Distributed Decision-making Algorithms for Inspection by Autonomous Robot Collectives

HARVARD UNIVERSITY

Graduate School of Arts and Sciences

DISSERTATION ACCEPTANCE CERTIFICATE

The undersigned, appointed by the

Harvard John A. Paulson School of Engineering and Applied Sciences
have examined a dissertation entitled:

"Distributed Decision-making Algorithms for Inspection by Autonomous Robot Collectives"

presented by: Julia Tenis Ebert

Signature _____
Typed name: Professor R. Nagpal

Signature _____
Typed name: Professor R. Wood

Signature _____
Typed name: Professor S. Gil

April 27, 2022

Distributed Decision-making Algorithms for Inspection by Autonomous Robot Collectives

A dissertation presented
by
Julia Tenis Ebert
to
The John A. Paulson School of Engineering and Applied Sciences

in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

in the subject of
Computer Science

Harvard University
Cambridge, Massachusetts
April 2022

Dissertation Advisor: Radhika Nagpal Julia Tenis Ebert

Distributed Decision-making Algorithms for Inspection by Autonomous Robot Collectives

Abstract

Inspection is a ubiquitous challenge, from bridges to farm fields to space stations. These tasks are typically dirty, dull, and dangerous, making them ideal candidates for automation. Researchers have already begun to develop algorithms for robotic inspection, but they are typically limited to a few robots performing planned coverage paths with global communication and centralized computation. This creates a single point of failure and scales poorly for larger groups and environments.

In contrast, non-inspection research in swarm robotics has developed algorithms for large groups of simple robots with limited sensing and communication, with distributed computation. However, many swarm algorithms solve tasks that share essential features with inspection: robots must (1) move through the environment, (2) sense a feature of their environment, and (3) map those observations to a classification. In this dissertation, I focus on closing the gap between inspection tasks and swarm robotics by developing distributed algorithms to solve two types of inspection tasks: global classification of the state of an environment, and locating faults within an environment.

I present two algorithms that allow a group of simulated Kilobot robots to perform binary classification of a black-and-white world and create a committed collective decision. These algorithms can be conducted without localization or coverage, and with low-bandwidth, small range communication. First, I demonstrate a bio-inspired algorithm built on quorum sensing and honey bee waggle dances, which I also extended with a task-switching strategy to classify multiple color features. Second, I show a Bayesian algorithm to solve the single-feature case, which provides a statistically-grounded strategy that incorporates uncertainty by modeling the world as a distribution.

For robotic target localization, I present a hybrid algorithm built on particle swarm optimization (PSO) to allow simulated robots to locate a value below a threshold in a continuous, monochrome world. I introduce a variable update rate to PSO to improve fault detection, and a dispersion-based movement to share information through the group. The robots are able to achieve a detection success rate comparable to coverage, but without needing to visit the whole environment. I also demonstrate that fault detection with a robot swarm can be applied to the real-world problem of space station fault detection. I employ a related PSO-based algorithm that allows soft-bodied Fer-robot robots to detect multiple vibration sources in a physics-based simulation, and demonstrate that the locomotion and vibration detection can be achieved by real robots in microgravity.

As infrastructure ages and robots become more capable, we can employ collective robotics to ensure safety through inspection. This dissertation demonstrates that we can create robust, interpretable inspection algorithms for large groups of simple robots, without relying on centralized computation or planned coverage. It also shows how a complex task such as inspection can be broken down into fundamental swarm behaviors to make a problem easier to solve; this can serve as an example for using robot swarms to solve other complex real-world tasks.

Contents

List of publications

Parts of this dissertation are based on and reproduced from manuscripts which have been published, as described below:

Chapter 4:
Bioinspired Site Inspection with a Minimal Swarm

J. T. Ebert, M. Gauci, and R. Nagpal, "Multi-feature collective decision making in robot swarms," in *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018).* IFAAMAS, 2018, pp. 1711–1719.

Chapter 5:
Bayesian Site Inspection with a Minimal Swarm

J. T. Ebert, M. Gauci, F. Mallmann-Trenn, and R. Nagpal, "Bayes Bots: Collective Bayesian Decision-Making in Decentralized Robot Swarms," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2020).* IEEE, 2020.

Chapter 6:
A Hybrid PSO Algorithm for Multi-robot Target Search and Decision Awareness

J. T. Ebert, F. Berlinger, B. Haghighat, and R. Nagpal, "A Hybrid PSO Algorithm for Multi-robot Target Search and Decision Awareness," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2022).* IEEE, 2022. (In review)

Chapter 7:
Decision-making Applied to Space Station Fault Inspection

B. Haghighat, **J. T. Ebert**, J. Boghaert, A. Ekblaw, and R. Nagpal, "A Swarm Robotic Approach to Inspection of 2.5D Surfaces in Orbit," in *5th International Symposium on Swarm Behavior and Bio-Inspired Robotics (SWARM5),* 2022.

B. Haghighat, J. Boghaert, Z. Minsky-Primus, **J. T. Ebert**, F. Liu, M. Nisser, A. Ekblaw, and R. Nagpal, "An Approach Based on Particle Swarm Optimization for Inspection of Spacecraft Hulls by a Swarm of Miniaturized Robots," in *13th International Conference on Swarm Intelligence (ANTS 2022)*, 2022. (In review)

List of figures

To Mom, who knew I could do this before I did.

Acknowledgments

First, thank you to my family. My dad and my sister, Wesley and Sophie Ebert, have been my lifelong supporters (Sophie and I can continue to talk computers over Dad's head!) Thank you to my mom, Karen Ebert: I wish you were still here to see me reach this milestone. You knew I could get here long before I ever had that confidence, and I know how proud you'd be. My amazing fiancé Clark Teeple has been with me through thick and thin, since we forged our way through AM215 in the first year of our PhDs. Your love and support have been my rock. In addition, the whole Teeple family have become my champions through this process.

I want to thank my PhD committee: Radhika Nagpal, Rob Wood, and Stephanie Gil. Radhika took a chance on me, coming in with enthusiasm for robots, but a background in neuroscience. You gave me the opportunity to develop into a roboticist and an independent researcher. Rob stepped up to help me finish my PhD, becoming my official co-advisor in my last semester and opening his lab to provide a welcoming and supportive community to help me get across the finish line.

I would also not have completed this degree without the support of everyone in the Self-organizing Systems Research group. In particular, Melvin Gauci was the mentor who set me on a path to success in the first years of my PhD, helping to develop my first algorithms, write my first paper, and debug C++ code. Florian Berlinger helped my robots flock and my papers flow. Helen McCreery helped turn gigabytes of data into beautiful graphs and statistics. With Bahar Haghighat, I learned how to flip in microgravity and how to turn high-voltage, high-current systems into robot locomotion (and only causing one EMP in the process). Jordan Kennedy was a source of unwavering support, confidence, and commiseration when times got tough.

I have also been fortunate to have collaborators who have helped make my PhD happen. Gabe Seymour was an REU student who helped to develop the Kilobot simulator, create new outreach activities, and gave me the chance to develop my mentoring skills. Johannes Boghaert was a powerhouse masters student from ETH whose masters thesis made the last chapter of this dissertation possible. I spent a summer at Lawrence Livermore National Laboratory (LLNL), where Michael Schneider's mentorship in the physics department kindled my excitement about space and gave me an immense boost in my software engineering skills. Richard Barnes stepped up to help me turn the Kilosim simulator into what it is today (something that helped me land a job!) Finally, our collaboration with the MIT Media Lab — in particular, Ariel Ekblaw and Fengzhang Liu — gave me the experience of a lifetime to create robots for space stations and test them on parabolic flights.

I have been lucky to have a supportive community that has helped me ride the ups and downs

over the past five years. My friends always had my back — Emma, Neetha, Katherine. Every Friday, I could clear my head on the ice with the Harvard curling team; I couldn't ask for a better crew. As part of the Department of Energy Computational Science Graduate Fellowship (DOE CSGF), I was lucky to be part of a community of like-minded graduate students and alumni ready to help navigate research, advising, and life after grad school; you shared my passion for good research code and helped empower me in my research.

I also never would have even started down this PhD path without the support of mentors before. In my masters at Imperial College, Ildar Farkhatdinov and Etienne Burdet helped me translate my knowledge of brains into skills for robotics research. Before that, my passion for research would never have developed with the support of Dagmar Sternad. I spent four years with Dagmar in the Action Lab at Northeastern, where she would always push me to become a better, stronger, more confident scientist.

1

Introduction

Imagine a space station in orbit: a feat of engineering responsible for supporting the life of its human inhabitants. To do so, the space station must retain structural integrity, free of leaks and defects that could cause a failure of the hull. In the harsh environment of space, the exterior is constantly subjected to potentially damaging debris, and regular inspection can ensure that structure remains safe. Currently, however, this would require sending a human on a space walk around the

station to perform a visual inspection — a dangerous and time-consuming task.

However, an alternative solution is on the horizon: using groups of robots for inspection. On a space station, these robots can operate autonomously while the humans remain safely inside. They can also use additional sensing modalities, such as ultrasound or vibration sensing, and they could perform inspections faster and more frequently than the astronauts. This could catch any faults early, preventing expensive or dangerous damage to the space station. Inspection with robot collectives can make space safer for astronauts, but it also has the potential to ensure safer infrastructure here on Earth in a variety of domains, if we develop the right algorithms to enable intelligent, autonomous robot inspection behaviors.

1.1 Motivation

Inspection is a ubiquitous challenge, from ensuring the safety of a bridge to evaluating a potential construction site. These are examples of, respectively, finding faults within a known space and classifying a relevant feature of an unknown area. These tasks tend to be dirty, dull, and dangerous — the trifecta that makes them ideally suited for robotic automation. As a result, researchers have already begun to develop robotic solutions for a broad range of inspection challenges, such as power lines [1,2], bridges [3], agriculture [4], pipelines [5], and other infrastructure [6]. Using robots for inspection is particularly important in locations that humans cannot reach (e.g., Mars), cannot inspect safely (e.g., hazardous material leaks), or where a task would be particularly time-consuming (e.g., large coverage area).

