close to the red-light source, they also started to flash, thereby reinforcing the *Alert* signal (Fig. 5.10). This search protocol is summarized in Algorithm 5. Implementation details on detection of the red-light source and LED flashings are discussed at the end of this section.

This search experiment demonstrates the ability to design composite behaviors using signaling as a simple visual communication method, combining both implicit collective behavior and simple explicit state signaling, which allows a leaderless group to work together efficiently on a complex task. The *Gather* behavior mimics the recruitment seen in natural collectives, for example ants recruiting to collectively transport large bait, or bees recruiting to high-value food sources[1]. A recent study suggests that flashlight fish may use bioluminescent flashing to signal during nighttime schooling[54]. For a single robot searching alone, expected red-light source detection time theoretically increases to 1024 seconds (see end of this section). Blueswarm completed the search operation efficiently, with all robots able to detect the source within ~90 s, getting significant mileage from their collaboration (Fig. 5.10 E).

5.4.1 Detection of LED Flashings and the Red-light Source

This composite behavior introduces signaling, where a robot flashes its LEDs at 15 Hz and other robots detect that a robot is flashing. *Flash detection* is achieved by an algorithm that is designed and tuned to be robust to noise and robot motion (see Appendix A.4 and Alg. 7 for further details). The Bluebot captures a rapid sequence of 30 images (in approximately 0.5 seconds) from each of its cameras. The two sequences of images are analyzed separately for the presences of flashes. The flash detection algorithm proceeds in three phases. Firstly, blob identification is performed on each image in the sequence. Secondly, outlier blobs are identified between each two successive images. Outliers are blobs that appear in some location in the first image and are not present in the second anywhere within a small radius of that location. Thirdly, streaks of outliers are identified. A streak is a sequence of outliers such that each outlier occurs within a small radius of the previous one. A flash detection is declared if a streak is sufficiently long. The threshold radii for outlier and streak detection and the minimum streak length for a flash detection were tuned empirically in the tank under experimental conditions to give a good balance between reliably identifying flashes and minimizing false positive errors.

In addition, the robots also must detect a target that emits red light. This detection was tuned to restrict the detection radius to be small (within 3 BL) so that robots would need to search the tank before finding the target. *Red-light source detection* works by comparing the red and blue channels of the image. For source detection, blob identification is performed not on the blue channel of the image, but on the average of the channels (i.e., a greyscale version of the image). After blob identification is performed, every blob larger than a predifined threshold N_b is considered a candidate to be the source.

Blobs that are smaller are not considered to avoid noise and to make sure that Bluebots can only detect the source when they are sufficiently close to it. For each candidate blob, the blue and red values of all the pixels within Moore distance D of the blob's centroid are summed up separately, and the ratio of the red total to the blue total is calculated. A blob is considered to be red (i.e., the source), if this ratio is larger than some value. In above experiments, this value was set to 1.2, and $N_b = 3$ and $D = 2$, respectively.

5.4.2 Expected Search Time for a Single Robot

The search operation exploits cooperativity to be efficient. We can approximate the expected time for red-light source detection if Bluebots do not collaborate, by using a Markov chain to model a random walk at 0.5 BL/s average speed on an undirected graph G. The vertices of G are the integers $0, \cdots, n$ and represent distance to the red-light source. The source is at vertex $n = 40$, the center of the water surface (initial Bluebot location) at vertex $j = 24$, and the furthest location to the source at vertex 0. For $1 \leq i \leq n - 1$, vertex i is connected to vertex $i - 1$ and vertex $i + 1$ at a distance of 0.5 BL (41×0.5 BL $= 20.5$ BL $\equiv$ tank diagonal). In each 1 second long time step, a robot is assumed to move toward or away from the source with equal probability. The expected time to find the source is $(n^2 - j^2)$ s $= 1024$ s (proof using linearity of expectations in Appendix A.4, cf. runtime of the randomized algorithm for 2-SAT). While a random walk is not the most efficient search pattern for an unknown space, our Bluebots are simple robots that so far lack the sensing/motion complexity necessary for more sophisticated methods (e.g., simultaneous localization and mapping).

6

Implicit Coordination with Headings: Bioinspired Evasion

Predation plays an important role in balancing ecosystems and is one driver of evolution, which pushes animal populations toward those individuals or groups which have evolved successful escape strategies[109,110]. Such escape can be witnessed from tiny plankton[111] to bird flocks[112] and fish schools[4,5]. Particularly impressive are social preys that cooperate through local interactions to evade predators collectively. Fish, for example, display several evasive behaviors, ranging from the fountain maneuver[4,5] to flash expansion and bait balls[113]. In doing so, schooling fish demonstrate collective vigilance and resilience that leverages the cognition and actions of all individuals[4,5].

We aimed to mimick the dynamic and parallel coordination of fish during predator attacks (Fig. 6.1), using the example of the fountain maneuver[4,5,114]. When a fish school performs a fountain maneuver, the fish typically encircle the predator and reunite behind it, taking advantage of its high inertia and letting it swim through the void. Many fish species use visual observations of nearby neighbors[6,46–48,115] and have evolved specialized visual patterns called schooling marks[50] for such group coordination that involves alignment before and dynamic evasion during an attack. Remarkably, many schooling behaviors including predator evasion work with limited and decentralized coordination as opposed to any one single fish permanently leading the school[1,7,52,92].

A wealth of experimental and theoretical work exists on flocking and alignment as it is observed in fish schools[24,58–61]. Biologists have described several advantages of schooling — among them protection from predatory attacks — and documented a range of evasive maneuvers[4–6]. Which maneuver fish choose depends on the direction of the attack[116]; split behavior like the fountain maneuver is triggered by attacks from behind. In contrast, the local mechanisms for evasive behaviors have been less well understood. Only a single descriptive model of the fountain maneuver exists and proposes that fish visually monitor the predator to maximize the rate of escape while minimizing the associated energetic cost[117]. To do so, fish are assumed to swim away from the predator at a constant angle determined by the rear limit of their visual field.

For robotic swarms, the ability to escape threats or moving obstacles exemplifies complex dynamic coordination that goes beyond simple and well-studied flocking[118–120]. With such coordination, a group of underwater robots deployed to monitor coastal environments like coral reefs and harbors, could, for instance, react cohesively to circumnavigate oncoming traffic. However, very few examples

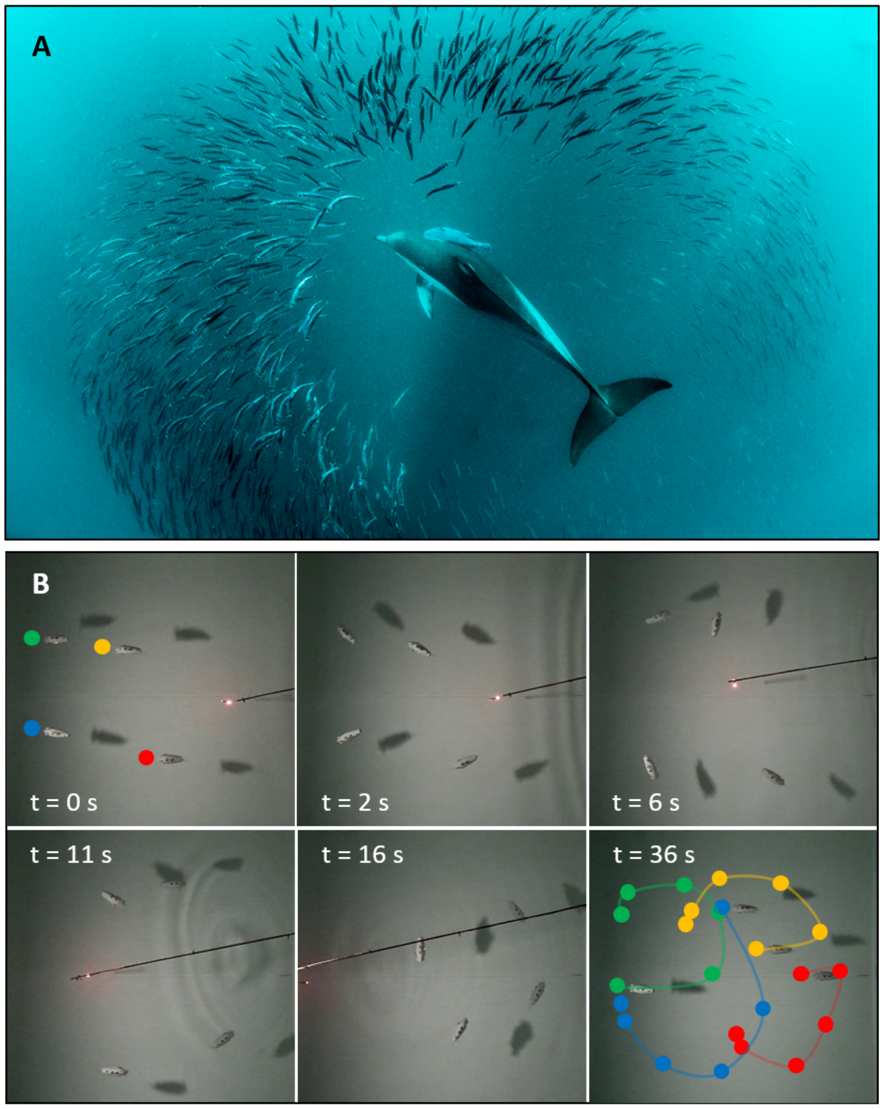


Figure 6.1: Evasive maneuvers. (A) A fish shoal reacts to a predatory dolphin by performing a fountain maneuver. (Source: iStock) **(B)** Four Bluebots (130 mm in length) react to a manually guided predator surrogate. Initially self-aligned and facing to the left ($t = 0$ s), the robots embark on fountain-shaped trajectories to evade the predator, which inadvertently moves through the middle of the shoal ($t = 2$ s until $t = 16$ s). Once the danger has passed, the Bluebots regroup and realign ($t = 36$ s). Color-coded initial positions and trajectories were added to the top left and bottom right snapshots, respectively.

exist of implementing bioinspired evasive manuevers in the robotics domain. Fish-inspired escape was demonstrated with centrally-controlled ground robots[121], and variations of flash expansion and fountain maneuvers were implemented for obstacle avoidance with LEGO robots[122]. In computer graphics, animations of the fountain maneuver were created, however the underlying algorithms used global knowledge that would not be available to fish or robots[123]. In contrast to evasive behaviors, basic heading alignment has been well studied; collective alignment can be achieved with simple averaging algorithms[124] and was shown with ground-based robots that use local perception to detect neighbor headings[94] and with aerial robots that exchange GPS headings wirelessly[12].

The behaviors discussed in the previous chapter all used at most the direction and distance of robots, which can be inferred from the two posterior LEDs. As part of the fountain maneuver, we experimented with alignment in order to achieve coordination into a swarm whose members are coherently pointed in the same direction. This behavior also relies on detecting the headings of neighbors based on additional information gained from the anterior LED.

6.1 A Behavioral Model of the Fountain Maneuver

During a fountain maneuver, fish typically encircle a predator and realign behind it (Fig. 6.1). We modelled the fountain maneuver triggered by a predatory attack as a finite-state machine: robots start in the *Align* state and transition into the *Evade* state if they *Detect* a predator (Fig. 6.2). As a predator, we used two vertically stacked red LEDs on rod that was moved by a human.

6.1.1 Align

During alignment, Bluebots detect and track neighboring robots to infer their headings and rotate toward the unweighted average heading. Following this simple averaging protocol, a convergence proof (by induction) is straightforward under the assumption of accurate and fast enough perception and

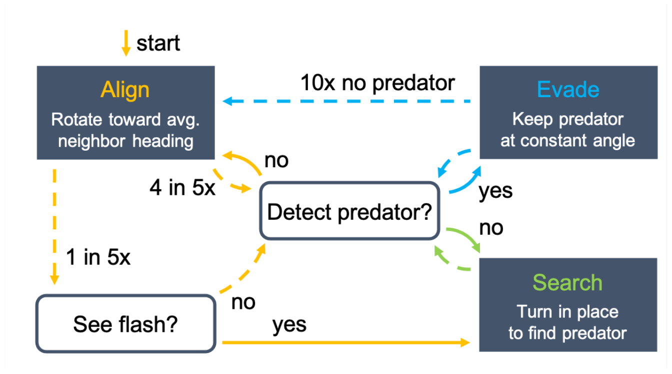

start
Align
Rotate toward avg. neighbor heading
10x no predator
Evade
Keep predator at constant angle
no
4 in 5x
Detect predator?
yes
no
1 in 5x
Search
Turn in place to find predator
See flash?
no
yes

Figure 6.2: Fountain maneuver finite-state machine. Yellow loop: Bluebots transition from the *Align* into the *Evade* state if they detect a predator, or the *Search* state if they see a flashing neighbor (checked in every fifth iteration). Green loop: Flash-alerted Bluebots remain in the *Search* state until they detect the predator themselves and switch into the *Evade* state. Blue loop: Bluebots transition from the *Evade* back into the *Align* state if they do not detect the predator for ten consecutive iterations.

locomotion that lead to monotonically decreasing deviations in robot headings[124]. Our experiments provide insight into how robustly alignment works with Bluebots' imperfect inference of headings, and how tight a convergence bound can be achieved. Here we used a Kalman filter to improve the parsing of LEDs and tracking of neighboring robots by estimation of their positions.

6.1.2 Detect

In order to distinguish the predator (two red LEDs) from fellow robots (three blue LEDs), a Bluebot derives the red-to-blue ratio of all LED blobs. A predator is detected if any two blobs have such a ratio greater than 1.2 and are vertically stacked (from which the distance is known as well). A Bluebot then stops aligning and starts evading the predator.

In addition, Bluebots that have detected a predator can flash their LEDs (at 15 Hz) to alert fellow robots, programmed to check for flashing in a series of 30 images taken in burst mode at 60 frames-per-second. Similar flashing alerts were observed in schools of anchovy, which roll their bodies to reflect sunlight off their shiny silver ventral sides[125].

The flashing signal as well as the distinction of LED colors have been used as well in Section 5.4.

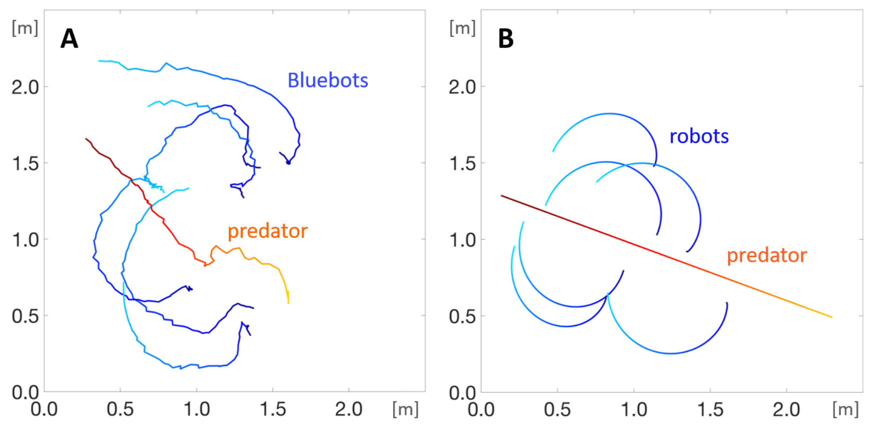


Figure 6.3: Fountain maneuver trajectories. (A) Tracked from an experiment with six Bluebots in the large tank. **(B)** Plotted from a simulation with six robots. **(A,B)** Time (0 s – 27.5 s) indicated by color progression from dark (early) to light (late); robots in blue, predator in red.

Here we tuned both to lean toward false negatives such that evasion is not triggered accidentally.

6.1.3 Evade

Inspired by a descriptive model of the fountain maneuver[117], Bluebots evade a predator by swimming away while keeping it visible at a constant angle of $|60\,°|$. The result is a fountain-shaped trajectory. Bluebots detecting the predator to their right (or left) evade in counterclockwise (or clockwise) direction, using the caudal and pectoral right (or left) fin. Once the predator can no longer be seen, Bluebots complete the maneuver by going back to alignment. We added hysteresis to provide robustness against sporadic misses of the predator (i.e., false negatives due to occlusions), which would result in premature (and potentially fatal) abandonment of evasion.

6.2 The Fountain Maneuver with Physical Robots

We ran experiments at two different test sites and demonstrated that self-organized fountain maneuvers are feasible despite limited perception quality and cognition speed paired with submerged but imperfect motion (Figures 6.1 B and 6.3–6.4). Our in-house testing environment is a square water tank of size $1.78 \times 1.78 \times 1.17$ meters or $13.7 \times 13.7 \times 9.0$ body lengths (henceforth the *small tank*). A larger circular water tank at the Olin College of Engineering had a diameter of 6.1 m and a depth of 1.5 m (henceforth the *large tank*).

When fish escape during a fountain maneuver, they dart away from the aligned school to keep the predator at a safe distance and within their visual field[117]. Accordingly, the three key metrics for Bluebots were: i) stable alignment with circular standard deviations $\sigma_\varphi \leq 0.5$; ii) convergence to a prescribed viewing angle Θ at which the predator is kept; iii) increased distance d to the predator compared to not taking action. Although this behavior has been described extensively in fish literature[4,5,113,117], it is mostly studied qualitatively without standard metrics for quantification.