While robotic inspection covers a broad variety of tasks, all require that the robots (1) move through their environment to (2) sense it, and then (3) map those observations to a classification of the environment. Inspection tasks often require covering large areas, such as a network of power

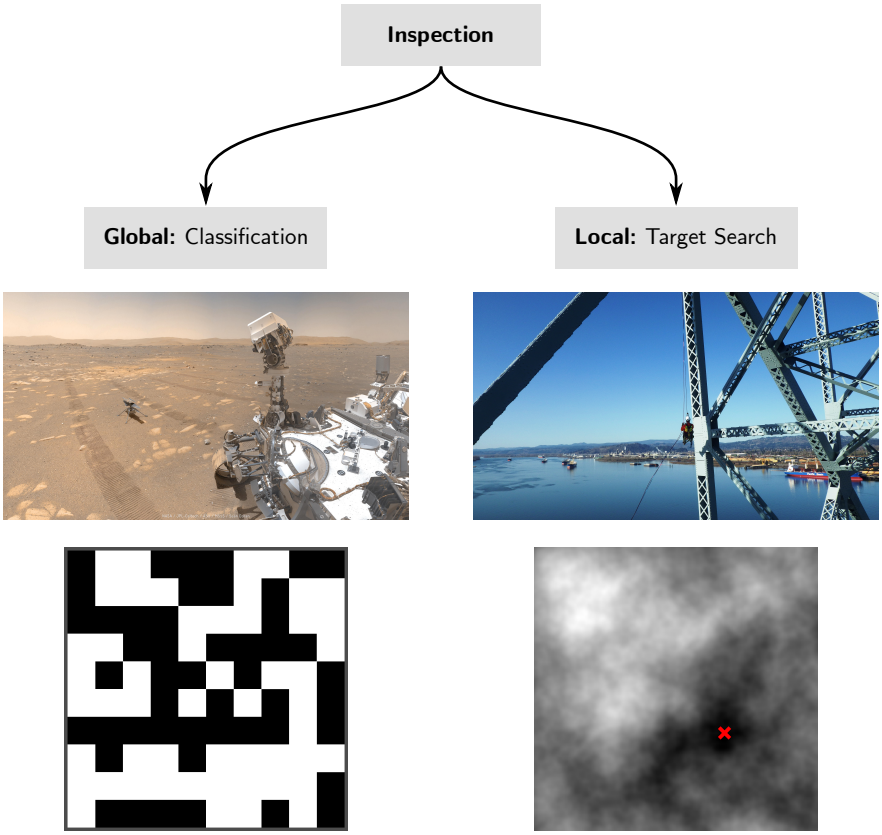


Figure 1.1: Inspection tasks can be broadly categorized as global classification of an environment (left), such as determining the suitability of a habitat site on Mars; or locating a target within an environment (right), such as finding faulty locations on a bridge. In the bottom row, we see abstractions of these problems: classifying an environment as mostly black or mostly white (left), and finding a position with a value below a threshold (right). (Images courtesy of NASA/JPL[7] and the Washington State Department of Transportation[8].)

lines[1,9,10] or a farm field[11,12,13], which are significantly larger than the inspecting robot. Currently, this is typically done using complex, purpose built robots planning a coverage path through the environment, then executing it[14,15,16]. These tasks are completed either by a single robot, or a small group of tightly coordinated robots. Current inspection-focused research also focuses on inspecting for a single feature in the environment, such as identifying cracks, rather than investigating multiple features in a single inspection run.

Using multiple robots can reduce the time to complete an inspection task, but movement coordi-

nation and data ingestion is generally done centrally, rather than distributed among the individual robots [17,18]. This can create a capable collective, but it relies on certain strong assumptions. Namely, all robots must be capable of global communication, as well as precise, coordinated movement to meet the coverage guarantees. In many cases, the target of global coverage may not even be necessary to identify certain types of faults, such as the source of leaks [19,20,21], but there are not currently any standard approaches within the inspection literature that use non-coverage movement algorithms [22,23,15,16,24]. If any member of the group fail — either the central computer or any of the individual — then the inspection task will fail to be completed as planned. A multi-robot system could also facilitates multi-feature inspection, by allowing efficient, dynamic allocation of robots between tasks; this has so previously been investigated in swarm robotics [25,26,27], but thus far it has had less investigation within inspection research [12].

Instead of complex, centralized robotic inspection systems, we can draw on algorithms in the robotics literature developed for non-inspection tasks. Here, we see a swarm robotics approach that can be applied to inspection tasks: large numbers of simple robots with limited sensing, communication range, and communication bandwidth, where computation and decision-making is distributed across all individuals, rather than centralized [28,29]. Many of these approaches are bioinspired, creating decision-making algorithms based on nest site selection in insects [30,31,32,33] or locating optima in an environment with variations on particle swarm optimization (PSO) [34,35,36]. A distributed approach provides certain advantages over a centralized system comprising fewer, more capable individuals; it is more robust to failures of individual robots, parallelization across more robots can speed up inspection, and the simplicity of the individuals makes them simpler to produce [37,38]. However, decentralization introduces challenges for coordinating movement and sharing information, especially when there is no global communication between robots.

In order for a group of robots to collectively inspect their environment, they must also move beyond individual decisions, to reaching a collective decision as a group. Inspection requires that

robots achieve committed, or "locked-in" decisions from the group of robots. However, the current robotic collective decision-making literature commonly aims to achieve only a transient consensus among the group [39,40,41,42]. Collective decision awareness will allow the robots to chain tasks together, such as repairing a fault themselves or returning to a collection point, or communicate the inspection results to a human. There is also typically a speed-accuracy tradeoff in such decision-making: the goal is for all robots to reach the same, accurate decision as quickly as possible.

To solve these problems, I divided inspection tasks into two categories that align well with existing robot algorithm literature, as shown in Fig. 1.1. This division is explained further in Chapter 2. First, there are global inspection tasks, which require classifying a site or environment. This can include determining if a site on Mars is suitable to construct a human habitat, or checking whether a lake is polluted. In each case, the robots classify one of more features of the environment from samples taken while moving through their environment. In many cases, this takes the form of a "go/no-go" problem: each robot must select the best of two possible choices based on some incomplete information available to them. In the case of Martian site inspector, this could mean determining whether the site has stable enough ground and whether there is enough water. Even groups of very simple robots are capable of completing this type of task, without localization and with very small communication ranges. To develop algorithms for site classification, we can abstract this problem to a color-based feature, shown in the bottom left of Fig. 1.1. Here, the robots must determine whether the arena is filled with mostly black or mostly white.

Second, there are local inspection tasks, where the robots' goal is to determine the location of a target within an environment, such as the location of faults on a bridge. These tasks require that the robot sense a feature through the environment and find positions where the value is past a certain threshold, which means the robots also must possess some method of localization within the environment. For bridge inspection, this might mean identifying where too much rust is present or where cracks are too large. The feature under investigation often has a non-convex distribution or

cannot be well-modeled. To guarantee locating the global minimum, the optimal solution is exhaustive coverage[43]. In many real-world situations, though, a location with a value past a threshold can be identified with non-coverage search algorithms. Like the global classification task, this problem can also be abstracted to represent any scalar feature using grayscale, as seen in the bottom right of Fig. 1.1, where robots locate a value close to the global minimum.

In my dissertation, I designed algorithms to solve both the classification and localization classes of inspection algorithms. These algorithms demonstrate a variety of different approaches to solving collective perception and decision-making problems, including an insect- and bacteria-inspired classification algorithm, Bayesian modeling of an environment, and hybrid combinations of existing algorithmic components to create new collective behavior. This work represents an important advance in understanding how to utilize collectives of robots for the real-world challenge of inspection, including the trade-offs involved in both algorithmic and swarm design.

1.2 Contributions

My work makes four main contributions to the area of collective robotic inspection: first, algorithms for global site classification; second, an algorithm for target localization; third, a demonstration of collective robotic space station inspection; and fourth, a simulator for investigating large-scale swarm behavior;

1. **Design algorithms to accomplish single- and multi-feature site evaluation with simple robots.** I present two distributed algorithms for site classification. I developed a bioinspired algorithm based on quorum sensing that allows a group of simple robots to make binary classifications of multiple features of their environment, demonstrated in both simulation and on Kilobot robots. I also created a Bayesian decision-making algorithm for site classification, resulting in faster, statistically-grounded decision-making. This work showed that simple

robots are capable of highly accurate inspection, even at modest densities relative to their environment scale.

2. **Design an algorithm to achieve fault target search and decision awareness.** I present an algorithm to locate a target in an environment with a continually sensed cue. This is representative of many inspection tasks requiring detecting the location of a fault by identifying a location where a feature value is past a safety threshold. My algorithm demonstrates the power of combining algorithm components in a computationally and physically distributed robot system — such as particle swarm optimization, flocking, and dispersion — to achieve collective detection and awareness of a fault target, without requiring coverage or precisely coordinated robot movement.

3. **Demonstrate the potential application of swarm inspection for space station fault detection.** Moving beyond abstract monochrome cases, I collaborated with an MIT-Harvard team to employ the above concepts toward a real-world application scenario: inspecting the exterior surface of the space station. First, we showed how a collective algorithm can be used to identify multiple vibratory faults in a 2.5D physics-based simulation. Second, we validated the simulation assumptions by showing that real soft-bodied robots in microgravity can detect characteristic vibrations and move efficiently on steel surface.

4. **Develop a simulator capable of handling the large scale simulations necessary for algorithm development and analysis.** To facilitate the development of swarm algorithms, I developed Kilosim, an abstracted simulator capable of quickly simulating large groups of arbitrary robots. This simulator was designed for *scale*, simulating up to hundreds of robots, and *speed*, simulating at up to 1000× realtime. This allows for more rapid algorithmic iteration than in physical hardware, as well as detailed study of the impacts of robot features and parameter choice.

Together, these contributions advance the capabilities of robotic inspection: develop new algorithmic approaches to multiple types of inspection tasks while connecting swarm robotics literature to the real-world challenge of inspection.

1.3 Outline

This dissertation is organized into eight chapters:

Chapter 2 — Related Work: Discusses prior work on robotic inspection and relevant algorithmic research on multi-agent decision-making. It also situates collective robotic inspection within the taxonomy of multi-agent behavior.

Chapter 3 — Kilosim: A High-performance Robot Swarm Simulator: Introduces a custom simulator created for studying large multi-agent collectives, including Kilobots. This simulator is used for work through the rest of this dissertation.

Chapter 4 — Bioinspired Site Inspection with a Minimal Swarm: Presents a bioinspired algorithm for allowing a swarm of Kilobot robots to classify multiple binary color features of their environment, including task switching.

Chapter 5 — Bayesian Site Inspection with a Minimal Swarm: Presents a statistically validated Bayesian algorithm for binary environmental classification in simulated Kilobots, demonstrating a performance improvement over the previous bioinspired approach.

Chapter 6 — A Hybrid PSO Algorithm for Multi-robot Target Search and Decision Awareness: Presents a hybrid PSO-based algorithm for locating a fault target within a

monochrome environment and disseminating this information through the group to achieve collective decision awareness.

Chapter 7 — Decision-making Applied to Space Station Fault Inspection: Demonstrates the application of swarm inspection to detecting vibratory faults on a space station surface, including physics-based simulation of multi-fault detection and validation of robot hardware in microgravity.

Chapter 8 — Conclusions and Future Work: Concludes and discusses future work, including further development toward solving real-world inspection tasks.

A robot may not injure a human being, or, through inaction, allow a human being to come to harm.

Isaac Asimov

2

Related Work

Inspection is a pervasive challenge across many domains, from outer space to factory machinery to environmental monitoring, but it has so far received only limited explicit attention in robotics research, particularly for multi-robot approaches. Research thus far has focused on particular applications, such as pipeline or inspection, rather than broad algorithmic understanding of the domain. However, other work within multi-robot algorithms research is not explicitly framed

as inspection, but nevertheless solves similar or related problems and could be applied to inspection. If we understand what characterizes an inspection task, we can then apply and adapt existing algorithms work to inspection problems.

In this chapter, I present robotic inspection research that allows us to define what features make a task an inspection task. With a definition of inspection, I then identify and related work in the multi-robot literature that does not describe itself as solving inspection tasks, but nevertheless solves tasks that meet our definition of inspection. Finally, I situate multi-robot inspection within a broader taxonomic understanding of multi-robot behaviors.

2.1 Defining Robotic Inspection

Inspection tasks vary broadly across many axes — for example: in environment, from agriculture [11] to space stations [44]; in feature modality, from magnetic flux [45] to computer vision-based optical object assessment [4]; and in the detecting agent, from a single fixed sensor to a robot swarm. Therefore, characterizing inspection tasks requires abstracting from the specifics of an individual case and recognizing the features that are consistent across these scenarios, and relevant sub-categorization of types of inspection tasks. In short, we must identify: What features make something a robotic inspection task?

Below, I present three features identified from literature that characterize robotic inspection tasks: sensing, classification, and mobility. I also present a categorization of inspection into two sub-tasks based on the task goal.

2.1.1 Environmental Sensing

The first feature of an robotic inspection task is that it requires the agents to sense at least one feature of the environment being inspected. This follows naturally from the definition of inspection

more broadly, and is observed consistently across the robotic inspection literature.

Different sensing modalities are used in different inspection regimes, depending on the feature under investigation. These are generally non-destructive evaluation (NDE) techniques, which allow for regular investigation without creating damage in the process of testing. Large scale infrastructure, such as bridges or dams, are concerned with the structural integrity of the constituent components. Ultrasound can detect rusting bridge supports with different responses to thick or thin supports[46]. Magnetic flux can detect variations in ferromagnetic materials that indicate faults, such as thinning bridge cable-stays[47] or pipe leaks[45]. Hot spots in power lines can indicate failures due to high resistance[10].

One of the most common inspection sensing modalities across domains is image and video data, since many faults are visible at the surface. This modality has been used for many inspection domains, such as power transmission lines[1], tunnels[48], building construction[49], bridges[50], and pipelines[5]. However, the processing of this image data varies dramatically by domain.

In some cases, multiple sensor modalities can be employed toward a single inspection goal. For example, the BETOSCAN system investigates reinforced concrete using many sensor modalities, including ultrasound, microwaves, optical analysis, and concrete resistance[51]. This is particularly beneficial for systems where there are different constituent components, like the rebar, cement, and aggregate of reinforced concrete. To detect both surface and subsurface faults, robots can combine visual surface inspection with subsurface sensing, like vibration or sound[52].

2.1.2 Mapping Sensing to Classification

What next distinguishes an inspection task is what happens to the sensed data. For inspection, the data is processed into a meaningful output: classifying the quality of the feature. Often, this takes the form of a binary classification: Is this faulty or healthy? However, further classification is possible, such as determining the magnitude of a fault.

The difficulty of mapping sensor data to classifications depends on the type of sensor data, as well as the task complexity. For example, Yu et al. [48] were able to identify cracks in a tunnel wall from camera data, which is often a difficult sensor modality, but the task was simplified by high contrast between cracks and uncracked walls. Temperature sensing on power lines is a one-dimensional signal with clear fault thresholds, resulting in straightforward classification [10]. More complex sensor data results in a more difficult sensor-to-classification mapping, which draws on traditional signal processing techniques, like wavelet decomposition of audio data [52].

More recently, this mapping has been achieved through machine learning techniques. With complex visual scenes, or when sensor fusion is required, traditional sensor fusion and hand-rolled techniques are insufficient and can fail to reach the accuracy of a human inspector. Machine learning has been employed to detect scratches on machine components from photos using naive Bayes classifiers [53], and modern convolutional neural networks (CNNs) were used to identify surface imperfections [54] and multiple types of defects on rail tracks [55].

Note that there are some cases of inspection that meet the first criteria for robotic inspection (robotic sensing), but the assessment of this sensor data is conducted by a human, as in Kawaguchi et al. [56]. We do not consider this to be a fully robotic inspection task, because the inspection is not happening autonomously. Similarly, it is possible for sensor data to be collected by autonomous robots, but it is only processed offline, after the robots have left the space. While this does not require human assessment, it does not represent self-contained robotic inspection.

2.1.3 Robot Mobility

The ability for an agent to sense and classify its environment are essential features of inspection, but this does not distinguish a static sensor or static sensor network from robotic inspection; robotic inspection is differentiated by the autonomous mobility of the agents. Within inspection, mobile sensors and mobile wireless sensor networks (MWSNs) typically meet this requirement, provided

that the sensors locomote, instead of being passively moved by their environment[57]. Here, we do not consider the modality of locomotion, but rather the choice of where and how to move, such as path planning, random walks, or reactive movement.

Coverage path planning (CPP) approaches provide a method to guarantee that an environment or structure has been fully inspected, and can be either pre-computed offline, or online in an unknown environment[22]. Such techniques are typically used to provide a complete reconstruction of an environment for outside assessment, and are evaluated on the completeness of the coverage, rather than evaluating whether the robots visited the most consequential locations within the environment[16]. The most common CPP approach for inspection is conducting a boustrophedon search, or a sweeping zig-zag path through the environment, but this problem is NP-hard in the general case, even before accounting for obstacles in the environment[22,58]. Various approaches to achieve near-optimal coverage have made this problem tractable, such as cellular decompositions[59,60], grid-based environmental models[61], and landmark-based topological coverage[62,63]. However, these remain computationally intensive and performance degrades in the presence of often-inevitable uncertainty in sensing and localization[64,65,23].

There are alternatives to path planning coverage. With time constraints, it is instead possible to create prioritized waypoints and solve a version of the traveling salesman problem[14], or employee ant colony optimization to plan good-enough path[66]. If heuristics are available, it is also possible to employ path planning techniques such as variations on A*; Lai et al.[67] designed center constraint weighted A* (CCWA*) to inspect a petrochemical plant with reduced computation required for such a large environment.

2.2 Categorizing Inspection Tasks

There is a wide variety of tasks that have all three features of robotic inspection described above. For example, a robot may be tasked with locating cracks in a pipe, identifying whether there are insects in a farm field, or detecting high resistance sections of power lines. However, I propose that these varied tasks can be broadly placed into two categories:

- **Global classification:** Categorize an entire site or environment. For example: Is the ground at this site stable?
- **Target search:** Find the location of any faults or sites of interest in a large area. For example: Locate the cracks in a surface.

There is, in fact, a spectrum between these two cases: for example, to inspect and classify sub-regions of a full environment, rather than the extremes of the whole space or a discrete point. However, for the purposes of clarity within this dissertation, I will focus on these distinct cases.

2.2.1 Global Classification Inspection

The goal of *global classification inspection* is to classify an entire site, region, or environment based on one or more spatially distributed features observed within it. Regardless of the distribution of this feature within the area, a single global classification will be reached for the whole area. Note that, for this to be a robotic inspection task, the area must be larger than can observed and classified by a single static; otherwise, it is not a *robotic* inspection task, as there is no mobility involved. For robots to solve this problem, they either need to see the entire site (i.e., coverage), or they need to observe enough of the cite that they are collectively sufficiently confident to make a decision about its state.

There are many types of global site inspection, many of which are not yet conducted by robots, such as assessing topography and climate[68], forest ecology[69], and seismic conditions at building sites[70]. We also observe this type of inspection in insects, where they must assess the quality of a potential nest site[71,72], which has also been applied to robotic systems[32,73]. Thus far, however, explicit robot site inspection exists primarily in the agriculture domain[74,11,75] — for example, to assess rice farm crop quality[76] or fruit yield estimation[77]

2.2.2 Target Search Inspection

In algorithmic research, the term "target search" covers a broad class of problems, including targets that can be dynamic, moving, and intelligent[78]. Here, we use the term to refer specifically to the types of target search that is encountered in inspection-type tasks: where the target is (1) in a fixed location (a fault will not migrate around the structure or environment), and (2) quasi-static (i.e., we do not anticipate significant changes within the duration of a single inspection).

Target search is the predominant form of inspection in the literature, comprising both single- and multi-target searches. For example, Yu et al. [48] sought to identify the locations of any tunnel cracks, and Jiang et al. [10] found the positions of high-resistance, high-temperature points along power transmission lines. It is also possible to locate multiple classes of faults. Inoue et al. [52] employed wavelet decomposition of auditory data to identify two types of faults in tile walls — tile separation and layer separation — as well as locations where both faults were present.

2.3 Multi-robot Inspection Considerations and Possibilities

Using multiple robots for inspection provides additional sensing capabilities, but it introduces additional challenges for all three of the above components of components of inspection: observations and classifications must be communicated throughout the group, and movements of individual

robots must be coordinated. This can be with a centralized controller, or distributed across the individual agents. When many simple robots are used with fully distributed control, this is considered a robot swarm [28,29].

Multi-robot inspection is a form of collective decision-making: robots acquire observations from their individual distributed sensors, fuse this information into a decision, and then communicate the decision across the group to reach a common decision. In the case of centralized control with global communication, these processes can all be conducted by a single computer using all knowledge acquired in the system. This reduces the required algorithmic complexity, but it decreases robustness by introducing a single point of failure and requiring robots to be within communication range of the single central controller for many operations [17,18]. In contrast, decentralized control requires coordination among all individuals, even if they have global communication. However, many simple robots used in swarm-scale applications have limited communication ranges and bandwidth [79,80,81]. This results in cheaper robots with lower energy requirements, but further increases the algorithmic challenge for information sharing and coordination.

When considering mobility in multi-robot inspection, the challenge expands to include coordinating robot movements. This makes offline planned paths — particularly coverage — less reliable, and coordinating online path planning is unreliable if communication is limited. Coverage algorithms have been proposed for multi-robot inspection, but they typically fail to consider the challenges of physical coordination [17] or communication and where the computation occurs [15,24]. They assume that robots are able to precisely follow arbitrary paths and do not consider the failure of coverage if this requirement cannot be met. These algorithms must also either be computed by a central controller with global communication, or additional complexity must be added for these algorithms to be computed decentralized. In either case, their assumptions are not met if global communication is unavailable. As a result, non-coverage and less planned movement (i.e., reactive movement) is more common in multi-robot inspection [82]. It is also possible to combine planned and reactive

movement, as Nejadfard et al. [18] demonstrated by planning one robot's movement with quadratic programming, then coordinating the positions of additional dome-inspecting robots using particle swarm optimization. However, there has thus far been limited research on multi-robot movement that is explicitly presented as inspection.

While research in inspection has generally focused on small groups of capable, centralized robots, research in swarm robotics has been developing distributed algorithms for tasks that are not called inspection, but nevertheless possess all three of the features specified above. Therefore, we can begin to apply distributed swarm algorithms to inspection problems. By developing algorithms for robot swarms, we will have the algorithmic capabilities necessary to successfully deploy simpler, cheaper, more robust robot collectives for inspection.

2.4 Algorithms for Multi-robot Global Classification

The task of global classification often takes the form of a "go/no-go" choice about one or more features of the environment: robots must select the best of two possible choices based on some incomplete information available to them. In swarm robotics, this challenge requires all robots quickly and accurately come to the same decision.

Swarm robotics often draws inspiration from biology for such distributed classification problems, since binary decisions are common in biology. Insect-based algorithms are inherently decentralized and scalable, making them well-suited for robotic collectives. Honeybees and ants are known for house-hunting, in which the entire colony must select between multiple possible new nest sites or risk splitting the colony [30,83,72,71,84,85]. These strategies typically use random pairwise interactions and positive feedback to push the group to a decision, in which higher-quality options are more heavily communicated and more visited, pushing the colony to consensus. In particular, a cross-inhibition mechanism was identified as a key for successful collective decision-making in

bees[86], and ants commonly use stigmergic pheromone trails to drive consensus[87,88].

Several groups have explored robotic decision-making inspired by insect house-hunting. This approach proved effective in a house-hunting task completed by Kilobot robots[31,32]. At any moment in time, each robot has an opinion about which option is best. The robot either explores the option, or exchanges information with its neighbors. In the latter case, the robot locally broadcasts its opinion for a duration that is proportional to the perceived quality of the preferred option, analogous to the honeybee waggle dance. The robot also monitors incoming messages for a fixed time period, and then updates its opinion using the majority rule. The robot then switches to exploring the potentially new option, and the process repeats indefinitely. The results showed that the collective approaches consensus with high accuracy.

Recently, Valentini et al.[41] studied a modified version of the collective decision-making problem on a robotic system, in the form of *feature detection*. The agents are able to sense color in a black-and-white environment and are required to estimate whether the environment contains more back or white area. Each agent makes individual estimates of the feature and shares it globally with all other agents (i.e., communication is global). The agents then aggregate the estimates of other agents to form a belief about which color is more prominent. Several aggregation mechanisms were investigated in simulation and on a group of 20 e-puck robots, resulting in different speed/accuracy trade-offs.