The fountain maneuver can be split into two main dynamic parts: *Align* and *Evade*. Alignment allows Bluebots to move from a shoaling into a schooling state, enhancing collective order and preparing the school for effective predator evasion. As such, the behavior is also useful for formation control, for example to migrate long distances efficiently.

In two experiments (red, blue) in the large tank, seven Bluebots aligned their headings φ after approximately 15 to 20 seconds, and achieved final average circular standard deviations σ_φ of 0.29 and 0.30, respectively (Fig. 6.4 A). During alignment, Bluebots detected 2.14 (red) and 2.19 (blue) neighboring robots on average (means from 2170 and 2143 samples with standard deviations 1.19 and 1.25). The means and standard deviations at different times over the course of a single alignment run are shown in Figure 6.5. Alignment worked robustly despite robot occlusions and robots that were discarded due to parsing uncertainty. Since the means fluctuated between approximately 1.5 and 3 visible neighbors with significant variation among robots, we conclude that neither the unaligned shoaling nor the aligned schooling state are advantageous for neighbor visibility. Consequently, visual interactions are not facilitated by group alignment.

Bluebot trajectories during the evasion part resembled the fountain maneuver (Fig. 6.3 A).

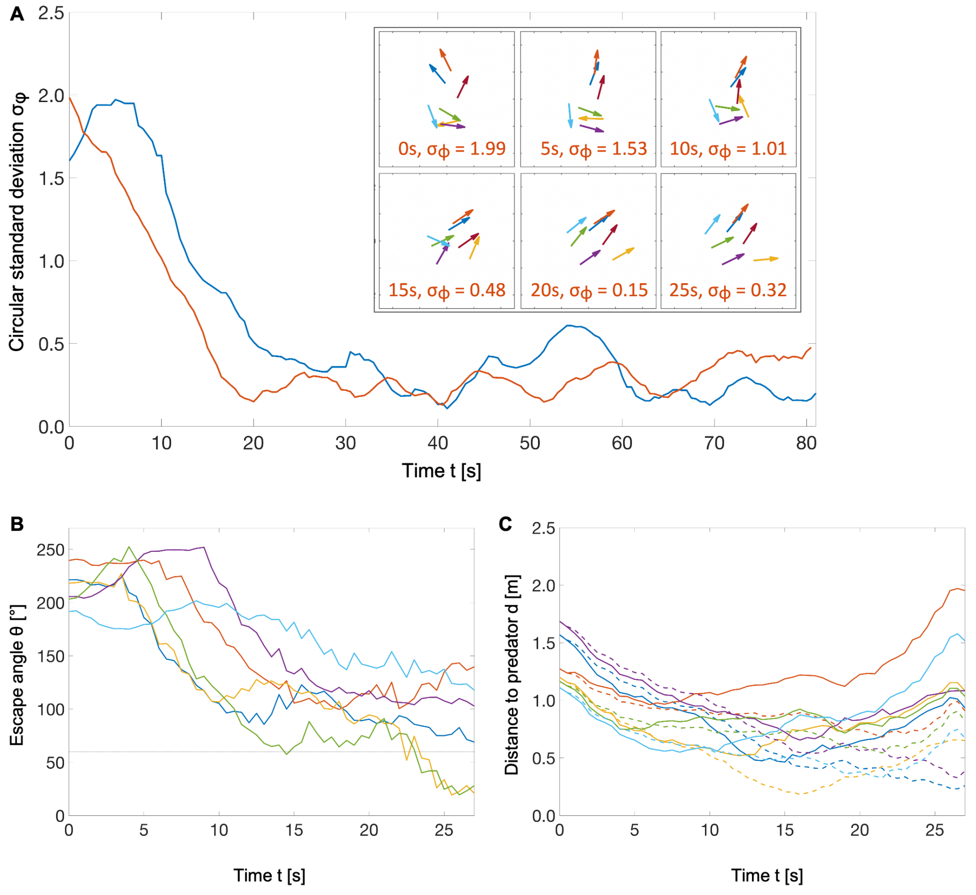


Figure 6.4: Blueswarm escapes the predator (hardware experiments). (A) Alignment with seven Bluebots: The circular standard deviations σ_φ of the headings converged to means of 0.29 (red) and 0.30 (blue) after approximately 15 to 20 seconds. The inlet shows individual headings at selected times during the red experiment. **(B–C)** Escape with six Bluebots (color-coded): (B) Escape angles Θ approached $|60\,°|$ (dashed line) during the fountain maneuver; (C) Distances d to the predator were effectively increased by active evasion (solid lines) compared to remaining idle at the initial position (dashed lines).

The robots rotated away from the predator and approached the prescribed $|60\,°|$ escape angle Θ (Fig. 6.4 B). They maintained an average and minimal distance of 0.93 m and 0.46 m from the predator during active evasion, as opposed to 0.73 m and 0.18 m when remaining idle at their initial positions, an increase of 27 % and 156 %, respectively (Fig. 6.4 C).

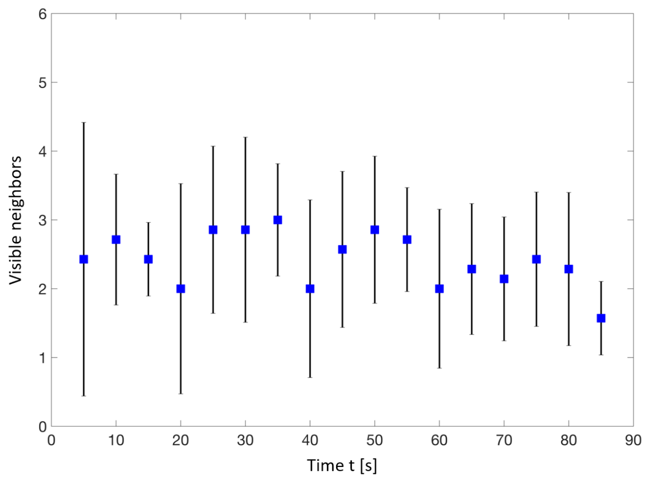


Figure 6.5: Bluebots do not see all neighbors at all times during alignment. Data shows the means (blue squares) and standard deviations (black lines) of $N = 7$ robots.

6.3 Robustness and Scalability from Simulation

Simulations allowed us to repeat experiments to statistically analyze robustness and scalability, as well as to inform and refine algorithmic implementations for the physical-robot experiments (Figures 6.6–6.9). We varied perception quality, neighbor visibility, cognition speed, and number of robots during alignment, and simulated different escape angles Θ.

6.3.1 Perception quality

Alignment relies on the accurate detection of neighbor headings. In simulation, we compared alignment with perfect perception against realistic perception, modelled after Bluebot and including noisy estimates for LED positions and robot headings (zero-mean Gaussians with standard deviations of 2 mm and 10 °, respectively), as well as imperfect LED parsing (Fig. 6.6). This realistic perception was used for the cognition and scalability simulations; alternative noise magnitudes are shown in Figure 6.7. As expected, we found that LED position errors are more severe than noisy headings

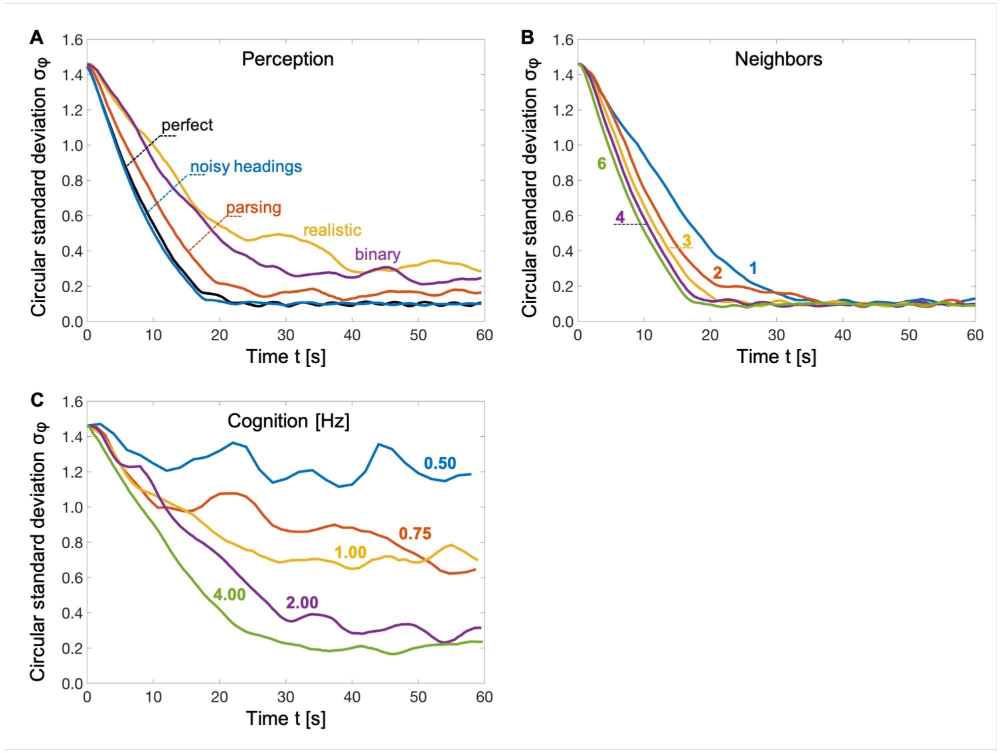


Figure 6.6: Robustness of alignment with seven simulated robots. (A) Perception quality matters critically for alignment. Black: perfect perception; blue: noisy headings with $\mathcal{N}(\mu = 0,\ \sigma = 10\,°)$; red: noisy headings and parsing; yellow: noisy headings, noisy LEDs with $\mathcal{N}(\mu = 0,\ \sigma = 2\,\mathrm{mm})$, and parsing (realistic Bluebot perception); purple: noisy headings, noisy LEDs, parsing, and reduction to binary directions (heading right/left). **(B)** Alignment works robustly, regardless of the number of visible robots, but takes longer with fewer visible robots. **(C)** Faster cognition results in faster alignment. Cognition speeds below $1\,\mathrm{Hz}$ impede the convergence of headings. All data points were averaged across $N = 10$ simulation runs.

(Fig. 6.6 A) because parsing relies on the heuristic that the posterior LEDs are stacked vertically, and wrongly parsed robot LEDs introduce arbitrary headings.

Noisy Headings

The simple averaging algorithm for alignment is surprisingly robust to zero-mean Gaussian noise with standard deviations σ_ϕ of up to $45\,°$ on the final headings (Fig. 6.7 A). We expect that due to the zero-mean and unbiased nature of the noise, such heading errors average out across the collective and time.

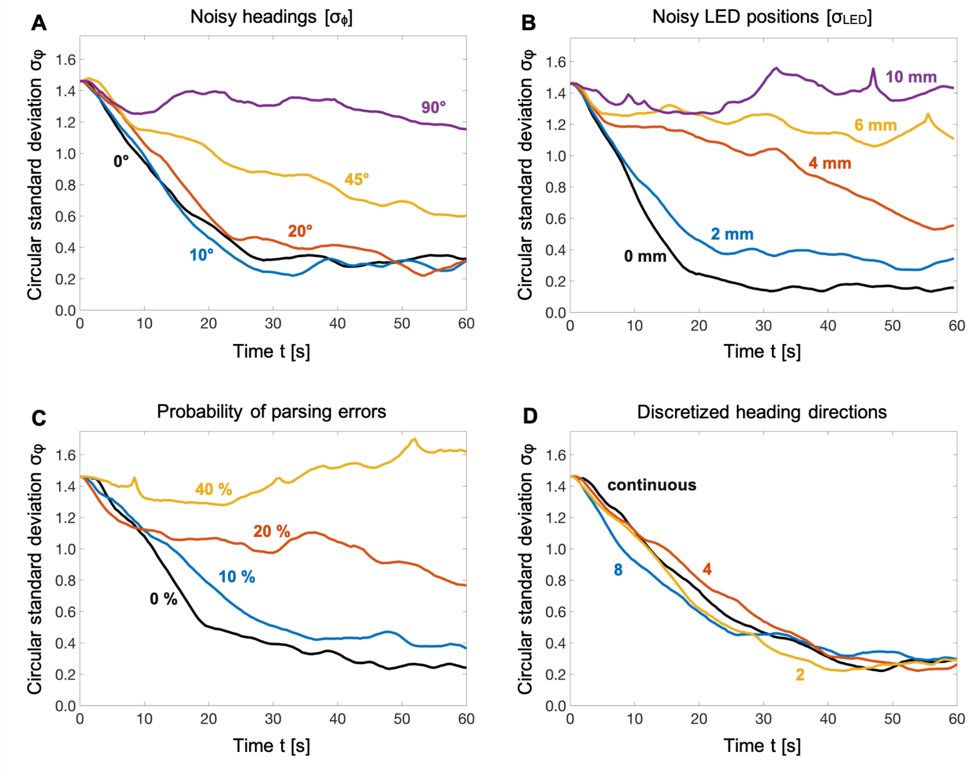


Figure 6.7: Perception quality affects alignment. **(A)** Zero-mean Gaussian noise on heading estimates. **(B)** Zero-mean Gaussian noise on LED position estimates. **(C)** Forced parsing errors at different probabilities. **(D)** Heading estimates reduced to eight, four, and two cardinal directions.

Furthermore, since perception is faster than locomotion, Bluebots motions act as a low-pass filter on noisy heading observations.

Noisy LED Positions

In order to obtain neighbor headings, a Bluebot has to detect robot LEDs and parse them into groups of three that belong to individual robots. Given three LEDs and the fact that the upper anterior LED 3 is on a horizontal circle of known radius around the upper posterior LED 1, the quadratic equation describing that circle can be solved, and the correct solution minimizes the vertical distance between LEDs 1 and 3 (see Eq. 3.2 in Section 3.1.2). Inaccurate LED position estimates can lead to arbitrary heading estimates, namely when causing the choice of the wrong solution. We simulated different levels of zero-mean Gaussian noise on the LED positions, and found that alignment works up to standard deviations σ_{LED} of 4 mm (Fig. 6.7 B).

Erroneous Parsing

On top of noise that may occur naturally due to imperfect sensing, we introduced artificial parsing errors to demonstrate the importance of accurate parsing algorithms. Alignment works as long as parsing errors occur with a probability lower than 20 % (Fig. 6.7 C). This sets a performance benchmark for parsing algorithms.

Discretized Perception

It is unlikely that fish single out individuals in a dynamic school and derive their headings as exactly as possible in order to align. Inspired by fish schooling, we simulated more minimalistic versions of heading-based alignment, in which a robot simply estimates in which of eight, four, or two cardinal directions each neighbor is heading. For instance, if perception was simplified to two cardinal directions, a robot only has to decide whether a neighbor is facing to the right ($0^\circ \leq \phi < 180^\circ$) or left ($-180^\circ \leq \phi < 0^\circ$). A neighbor heading to the right gets assigned a heading angle of 90°, and a neighbor heading to the left a heading angle of -90°. Following the standard alignment algorithm, the robot then takes the average of all neighbor headings. Interestingly, simulation indicates that precise heading information is not required for alignment: convergence with realistic perception was similarly good for exact headings and headings that were reduced to eight, four, or two directions (Fig. 6.7 D).

6.3.2 Neighbor visibility

During alignment with seven robots, Bluebots detected approximately 2.2 out of 6 possible neighbors on average with expected loss due to occlusion, occasional misidentified robots, and information loss during conservative parsing. In simulations with seven robots, we assigned a random subset of visible neighbors to each robot in each iteration, and compared the effect of the subset size on alignment. In this case, there was no parsing and no noise on the LED positions. The convergence of robot headings was accelerated if more than 2.2 neighbors were visible; conversely, it slowed down and almost doubled if one single neighbor was visible only (Fig. 6.6 B). The final circular standard deviation σ_φ was unaffected by the number of visible neighbors. As with schooling fish that exhibit group-level coordination, a limited[48,92] and noisy[126] count of neighbors was sufficient to achieve alignment consensus in our experiments.

6.3.3 Cognition speed

Faster cognition — achievable with more powerful hardware or more performant algorithms — allows for more frequent sensing, which generally results in more accurate state estimation and control. Bluebots' cognition speed is most severely affected by the complexity of image processing. Average sensing iteration frequencies of 3.42 Hz were measured during alignment only, and dropped to 0.96 Hz when predator detection was added. Simulations with seven robots showed that the quality of alignment deteriorates for frequencies below 1 Hz (Fig. 6.6 C). This prevented us from using a previously developed[127] and bioinspired[125] flashing mechanism to warn fellow Bluebots against the predator; while effective for predator alert, flash detection slowed down and damaged alignment significantly.