One limitation of the work in[41] is that it relies on global communication, whereas collective systems in nature exploit the use of only local communication to achieve scalability[89]. Moreover, the work in[41] — as well as all the others mentioned above — only investigate the case where the collective is asked to make a single decision. However, collective systems in nature are capable of handling multiple tasks simultaneously, adaptively allocating individuals as required to complete these tasks[87,90,91,92].

Other groups have studied the collective perception and decision-making problem more broadly

as a "best-of-N" task where robots decide between $N \geq 2$ options, emphasizing the techniques used to ensure collective agreement on the result. Parker and Zhang[73] studied a scenario in which a group of robots is expected to choose the best out of a number of unequal options. The robots employ an active recruitment strategy that relies on inter-robot communication. The robots start by looking for options and advocating them to each other, always switching their selection to the best of the known options. Once a robot's selection becomes sufficiently popular (reaching a *quorum*), the robot becomes committed to it. This enables the group to reach a consensus where all robots have locked in decisions. Hamann et al.[33] studied how a homogeneous group of robots can collectively choose between two global maxima in a light-intensity field. In their algorithm, each robot moves in a straight line until it encounters another robot. Then, it stops and counts the total number of robots in its neighborhood. If this is above some threshold, the robot measures the light intensity and waits for a time proportional to this intensity. This creates a positive feedback effect which enables symmetry breaking between the two options.

In inspection tasks, it is not sufficient to simply approach consensus. At some point, the decision must be 'locked in', allowing the collective to move on to the next task; for example, the bees or ants must start emigrating and moving their larvae to the new site. In the context of bacterial colonies, this process is known as quorum sensing[93,94]. When the concentration of an extracellular signaling molecule produced by the bacteria crosses a threshold, the colony can move from stasis to an active state.

In contrast to a bioinspired approach, it is also possible for robots to make a committed decision using statistical models to make probabilistic guarantees about decision accuracy. Bayesian algorithms provide a statistically grounded approach to this challenge, particularly when fusing data from multiple agents in distributed sensor networks and multi-agent systems. Many agents collect samples of information that must be integrated to form a single estimate or decision. Early approaches involved distributed sensors, but decision-making was centralized[95]. More recently,

decentralized Bayesian information fusion has been successful across a variety of domains, such as target tracking[96], source localization[97], self-localization[98], and event classification[99], demonstrating its broad applicability across tasks and domains. While many of these approaches scale across the number of agents[100], they often rely on assumptions such as maintaining a fixed or strongly connected network[101], or their goal is continuous state estimation, rather than a go/no-go decision.

2.5 Algorithms for Multi-robot Fixed Target Search

Search and localization problems are pervasive in robotics, though often not explicitly presented as inspection tasks. One common framing is gas source[102,103,104] and odor source localization[105,106,107]. The framing of these tasks reflects a breakdown of inspection search tasks more broadly, where the problem is viewed as three sub-tasks[108,105,109]: (i) *cue finding,* where the relevant signal (or change in signal) is first detected; (ii) *cue tracing,* where some strategy is used to follow that signal; and (iii) *source declaration,* where a location is determined to be a source (or in inspection, parlance, a fault).

Perhaps the simplest source search strategy tracing a cue with gradient descent[110]; when used in the context of following a chemical trail, this is chemotaxis[111]. In its pure form, this reactive approach is possible only if the signal can be detected at all points in the environment, it distribution is convex, and if the agents are capable of sensing a gradient. In some cases these limitations can be overcome by generating a pseudo-gradient, such as by using multiple sensors[103,112], but can still be thwarted by highly non-convex or sparse signals in the environment.

When a feature can only be sensed intermittently, but can be modeled, this model can be used in a probabilistic search algorithm[105]. One common such strategy is following information gradients to find an odor source using infotaxis[113,114,107] Simple approaches such as a reactive, gradient-based chemotaxis cannot be used because of the complex flow pattern and intermittent sensing of scent particles. Instead, the agent employs a model of the behavior of scent particles to predict that any

location in the environment is the source. Notably in this strategy, the agents do not move directly to the most likely source location, but rather, as the infotaxis name suggests, in the direction that will maximize the expected information gain (or reduction in entropy). This results in wide sweeping behaviors, similar to that observed by moths [115]. However, this approach relies heavily on an accurate model of the environment and particle behavior. If the model diverges too strongly from reality, the success at finding the source drops dramatically. It also requires maintaining and updating a probability distribution for the likelihood that every cell is the source, which scales poorly with larger environments. While this model was first developed for a single agent, it has since been extended to distributed multi-agent variations by adapting agent behavior and source likelihoods based on the intersections of their respective distributions [116,117]. Notably, however, some multi-agent adaptations require that agents require maintaining a connected graph [118]; when robots are sparse or have limited communication range, this is a non-trivial requirement.

When an environment cannot be modeled and is too complex for reactive approaches, this leaves heuristic search strategies. The most common multi-robots heuristic approaches are built on variations particle swarm optimization (PSO). In the original form of PSO, abstract particle agents employ a biased random walk toward their individual and collective best observations, resulting in non-guaranteed convergence at a global optimum [119]. PSO has been applied to real multi-robot systems; [34,120] demonstrated that PSO could be used to identify a source with convex feature distribution, despite limited movement speed and communication range. There are also hybrid algorithms that incorporate PSO into their search, including hybrid ant colony optimization (ACO/PSO), where virtual pheromone deposits augment direct communication [35]; PSO plus fruit fly optimization (MFPSO) to avoid local minima and improve search speed [36]; and adaptive robotics PSO (AR-PSO), which considers obstacle avoidance and a mechanism to escape local minima.

These search strategies typically have one of two possible stopping conditions: (1) when a single robot has located the target, or (2) in PSO, when all robots converge at the target. Without global

communication, the first termination condition does not consider knowledge across the rest of the collective. PSO-based convergence requires all robots to identify the target by travelling to the location, rather than learning indirectly by communication. Without global communication, this limits the reach of the already-converged robots to aid convergence of the remaining robots.

To disseminate knowledge of the target without requiring convergence, robots could travel as a connected network. This can be accomplished with distributed spanning trees [121] or creating a k-connected network to maintain at least k neighbors [122]. Both of these algorithms can guarantee maintaining a network, but they are computationally heavy and unrealistic to implement on simple robots. A simpler alternative is Boids flocking [123], where robots create a flock with a target neighbor distance smaller than their communication range. This creates a looser, easier to implement network, though without guarantees.

2.6 Inspection within Multi-agent Behavior Taxonomy

As research and applications of multi-robot systems have grown, researchers have endeavored to develop a systematic, hierarchical classification of their fundamental behaviors. This evolving understanding of multi-agent behavior provides a lens through which to structure and compare varied framings of similar behaviors. Understanding the basic underlying behaviors allows for more intentional design of complex, application-oriented behaviors such as inspection and repair tasks. Multiple iterations of swarm behavior taxonomy have been proposed [26,81,89], generally building on the prior versions. The most recent version, proposed by Schranz et al. [81] in 2020, is shown in an adapted form in Fig. 2.1.

Within the taxonomical framework, it becomes apparent that inspection is a complex task comprised of multiple behaviors. Noted in green in this figure, inspection requires both collective exploration and collective perception — that is, they must move through the environment to collectively

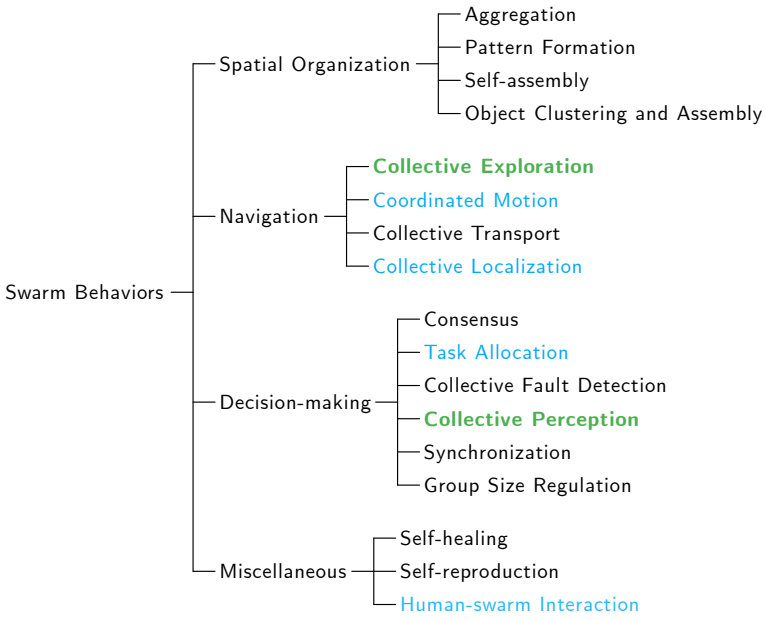


Figure 2.1: Taxonomy of swarm behaviors, adapted from Brambilla et al. [89], Schranz et al. [81]. Behaviors essential for collective inspection are in bold green. Behaviors that could be applicable to collective inspection in some contexts are in blue.

make their observations, and then combine those observations into a larger decision. By recognizing that inspection is composed of multiple types of behaviors, we can draw on the algorithmic literature from these areas to develop new approaches to solving inspection tasks.

In many cases, however, additional swarm behaviors from this taxonomy may improve inspection, or be necessary for certain inspection tasks; these are noted in blue in Fig. 2.1. Typically, a real-world inspection task will begin with a problem, then look for tools to solve it; this taxonomy can be used to identify sub-problems within an inspection task, then look for existing approaches in the literature that solve these sub-problems, rather than creating a new solution from scratch for every new inspection problem.

For example, it may be necessary for a human to command a robot swarm to inspect, or for robots to report their findings to a human for further action to be taken (e.g., to repair a discovered

fault or report that no faults were found). This is a straightforward form of *human-swarm interaction.* In Chapter 4, we demonstrate how *task allocation* can allow a swarm to conduct multi-feature inspection. In Chapter 6, we show how flocking, a form of *collective movement*, affects propagation of information through a robot swarm. If the task goal is a target search problem (as defined in Section 2.5) but the robots do not natively have a coordinate system, *collective localization* allows the group to establish a local coordinate system to establish target locations.

A behavioral taxonomy provides a framework by which we can break down complex inspection tasks into fundamental behaviors, then identify how to leverage existing solutions toward the larger task goal. Within these categories, it is also possible to develop new behavioral solutions designed for inspection, but which can in turn be leveraged for other swarm behaviors. The swarm behavior taxonomy provides a modular framework through which to view complex behavior, potentially simplifying the development of algorithmic approaches to a variety of real-world swarm robotic application.

To me programming is more than an important practical art. It is also a gigantic undertaking in the foundations of knowledge.

Grace Hopper

3

Kilosim: A High-performance Robot Swarm Simulator

Implementing collective robotic algorithms in physical robots is important to understand all of the nuances of their behavior, and how they operate in the context of real-world noise and variability that cannot easily be modeled. However, developing complex algorithms exclusively

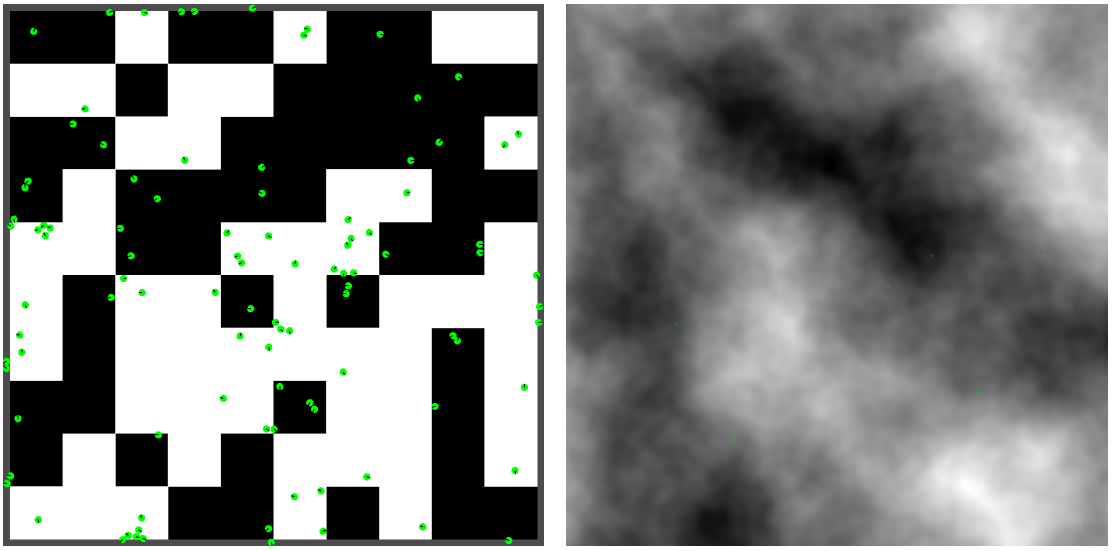
on physical hardware is difficult: you are limited to operating in realtime, subject to breakdowns and battery limitations of the hardware, and cannot parallelize the workload beyond the number of robots that you have. Therefore, it is valuable to have a simulation platform that can simplify and speed up algorithm development by overcoming these limitations. A simulation platform can also provide the algorithm developer additional insight through visualization, detailed logging, and additional debugging information not available on the respective hardware platform. With any simulation platform, however, there is an abstraction tradeoff between the complexity and accuracy of the simulation versus speed, simplicity, and ease of development.

I present Kilosim [124], a 2D robot simulator developed to quickly simulate large groups of simple robots, seen in Fig. 3.1. I developed this simulator to provide fast, high abstraction simulation of physical robots like the Kilobots [80] — which are used as a model platform throughout this thesis — and abstract robot representations, such as the common abstraction of a robot moving on a grid — which is used in Chapter 6. This allows for large-scale simulations, on the order of hundreds of thousands of trials, to thoroughly investigate an algorithmic parameter space. In addition, this open source simulator is modularly designed to allow users to easily implement and integrate their own robot models to gain the performance benefits of the simulation platform.

This simulator was used for all the simulations conducted in Chapters 4–6, enabling large-scale parameter sweeps and many trials to produce robust results.

3.1 Comparison to Existing Simulation Platforms

A multitude of robotic simulators have been created for various application domains and levels of abstraction. Of particular interest here are simulators used particularly for multi-robot applications. Some such simulators are highly specific to a particular application domain, such as the Übersim multi-robot soccer simulator [125] or the MARS simulator for marine swarm robotics [126]. It is also



(a) Kilobots **(b)** Gridbots

Figure 3.1: Viewer output of Kilobot and Gridbot robots in simulated Kilosim environments.

possible to create simulators developed exclusively for a particular robot hardware platform, like Bluesim for Bluebots [127], or Kilombo [128] and the Kilobot Gym [129] for Kilobots . These tend to be designed by researchers for a narrow research scope. They often received little attention or use beyond the group in which they were developed, or the extent to which the hardware platform is used. In many cases, these are built as layers on top of existing physics or game engines like Bullet [130] and Unity [131,132]. This simplifies development for researchers by providing pre-optimized libraries for the most computational intensive portions of simulator development, such as physics and graphics.

On the other end of the spectrum are powerful, widely adopted, general purpose robot simulators such as Webots [133], Gazebo [134], V-REP [135], and more recently NVIDIA's Isaac Sim [136]. These simulators implement a complete 3D physics engine and natively support a wide variety of available robots, with the ability for users to implement their own robot models. This allows for a small simulation-to-reality gap, and the large user base and professional development make these reliable choices. However, this simulation fidelty comes at a cost; namely, while the physics modeling comes

with a large time cost, limiting their usability for large collectives and high throughput simulations, such as parameter sweeps. These simulators also often have high hardware requirements, like Isaac Sim's requirements of an NVIDIA RTX graphics card and 64 GB of RAM. In comparison to Kilosim, it is notable that Pitonakova et al. [137] found that V-REP could not feasibly complete simulations with 50 robots, even for small scenes. In general, performing experiments in these complex 3D simulators with more than dozens of robots reduces the performance to less than realtime [138]. However, most of these simulators are commercially supported products, or are have a large development team, which increases the reliability and feature set of the simulator products.

Between these extremes simulators targeted at particular application domains. Most notable for our purposes are Enki [139] and ARGoS [138], both of which are targeted at swarm simulation. Enki is a 2D simulator that comes packaged with five robot models, with the ability for user to implement their own, and computes basic collision physics. ARGoS is capable of 2D or 3D simulation, and is designed around the requirements of flexibility (implementing new robots and sensors) and runtime efficiency. In comparison to V-REP and Gazebo, Pitonakova et al. [137] found that ARGoS performed fastest with the largest group of robots tested (50 robots). ARGoS achieves this by dividing the environment to perform highly parallelized physics computations, allowing it to perform 2D simulations even with 10,000 e-puck robots.

3.2 Simulator Architecture

Kilosim is composed of modular classes built around the `World`. A simplified Unified Modeling Language (UML) diagram of the simulator architecture can be seen in Fig. 3.2. All code is written in C++11.

A user implements their code by developing a `main` file and function, then importing and instantiating the components of the Kilosim libraries. All simulations are built around a single in-

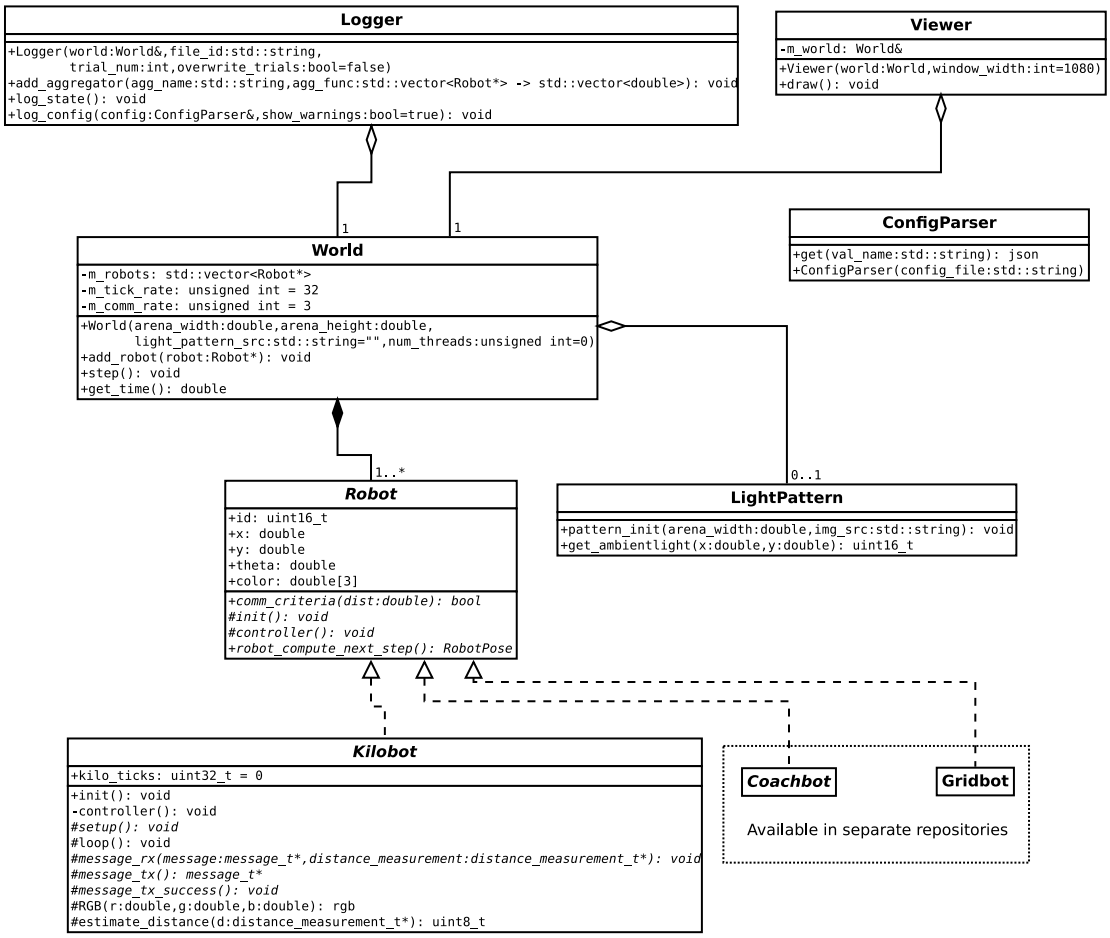


Figure 3.2: UML diagram of the Kilosim software architecture. For brevity, only the most relevant attributes and operations are included.

stance of the `World` class; the `World` contains all the state information for the simulation and is responsible for performing all the discrete step operations to iterate through the simulation, including communication, movement, sensing, collision handling, and calling robot controller functions. When a user creates the `World`, or after creation, they can optionally include a `LightPattern`, which provides an image to map onto the 2D environment surface. This creates values for robots to sense. Colors can also be used to represent obstacles, depending on the robot controller implementation.

After creating a `World`, `Robot` objects can be instantiated and added to the World. Particular robot models are discussed further in Section 3.4. Kilosim provides the abstract base class `Robot` as an interface of the required functions to interface with the `World`. Multiple different robot models can be used in a single simulation. Each robot class provides functions for: determining whether it can communicate with other robots (`comm_criteria`, primarily distance-based); an `init` function that is called when the `Robot` is placed in the `World`; and a `controller` that represents the physical operations of the robot, such as motor control and message handling. The robot model serves as an abstraction layer between the simulator and the user code; it frees the user to write algorithmic implementation details without handling the low-level robot operations.

Kilosim also provides a `Viewer` for visualizing the behavior of robots. Robots also have an interface to dynamically set their color for visualizing behavior in the Viewer. The Viewer uses the Simple and Fast Multimedia Library (SFML)[140] to provide an interface for viewing the robot behavior. This GUI does introduce significant computational overhead, and is therefore recommended only for development purposes.

Kilosim contains `ConfigParser`, a configuration parser for loading configuration data from JSON files. Configuration data can be loaded with a single function, without the user needing to directly interface with the complexities and varying data types of JSON content. It also prevents hard-coding of parameter information into the user's compiled C++ code. This simplifies the process of creating and running different experimental conditions.

Finally, Kilosim provides a flexible and feature-rich logging utility through the `Logger`; this is an essential research tool for producing comprehensive and interpretable data, with safeguards to prevent overwriting data. The Logger saves data in the hierarchical data format HDF5[141], which provides a folder-like structure within a file. HDF5 was selected for its language-agnostic file format (in contrast to language-specific pickled files) and its portable, self-contained nature (in contrast to a database). Its binary representation prevents loss of precision that comes with text-based encoding

Algorithm 3.1 Pseudo-code of a simulation step of the World in Kilosim: `World.step()`

```
for all robots do
    Run robot controllers (motor and battery state)
    Run user-implemented robot code
for all robots do
    Communicate between robots
for all robots do
    Compute next positions, neglecting collisions
for all robots do
    Find collisions, between robots and boundaries
for all robots do
    Move robots, accounting for collisions
Increment time
```

like CSV files, and the hierarchical nature allows for easy saving of metadata and parameters with the simulation logs. A single function (`log_config`) will save all configuration metadata in an experiment log file. To log simulation data, a user creates an aggregator function, which maps from a list of robots to a vector of data, which typically contains some state information of the robots. For example, this can be used to log pose and orientation of the robots, or summary statistics like the eigenvalues of the robots' connectivity matrix. Each of these aggregator functions is added to the Logger once with `add_aggregator` . Thereafter, a simulation loop only needs to call the function `log_state` to save all time-stamped data from the robots.