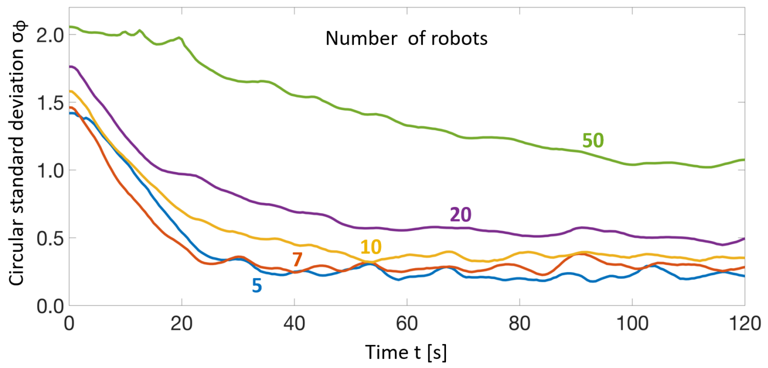


Figure 6.8: Scalability of alignment. Convergence of headings takes longer and settles at higher circular standard deviations σ_φ with an increasing number of robots. All data points were averaged across $N = 10$ simulation runs.

6.3.4 Number of robots

We simulated alignment with 5–50 robots to assess whether our implementation scales well to larger collectives. Perception and cognition of these simulated robots matched the Bluebots. For headings, the convergence time grew and the final circular standard deviation σ_φ deteriorated with the number of robots (Fig. 6.8). This confirms our expectation that with more robots, occlusions become more frequent and inferring individual headings more challenging. As a result, small alignment errors add up from one end of the collective to another.

6.3.5 Escape angle

The fountain maneuver takes up more space with larger escape angles (Fig. 6.9 A); an angle of $|60\,°|$ was chosen for physical experiments due to space constraints in our testing environment. In comparison, the same maneuver with $\Theta = |150\,°|$ — the rear limit of the visual field of gadoid fish[117] — requires nearly ten times as much space (compare Figures 6.9 B and 6.3 B). Simulated robots were able to reach and hold larger escape angles more easily.

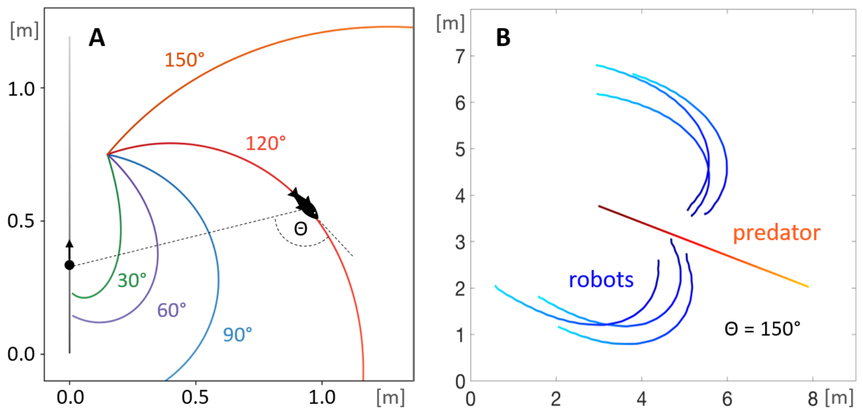


Figure 6.9: Escape angles. (A) The larger the escape angle Θ, the longer are the escape trajectories (colored). The predator moves from bottom to top and is indicated by the vertical line on the left (greyscale). **(B)** A fountain maneuver with six simulated robots and an escape angle of $|150\,^\circ|$.

7
Toward Open Water Swarming

Coming soon, because major parts of this chapter yet have to be published in a journal/conference paper.

8 Conclusion

Self-organized collective behaviors without external assistance are hard to engineer, especially in 3D space which puts high requirements on autonomous perception and maneuverability. To complicate matters further, many existing above-ground sensing and explicit communication methods are unavailable underwater. As a consequence, capable underwater robot collectives that demonstrate dynamic decentralized coordination are missing in the literature. By contrast, schooling fish that use vision as their dominant sensory modality[53] display mesmerizing underwater coordination at scale, and inspired us to design a novel underwater robot swarm that can self-organize with local implicit vision-based coordination only.

8.1 Achievements

Blueswarm marks a major milestone in the experimental investigation of self-organized collective behaviors in 3D space. This work on the design and strong experimental validation of autonomous underwater robots with 3D perception and 3D locomotion represents the first example of fully decentralized 3D underwater coordination in the field of robotics.

The miniature underwater robots use only local and implicit coordination mediated through blue light. The Bluebot cameras paired with the LEDs result in a minimalist but versatile visual system that is used to rapidly infer the positions of neighboring robots but also enables synchronization and signaling. The fish-inspired body design with multiple fins offers a high degree of maneuverability at a low footprint to allow for fast response to neighbor actions or precise maneuvers in more complex environments.

Blueswarm shows that a range of complex and dynamic 3D collective behaviors — firefly-inspired synchrony, molecular aggregation-dispersion, dynamic circle formation, search-capture, evasive fountain maneuvers — can be achieved by sensing minimal, noisy impressions of neighbors, without any centralized intervention. To the best of my knowledge, this is the first significant demonstration of self-organized 3D collective coordination underwater, without any external sensing and control assistance.

The results illustrate the power of decentralized autonomy, and provide new insights into the role of implicit coordination as a key mechanism for achieving coordination in challenging environments. This work provides for the first time strong experimental evidence for multiple complex and robust 3D underwater collective behaviors, achievable entirely through self-organized coordination. It advances the potential for future underwater robots that display collective capabilities on par with fish schools for applications such as environmental monitoring in coral reefs and large-scale search operations. In addition, Bluebots are also well-suited as an experimental test bed for investigating natural collective behaviors and biomimicry, for example studying energy savings for different formations in schooling fish, or collective predator behaviors exhibited by wolf packs or dolphins[1,7,27].

8.2 Future Work

While the Blueswarm platform is currently limited to a laboratory setting, the introduction of progressively more powerful and smaller microcomputers, underwater cameras, and new actuators will enable such robots in more complex natural environments[36,39]. Future camera-equipped miniature underwater robots may additionally record videos and take images, for instance to inspect coral reefs or man-made underwater structures. Instead of LEDs or simple colors, such robots may use deep-learning generated visual patterns (i.e., “artificial” schooling marks) to recognize neighbor pose[67] and navigate underwater.

Insights from real-robot underwater experiments will contribute toward future unsupervised versions of coordinated maneuvers of unmanned vehicles, making it possible to combine multiple robot modalities (aerial, water-surface, and underwater) to achieve scalable and robust realizations of ventures such as collective search of missing aircrafts, vessels, and persons in water[16]. To realize these ventures, I prioritize future work in the following three key areas:

1. **Collective mapping, sampling, and localization in unknown environments**
   Vast parts of our oceans are still relatively unexplored. Other parts are endangered for a variety of reasons including overfishing, pollution, and the effects of climate change. Blueswarm can help in stepping up conservation efforts to protect important and fragile underwater habitats

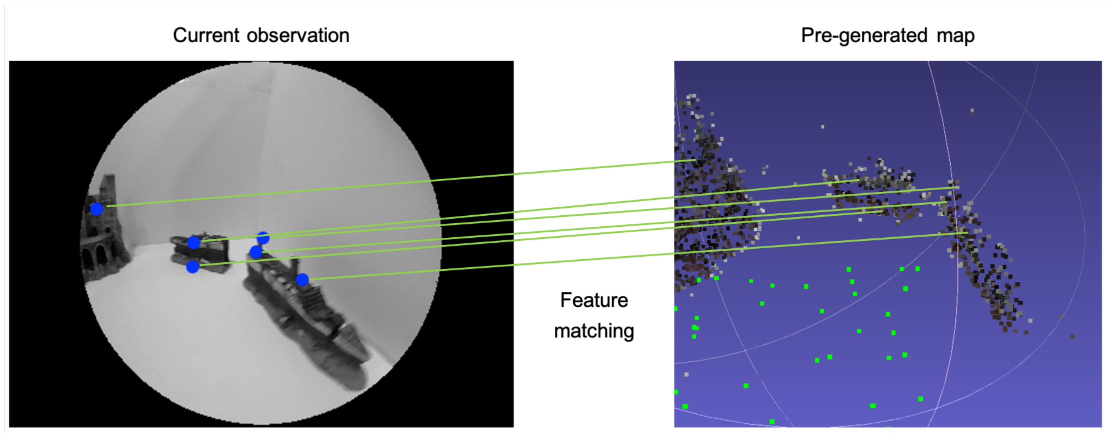


Figure 8.1: Navigating unknown territory. A robot extracts key features from a current image (left) and matches them against a pre-generated map (right) in order to localize itself.

like coral reefs. Small, agile, and innocuous robots inspired by Bluebot can inform such efforts by gathering relevant data to better understand the state of an ecosystem.

The envisioned approach will make use of existing methods such as structure from motion (SfM)[128–130] and visual odometry[131,132], and gear them toward running fast on microcomputers onboard of autonomous underwater robots. In order to map new environments, a swarm of robots will first take a large amount of images. With SfM applied to this image set, the environment can be reconstructed offline and represented as a 3D map even if the locations where the images were taken are unknown[133]. Key features will then be extracted from the images and linked to their corresponding locations in the 3D map. Back in the water, a robot can now extract features from fresh images and match them against the pre-generated feature map to localize itself. We already demonstrated successful feature matching in preliminary experiments in air (Fig. 8.1).

2. **Complex behaviors with heterogeneous swarms**

Many open water tasks will be of high complexity and require collectives that master a variety of challenges. An interesting approach that was successfully demonstrated above-ground[33] are heterogeneous systems with several complementary robots that fulfill specific roles and are specialized in particular tasks. In addition, such robots might sporadically communicate explicitly to exchange information coming from different sensors. Bluebots already carry an onboard radio with a communication range of at least 20 body lengths underwater that can be used to develop explicit communication.

3. **Blueswarm as an open-source platform**

   I designed Bluebots to be easy to use, not only by myself, but also by other researchers. Collaborations on milling, search, and evasive maneuvers showed that postdocs and students can operate Bluebots without lengthy introductions. In order to launch Bluebot as an open-source platform — our group did that previously with the Kilobot[10] — the hardware design and software starter package will have to be further optimized to be easy to fabricate and handle, durable, and reliable. Such open-source platform will give researchers a much-needed tool to systematically investigate self-organized 3D collective behaviors in the laboratory.

In addition to studying collective behaviors, the Blueswarm platform lends itself for biomimicry and the development of soft propulsors. In a first collaboration, I looked at nature to refine a Bluebot variant — the Finbot — such that it accurately mimics three key characteristics of fish swimming: i) linear speed-frequency relationships, ii) U-shaped CoT, and iii) reverse Kármán wakes[75] (Appendix B). Finbot then served as a fish-surrogate to experimentally validate a novel hypothesis on the energy savings of in-line schooling[57]. Originating from bioinspired design ideas, biomimetic robots can in turn be useful tools (e.g., in evolutionary and comparative biology) to explore and reflect on animal physiology and behavior.

In a second collaboration, I demonstrated soft, muscle-like actuators on a Bluebot variant[134] (Appendix C). Thanks to recent advances in materials and fabrication processes, soft propulsion can be combined with soft bodies to create entirely soft and autonomous robots[135]. A future challenge will be to reduce the complexity and increase the durability of soft designs, such that many small robots can be manufactured reliably.

8.3 The Importance of Understanding Collective Behaviors at Large

Collective behaviors emerge not only in fish schools, bird flocks, or ant colonies[1], but also in human social dynamics ranging from information spread on social networks[136] to trading activities in market places[137]. Collectives can solve problems that may be insurmountable for any individual member, capitalizing on shared information and coordinated actions[2,3,138]. In human societies, we all contribute to and benefit from collective welfare by organizing political and economic institutions, tackling global challenges such as climate change or disease spread, and changing social norms for the better[139–141]. Understanding collective behaviors allows us to create and maintain stable social systems, and engineer novel solutions for hitherto complex and vast problems.

The research field of collective robotics draws inspiration from natural systems and aims to engineer their attractive properties into de novo artificial systems[8]. Such work has far-reaching impacts on human society. The reliability of large-scale infrastructure systems (e.g., traffic, finance, or electric power) is of growing importance in a globally connected as well as increasingly automated and accelerated world. Failures of such systems may critically affect millions of people. Decentralized and self-organized designs can improve the robustness of largely autonomous systems while also increasing their scalability; such systems become more redundant and less vulnerable to human error and malicious attacks. In addition, self-organized systems may enable new ventures, which are currently unfeasible due to impaired real-time communication with, or the safety of, human operators. For instance, teams of robots could provide shelter and harvest energy on distant planets long before humans arrive, or speed up search and rescue missions in hazardous areas here on earth. More immediately, the insights from research on collective robotics are expected to have a direct impact on the organization and coordination of large automated systems, spanning from automated warehouses and self-driving cars to high-frequency trading and smart communication and power grids.

A

Main Supplement

A.1 Rapid LED Blob Detection

A Bluebot detects the presence of fellow robots by locating their LEDs on its vision system. The Bluebot's camera settings (e.g., brightness and contrast) are set such that under laboratory experiment conditions, everything in the images appears dark except for the LEDs of other Bluebots, which are the brightest light sources in the environment. These settings facilitate the process of detecting LEDs

Algorithm 6: Rapid LED blob detection (RapidLED)

```
1 Input: a binary image in $m \times n$ matrix format with only black and white pixels
2 Store all white pixels in a list, recording their $mn$ coordinates.
3 Sort the list of white pixels by $m$ coordinate.
4 Split this sorted list into sub-lists wherever there is a gap larger than $L$ between
    successive $m$ coordinates.
5 for list in sub-lists do
6     Sort the pixels by $n$ coordinate.
7     Split this sorted list into sub-lists wherever there is a gap larger than $L$ between
        successive $n$ coordinates.
8 Each sub-list now corresponds to an LED blob. Calculate its centroid and add it to
    the list of blob centroids.
9 return the list of blob centroids
```

from other Bluebots, because they appear as more or less circular blobs in the image. The process of detecting contiguous sets of pixels in an image is often referred to as blob detection; in order to achieve fast processing times, I designed a simple custom algorithm (Alg. 6).

The blob detection process starts by thresholding the image to obtain a binary image with only black and white pixels. The threshold is set heuristically to achieve a balance between sensing range and sensitivity to false-positive errors. I define blobs as continuous regions of white pixels in a binary image. The notion of continuity depends on how neighbor pixels are defined. Common definitions include 4-connectivity and 8-connectivity. I use the definition that two pixels are connected if their Moore distance is smaller than or equal to two. This allows for gaps of size one pixel within a blob.

Many algorithms exist for blob detection. A simple but effective method is to search for blobs via a depth- or breadth-first search (DFS, BFS) starting from every unvisited white pixel. This method will correctly identify all blobs in every case, and require O(3mn) steps, where mn is the number of pixels in the image. However, the Raspberry Pi Zero W's limited computational power renders this method too slow to be useful for real-time control on the Bluebots, even at the reduced image resolution of 192×256 pixels.

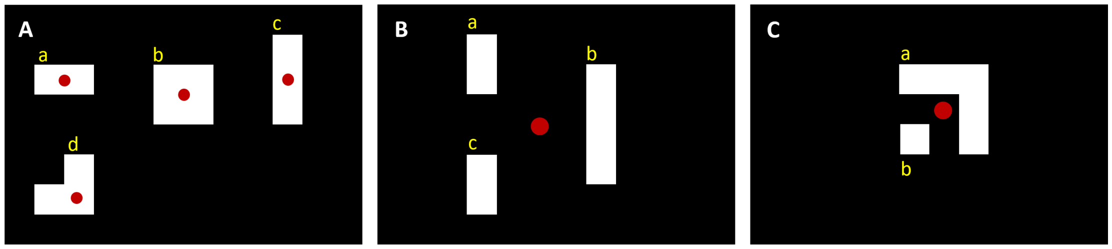


Figure A.1: Correctness of rapid LED blob detection for three different cases. (A) Non-pathological: All four blobs can be detected and their individual red centroids will be returned. **(B)** Solvable pathological: Blobs a and c could be separated with a second pass of LED blob detection. **(C)** Non-solvable pathological: Blobs a and b will always be mistakenly returned as a single blob. This case, however, is highly unlikely for circular LED blobs.

I therefore opt for a trade-off with an algorithm that runs an order of magnitude faster than DFS and BFS, but is not guaranteed to identify all blobs in every case. In practice, successful experiments demonstrate that this detection system is sufficient. The algorithm is based on identifying continuous pixels first in one direction and then in the other, and inherently considers pixels to be connected or not according to their Moore distance. I introduce a parameter L that defines the maximum Moore distance for two pixels to be considered connected. Setting $L = 1$ reduces to 8-connectivity and increasing L allows for some black pixels within white blobs.

A.1.1 Correctness

In the following analyses, I assume $L = 1$ for simplicity. Figure A.1 A shows a non-pathological case for the blob detection algorithm (Alg. 6). In step 4, blob d is isolated, but blobs a, b, and c are grouped together because their vertical m coordinates overlap. Blobs a, b, and c are then separated in step 7, because their horizontal n coordinates are not continuous.