3.3 Performant Implementation

Various considerations were made in the design and implementation of Kilosim to improve its performance. This includes design choices about the abstractions, as well as the structure of the code itself.

The most significant impact on performance comes from the design of the `step` in the World,

as this is where nearly all of the computation time is spent; this can be seen in Algorithm 3.1. This is divided into separate loops over the robots that allow for parallelization. Because all computations must be computed over all n robots, these operations cannot be faster than $O(n)$. However, it is notable that we separate the computation of the next positions (line 7) from collisions (line 9). Because there can be collisions between robots, this allows for pairwise comparison of all potentially-colliding new robot positions. First, the computation of all new robot positions is parallelized across threads by OpenMP [142]. Then, the collisions between these new robot positions and any boundaries or obstacles are also computed in parallel, and the collision information is used for a parallelized update of robot poses. When running a single instance of Kilosim, this parallelization results in significant performance increases, dependent on the number of cores available in the computer used. However, having an eight core computer does not result in $8\times$ faster performance, due to the overhead of dividing and merging the tasks, as well as non-parallelized components of the simulation. When running many simulations, such as when conducting a parameter sweep, performance is best when running multiple single-threaded instances.

Because of pairwise interactions between robots, a naive approach to implementing collision detection and communication would result in $O(n^2)$ scaling with the number of robots. This is problematic when our goal is to simulate large groups of robots. Instead, we reduce this complexity by creating small spatial bins around the robot into which robots are placed. For collisions, these bins are $2\times$ the robot radius; for communication, this is $2\times$ the communication radius. Robots then only need to check for collisions and communication very locally, rather than against all other robots, and the complexity will scale linearly with the environment size and collisions will scale linearly the number of robots.

In many simulators, one of the most computationally taxing components is realistic physics. To circumvent this issue, Kilosim is designed without a physics engine; instead, it uses *pseudo-physics*. Using accurate physics models is important for simulators that aim to provide a highly accurate

representation of the physical world. Kilosim, however, is designed to simulate a higher level of abstraction, providing a sufficiently accurate physical representation that algorithmic trends reflect the real world, at a lower computational cost than a physics engine. Therefore, Kilosim's pseudo-physics provide accurate detection of collisions, with a simple implementation of collision responses. In the default collision response model, robots that collide are assumed to be circular, and they turn slowly in a random direction. This arises from the behavior observed in collisions of real Kilobot robots: they slowly slide against each other and collisions eventually naturally break apart. Consistent with the modular design of Kilosim, it is also possible to disable collisions (i.e., allow robots to intersect) and for users to implement their own behavior in response to collisions within a `Robot` model.

3.4 Robot Models

Currently, Kilosim comes packaged with a single robot model implementation: Kilobots. However, I have also implemented models for two additional types of robots, which are available in separate repositories. To be used by Kilosim, robot models must implement the abstract methods of the `Robot` base class, as seen in Fig. 3.2. Namely, the `init`, `controller`, and `comm_criteria` functions. In addition, any robot API functions will be defined within the robot model's class.

3.4.1 Kilobots

Kilobots have been a popular robot platform for swarm behavior since shortly after their inception in 2013. However, conducting large-scale simulations with these robots is difficult: they have limited battery life, noisy movement, and debugging can only be conducted by a single RGB LED and a wired serial connection. An appropriate simulation platform can therefore significantly speed algorithm development for this platform. In Fig. 3.1a, we can see what these simulated Kilobots look like within Kilosim.

The Kilobot model within Kilosim is well-suited for Kilobot development for a three primary reasons. First, it fully implements the Kilobot Library API (Kilolib)[143], providing the full complement of capabilities available on the Kilobot robot, including ambient light sensing. Because Kilosim is built in C++, it is also possible to develop code in Kilosim that will directly transfer to the Kilobots, for which code is written in C.

In addition, we have also validated that the behavior of Kilosim Kilobots qualitatively matches the behavior of physical Kilobot robots. This can be seen in Chapter 4, where our bio-inspired classification algorithm is able to achieve equivalent success in simulation and hardware. The simulation resulted in significantly faster task completion times than were observed in the physical robots. This is likely due to the simplification of collision physics, as well as less consistent straight movement of the physical robots on slightly uneven surfaces. In Chapter 5, Kilosim Kilobots were used to conduct a parameter sweep requiring over 700,000 individual simulation trials with 100 robots. If conducted at sequentially at realtime speed, this would have required over 230 years to complete. Instead, Kilosim was able to complete the trials at over 700× realtime, parallelized over 32 threads, to finish in under four days.

We also validated Kilosim by simulating pattern formation with morphogenesis[144]. This behavior was achieved on physical Kilobots by Slavkov et al. [145], and we were able to replicate the behavior in Kilosim without algorithmic changes[146].

3.4.2 Gridbots

A grid-based world is a common abstraction within artificial intelligence and simulated robotics. As such, it is a logical robot model to implement within Kilosim. In the typical grid world, the environment is a rectangular grid of cells, some of which may be blocked as obstacles, that operates in discrete time ticks. A robot within the grid occupies a single cell, and can move to neighboring cells (either in the four cardinal directions or to any of the eight neighbors). Robots may or may not be

able to occupy the same cell, depending on the scenario.

We developed an implementation of this model in Kilosim, called Gridbots[147]. This model is abstracted to allow for either type of movement described above, as well as parameterized communication range to allow for investigating the effect of this parameter. In addition, it provides a user API to specify movements to neighboring cells, set a path to a target location, access location data, and sense RGB colors within their environment. While this seems a simple abstraction, the same scaling challenges occur as with more complex robot models. In fact, a naive Python implementation achieved performance almost 100× slower than an equivalent representation within Kilosim[148].

The Gridbots robot model was used in Chapter 6 of this dissertation, for which we were able to conduct over 500,000 simulations in Kilosim. These included groups with up to 64 robots and global communication. In Fig. 3.1b, we see an example of these Gridbots in a 384 × 384 cell grid environment.

3.4.3 Coachbots

In addition to implementing the robot models used in this dissertation, I also created a simulated representation of the Coachbot robots[149,150]. This collective of 100 robots, developed at Northwestern University, was created to study behavior that could be achieved by more capable hardware than found in the Kilobots. Namely, these robots possess: greater computational abilities, powered by a Raspberry Pi computer; localization using the HTC Vive lighthouse; high bandwidth global communication over Wi-Fi; and faster, more accurate movement with differential drive wheels.

All of these Coachbot capabilities can be implemented within the Kilosim robot framework, while providing the existing Coachbot API to the developer in a `Coachbot` class. The complete implementation is available on GitHub[151]. Although we have not yet used this particular model for algorithm development, its implementation demonstrates the flexibility of the Kilosim framework to easily implement additional robot models. For example, this model differs from Kilobots in that

its movement is driven by differential drive of a pair of wheels; this is accounted for by defining the robot's `robot_compute_next_step` to be determined by the forward kinematics of the motor configuration. In addition, the Coachbot maintains a queue of string-based JSON-like messages that can be checked asynchronously. Because Kilosim uses void pointers to transmit messages, any message type can be used, as long as the receiver can correctly dereference the pointer. In addition, this message queue can be stored within the Coachbot class, and the end user of the robot model does not need to perform any message-handling beyond the access that is defined in the Coachbot API.

3.5 Conclusion

In this chapter I have presented an abstracted, high-performance robot simulator capable of high-throughput simulations of a variety of robot models. I have demonstrated that Kilosim is able to model existing swarm robot platforms like Kilobots and Coachbots, at up to hundreds of times realtime, while providing a modular interface for creating additional robot models.

While Kilosim lacks the physical fidelity of general purpose simulators like Gazebo or Isaac Sim, it can be run quickly on a lower-specification consumer computer with up to hundreds of robots. Its modular design and extensibility also makes is more general-purpose than some research simulators that focus only on a specific hardware model and application. In addition, its inclusion of a viewer, configuration parser, and powerful logging tools make Kilosim a powerful research tool for algorithm development, debugging, and data collection.

When you have a single large robot that does everything, if that robot breaks down, you lose your ability to do anything.

Kelly & Zach Weinersmith

4

Bioinspired Site Inspection with a Minimal Swarm

Collective systems in nature are ones in which a large number of relatively simple agents interact with each other to produce complex behaviors[152]. Decision-making is a key behavior that appears across many of these systems and has been extensively studied over the past few decades[153]. In

general, the agents are required to choose one out of multiple options present in their environment. They have some means of obtaining noisy information about these options and of influencing each other directly through communication or indirectly through stigmergy to achieve consensus on a single decision [154]. This is the same type of problem we aim to solve with robotic collectives in site inspection: determining the quality of a site through observations, and mapping this to a collective decision about the state or quality of the site.

In both natural and robotic systems, it is also common to require observing multiple features of the environment to determine its quality. This typically requires adaptively allocating individuals to complete these sub-tasks [87,90,91,92]. Valentini et al. [41] presented a problem in which a group of robots was tasked with classifying a single feature of their environment: whether the world was mostly black or mostly white. However, robots were able to move into and out of a transient consensus, rather than making the type of committed decision we require for inspection tasks. We add this commitment, extend the problem to investigate *multiple* features of the environment, and introduce communication constraints that are more representative of biological and simple robot systems.

In this work, we investigate the established *single-feature decision-making problem*, and we also introduce and investigate the *multi-feature collective decision-making problem* for inspection. We present a scenario in which the collective must make decisions about either one or three color features in an environment using only local communication. The single-feature problem extends the scenario in [41] by introducing goal of making committed decisions, as well as solving the task with more limited communication. In the multi-feature task, we also demonstrate a decision-making rule and a dynamic task allocation strategy that allow the agents to lock in decisions in finite time.

This work was conducted in collaboration with Dr. Melvin Gauci. It was previously published at the 2018 conference on Autonomous Agents and Multiagent Systems (AAMAS) [155].

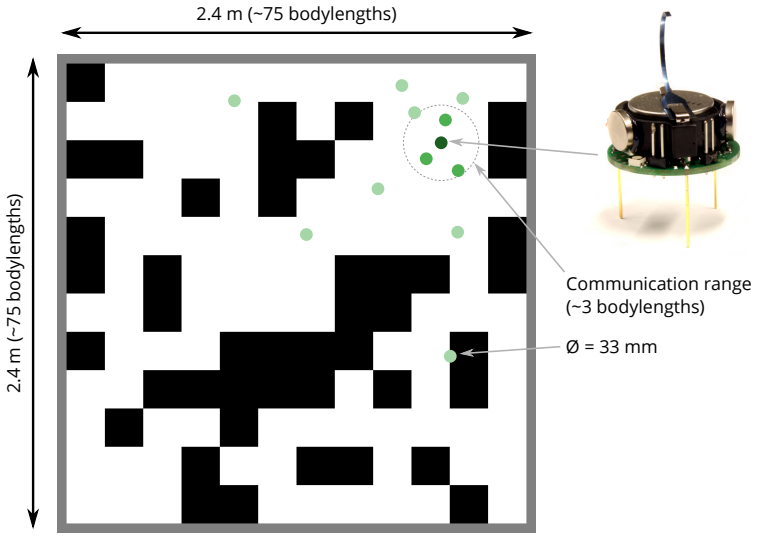


Figure 4.1: An example of Kilobots in a 2.4×2.4 m (75×75 bodylengths) arena. Kilobots can communicate to others within a radius of approximately 3 body lengths.

4.1 Problem Definition and Motivation

We consider N agents that move in a 2-dimensional environment, with boundaries that are detectable by the agents. The environment contains M features that individual agents have a means of estimating. The features may either be inherently binary-valued, or else the agents must have some agreed-upon threshold for making them as such. The features can thus be represented as functions:

$$f_i : \mathcal{S} \to \{0, 1\}, \quad i \in \{1, 2, \ldots, M\} \tag{4.1}$$

where $\mathcal{S}$ is the environment. We can group the features into a vector-valued function over $\mathcal{S}$:

$$\mathbf{f}(\mathcal{S}) = [f_1(\mathcal{S}), f_2(\mathcal{S}), \ldots, f_M(\mathcal{S})]^T \tag{4.2}$$

The features are defined over the environment *as a whole*. Examples of features include the ratio

of white area to the total area in a black-and-white environment (as in [41]), the entropy of a pattern, and the amount of curvature present in a pattern. It is assumed that the agents have some means of making (noisy) estimates of the features. Considering the above examples, the color fill ratio feature can be estimated by calculating the ratio observed over a random walk; the entropy ratio might be estimated by calculating the amount of regularity observed over a straight-line motion; and the curvature might be estimated by performing edge following whenever an edge is detected, and calculating the average radius of curvature. In this work, we focus on features of the type of the first example discussed above (color fill ratio). Specifically, the feature is *locally defined at every point* (x, y) in the environment, mapping the point onto a value in $\{0, 1\}$. The binary value of the feature *over the whole environment* is defined as the rounded value of the fill ratio:

$$f_i(\mathcal{S}) = \text{round}\left\{ \frac{1}{|\mathcal{S}|} \iint_S f_i(x, y)\, dA \right\}, \tag{4.3}$$

where $|\mathcal{S}|$ is the area of the environment and $\text{round}(*)$ is the usual rounding function, outputting 0 if $* < 0.5$, and 1 otherwise. In other words, $f_i(\mathcal{S})$ assumes whichever one of the two values is more prevalent in the environment. Obtaining estimates for features of this type is straightforward: the agent simply samples a subset of points over the course of a random walk.

The agents are limited in what actions they may take. Each agent is capable of estimating every feature in the environment, but is restricted to estimating only one feature during each *observation period* (defined later in Sec. 4.3.1). This limitation is mainly imposed because in general, estimating different features may require different motion patterns; moreover, in practice the computational power available on simple physical agents may not be enough to sense and process all the features at once.

The agents are able to communicate with each other. They may listen to messages from other agents continuously, but are only allowed to transmit messages while they are not observing the en-

vironment. Once again, this limitation is imposed because of the different motion patterns that may be required to estimate the different features; for example, if agents were to disseminate while edge following, this might bias their transmission towards other agents that happen to be performing the same motion pattern in the same region, and so a random walk would be a more desirable motion pattern during transmission.

The problem for the collective is to decide on the value of each feature over the environment as a whole; in other words, to compute Eq. 4.2. The collective is required to not only have an estimate that converges to Eq. 4.2 over time, but also to make a unanimous decision in finite time. When collective decision-making is used as a primitive component in a composite behavior, this allows the collective to move on to the next action.

4.2 Experimental Methods

4.2.1 Agent Model: The Kilobot Robot

We use the Kilobot robot[80] as a basis for our agents. The Kilobot, shown in Fig. 4.1, is a miniature mobile robot with a circular body of diameter 33 mm, developed specifically for use in collectives.

Motion The Kilobot is capable of noisy locomotion in a straight line at approximately 1 bodylength/s and turning on the spot in both directions, completing a full turn in approximately 10 s. Kilobots are individually calibrated for motion, but in practice straight-line motion is inaccurate over long ranges, and both straight-line and turning speeds exhibit significant variation among units.

Communication Kilobots can communicate with each other locally, transmitting to and receiving from neighbors within a 3 body length radius. Each robot is assigned a unique ID at initialization, allowing their messages to be identified by receivers. They are capable of communicating at

a rate of up to 10 messages/s, although this rate decreases at higher robot densities.

LIGHT SENSING The only environmental sensor on the Kilobot is an ambient light sensor. Therefore, in our setup we use light to represent the environmental features to be estimated.

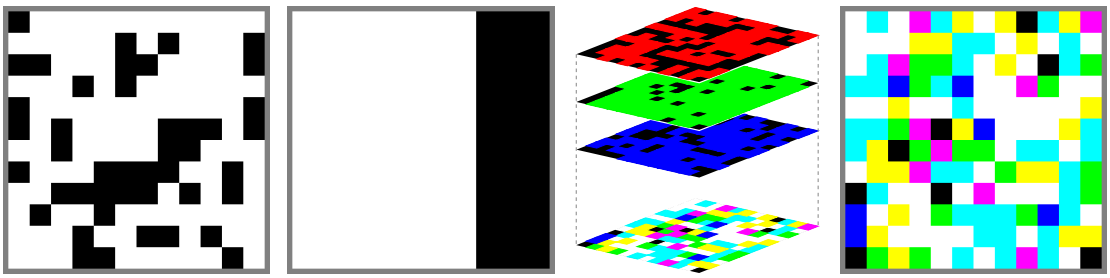


Figure 4.2: Far left: Single-feature homogeneous distribution with a 0.7 fill ratio (i.e., proportion of white cells). **Middle left:** Single-feature non-homogeneous distribution with a 0.7 fill ratio. **Middle right:** Generation of a multi-feature environment by overlaying 3 single-feature distributions. **Far right:** multi-feature environment with RGB fill ratios (0.55, 0.8, 0.65). Colors are combined according to the standard RGB color model.

4.2.2 Simulation Platform

To perform experiments in simulation, we used the Kilosim simulator described in Chapter 3. The ambient light sensor is emulated by directly feeding the Kilobot the light intensity at its position, and for single-feature estimation, we use black and white (i.e., minimum and maximum brightness) areas to represent the two feature values. For multi-feature estimation, we virtually extended the light sensor to detect three different colors: red, green, and blue (see Fig. 4.2). All simulations were conducted in a 2.4×2.4 m environment (approximately 75×75 bodylengths). This environment is padded by a 50 mm thick border with a gray light value, such that the agents have a means of knowing then they have left the environment.

4.2.3 Physical Platform

Physical Kilobot robots move using two vibration motors, based on the principle of stick-slip locomotion. Their communication channel is implemented using an infrared transceiver located at the bottom of each robot. Channel sharing is achieved using a CSMA/CD protocol. The light sensor on the physical robots reports the ambient light intensity with a 10-bit resolution.

Physical experiments were run on a whiteboard surface of 1.2×1.2 m An overhead projector was mounted over the surface, which allowed us to project a light pattern onto the surface. The Kilobot's light sensor is sensitive enough to distinguish three levels of brightness: we used two levels to implement the pattern (black and white), and one to define the boundary of the arena (gray). The thresholds for distinguishing light levels were automatically calibrated for each robot before each experiment. An overhead camera was used to record the experiments.

4.3 Single-Feature Decision-Making

4.3.1 Algorithm

The goal of the single-feature algorithm is for the collective of agents to combine their noisy estimates of a binary-valued environment feature, arrive at a consensus over its value, and lock in a final decision in finite time; that is, 100% of the agents must agree on the same answer. Our algorithm consists of five components that we detail below. An overview of the agent behavior is shown in Fig. 4.3. In the following description, we represent the binary-valued feature in terms of colors, with black and white corresponding to the values 0 and 1, respectively.

1. Individual Motion For the duration of the experiment, agents move in a random walk, with a straight component drawn from an exponential distribution with a mean of 240 s, followed by an on-the-spot rotation sampled uniformly from $[-\pi, \pi)$ rad. If an agent enters the gray border

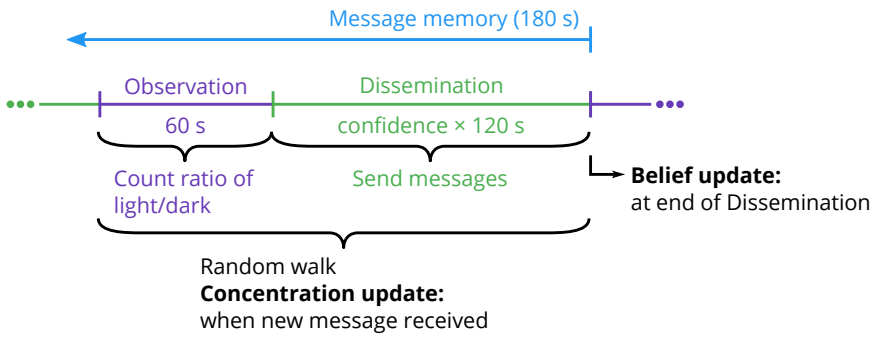


Figure 4.3: Timeline representing agent behavior during the collective decision-making algorithm.

region, it returns to the arena by turning until it no longer detects gray. Recall that agents move forward at 1 bodylength/s and turn at approximately 0.63 rad/s.

2. Estimate and Confidence An agent makes an *estimate* of the feature in the environment during a 60 s *observation window* by counting the time spent detecting black or white; it pauses its observation timer during any time spent in the gray border region. At the end of an observation period, the agent computes the ratio of white (n_1) to total ($n_0 + n_1$) observation duration. The *confidence* in this estimate is set to be minimum when the observation duration ratio is 0.5, and maximum when it is 0 or 1. It scales linearly between these points. Formally, we define:

$$e = \text{round}\left\{\frac{n_1}{n_0 + n_1}\right\}, \quad c = \frac{\max\{n_0, n_1\}}{n_0 + n_1}$$

An agent then enters a *dissemination period* during which it sends messages containing its ID and feature estimate. The duration of the dissemination period is set to $c \times 120$ s; that is, the more confident an agent is in its estimate, the longer it disseminates it. An analog of this concept is demonstrated in nature, such as the waggle dance in honeybees [156]. Following a dissemination period, an agent begins a new observation period.

3. Belief During both observation and dissemination periods, an agent receives estimates from other agents. It stores the agent ID and estimate in memory for 180 s. If an agent is heard from more than once within 180 s, only its most recent estimate is kept. At the end of a dissemination period, an agent computes its belief to match the majority of estimates in its 180 s memory, selecting a random belief if the count of each is equal or maintaining its current belief if its memory is currently empty. In essence, the agent is integrating information over the space covered by its neighbors' random walks. In every dissemination period except for the first one, the agent transmits messages containing its belief in addition to its ID and estimate.

4. Concentration Each agent also maintains a belief concentration C of the feature, which is a moving average that represents its understanding of the collective's feature belief. The concentration is initialized at 0.5 and can range from $0 - 1$. When an agent receives a message containing a belief b, it updates its concentration if the sending agent is not stored in memory, for a new concentration C^*:

$$C^* = 0.9C + 0.1b$$

The concentration represents an integration of the spatial belief over time, forming a longer-term history than the transient beliefs.

5. Decisions When the concentration for a feature crosses a threshold of 0.1 from the extrema and remains there for 30 s, an agent makes a non-reversible decision about the feature. A concentration below 0.1 results in a decision of 0 (mostly black), while a concentration above 0.9 yields a decision of 1 (mostly white). Increasing this threshold of After making a decision on the feature, an agent changes from disseminating its belief to disseminating its decision; agents receiving this value interpret it the same as a belief and use it to update their concentrations. This causes positive feedback that will increase the average concentration of the collective and push additional agents toward

a decision.

Our algorithm above — in particular components 2 and 3 — builds on the work of Valentini et al. [41]. The most important distinction is that our algorithm uses local rather than global communication. It also changes some stochastic computations (e.g., the observation and dissemination period duration) into deterministic ones; pilot experiments confirmed that this does not degrade performance. The notions of concentration and decision-making are not present in [41]; we introduced these inspired by quorum sensing in natural collective systems [93,94].