Figure A.1 B shows the first pathological case for the blob detection algorithm. In the vertical m coordinate pass of step 4, all three blobs are added to the same sub-list because their m coordinates are continuous. In step 7, blob b is isolated, but blobs a and c remain lumped together because their horizontal n coordinates are continuous. This pathological case can be solved by recursively calling

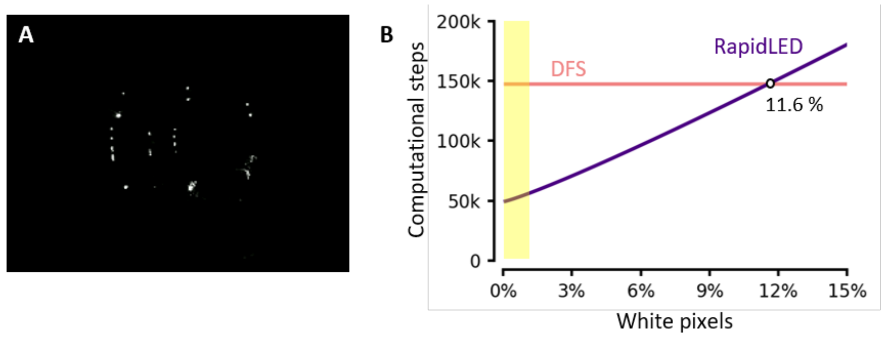


Figure A.2: Time complexity of rapid LED blob detection at an image resolution of 192×256. **(A)** A typical image taken with a Bluebot during aggregation with seven robots, which contains $0.35\ \%$ white pixels. **(B)** Rapid LED blob detection (RapidLED) is significantly faster than depth-first search for low ratios of white LED blob pixels in a given image. The ratio of white pixels in Bluebot images typically falls into the yellow left region. RapidLED would process example image (A) with approximately 2.9 fewer computational steps than DFS.

the algorithm on all sub-lists after step 7: once blob *b* is removed, blobs *a* and *c* are distinguished by another pass of the algorithm. In general, the algorithm would have to be called recursively whenever any splits are made in step 7. I opted not to use this recursion in my implementation, because it slowed down the algorithm significantly and did not provide a substantial benefit in practice. Even if recursion is used, there are still some pathological cases that cannot be resolved by this algorithm, as shown in Figure A.1 C. These two blobs will always be identified as one because they are continuous in both *m* and *n* coordinates. Given the circular nature of the LEDs, pathological cases were a non-issue in experiments. Conversely, there are many more sources of error and noise, e.g., reflections and occlusions, that play a more significant role in reducing the accuracy of neighbor detection.

A.1.2 Time Complexity

Depth-first search (DFS) visits every node and edge in a graph exactly once. Here, an image has mn nodes (pixels), and $(m-1)n + m(n-1)$ edges (links between adjacent pixels). The resulting time complexity of DFS is O($3mn$), regardless of how many white LED blob pixels there are in any given image.

Rapid LED blob detection (RapidLED) also visits all mn pixels once to identify white pixels above a heuristically tuned threshold (step 2 in Alg. 6). Let's denote the ratio of white pixels in an image as w (e.g., an image with $w = 0.005$ contains $0.5\,\%$ white pixels). For each of steps 3 and 6 in Algorithm 6, RapidLED has to sort at most wmn pixels in total, and sorting takes time $\mathrm{O}(wmn \log(wmn))$. The resulting time complexity of RapidLED is $\mathrm{O}(mn + 2wmn \log(wmn))$, and depends on the ratio of white pixels in any given image.

Figure A.2 B shows the number of computational steps required for different ratios of white pixels for a Bluebot image of size 192×256. As long as there are fewer than $11.6\,\%$ white pixels, RapidLED requires fewer steps than DFS. Typical Bluebot images have less than $1\,\%$ white pixels (yellow left region), and are processed significantly more efficiently with RapidLED (e.g., 2.9 times fewer computational steps for a typical image as shown in Figure A.2 A). The mean and standard deviation of white pixels taken from 453 images over the course of a representative aggregation-dispersion experiment were $\mu = 0.27\,\%$ and $\sigma = 0.26\,\%$, respectively. At most, there were $1.86\,\%$ white pixels in an image showing a very nearby robot, and only 14 images out of 453 had more than $1\,\%$ white pixels.

A.2 3D Tracking of Laboratory Blueswarm Experiments

A custom tracking system allows for each Bluebot's individual 3D trajectory to be fully tracked in experiments with multiple Bluebots. The tracking system works by combining video data from an overhead camera with depth measurements from each Bluebot's pressure sensor. The overhead camera is a Canon Rebel T5i equipped with a Sigma 35 mm f/1.4 DG HSM Art Lens. This lens provides a sufficiently wide angle to cover the entire 1.78 m $\times$ 1.78 m tank area with negligible distortion. The Bluebot's pressure sensor (TE connectivity MS5803-02BA) provides sub-millimeter precision in depth measurements. The tracking software is written in `MATLAB`.

A.2.1 Experimental Protocol

After an experiment, each Bluebot's pressure sensor log file is downloaded to a computer via a secure shell (SSH). The video from the overhead camera is downloaded to the same computer via a USB connection.

It is necessary to synchronize the time stamps from the pressure sensor log files with the time of the video. This is achieved by the following protocol:

a) At the start of an experiment, the Bluebots are placed at the surface of the tank in the desired configuration and are put into a waiting state, in which they continuously sample the reading of their photodiode (VTP1112H) and wait for a light-to-dark transition in the environment.

b) Recording on the overhead camera is started.

c) The laboratory lights are turned off, which simultaneously signals all Bluebots to transition to the initialization state and start periodically logging their pressure sensor data with time stamps beginning at zero.

d) In post-processing, the time in the video at which the light-to-dark transition occurs is used as the synchronization point for the pressure sensor data.

Apart from temporal synchronization, it is critical to spatially match each pressure sensor log file with its originating robot in the overhead video. This is achieved by the following protocol:

a) After a short period (~1 s) from the light-to-dark transition, each Bluebot flashes its LEDs a unique number of times, as given by its identifier (ID). Bluebots with lower IDs wait until the Bluebot with the highest ID is finished flashing before transitioning to the experiment state. This way, all Bluebots start executing the experiment code simultaneously.

b) In post-processing, the tracking software automatically detects the number of flashes for each Bluebot, and this information is used to assign pressure sensor log files to the robots in the starting frame.

A.2.2 Multi-robot Tracking

Tracking proceeds in five phases: i) initialization; ii) 2D position identification; iii) 2D trajectory isolation; iv) 3D data fusion; v) post-processing. In the **initialization** phase, the pressure sensor data is temporally and spatially synchronized with the video data as explained in the previous section. The video is trimmed to start when the Bluebots begin executing the experiment, which occurs shortly after the Bluebot with the highest ID has finished flashing its LEDs.

In the **2D position identification** phase, the positions of all N Bluebots are identified in each frame of interest (disregarding identities). Tracking is performed every 0.2 s, or every six frames since the video is recorded at 30 frames per second (FPS). In each frame of interest (henceforth simply "frame"), the tracking software attempts to perform position identification automatically, and asks the user for input if it cannot perform this task unambiguously. The automated attempt proceeds as follows and is repeated for each frame until the end of the video:

a) The image is converted to binary according to a user-specified threshold value. This value can be heuristically set to improve the chances of successful automatic identification in a given video. However, decreasing this value too much may result in genuine Bluebots being missed (false negative detections), while increasing it too much may result in non-existent Bluebots being identified (false positive detections).

b) White regions smaller than a certain number of pixels are removed using `MATLAB`'s `bwareaopen` function. The number of pixels can be adjusted by the user to optimize performance.

c) `MATLAB`'s `regionprops` function is used to identify connected white regions (blobs), which represent Bluebots.

d) If more than *N* blobs are identified, a correction attempt is made by performing a minimum-distance bipartite matching with the positions already identified in the previous frame. The *N* blobs in the current frame contained in the matching are proposed as the current Bluebot positions.

e) If fewer than *N* blobs are identified, no correction attempt is made, and the result is presented to the user for correction.

f) Regardless of whether or not a correction has been attempted or not, the result is shown to the user for correction/verification by plotting each identified position on the original (i.e., non-binary) frame. If necessary, the user can make adjustments by clicking on existing positions to remove them and clicking anywhere else in the frame to add a new position. When the user is satisfied with the result, they press `Enter` to proceed to the next frame. The tracking software will not proceed to the next frame until *exactly N* positions have been identified. Moreover, the tracking software performs a minimum-distance bipartite matching between the positions in the current frame and the previous frame and warns the user if the sum of distances is larger than a threshold value. This value is set according to the maximum distance that a Bluebot can move between successive frames.

We now have a set of *N* Bluebot positions per frame, but these positions are not correlated between frames; i.e., for any specific Bluebot in some frame, it is not known which position corresponds that Bluebot in the following frame(s). The **2D trajectory isolation** phase addresses this issue, which is important for two reasons:

i) Extracting the trajectories of individual Bluebots, rather than simply uncorrelated positions, is useful data that can help in analyzing experimental outcomes.

ii) Since pressure sensor data is used for extracting the vertical dimension of the Bluebot's position, it is critical to know the identity of each Bluebot throughout the video. Note that the 2D positions obtained from tracking the video are *not* a faithful representation of the Bluebot positions, even if one is only interested in the horizontal plane and not in the vertical dimension. This is because every point in the 2D frame corresponds to a *non-vertical* set of possible positions in 3D space. The depth must be known for resolving this set of possible positions to a single one, as shown in Figure A.3.

Trajectory isolation is performed by traversing the entire video for each robot, one by one. Given the position of the Bluebot in Frame k (starting at Frame 1), a new position is searched for in Frame $k+1$ within a fixed radius of the previous position. If exactly one new position is found, this becomes the current position and the tracking software automatically proceeds to the next frame. If no new position is found, or multiple candidate new position are found, the user is shown all the available new positions and asked to select the correct one. When a new position is identified (whether automatically or by the user), it is marked as "used", and may not be assigned to any subsequent Bluebot. This step makes the trajectory isolation increasingly automated for successive Bluebots, because there is less potential for conflicting new positions.

The search radius can be adjusted by the user but should roughly correspond to the maximum distance that a Bluebot can be expected to move between successive frames. Too small a radius results in a higher risk of no new position being identified, while too large a value will result in a higher risk of multiple candidate new positions being identified.

As explained above, the 2D trajectories obtained so far are not a faithful representation of the Bluebots' movements, even if one is interested only in the horizontal plane. In the **3D data fusion** phase,

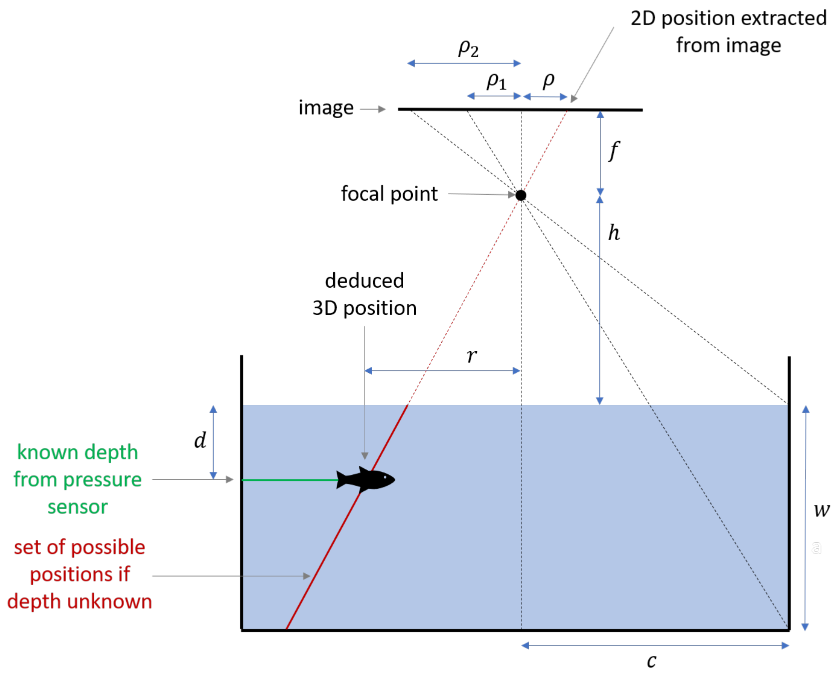

2D position extracted from image
ρ_2
ρ_1
ρ
image
f
focal point
h
deduced 3D position
r
d
known depth from pressure sensor
w
set of possible positions if depth unknown
c

Figure A.3: 3D tracking — side view. Schematic side view of the tank showing the overhead camera and the salient parameters. Note that all the dimensions are shown as if they are projected onto a flat surface; in reality, they are 3-dimensional (cf. Fig. A.4).

these 2D trajectories are fused with the Bluebots' pressure sensor data to obtain 3D trajectories. Recall that spatial and temporal synchronization between the video and pressure sensor data was achieved in phase 1. As such, for every $\hat{p}_i^k$ (2D position of Bluebot i in Frame k), we now have a corresponding d_i^k (depth of Bluebot i in Frame k). The fusion proceeds Bluebot by Bluebot and frame by frame. In the following, we assume a general pair $\hat{p}_i^k$ and d_i^k and drop the sub- and superscripts to simplify notation. Note that $\hat{p} = (m, n)$ is referenced from the center of the image with the conventional axis directions, while d is referenced from the water surface pointing downwards (see Fig. A.3).

Given $\hat{p} = (m, n)$ and d, we wish to find the 3D position of the Bluebot in the tank. The origin is taken to be the center of the tank at the water surface, with the z-axis pointing downwards. Therefore, the Bluebot's 3D position takes the form $p = (x, y, d)$, and the problem reduces to finding x and y in

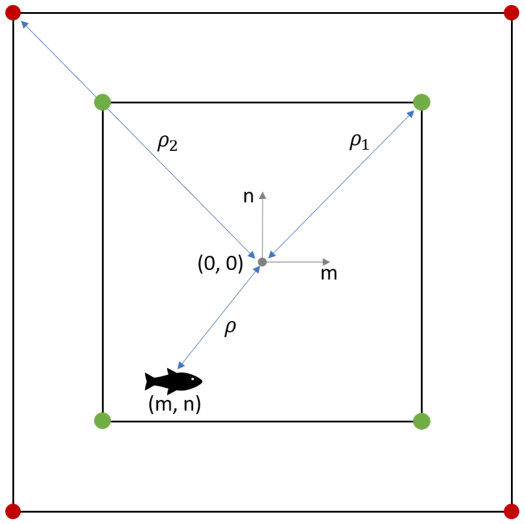


Figure A.4: 3D tracking — overhead view. Schematic of the tank view as seen from the overhead camera. The square with red corners corresponds to the water surface, while the square with green corners corresponds to the bottom of the tank. The image coordinate system has its origin at the center of the tank with the m and n axes pointing right and upwards, respectively. Distances in this coordinate system are measured in pixels. The distance from the Bluebot to the origin in this overhead image is denoted by ρ.

terms of m and n. To solve this problem, we must make a number of measurements in the image and in the real world.

Before data fusion begins, the user is presented with a frame from the video and is asked to click on eight points; namely, the four corners of water surface and the four corners of the bottom of the tank. These points are shown in Figure A.4 in red and green, respectively. From these points, the tracking software automatically computes the average image distance (in pixels) from the center of the tank to the corners of the water surface (ρ_1) and to the corners of the bottom of the tank (ρ_2).

As shown in Figure A.3, let c represent the horizontal cross-sectional distance from the center of the tank to one of its corners and w the water depth. These values are known from direct measurement.

Furthermore, let f represent the focal distance and h the distance from the focal point to the water surface. These values are harder to measure directly since the exact focal point is unknown, so we calculate them from the above known values. From Figure A.3, observe from similarity of triangles that:

$$\frac{f}{\rho_1} = \frac{h+w}{c} \qquad \text{and} \qquad \frac{f}{\rho_2} = \frac{h}{c},$$

from which we compute:

$$f = \frac{w\rho_1\rho_2}{\rho_2 - \rho_1}$$
$$h = \frac{wc\rho_1}{\rho_2 - \rho_1}.$$

Given a Bluebot's 2D position $p = (m, n)$ and depth d, we define the 2D distance of the Bluebot from the center of the image as:

$$\rho = \sqrt{m^2 + n^2},$$

(see Fig. A.4) and from observation of Figure A.3 we can calculate the horizontal cross-sectional distance of the Bluebot from the center of the tank, r:

$$r = \frac{\rho(h+d)}{f}.$$

We now scale m and n by a factor $r\rho$ to obtain x and y:

$$x = \frac{r}{p}m$$
$$y = \frac{r}{p}n.$$

and we have the final, 3D position of the Bluebot in the tank:

$$p = (x, y, d).$$

The **post-processing** phase consists of a simple exponential moving average filter that reduces noise in the Bluebot positions due to tracking errors (e.g., inaccurate user clicks, noise on the pressure sensor). Denoting the unfiltered position of Bluebot i at Frame k as p_i^k, we obtain the filtered version $\tilde{p}_i^k$ by applying:

$$\begin{aligned} \tilde{p}_i^1 &= p_i^1 \\ \tilde{p}_i^k &= \alpha \tilde{p}_i^{k-1} + (1-\alpha) p_i^k \qquad k = 2, \cdots, K \end{aligned}$$

where K is the number of frames and α is the filter parameter that can be set by the user in the range $[0, 1)$. Smaller values of α place more emphasis on the measured values (i.e., less filtering), while higher values place more emphasis on past measurements. As a guideline, α should be set such that no spikes in the filtered data occur that are beyond the dynamic capabilities of the robot (due to inertia, hydrodynamic drag, and actuator placement and power limitations). We use a value of $\alpha = 0.7$ in all cases.

A.3 Dynamic Circle Formation and Milling

The dynamic circle formation and milling algorithm places strikingly few requirements on the part of the robots. It is only assumed that:

- Each robot has a line-of-sight or a field-of-view binary sensor that can detect the presence of other robots. The sensor returns 0 if it does not detect other robots and 1 if it does. The robot does not need to know the number or exact positions of robots within the field-of-view, thus placing very little load on the image processing and working with minimal information.

- The robots are capable of moving in a planar circle of a fixed radius at a fixed speed in either direction (i.e., clockwise or counterclockwise). For dynamic cylinder formation, we also assume that robots can independently control their depth in the tank while moving clockwise or counterclockwise in the plane.

The algorithm itself is also extremely simple. Each robot follows the following protocol:

- If the sensor returns 0, rotate clockwise along a circle of radius r_0 with a fixed speed.
- If the sensor returns 1, rotate counterclockwise along a circle of radius r_1 with a fixed speed.

Previous work in simulated ground robots suggested that this algorithm generates an emergent rotating circle formation, where ideal robots (with no sensory or locomotion noise) are equally distributed along the circle and the size of the circle increases with the number of the robots.

Here we prove that the original algorithm for simulated ground robots with line-of-sight sensors has a unique stable state with a circle radius R, whose formula is derived, and we also provide bounds on necessary parameters to achieve a stable configuration. We then extend this proof to the Bluebots, which use an angular field-of-view sensor.

In the following analysis, we make several simplifications:

- The sensor-actuator loop is fast enough that it can be considered continuous.
- There is no noise on the robots' sensor readings and locomotion.
- The robots have no inertia and can change directions (i.e., from rotating clockwise to counterclockwise or vice versa) instantaneously. This simplification does not hold strongly for the Bluebots, but it makes the analysis tractable, and simulation and physical results show that milling with the Bluebots is indeed possible.

- The robots are on the same horizontal plane. This simplification leads to no loss of generality as long as the robots' sensors are implemented in such a way that they are invariant to other robots' altitudes/depths.

A.3.1 Overview of Theoretical Results

In this section we introduce the main parameters and present a short overview of the theorems and insights that will follow in more detail in the subsequent sections.

We are given a number of robots N and assume that the robots have an approximately circular body of radius ρ (for the purposes of being sensed by other robots). We begin by considering line-of-sight sensors, which will then easily extend to the more general case of field-of-view sensors.

Given N and ρ, we show that there is a unique stable milling configuration with radius:

$$R = \frac{\rho}{1 - \cos\frac{2\pi}{N}},$$

provided certain conditions are satisfied, namely:

- The robots' turning radius when the sensor reports no other robots, r_0, must be smaller than or equal to the milling radius: $r_0 \leq R$. Milling is smoother if the two values are similar, i.e., if r_0 is not much smaller than R.

- The robots' sensors must have sufficient range, and a lower-bound on this sensing range is theoretically calculated.

We then generalize the above results to the case of a field-of-view sensor. The sensor's half field of view is denoted by α. The line-of-sight sensor results are all recoverable from the field-of-view sensor results by setting $\alpha = 0$.

$$R' = \frac{\rho}{\cos\alpha - \cos\left(\frac{2\pi}{N} - \alpha\right)}$$

For a field-of-view sensor, the milling radius R' now depends on N, ρ, and α. The same conditions as for line-of-sight sensors still hold, but the lower bound on the required sensing range now also depends on α. Additionally, there is also a theoretically calculated upper bound on α to achieve a finite steady-state milling radius. For values of α larger than this upper bound, the milling radius grows until the robots can no longer see each other (or indefinitely if the sensing range is unlimited).

In the following subsections, we provide proofs for each of the claims made above.

A.3.2 Milling with Line-of-Sight Sensors

Steady-State Circle Formation and Milling Radius

The first theorem shows that a milling formation can be maintained if robots start from an idealized configuration and an optimal r_0 is chosen.

Theorem A.3.1. *Given a free choice of r_0, N robots of radius ρ can be placed on the vertices of a regular N-sided polygon such that upon moving, no robot detects another robot, and the polygon formation is maintained indefinitely. The radius of this polygon is:*

$$R = \frac{\rho}{1 - \cos\frac{2\pi}{N}},$$

and we choose $r_0 = R$.

Proof. Consider the configuration in Figure A.5 A, where N robots are placed on the vertices of a regular N-sided polygon and each robot is oriented such that its sensor is tangential to the subsequent robot. In this configuration, no robot detects another robot. Now, if the robots rotate clockwise with

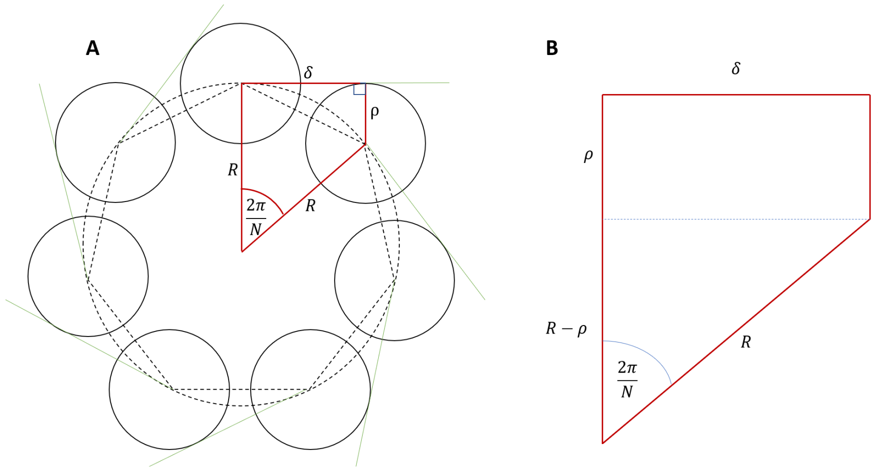


Figure A.5: Steady-state milling radius. **(A)** Seven robots arranged in a regular polygonal formation with line-of-sight sensors tangential to each other. **(B)** Expanded view of the red quadrilateral.

the same radius as the radius of the polygon (i.e., $r_0 = R$, as shown by the dotted circle), we have the invariant property that no robot will ever see another robot as they move. It remains to calculate the radius of the polygon, R.

Figure A.5 B shows an expanded view of the red quadrilateral in Figure A.5 A. Since the polygon is regular, the angle spanning from the center of the polygon to any two adjacent robots is given by $\Theta = 2\pi/N$. Observe from the triangle in Figure A.5 A that:

$$\cos\frac{2\pi}{N} = \frac{R-\rho}{R},$$

which can be rearranged to give:

$$R = \frac{\rho}{1-\cos\frac{2\pi}{N}}$$

■

Unique Steady-State Configuration

We do not know of a method to prove or guarantee the convergence to the milling formation starting from an arbitrary initial configuration of robots. However, assuming an unlimited sensing range and $r_0 = R$, it is easy to show that the milling formation of Figure A.5 A is the *only* stable formation. By stable, we mean that no robot will ever see another robot, and therefore the formation will never change (except for the robots rotating indefinitely).

Theorem A.3.2. *The regular polygon formation of Theorem A.3.1 is the unique stable formation if the sensing range is unlimited and $r_0 = R$.*

Proof. In Figure A.6, robot A rotates along a circle of radius $r_0 = R$. For contradiction, assume a formation not identical to the one of Theorem A.3.1. Then, at least one other robot, say robot B, does not have its center lying on the circle traversed by robot A. Therefore, robot B traverses a circle different to the one traversed by robot A. It is clear that as both robots move along their respective circles, each robot's sensor must at some point intersect the other robot. Thus, the configuration is not stable. ■

Previous work's simulation results and our own MATLAB simulations suggest that the stable milling formation in Figure A.5 A with the corresponding milling radius is achieved from any initial configuration with $r_0 = R$, even though a proof of convergence does not yet exist. Furthermore, these simulation results suggest that this algorithm works for $r_0 \leq R$ and with sufficient (not just unlimited) sensing range. These two conditions are expanded on in the following two sections.

Effects of r_0 and r_1 on Milling Formation

Theorems A.3.1 and A.3.2 assume that r_0, the turning radius of the robots when their sensor reads 0, is equal to the milling radius R. Under this condition, no robot in the formation will ever see another

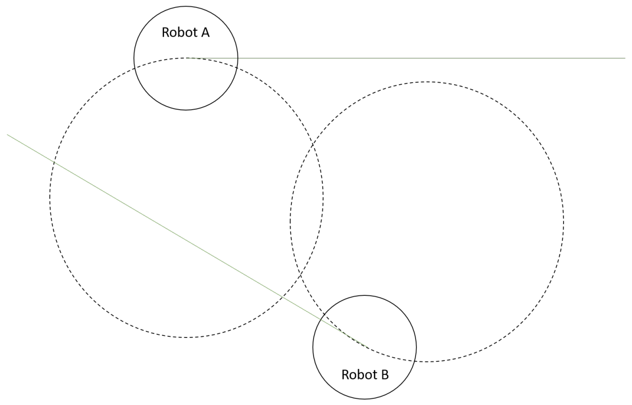


Figure A.6: Unique steady-state circle. Example configuration of two robots to show that the regular polygon formation of Theorem A.3.1 is the only stable formation.

robot, and the formation is maintained indefinitely. In practice, this is achievable if the number of robots is known and the turning radius is easily adjustable, but it is not possible if the turning radius is fixed and the number of robots is unknown or variable.

To examine the effect on r_0 on maintaining the formation in Figure A.5 A, consider one of the robots and consider its position after a short amount of time Δt. If $r_0 = R$, then the robot's center will still be on the dotted circle, only shifted slightly clockwise. Moreover, its sensor will still be tangential to the next robot, and therefore its sensor will still read 0 and it will continue rotating clockwise.

Assume now that $r_0 < R$. The robot's center after a short period of time Δt will now lie slightly inside the dotted circle, and as a result, its sensor will now intersect with the next robot, returning a reading of 1. Consequently, the robot will now rotate counterclockwise with radius r_1. This creates a negative feedback mechanism that, on average, maintains the formation of Figure A.5 A. Naturally, the closer r_0 is to R the more smoothly the formation can be maintained, but simulation results show that r_0 can be reduced by around an order of magnitude before any noticeable deviations start to occur, and by around two orders of magnitude before the overall formation is seriously compromised. Simulation results also show that setting $r_1 = r_0$ in all cases is a reasonable choice.

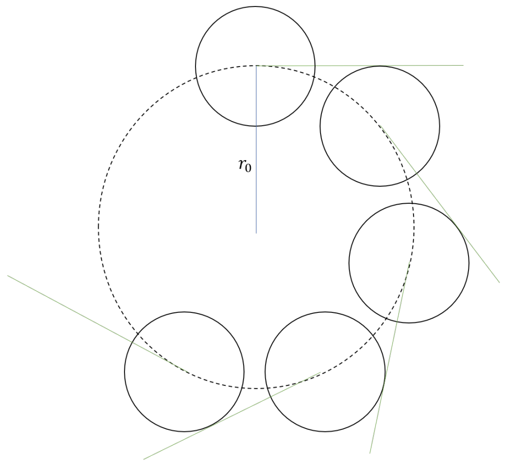


Figure A.7: A pathological initial circle configuration. In the case of $r_0 > R$, the five robots, which do not see each other, will continue rotating along a circle of radius r_0 and never converge to a regular pentagon.

Assume that $r_0 > R$. In this case, we can guarantee that it is not possible to achieve the formation of Figure A.5 A starting from any initial configuration (but it might be possible starting from some initial configurations). We state this as a theorem.

Theorem A.3.3. *If $r_0 > R$, it cannot be guaranteed that the robots will converge to a regular polygon of radius R.*

Proof. We prove this statement by providing a pathological initial configuration. We only need to redraw the formation of Figure A.5 A, but with R replaced by r_0 and some of the robots removed, as shown in Figure A.7.

None of the five robots see each other, and therefore they will continue rotating along the circle of radius r_0, never converging to a regular pentagon. ■

The above analysis, coupled with simulation results, suggest that in practice it is advisable to set:

$$\frac{R}{10} < r_0 < R$$

$$r_1 = r_0$$

Lower-bound on the Sensing Range

If $r_0 = R$, the robots in Figure A.5 A will maintain their formation regardless of the sensing range, because no robot will ever see another robot in any case. However, if $r_0 < R$, then maintaining the formation relies on the negative feedback effect discussed in the previous section, which in turn relies on each robot being able to sense the adjacent robot. Therefore, a lower-bound on the sensing range required to maintain the formation is given by calculating the value of δ in Figure A.5 B. We see that:

$$\delta = R \sin \frac{2\pi}{N},$$

or, substituting in the expression for R calculated in Theorem A.3.1:

$$\delta = \frac{\rho \sin \frac{2\pi}{N}}{1 - \cos \frac{2\pi}{N}}$$

Note that this is only a lower-bound on the required sensing range and does not say anything about the ability of robots being able to *form* the milling formation from any initial configuration. The ability to form the milling formation depends on the compactness of the initial configuration. We do not know of any method to guarantee the milling formation based on the sensing range and the initial configuration, but simulation results suggest that milling is almost always successful if the robots initially form a strongly-connected graph (with two robots being connected if they are within each other's sensing range).

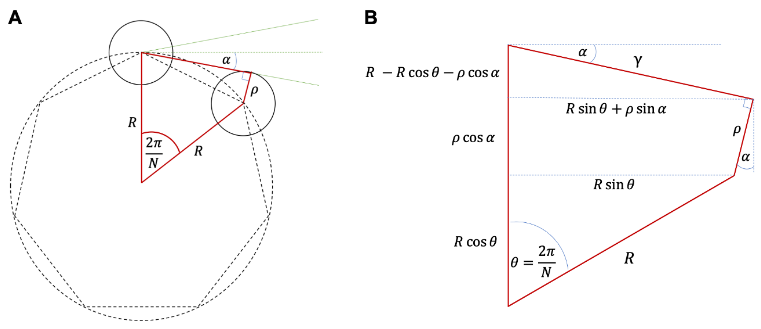


Figure A.8: Milling with field-of-view sensors. (A) A regular polygon configuration with field-of-view sensors (cf. Fig. A.5 A for line-of-sight sensors). Only two robots are shown for simplicity; additional robots lie on each vertex of the polygon. **(B)** Geometry for calculating the milling radius with field-of-view sensors.

A.3.3 Milling with Field-of-View Sensors

We now generalize the results of the previous section to the case of a field-of-view sensor. In practice, field-of-view sensors are more straightforward to implement than line-of-sight sensors (strictly speaking, even a column of width 1 pixel corresponds to a field-of view). Moreover, having a field of view allows us to control the radius of the formation. The only two results that must be modified are the milling radius in Theorem A.3.1 and the lower-bound on the sensing range.

Recall that we define α as the half field of view of the sensor to obtain simpler equations, but for brevity we refer to it as the field of view.

Figure A.5 must be modified to account for the field of view. The resulting configuration is shown in Figure A.8 A (only two robots are shown for simplicity; additional robots lie on each vertex of the polygon).

Milling Radius

Figure A.8 B shows an expanded view of the red quadrilateral in Figure A.8 A, with additional annotations and measurements. The additional measurements follow from the ones in Figure A.8 A by simple geometrical observations. Note that we have defined $\Theta = 2\pi/N$ to keep the equations

simpler.

Now, referring to the uppermost triangle in Figure A.8 B, we have:

$$\tan\alpha = \frac{\sin\alpha}{\cos\alpha} = \frac{R - R\cos\Theta - \rho\cos\alpha}{R\sin\Theta + \rho\sin\alpha},$$

which can be rearranged to give:

$$R = \frac{\rho}{\cos\alpha - \cos\Theta\cos\alpha - \sin\Theta\sin\alpha}$$

The denominator can be further simplified via trigonometric identities, and reintroducing $\Theta = 2\pi/N$ we finally obtain:

$$R = \frac{\rho}{\cos\alpha - \cos\left(\frac{2\pi}{N} - \alpha\right)}$$

As a check, observe that letting $\alpha = 0$ in the above equation, we recover the equation for the line-of-sight sensor given in Theorem A.3.1.

Maximum Field of View

Theorem A.3.4. *The maximum field of view for achieving a finite-radius, steady-state milling formation is $\alpha = \pi/N$.*

Proof. Observe from the above expression for R that the denominator tends to 0 as $\alpha \to \pi N$ from below, meaning that $R \to \infty$. For $\alpha > \pi/N$, we have $R < 0$. In other words, the expression only provides admissible (i.e., positive) values for R when $\alpha \in [0, \pi/N]$. ■

The intuitive explanation of this result is that for a given number of robots, if the field of view is set too large it becomes geometrically impossible to form a finite-radius regular polygon formation where no robot sees another robot.

Simulation results show that if $\alpha > \pi/N$, a regular polygon formation is still formed, but its radius grows indefinitely. This is because as robots see other robots, they rotate counterclockwise making the formation larger.

Lower-bound on the Sensing Range

It is straightforward to calculate the lower-bound on the sensing range as we did for the case of line-of-sight sensors. In particular, we only need to apply the Pythagorean theorem to the upper triangle in Figure A.8 B to find the length of the hypotenuse γ:

$$\gamma = \sqrt{(R - R\cos\Theta - \rho\cos\alpha)^2 + (R\sin\Theta + \rho\sin\alpha)^2}$$

A.4 Multi-behavior Collective Search

A.4.1 Detection of LED Flashings

During collective search, Bluebots must be able to detect flashing robots and distinguish them from other Bluebots whose lights are continuously on. The Raspberry Pi camera's rapid sequence capture mode, which operates at a rate of up to 60 frames per second, makes it possible in principle to detect a flashing light up to a rate of 30 Hz (by the Nyquist sampling theorem). The underlying idea of flash detection is to identify blobs in the field of view that alternately appear and disappear at the expected rate. While straightforward in principle, this process is made challenging in practice by three main factors:

i) The collective clutters the visual field of each robot. Lights from the same or separate robots may appear close to each other or overlap, especially at the extremes of the visual field. A light that is intermittently merging and unmerging with another one, due to noise, may trigger an

Algorithm 7: Flashing LED detection (FlashLED)

```
1  Input: a list of $N = 30$ binary images in $m \times n$ matrix format with only black and
     white pixels (captured in burst mode within 0.5 s)
2  Blob detection: Run Rapid LED blob detection (RapidLED, Alg. 6) on each image.
3  Outlier detection: Outliers (might) correspond to an LED that was on in a given
     image and turned off in the subsequent image, signifying a flash.
4  for each pair of successive images $k$ and $k + 1$ do
5      Sort pairs of blobs from images $k$ and $k + 1$ by their distance.
6      Match pairs that are closer than a threshold distance (in pixels).
7      Designate remaining blobs in image $k$ as outliers for image $k$, i.e., blobs in image $k$
         that have no counterpart in image $k + 1$.
8  Streak detection: A streak corresponds to a flashing light that might be moving
     while the burst sequence is taken, manifested as outliers that appear in multiple
     images such that successive appearances are not separated by more than some
     threshold distance.
9  Initialize the set of streaks to the empty set.
10 for each image do
11     for outliers in given image do
12         Given current outlier, search match in the set of streaks:
13         if outlier is within a threshold distance of the last outlier that was added to a
             given streak then
14             Declare match, add outlier to given streak.
15         else
16             Start a new streak with the given outlier.
17 return the longest streak, which corresponds to a flashing robot if longer than some
     fixed threshold
```

incorrect flash detection. This phenomenon makes it challenging to detect one or a few flashing robots among several more non-flashing ones reliably.

ii) While the capture sequence is in progress, both the image-taking robot and the robots in its visual field may be moving. As a result, lights do not, in general, appear at the same location in successive images.

   - To the image-taking robot, the movement of other robots is unpredictable. Moreover, the effect of this movement on the locations of lights in the images depends on the distances to these robots, which may be unknown.
   - In principle, odometry can compensate for the movement of the image-taking robot itself. In practice, however, the feasibility of this strategy is severely limited for two reasons: i) the noisy nature of the fin-based actuation modality, and ii) undetectable influences from other robots, in the form of hydrodynamic interference or physical impacts. All forms of motion by the image-taking robot cause lights in successive images to move, but yawing is especially disruptive as it causes all lights to move significantly, regardless of distance.

iii) An inherent trade-off for the robot's quasi-omnidirectional field-of-view is that objects appear small even when they are close. This limitation becomes exacerbated when using a limited picture resolution for faster computational processing. At our default resolution of 192×256 pixels, a robot starts to perceive another robot's light as only one pixel at a modest distance of around 10 body lengths (but the range of perception is significantly longer: up to around 30 body lengths). If we use a simple thresholding protocol for deciding which pixels are on or off, a light that is at the critical distance may be perceived to move in and out of the frame rapidly, due to noise. This phenomenon is highly prone to misclassification as flashing, leading to false-positive errors.

To address the challenges above we use an algorithm (FlashLED, see Alg. 7) that is robust against false positive errors while being sufficiently sensitive to detect true flashing. In each control cycle, the algorithm is run once for each camera. All the threshold values were tuned empirically.

A.4.2 Expected Search Time for a Single Robot — Proof

I approximate the expected time for source detection if Bluebots do not collaborate with a random walk on an undirected graph G. The vertices of G are the integers $0, \cdots, n$ and represent distance to the source; consecutive vertices are connected. I calculate the expected number of steps h_j to reach the target node n (the source) when starting from initial node j. I reach the source at vertex n, so $h_n = 0$; I can only move closer to the source from vertex 0, so $h_0 = h_1 + 1$. For all intermediate vertices $1 \leq j \leq n-1$, I use linearity of expectations to find h_j. I introduce Z_j, a random variable for the number of steps to reach n starting from j. For a random walk, $Z_j = Z_{j-1} + 1$ and $Z_j = Z_{j+1} + 1$ are equally likely with probability $1/2$ each. Hence:

$$\mathbf{E}[Z_j] = \mathbf{E}\left[\frac{1}{2}(Z_{j-1}+1) + \frac{1}{2}(Z_{j+1}+1)\right],$$

where $\mathbf{E}[Z_j] = h_j$ and by linearity of expectations, $h_j = h_{j-1}/2 + h_{j+1}/2 + 1$. This gives me the following system of equations:

$$\begin{aligned} h_n &= 0, \\ h_j &= \frac{h_{j-1}}{2} + \frac{h_{j+1}}{2} + 1 \qquad \forall 1 \leq j \leq n-1, \\ h_0 &= h_1 + 1, \end{aligned}$$

for which I can easily verify that $h_j = n^2 - j^2$ is the correct solution. Since the system has exactly $n+1$ linearly independent equations and $n+1$ unknowns, this solution is also unique.

B

On Robotics and Fish Swimming

Fish are excellent swimmers, navigating cluttered environments such as coral reefs in search of food, migrating long distances in the ocean, or swimming up rivers to reproduce. In doing so, fish combine a high degree of maneuverability with effective long-distance swimming, a combination that is highly attractive from an engineering perspective. Fish can achieve this level of performance in part by using multiple fins to vector forces in three dimensions, supported by sensory systems for closed-

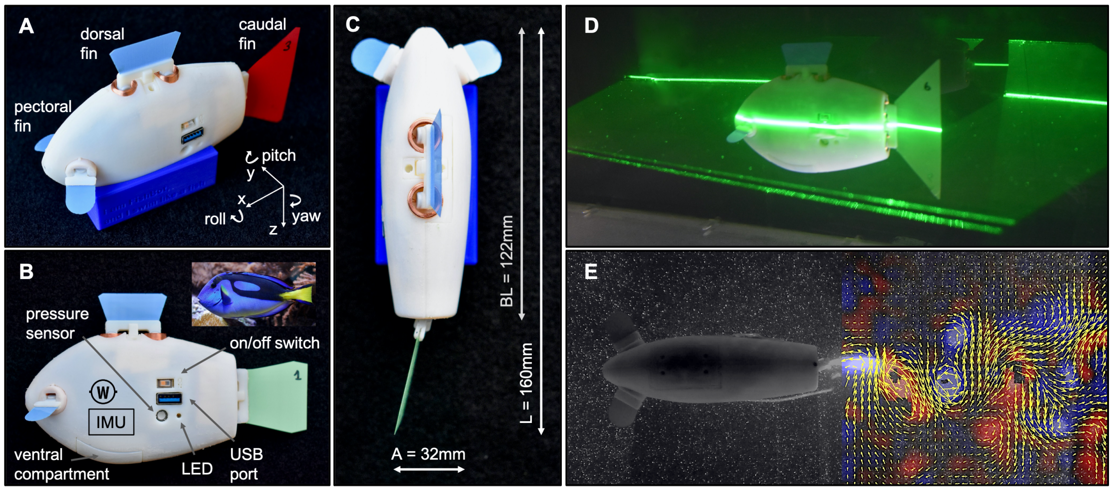


Figure B.1: Finbot: An autonomous and biomimetic experimental platform. **(A)** Finbot was designed to enable fish-like autonomous swimming. Four independently controllable flexible fins enable precise maneuvers in 3D space (x, y, z corresponding to surge, sway, heave). The caudal fin connects magnetically to the main body and can be exchanged easily to alter swimming speed and cost of transport. **(B)** The rigid and streamlined body is inspired in shape by fish such as the blue tang (depicted top right, credit: by DerHans04, used under CC BY-SA 3.0; mirrored, rotated, cropped). Finbot swims autonomously in 3D, using an inertial sensor (IMU) to control heading and a pressure sensor to control depth, and can monitor its own power consumption. **(C)** Tail beat amplitude A was tuned to approximately $20\,\%$ of the body length L (including fin) according to the observed value among a diverse species of fish[142]. **(D)** Finbot in a laser light imaging with particle image velocimetry, which was used to visualize wake structures. **(E)** A trailing wake behind Finbot; several wakes were similar to reverse Kármán streets, which are characteristic of fish swimming.

loop control of fin kinematics.

In a multidisciplinary collaboration with experts on fish swimming and fluid dynamics spearheaded by George Lauder and Mehdi Saadat, we demonstrated fish-like swimming including properties such as U-shaped costs of transport and reverse Kármán wakes with a custom, more biomimetic version of our underwater robots named Finbot (Fig. B.1). We then used Finbot as a fish surrogate to experimentally validate a new thrust-based hypothesis on the hydrodynamic advantages of in-line schooling.

In the following, the abstracts of the two publications coming out of this collaboration are reprinted. Details are found in the full manuscripts.

B.1 Fish-like Three-dimensional Swimming with an Autonomous, Multi-fin, and Biomimetic Robot

Fish migrate across considerable distances and exhibit remarkable agility to avoid predators and feed. Fish swimming performance and maneuverability remain unparalleled when compared to robotic systems, partly because previous work has focused on robots and flapping foil systems that are either big and complex, or tethered to external actuators and power sources. By contrast, we present a robot — the Finbot — that combines high degrees of autonomy, maneuverability, and biomimicry with miniature size (160 cm^3). Thus, it is well-suited for controlled three-dimensional experiments on fish swimming in confined laboratory test beds. Finbot uses four independently controllable fins and sensory feedback for precise closed-loop underwater locomotion. Different caudal fins can be attached magnetically to reconfigure Finbot for swimming at top speed (122 mm/s $\equiv$ 1 BL/s) or minimal cost of transport (CoT $=$ 8.2) at Strouhal numbers as low as 0.53. We conducted more than 150 experiments with 12 different caudal fins to measure three key characteristics of swimming fish: i) linear speed-frequency relationships, ii) U-shaped costs of transport, and iii) reverse Kármán wakes (visualized with particle image velocimetry). More fish-like wakes appeared where the cost of transport was low. By replicating autonomous multi-fin fish-like swimming, Finbot narrows the gap between fish and fish-like robots and can address open questions in aquatic locomotion, such as optimized propulsion for new fish robots, or the hydrodynamic principles governing the energy savings in fish schools.

B.2 Hydrodynamic Advantages of In-line Schooling

Fish benefit energetically when swimming in groups, which is reflected by lower tail-beat frequencies for maintaining a given speed. Recent studies further show that fish save the most energy when swimming behind their neighbor such that both the leader and the follower benefit. However, the mechanisms underlying such hydrodynamic advantages have thus far not been established conclusively. The long-standing drafting hypothesis — reduction of drag forces by judicious positioning in regions of reduced oncoming flow — fails to explain advantages of in-line schooling described in this work. We present an alternate hypothesis for the hydrodynamic benefits of in-line swimming based on enhancement of propulsive thrust. Specifically, we show that an idealized school consisting of in-line pitching foils gains hydrodynamic benefits via two mechanisms that are rooted in the undulatory jet leaving the leading foil and impinging on the trailing foil: i) leading-edge suction on the trailer foil, and ii) added-mass push on the leader foil. Our results demonstrate that the savings in power can reach as high as 70 % for a school swimming in a compact arrangement. Informed by these findings, we designed a modification of the tail propulsor that yielded power savings of up to 56 % in a self-propelled autonomous swimming robot. Our findings provide insights into hydrodynamic advantages of fish schooling, and also enable bioinspired designs for significantly more efficient propulsion systems that can harvest some of their energy left in the flow.

C
On Smart Materials for Soft Robots

Dielectric elastomer actuators (DEAs) are compliant capacitors that operate as soft electromechanical transducers and convert electrical energy into mechanical work (Fig. C.1 A). DEAs have several attractive properties compared to other smart materials (e.g., use of soft elastomers, or embedded and direct electrical-to-mechanical actuation), which make them particularly suitable for soft robotics applications. Unfortunately, most DEAs also have severe limitations including low output

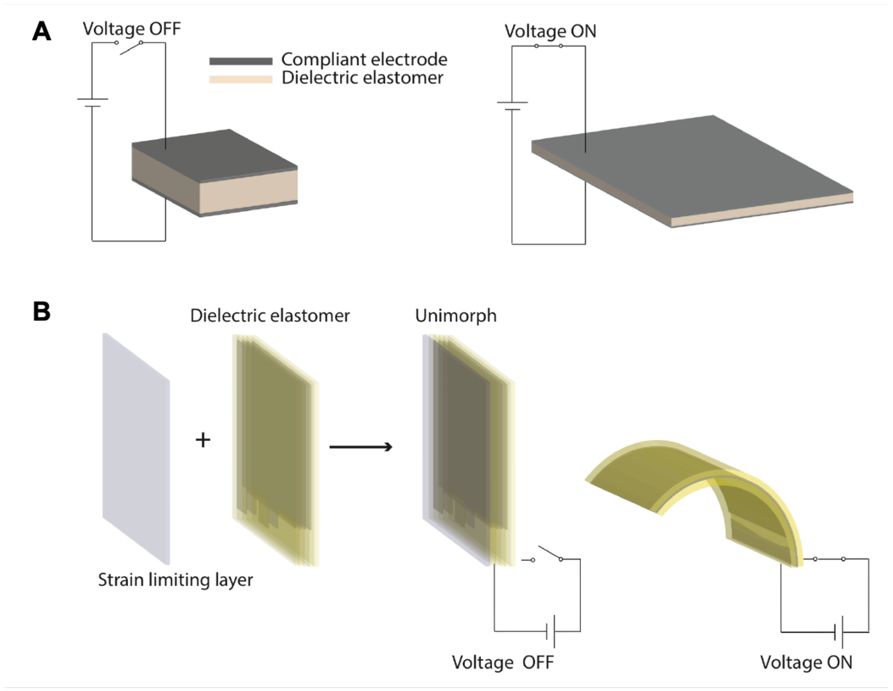


Figure C.1: Operation of a dielectric elastomer actuator. (A) An electric field applied between two outer compliant electrodes causes a Maxwell stress. The outer electrodes attract and compress the inner soft dielectric elastomer. **(B)** Directional control of such actuation is achieved by addition of strain limiting elements. Here, a single strain limiting layer prevents biaxial expansion and causes the unimorph to bend, as is useful in a propulsive fish-like fin, or an impulsive jumping mechanism. (Credit: Mihai Duduta[143])

forces, the need for a rigid frame, high actuation voltages, slow response times, low energy density, and challenging integration with other components.

In a multidisciplinary team of roboticists and material scientists spearheaded by Mihai Duduta, we addressed the limitations of DEAs. In a novel design approach, we were able to avoid pre-stretch of the elastomers, and multiply actuation forces by stacking several actuation layers. Such DEAs were strong enough to act as propulsive fins on an underwater robot (Fig. C.1 B) in one of the first demonstrations of autonomous DEA-driven vehicles[134].

We then explored soft DEA robots conducive to multi-modal locomotion[144,145]. Our bending beam design approach enabled multiple gaits including crawling, hopping, jumping, and rolling

(Fig. C.1 B). Gaits were selected and controlled by choice of actuation pattern, i.e., the frequency and magnitude of the applied input voltage to the DEA. Such DEA robots can be used as versatile experimental devices to validate locomotion models, in both natural and engineered systems.

In the following, the abstracts of the three publications coming out of this collaboration are reprinted. Details are found in the full manuscripts.

C.1 A Modular Dielectric Elastomer Actuator to Drive Miniature Autonomous Underwater Vehicles

In this paper we present the design of a fin-like dielectric elastomer actuator (DEA) that drives a miniature autonomous underwater vehicle (AUV). The fin-like actuator is modular and independent of the body of the AUV. All electronics required to run the actuator are inside the 100 mm long 3D-printed body, allowing for autonomous mobility of the AUV. The DEA is easy to manufacture, requires no pre-stretch of the elastomers, and is completely sealed for underwater operation. The output thrust force can be tuned by stacking multiple actuation layers and modifying the Young's modulus of the elastomers. The AUV is reconfigurable by a shift of its center of mass, such that both planar and vertical swimming can be demonstrated on a single vehicle. For the DEA we measured thrust force and swimming speed for various actuator designs ran at frequencies from 1 Hz to 5 Hz. For the AUV we demonstrated autonomous planar swimming and closed-loop vertical diving. The actuators capable of outputting the highest thrust forces can power the AUV to swim at speeds of up to 0.55 body lengths per second. The speed falls in the upper range of untethered swimming robots powered by soft actuators. Our tunable DEAs also demonstrate the potential to mimic the undulatory motions of fish fins.

C.2 Electrically-Latched Compliant Jumping Mechanism Based on a Dielectric Elastomer Actuator

Jumping mechanisms are useful in robotics for locomotion in unstructured environments, or for self-righting abilities. However, most rigid robots rely on impact with the ground to jump, thereby requiring a relatively rigid, and flat environment. Moreover, they need to be able to absorb high impact forces during landing in order to maintain structural integrity. In this paper we investigate soft systems, capable of jumping repeatedly in unstructured environments with no need for precise landings. Our impulsive approach is based on a soft electro-mechanical transducer, a dielectric elastomer actuator (DEA). The design is inspired by click-beetles and simple bio-mechanical models, which convert the flexing around a hinge into jumping. Our actuator is power amplified by the addition of a stiffer strip, allowing for rapid shape transitions (22 ms) between flat and curved states. The transition is controlled by an electric latch: the DEA is discharged faster than the actuator can deform. The mechanical energy stored in the composite beam is released rapidly, leading to impulsive motions (jumps of a full body length: i.e. 5 cm). This demonstration of an electrically-latched power amplification mechanism shows that relatively simple electro-mechanical systems can exhibit impulsive behavior which may enable new types of locomotion in compliant machines.

C.3 Tunable Multi-modal Locomotion in Soft Dielectric Elastomer Robots

Soft robots require strong, yet flexible actuators for locomotion and manipulation tasks in unstructured environments. Dielectric elastomer actuators (DEAs) are well suited for these challenges in soft robotics because they operate as compliant capacitors and directly convert electrical energy into mechanical work, thereby allowing for simple design integration at a minimal footprint. In most demonstrations, DEA-based robots are limited to a single mode of locomotion, for example crawling, swimming, or jumping. In this work, we explored a range of actuation patterns in combination with a novel actuator design to enable multi-modal locomotion, whereby an actuation pattern is defined by an actuation voltage (proportional to the applied electric field) and frequency (the actuation rate). We present a DEA robot capable of three different gaits including crawling, hopping, and jumping. In addition, our robot can set itself upright by performing a roll, for example to prepare for the next jump after landing on its side. These results demonstrate that DEAs can be used as versatile experimental devices to validate locomotion models, in both natural and engineered systems.

References

[1] Scott Camazine, Jean-Louis Deneubourg, Nigel R Franks, James Sneyd, Eric Bonabeau, and Guy Theraula. *Self-organization in biological systems*. Princeton university press, 2003. ISBN 0691116245.

[2] Chris R Reid, David JT Sumpter, and Madeleine Beekman. Optimisation in a natural system: Argentine ants solve the towers of hanoi. *Journal of Experimental Biology*, 214(1):50–58, 2011.

[3] Randolf Menzel, Jacqueline Fuchs, Andreas Kirbach, Konstantin Lehmann, and Uwe Greggers. Navigation and communication in honey bees. In *Honeybee Neurobiology and Behavior*, pages 103–116. Springer, 2012.

[4] Brian L Partridge. The structure and function of fish schools. *Scientific american*, 246(6): 114–123, 1982.

[5] Evelyn Shaw. Schooling fishes: the school, a truly egalitarian form of organization in which all members of the group are alike in influence, offers substantial benefits to its participants. *American Scientist*, 66(2):166–175, 1978.

[6] Tony J Pitcher. Functions of shoaling behaviour in teleosts. In *The behaviour of teleost fishes*, pages 294–337. Springer, 1986.

[7] Julia K Parrish, Steven V Viscido, and Daniel Grunbaum. Self-organized fish schools: an examination of emergent properties. *The biological bulletin*, 202(3):296–305, 2002.

[8] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013. ISSN 1935-3812.

[9] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, 2014.

[10] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[11] Aaron Weinstein, Adam Cho, Giuseppe Loianno, and Vijay Kumar. Visual inertial odometry swarm: An autonomous swarm of vision-based quadrotors. *IEEE Robotics and Automation Letters*, 3(3):1801–1807, 2018.

[12] Gábor Vásárhelyi, Csaba Virágh, Gergő Somorjai, Tamás Nepusz, Agoston E Eiben, and Tamás Vicsek. Optimized flocking of autonomous drones in confined environments. *Science Robotics*, 3(20):eaat3536, 2018.

[13] Gábor Vásárhelyi, Cs Virágh, Gergo Somorjai, Norbert Tarcai, Tamás Szörényi, Tamás Nepusz, and Tamás Vicsek. Outdoor flocking and formation flight with autonomous aerial robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3866–3873. IEEE, 2014.

[14] Sabine Hauert, Severin Leven, Maja Varga, Fabio Ruini, Angelo Cangelosi, Jean-Christophe Zufferey, and Dario Floreano. Reynolds flocking in reality with fixed-wing robots: communication range vs. maximum turning rate. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5015–5020. IEEE, 2011.

[15] KN McGuire, C De Wagter, K Tuyls, HJ Kappen, and GCHE de Croon. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, 4(35):eaaw9710, 2019.

[16] Geert De Cubber, Daniela Doroftei, Daniel Serrano, Keshav Chintamani, Rui Sabino, and Stephane Ourevitch. The eu-icarus project: developing assistive robotic tools for search and rescue operations. In *2013 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, pages 1–4. IEEE, 2013. ISBN 1479908800.

[17] Yanwu Zhang, John P Ryan, Brett W Hobson, Brian Kieft, Anna Romano, Benedetto Barone, Christina M Preston, Brent Roman, Ben-Yair Raanan, Douglas Pargett, et al. A system of coordinated autonomous robots for lagrangian studies of microbes in the oceanic deep chlorophyll maximum. *Science Robotics*, 6(50), 2021.

[18] Jules S Jaffe, Peter JS Franks, Paul LD Roberts, Diba Mirza, Curt Schurgers, Ryan Kastner, and Adrien Boch. A swarm of autonomous miniature underwater robot drifters for exploring submesoscale ocean dynamics. *Nature communications*, 8(1):1–8, 2017.

[19] Tyler Maccready, Anthony White, and Drew Franklin Heltsley. Water drone, July 3 2018. US Patent 10,011,337.

[20] Tyler Maccready, Sankaran Ramakrishnan, and Thomas Zambrano. Swarm autopilot, June 4 2019. US Patent 10,310,518.

[21] Guorui Li, Xiangping Chen, Fanghao Zhou, Yiming Liang, Youhua Xiao, Xunuo Cao, Zhen Zhang, Mingqi Zhang, Baosheng Wu, Shunyu Yin, et al. Self-powered soft robot in the mariana trench. *Nature*, 591(7848):66–71, 2021.

[22] Enrica Zereik, Marco Bibuli, Nikola Mišković, Pere Ridao, and António Pascoal. Challenges and future trends in marine robotics. *Annual Reviews in Control*, 46:350–368, 2018.

[23] Thomas Schmickl, Ronald Thenius, Christoph Moslinger, Jon Timmis, Andy Tyrrell, Mark Read, James Hilder, Jose Halloy, Alexandre Campo, Cesare Stefanini, et al. Cocoro–the self-aware underwater swarm. In *2011 Fifth IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 120–126. IEEE, 2011.

[24] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, 1987.

[25] Iain D Couzin, Jens Krause, Richard James, Graeme D Ruxton, and Nigel R Franks. Collective memory and spatial sorting in animal groups. *Journal of theoretical biology*, 218(1):1–12, 2002.

[26] Daniel S Calovi, Ugo Lopez, Sandrine Ngo, Clément Sire, Hugues Chaté, and Guy Theraulaz. Swarming, schooling, milling: phase diagram of a data-driven fish school model. *New Journal of Physics*, 16(1):015026, 2014. ISSN 1367-2630.

[27] Daniel Strömbom. Collective motion from local attraction. *Journal of theoretical biology*, 283 (1):145–151, 2011.

[28] Shuguang Li, Richa Batra, David Brown, Hyun-Dong Chang, Nikhil Ranganathan, Chuck Hoberman, Daniela Rus, and Hod Lipson. Particle robotics based on statistical mechanics of loosely coupled components. *Nature*, 567(7748):361, 2019. ISSN 1476-4687.

[29] I Slavkov, D Carrillo-Zapata, N Carranza, X Diego, F Jansson, J Kaandorp, S Hauert, and J Sharpe. Morphogenesis in robot swarms. *Science Robotics*, 3(25), 2018.

[30] Melvin Gauci, Jianing Chen, Tony J Dodd, and Roderich Groß. *Evolving aggregation behaviors in multi-robot systems with binary sensors*, pages 355–367. Springer, 2014.

[31] James A Preiss, Wolfgang Honig, Gaurav S Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3299–3304. IEEE, 2017.

[32] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35(4):287–300, 2013.

[33] Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, et al. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71, 2013.

[34] Junku Yuh. Design and control of autonomous underwater robots: A survey. *Autonomous Robots*, 8(1):7–24, 2000.

[35] Payam Zahadat and Thomas Schmickl. Division of labor in a swarm of autonomous underwater robots by improved partitioning social inhibition. *Adaptive Behavior*, 24(2):87–101, 2016.

[36] Felix Schill, Alexander Bahr, and Alcherio Martinoli. *Vertex: A New Distributed Underwater Robotic Platform for Environmental Monitoring*, pages 679–693. Springer International Publishing, Cham, 2018. ISBN 978-3-319-73008-0. doi: 10.1007/978-3-319-73008-0_47. URL https://doi.org/10.1007/978-3-319-73008-0_47.

[37] Iuliu Vasilescu, Keith Kotay, Daniela Rus, Matthew Dunbabin, and Peter Corke. Data collection, storage, and retrieval with an underwater sensor network. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 154–165, 2005.

[38] Naomi E Leonard, Derek A Paley, Russ E Davis, David M Fratantoni, Francois Lekien, and Fumin Zhang. Coordinated control of an underwater glider fleet in an adaptive ocean sampling field experiment in monterey bay. *Journal of Field Robotics*, 27(6):718–740, 2010.

[39] Robert K Katzschmann, Joseph DelPreto, Robert MacCurdy, and Daniela Rus. Exploration of underwater life with an acoustically controlled soft robotic fish. *Science Robotics*, 3(16): eaar3449, 2018. ISSN 2470-9476.

[40] David JT Sumpter. *Collective animal behavior*. Princeton University Press, 2010.

[41] Magnus Delight, Sankaran Ramakrishnan, Thomas Zambrano, and Tyler MacCready. Developing robotic swarms for ocean surface mapping. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5309–5315. IEEE, 2016.

[42] Miguel Duarte, Vasco Costa, Jorge Gomes, Tiago Rodrigues, Fernando Silva, Sancho Moura Oliveira, and Anders Lyhne Christensen. Evolution of collective behaviors for a real swarm of aquatic surface robots. *PloS one*, 11(3), 2016.

[43] Malcolm A MacIver, Ebraheem Fontaine, and Joel W Burdick. Designing future underwater vehicles: principles and mechanisms of the weakly electric fish. *IEEE Journal of Oceanic Engineering*, 29(3):651–659, 2004. ISSN 0364-9059.

[44] Ronald Thenius, Daniel Moser, Joshua Cherian Varughese, Serge Kernbach, Igor Kuksin, Olga Kernbach, Elena Kuksina, Nikola Mišković, Stjepan Bogdan, Tamara Petrović, et al. subcultron-cultural development as a tool in underwater robotics. In *Artificial Life and Intelligent Agents Symposium*, pages 27–41. Springer, 2016.

[45] Tony J Pitcher, Brian L Partridge, and CS Wardle. A blind fish can school. *Science*, 194(4268): 963–965, 1976.

[46] Brian L Partridge and Tony J Pitcher. The sensory basis of fish schools: relative roles of lateral line and vision. *Journal of comparative physiology*, 135(4):315–325, 1980.

[47] Ron H Douglas and Craig W Hawryshyn. Behavioural studies of fish vision: an analysis of visual capabilities. In *The visual system of fish*, pages 373–418. Springer, 1990.

[48] Li Jiang, Luca Giuggioli, Andrea Perna, Ramón Escobedo, Valentin Lecheval, Clément Sire, Zhangang Han, and Guy Theraulaz. Identifying influential neighbors in animal flocking. *PLoS computational biology*, 13(11):e1005822, 2017.

[49] DM Guthrie. Role of vision in fish behaviour. In *The behaviour of Teleost fishes*, pages 75–113. Springer, 1986.

[50] Quentin Bone and Richard Moore. *Biology of fishes*. Taylor & Francis, 2008. ISBN 1134186312.

[51] Laurie Landeau and John Terborgh. Oddity and the 'confusion effect'in predation. *Animal Behaviour*, 34(5):1372–1380, 1986. ISSN 0003-3472.

[52] J Krause, D Hoare, S Krause, CK Hemelrijk, and DI Rubenstein. Leadership in fish shoals. *Fish and Fisheries*, 1(1):82–89, 2000.

[53] Tony J Pitcher. *The behaviour of teleost fishes*. Springer Science & Business Media, 2012.

[54] David F Gruber, Brennan T Phillips, Rory O'Brien, Vivek Boominathan, Ashok Veeraraghavan, Ganesh Vasan, Peter O'Brien, Vincent A Pieribone, and John S Sparks. Bioluminescent flashes drive nighttime schooling behavior and synchronized swimming dynamics in flashlight fish. *PloS one*, 14(8), 2019.

[55] DH Cushing and FR Harden Jones. Why do fish school? *Nature*, 218(5145):918–920, 1968.

[56] Stefano Marras, Shaun S Killen, Jan Lindström, David J McKenzie, John F Steffensen, and Paolo Domenici. Fish swimming in schools save energy regardless of their spatial position. *Behavioral ecology and sociobiology*, 69(2):219–226, 2015.

[57] Mehdi Saadat, Florian Berlinger, Artan Sheshmani, Radhika Nagpal, George V Lauder, and Hossein Haj-Hariri. Hydrodynamic advantages of in-line schooling. *Bioinspiration & Biomimetics*, 16(4):046002, 2021.

[58] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. *Physical review letters*, 75(6):1226, 1995.

[59] Daniel S Calovi, Alexandra Litchinko, Valentin Lecheval, Ugo Lopez, Alfonso Pérez Escudero, Hugues Chaté, Clément Sire, and Guy Theraulaz. Disentangling and modeling interactions in fish with burst-and-coast swimming reveal distinct alignment and attraction behaviors. *PLoS computational biology*, 14(1):e1005933, 2018.

[60] M Adioui, JP Treuil, and O Arino. Alignment in a fish school: a mixed lagrangian–eulerian approach. *Ecological Modelling*, 167(1-2):19–32, 2003.

[61] Eric Carlen, Maria C Carvalho, Pierre Degond, and Bernt Wennberg. A boltzmann model for rod alignment and schooling fish. *Nonlinearity*, 28(6):1783, 2015.

[62] David JT Sumpter, Jens Krause, Richard James, Iain D Couzin, and Ashley JW Ward. Consensus decision making by fish. *Current Biology*, 18(22):1773–1777, 2008.

[63] Ashley JW Ward, David JT Sumpter, Iain D Couzin, Paul JB Hart, and Jens Krause. Quorum decision-making facilitates information transfer in fish shoals. *Proceedings of the National Academy of Sciences*, 105(19):6948–6953, 2008.

[64] Derek J Hoare, Iain D Couzin, J-GJ Godin, and Jens Krause. Context-dependent group size choice in fish. *Animal Behaviour*, 67(1):155–164, 2004.

[65] Duarte Dias, Rodrigo Ventura, Pedro Lima, and Alcherio Martinoli. On-board vision-based 3d relative localization system for multiple quadrotors. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1181–1187. Ieee, 2016.

[66] Jon C van Olst and John R Hunter. Some aspects of the organization of fish schools. *Journal of the Fisheries Board of Canada*, 27(7):1225–1238, 1970. ISSN 0706-652X.

[67] Oleg Grinchuk, Vadim Lebedev, and Victor Lempitsky. Learnable visual markers. In *Advances In Neural Information Processing Systems*, pages 4143–4151, 2016.

[68] Zhifang Fan, Jack Chen, Jun Zou, David Bullen, Chang Liu, and Fred Delcomyn. Design and fabrication of artificial lateral line flow sensors. *Journal of micromechanics and microengineering*, 12(5):655, 2002.

[69] Ahmad T Abdulsadda and Xiaobo Tan. An artificial lateral line system using ipmc sensor arrays. *International Journal of Smart and Nano Materials*, 3(3):226–242, 2012.

[70] Wayne J Korzan, Rex R Robison, Sheng Zhao, and Russell D Fernald. Color change as a potential behavioral strategy. *Hormones and behavior*, 54(3):463–470, 2008.

[71] Michael Sfakiotakis, David M Lane, and J Bruce C Davies. Review of fish swimming modes for aquatic locomotion. *IEEE Journal of oceanic engineering*, 24(2):237–252, 1999.

[72] WS Alevizon. Mixed schooling and its possible significance in a tropical western atlantic parrotfish and surgeonfish. *Copeia*, 1976(4):796–798, 1976.

[73] Florian Berlinger, Jeff Dusek, Melvin Gauci, and Radhika Nagpal. Robust maneuverability of a miniature, low-cost underwater robot using multiple fin actuation. *IEEE Robotics and Automation Letters*, 3(1):140–147, 2017.

[74] Katerina Soltan, Jamie O'Brien, Jeff Dusek, Florian Berlinger, and Radhika Nagpal. Biomimetic actuation method for a miniature, low-cost multi-jointed robotic fish. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–9. IEEE, 2018.

[75] Florian Berlinger, Mehdi Saadat, Hossein Haj-Hariri, George V Lauder, and Radhika Nagpal. Fish-like three-dimensional swimming with an autonomous, multi-fin, and biomimetic robot. *Bioinspiration & Biomimetics*, 16(2):026018, 2021.

[76] Vladislav Kopman and Maurizio Porfiri. Design, Modeling, and Characterization of a Miniature Robotic Fish for Research and Education in Biomimetics and Bioinspiration. *IEEE/ASME Transactions on Mechatronics*, 18(2):471–483, November 2012.

[77] Anthony J Clark, Xiaobo Tan, and Philip K Mckinley. Evolutionary multiobjective design of a flexible caudal fin for robotic fish. *Bioinspiration and Biomimetics*, 10(6):1–17, November 2015.

[78] Matteo Aureli, Vladislav Kopman, and Maurizio Porfiri. Free-Locomotion of Underwater Vehicles Actuated by Ionic Polymer Metal Composites. *IEEE/ASME Transactions on Mechatronics*, 15(4):603–614, September 2009.

[79] Yogo Takada, Keisuke Koyama, and Takahiro Usami. Position Estimation of Small Robotic Fish Based on Camera Information and Gyro Sensors. *Robotics*, 3(2):149–162, June 2014.

[80] Simon A Watson and Peter N Green. Propulsion systems for micro-Autonomous Underwater Vehicles (μAUVs). In *2010 IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 435–440, 2010.

[81] S Mintchev, E Donati, S Marrazza, and C Stefanini. Mechatronic design of a miniature underwater robot for swarm operations. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2938–2943, 2014.

[82] The Jessiko project. http://www.robotswim.com/index.php. Accessed: 2021-03-26.

[83] Yonghui Hu, Wei Zhao, Guangming Xie, and Long Wang. Development and target following of vision-based autonomous robotic fish. *Robotica*, 27(07):1075–1089, 2009.

[84] S Bhattacharyya and HH Asada. Control of a compact, tetherless ROV for in-contact inspection of complex underwater structures. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2265–2272, 2014. doi: 10.1109/IROS.2014.6942868.

[85] James McLurkin, Jennifer Smith, James Frankel, David Sotkowitz, David Blau, and Brian Schmidt. Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots. In *AAAI spring symposium: to boldly go where no human-robot team has gone before*, volume 72. Palo Alto, CA, 2006.

[86] John Buck. Synchronous rhythmic flashing of fireflies. ii. *The Quarterly review of biology*, 63 (3):265–289, 1988. ISSN 0033-5770.

[87] Renato E Mirollo and Steven H Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM Journal on Applied Mathematics*, 50(6):1645–1662, 1990. ISSN 0036-1399.

[88] GM Ramırez Ávila, Jean-Luc Guisset, and Jean-Louis Deneubourg. Synchronization in light-controlled oscillators. *Physica D: Nonlinear Phenomena*, 182(3-4):254–273, 2003.

[89] Geoffrey Werner-Allen, Geetika Tewari, Ankit Patel, Matt Welsh, and Radhika Nagpal. Firefly-inspired sensor network synchronicity with realistic radio effects. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 142–153, 2005.

[90] Anders Lyhne Christensen, Rehan OGrady, and Marco Dorigo. From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation*, 13(4):754–766, 2009.

[91] Agata Bareiś, Michał Bareiś, and Christian Bettstetter. Robots that sync and swarm: a proof of concept in ros 2. In *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 98–104. IEEE, 2019.

[92] Iain D Couzin, Jens Krause, Nigel R Franks, and Simon A Levin. Effective leadership and decision-making in animal groups on the move. *Nature*, 433(7025):513–516, 2005.

[93] Herbert G Tanner, Ali Jadbabaie, and George J Pappas. Stable flocking of mobile agents part ii: Dynamic topology. *IEEE International Conference on Decision and Control*, 2003.

[94] Ali E Turgut, Hande Çelikkanat, Fatih Gökçe, and Erol Şahin. Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2(2-4):97–120, 2008.

[95] JE Lennard. On the determination of molecular fields. *Proc. R. Soc. London*, 106:441–477, 1924.

[96] Carlo Pinciroli, Mauro Birattari, Elio Tuci, Marco Dorigo, Marco del Rey Zapatero, Tamas Vinko, and Dario Izzo. Self-organizing and scalable shape formation for a swarm of pico satellites. In *2008 NASA/ESA Conference on Adaptive Hardware and Systems*, pages 57–61. IEEE, 2008.

[97] TJ Pitcher and BL Partridge. Fish school density and volume. *Marine Biology*, 54(4):383–394, 1979.

[98] William L Romey and Alicia R Lamb. Flash expansion threshold in whirligig swarms. *PloS one*, 10(8):e0136467, 2015.

[99] Ugo Lopez, Jacques Gautrais, Iain D Couzin, and Guy Theraulaz. From behavioural analyses to models of collective motion in fish schools. *Interface focus*, 2(6):693–707, 2012.

[100] Sara Arganda, Alfonso Pérez-Escudero, and Gonzalo G de Polavieja. A common rule for decision making in animal collectives across species. *Proceedings of the National Academy of Sciences*, 109(50):20508–20513, 2012.

[101] William Bialek, Andrea Cavagna, Irene Giardina, Thierry Mora, Edmondo Silvestri, Massimiliano Viale, and Aleksandra M Walczak. Statistical mechanics for natural flocks of birds. *Proceedings of the National Academy of Sciences*, 109(13):4786–4791, 2012.

[102] Melvin Gauci, Jianing Chen, Wei Li, Tony J Dodd, and Roderich Groß. Self-organized aggregation without computation. *The International Journal of Robotics Research*, 33(8):1145–1161, 2014.

[103] Melvin Gauci, Jianing Chen, Wei Li, Tony J Dodd, and Roderich Gross. Clustering objects with robots that do not compute. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 421–428, 2014.

[104] Jianing Chen, Melvin Gauci, Wei Li, Andreas Kolling, and Roderich Groß. Occlusion-based cooperative transport with a swarm of miniature mobile robots. *IEEE Transactions on Robotics*, 31(2):307–321, 2015.

[105] TJ Pitcher, AE Magurran, and IJ Winfield. Fish in larger shoals find food faster. *Behavioral Ecology and Sociobiology*, 10(2):149–151, 1982. ISSN 0340-5443.

[106] Steven L Lima. Back to the basics of anti-predatory vigilance: the group-size effect. *Animal Behaviour*, 49(1):11–20, 1995. ISSN 0003-3472.

[107] Gilbert Roberts. Why individual vigilance declines as group size increases. *Animal behaviour*, 51(5):1077–1086, 1996. ISSN 0003-3472.

[108] Carlo Pinciroli and Giovanni Beltrame. Swarm-oriented programming of distributed robot networks. *Computer*, 49(12):32–41, 2016.

[109] Alan A Berryman. The orgins and evolution of predator-prey theory. *Ecology*, 73(5):1530–1535, 1992.

[110] Takehito Yoshida, Laura E Jones, Stephen P Ellner, Gregor F Fussmann, and Nelson G Hairston. Rapid evolution drives ecological dynamics in a predator-prey system. *Nature*, 424 (6946):303–306, 2003.

[111] Thomas M Zaret and J Samuel Suffern. Vertical migration in zooplankton as a predator avoidance mechanism 1. *Limnology and oceanography*, 21(6):804–813, 1976.

[112] Claire L Devereux, Mark J Whittingham, Esteban Fernández-Juricic, Juliet A Vickery, and John R Krebs. Predator detection and avoidance by starlings under differing scenarios of predation risk. *Behavioral Ecology*, 17(2):303–309, 2006.

[113] Tony J Pitcher and Christopher J Wyche. Predator-avoidance behaviours of sand-eel schools: why schools seldom split. In *Predators and prey in fishes*, pages 193–204. Springer, 1983.

[114] Florian Berlinger, Paula Wulkop, and Radhika Nagpal. Self-organized evasive fountain maneuvers with a bioinspired underwater robot collective. *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[115] John R Stowers, Maximilian Hofbauer, Renaud Bastien, Johannes Griessner, Peter Higgins, Sarfarazhussain Farooqui, Ruth M Fischer, Karin Nowikovsky, Wulf Haubensak, Iain D Couzin, et al. Virtual reality for freely moving animals. *Nature methods*, 14(10):995–1002, 2017.

[116] Stefano Marras, Robert S Batty, and Paolo Domenici. Information transfer and antipredator maneuvers in schooling herring. *Adaptive Behavior*, 20(1):44–56, 2012.

[117] SJ Hall, CS Wardle, and DN MacLennan. Predator evasion in a fish school: test of a model for the fountain effect. *Marine biology*, 91(1):143–148, 1986.

[118] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control*, 51(3):401–420, 2006.

[119] Herbert G Tanner, Ali Jadbabaie, and George J Pappas. Stable flocking of mobile agents, part i: Fixed topology. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 2, pages 2010–2015. IEEE, 2003.

[120] Nima Moshtagh and Ali Jadbabaie. Distributed geodesic control laws for flocking of nonholonomic agents. *IEEE Transactions on Automatic Control*, 52(4):681–686, 2007.

[121] H. Min and Z. Wang. Group escape behavior of multiple mobile robot system by mimicking fish schools. In *2010 IEEE International Conference on Robotics and Biomimetics*, pages 320–326, 2010. doi: 10.1109/ROBIO.2010.5723347.

[122] Razvan-Dorel Cioarga, Mihai V Micea, Vladimir Cretu, and Voicu Groza. Evaluation of fish shoal inspired movement in collaborative robotic environments. In *2010 IEEE Instrumentation & Measurement Technology Conference Proceedings*, pages 1539–1544. IEEE, 2010.

[123] Sahithi Podila and Ying Zhu. Animating multiple escape maneuvers for a school of fish. In *Graphics Interface*, pages 140–147, 2017.

[124] Ali Jadbabaie, Jie Lin, and A Stephen Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on automatic control*, 48(6):988–1001, 2003.

[125] François Gerlotto, Sophie Bertrand, Nicolas Bez, and Mariano Gutierrez. Waves of agitation inside anchovy schools observed with multibeam sonar: a way to transmit information in response to predation. *ICES Journal of Marine Science*, 63(8):1405–1417, 2006.

[126] Christian Agrillo, Marco Dadda, Giovanna Serena, and Angelo Bisazza. Do fish count? spontaneous discrimination of quantity in female mosquitofish. *Animal cognition*, 11(3):495–503, 2008.

[127] Florian Berlinger, Melvin Gauci, and Radhika Nagpal. Implicit coordination for 3d underwater collective behaviors in a fish-inspired robot swarm. *Science Robotics*, 6(50), 2021.

[128] Matthew J Westoby, James Brasington, Niel F Glasser, Michael J Hambrey, and Jennifer M Reynolds. 'structure-from-motion'photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179:300–314, 2012.

[129] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.

[130] Changchang Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 127–134. IEEE, 2013.

[131] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. Ieee, 2004.

[132] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE, 2014.

[133] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[134] Florian Berlinger, Mihai Duduta, Hudson Gloria, David Clarke, Radhika Nagpal, and Robert Wood. A modular dielectric elastomer actuator to drive miniature autonomous underwater vehicles. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3429–3435. IEEE, 2018.

[135] Michael Wehner, Ryan L Truby, Daniel J Fitzgerald, Bobak Mosadegh, George M Whitesides, Jennifer A Lewis, and Robert J Wood. An integrated design and fabrication strategy for entirely soft, autonomous robots. *Nature*, 536(7617):451–455, 2016.

[136] Duncan J Watts. The "new" science of networks. *Annu. Rev. Sociol.*, 30:243–270, 2004.

[137] Rama Cont and Jean-Philippe Bouchaud. Herd behavior and aggregate fluctuations in financial markets. *arXiv preprint cond-mat/9712318*, 1997.

[138] Francis Galton. Vox populi, 1907.

[139] Jianguo Liu, Thomas Dietz, Stephen R Carpenter, Marina Alberti, Carl Folke, Emilio Moran, Alice N Pell, Peter Deadman, Timothy Kratz, Jane Lubchenco, et al. Complexity of coupled human and natural systems. *science*, 317(5844):1513–1516, 2007.

[140] Simon Levin, Tasos Xepapadeas, Anne-Sophie Crépin, Jon Norberg, Aart De Zeeuw, Carl Folke, Terry Hughes, Kenneth Arrow, Scott Barrett, Gretchen Daily, et al. Social-ecological systems as complex adaptive systems: modeling and policy implications. *Environment and Development Economics*, 18(2):111–132, 2013.

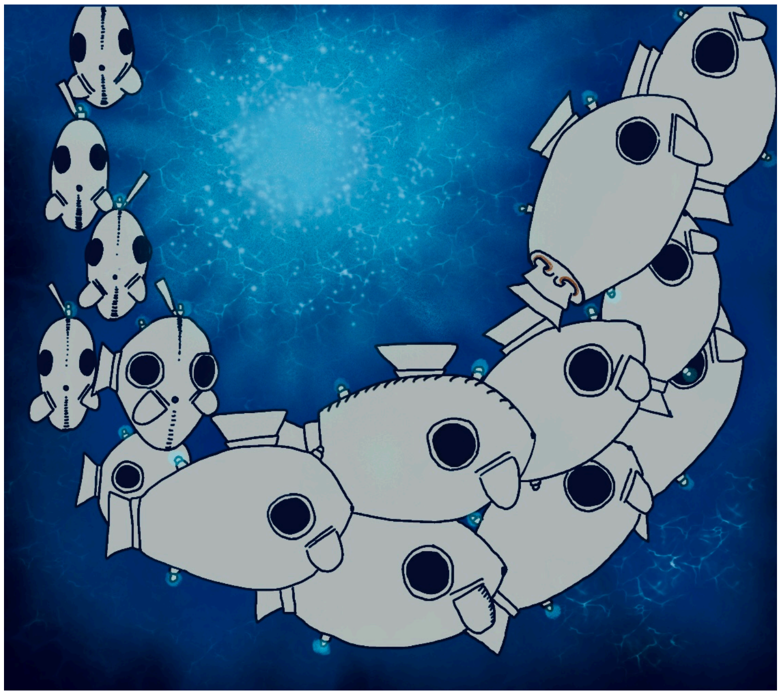
[141] Robert M May, Simon A Levin, and George Sugihara. Ecology for bankers. *Nature*, 451 (7181):893–894, 2008.

[142] M Saadat, Frank E Fish, AG Domel, V Di Santo, GV Lauder, and H Haj-Hariri. On the rules for aquatic locomotion. *Physical Review Fluids*, 2(8):083102, 2017.

[143] Mihai Duduta. *Dielectric Elastomer Actuators as Artificial Muscles for Soft Robotic Applications*. PhD thesis, Harvard University, 2019.

[144] Mihai Duduta, FCJ Berlinger, Radhika Nagpal, DR Clarke, RJ Wood, and F Zeynep Temel. Electrically-latched compliant jumping mechanism based on a dielectric elastomer actuator. *Smart Materials and Structures*, 28(9):09LT01, 2019.

[145] Mihai Duduta, Florian Berlinger, Radhika Nagpal, David R Clarke, Robert J Wood, and F Zeynep Temel. Tunable multi-modal locomotion in soft dielectric elastomer robots. *IEEE Robotics and Automation Letters*, 5(3):3868–3875, 2020.



"Any sufficiently advanced technology is indistinguishable from magic." *Arthur C. Clarke*

And nature, too. I was lucky enough to experience the magical underwater worlds while diving in the Galápagos Islands a year before I started my PhD. Today, I am more fascinated than ever about how schooling fish coordinate their behaviors to seemingly swim like a single entity. In a bid to replicate some of this magic, I built and experimented with Blueswarm. Above illustration by Jessica Paz shows Bluebots milling about, much like the school of barracudas I witnessed in the Galápagos.