4.3.2 Simulation Results

We tested our algorithm in simulation on both (quasi-)homogeneous and non-homogeneous feature distributions. In a homogeneous feature distribution, each individual agent estimate arising from a random walk is expected to represent a good approximation of the true value. As the environment becomes less homogeneous, individual agent estimates are expected to become, on average, less reflective of the true value, and the variance among them is expected to increase.

Recall that in our environment, the feature values 0 and 1 correspond to the colors black and white, respectively. Therefore, the fill ratio r of the feature is given by the proportion of white area present within the environment.

Homogeneous Feature Distribution

To create homogeneous feature distributions, the square environment was divided into a grid of 12×12 cells of equal size. To create a fill ratio of $r \in [0, 1]$, each cell was randomly assigned a value of black or white with probabilities $1 - r$ and r, respectively, independently of the other cells; an example is shown in Fig. 4.2.

We ran simulations with 100 agents, which covers approximately 1.5% of the environment. The

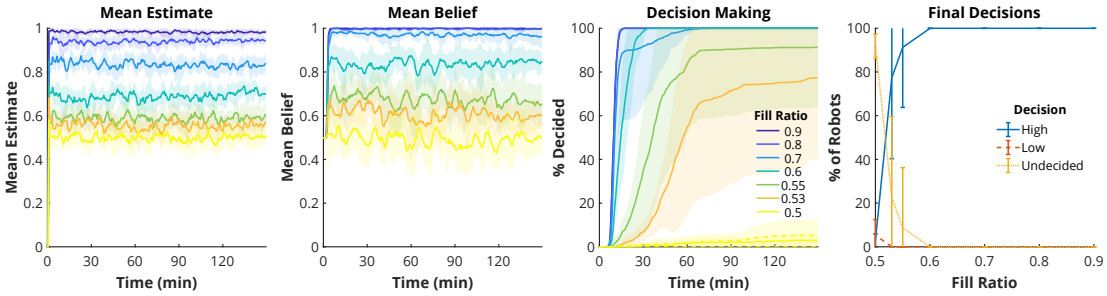


Figure 4.4: Simulated single-feature decision-making with a homogeneous feature distribution (shading and bars indicate standard deviation.) **Far left:** Mean feature estimate for all agents over time, for differing fill ratios. **Middle left:** Mean belief for all agents over time. **Middle right:** Mean percentage of robots that have made a decision over time. **Far right:** Percentage of robots at end of trial that decided high (white), low (black), or remained undecided.

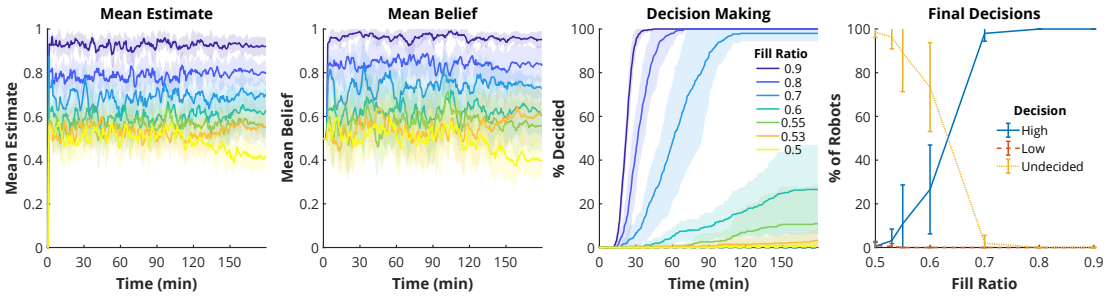


Figure 4.5: Simulated single-feature decision-making in an environment with a non-homogeneous feature distribution. (Shading and bars show standard deviation.) **Far left:** Mean feature estimate for all agents over time, for differing fill ratios. These are lower (closer to 0.5) than in homogeneously distributed environments. **Middle left:** Mean belief for all agents over time. These are also closer to 0.5 than in homogeneously distributed environments. **Middle right:** Mean percentage of robots that have made a decision over time. Compared to homogeneous environments, decision-making was slower, and fewer agents were able to make decisions with fill ratios closer to 0.5. **Far right:** Percentage of robots at end of trial that decided high (white), low (black), or remained undecided. After 180 min., fewer robots made decisions for fill ratios closer to 0.5 than in the homogeneous case.

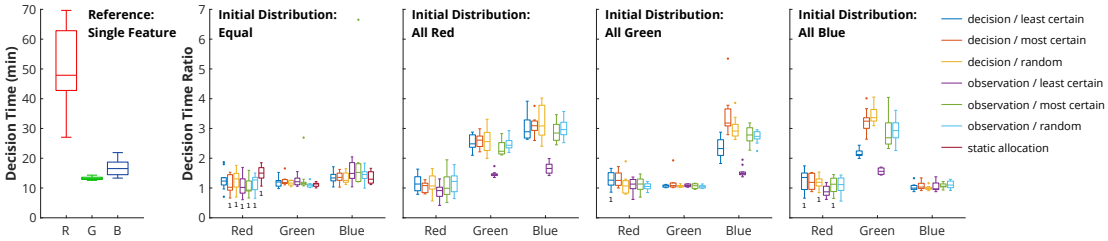


Figure 4.6: Comparison of time to reach 98% decided in multi-feature decision-making, with different approaches to task-switching. Boxes show median and 25th and 75th percentiles; dots are outliers. Numbers below boxes are the number of trials (out of 10) that did not reach 98% decided during the 150 min. trial. **Far left:** Time to decide with 100 agents on a single feature. **Others:** Decision time as a ratio of reference time for each feature. No feature-switching technique shows an advantage when the agents are initially equally distributed between the features. When all agents begin detecting a single feature, easy features are decided faster when they switch to the least certain feature after each observation.

initial distribution of agents was equally spaced within the environment with random orientations; at this density, agents are roughly 6 bodylengths apart and must move in order to communicate with each other. We conducted 10 simulations for various fill ratios ranging from 0.5 to 0.9. We also tested fill ratios below 0.5 to verify the symmetry of the decision-making, but for clarity we only present the upper half of the range.

Fig. 4.4 shows the results of 10 simulations for various fill ratios. The mean feature estimate and belief stabilize within minutes, but with an average higher belief than estimate. The individual agents' concentrations rise more slowly over time; in higher fill ratios with a higher mean belief, the concentration rises faster and this leads to faster decisions. For fill ratios of at least 0.53, no incorrect decisions are made; when the fill ratio is at least 0.6, all agents reach the correct decision within the 150 min. trial.

With a fill ratio of 0.5, no collective decision is made. The mean belief of approximately 0.5 results in concentrations that do not reach the threshold at either extreme. This means that our algorithm does not perform symmetry-breaking for truly ambiguous features, which may or may not be desirable, depending on the application.

Non-Homogeneous Feature Distribution

We implement a non-homogeneous feature distribution simply by splitting the environment into two strips, one of each color; an example is shown in Fig. 4.2. To achieve a fill ratio of $r \in [0, 1]$, the division line is set such that the area of the white strip as a fraction of the whole area is r, while the remaining $1 - r$ fraction of the whole area is black.

We conducted the same experiments as in the homogeneous feature distribution case. Fig. 4.5 shows that the resulting mean estimate and belief are lower than for the homogeneous environment, resulting in slower decision-making. Agents were also less capable of classifying fill ratios closer to 0.5, with the collective only consistently making complete decisions for fill ratios of at least 0.7.

The results in Fig. 4.5 (far left) confirm the expectation that individual agent estimates in a non-homogeneous environment will exhibit a larger variance than in a homogeneous environment. In our two-section environment, the agents can only make accurate estimates of the fill ratio when their random walk is close to the color interface. A random walk that happens to spend most of its time in one of the two areas will heavily bias the estimate towards that area, and will incorrectly increase the agent's confidence in its estimate. This exacerbates the propagation of the noise in the estimates into the beliefs, as shown in Fig. 4.5 (middle left); in turn, this leads to slower and less accurate decision-making.

4.3.3 Physical Results

We conducted experiments with 30 physical robots in a 1.2×1.2 m environment. In each of 5 trials, we projected onto the surface a randomly-generated homogeneous environment using a grid of 8×8 equally-sized cells (as in Sec. 4.3.2). The fill ratio was 0.7. This physical environment has 25% the area of the simulation environment, but uses 50% as many robots. The parameters of the random walk conducted by the robots was modified from the simulation case to have a straight

Trial	Time to first decision (min.)	Time to last decision (min.)
1	7:20	42:15
2	7:15	41:00
3	19:35	53:20
4	8:15	34:25
5	11:30	28:10
Mean (SD):	10:47 (5:13)	39:50 (9:25)

Table 4.1: Physical Experiment Results

component drawn from an exponential distribution with mean 60 s and a turning component drawn from a uniform distribution between $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right)$ rad. This shorter, more correlated random walk allowed robots to move more quickly after colliding (in the simulator, collided robots become unstuck quickly due to the pseudo-physical collision resolution).

For each trial, we recorded the time after which the first agent made a decision, and the time until all the agents had made a decision. The results are shown in Table 4.1. No robots made a wrong decision in any of the trials. The collective successfully classified the environment in, on average, less than 40 min., with the first decision appearing, on average, in just under 11 min.. There was significant inter-trial variability in both the time for the first robot to reach a decision and the time for all robots to decide. This can likely be attributed to the variation in the feature distributions between randomly-generated environments, as well as the random nature of the Kilobots' motion.

The physical robots exhibited some differences from their simulated counterparts. Their movement was less consistent than in simulation, with a straight-line movement that curved to varying degrees, and an inconsistent turning speed. When robots collided, they often failed to separate unless they changed to a turning state, resulting in transient clusters of robots in the environment. Both of these factors increase the locality of robot movement and decrease mixing. In addition, the simulator did not account for the noisy light sensing that was observed on the physical Kilobots.

These differences in movement and sensing likely combine to decrease the accuracy of the robots' estimates and beliefs. A video of the physical experiments is available in the online supplementary material[157]; note that due to some minor imperfections in the surface, and the Kilobots' locomotion mechanism (stick/slip with vibration motors), some robots become stuck and rotate around a single point.

4.4 Multi-Feature Decision-Making

4.4.1 Algorithm

The algorithm for multi-feature decision-making extends that for single-feature decision-making, with each agent keeping a belief, concentration, and decision for each of the three color features in our simulation. Each agent observes a single feature and disseminates its estimate of that feature in addition to the index of the feature. Agents receive and store estimates for all features from other agents, which they use to update beliefs for each feature at the end of their own dissemination period. Agents then transmit all beliefs in their future messages. New messages containing beliefs will therefore trigger a concentration update for all features. A decision on each feature is made from its respective concentration, independently of the other features.

Feature Switching On its own, the algorithm described above would be extremely sensitive to the initial allocation of agents to features; in the worst case, a feature would never be decided upon if no agents are allocated to it. Intuitively, it would make sense to allocate more agents to features that are harder to decide. However, we assume that no *a priori* information is available about this, and we therefore introduce a dynamic task allocation mechanism into the algorithm. While agents can only estimate one feature at a time, they are allowed to switch between estimating different features. We consider two options for when this switching may happen: either before each observation

period, or only after a decision has been made on the current feature. Moreover, we consider three possibilities for choosing which feature to switch to: the feature that has a concentration closest to 0.5 (the least certain feature), the feature with a concentration furthest from 0.5 (the most certain feature), or a random feature. An agent may not switch to a feature on which it has already made a decision. If there are no more undecided features, an agent remains allocated to its current feature.

4.4.2 Simulation Results

To create multi-feature environments, we overlaid single-feature environments. Three homogeneous feature distributions were first independently generated as described in Sec. 4.3.2, but with the color white replaced with one of red, green, or blue. These three feature distributions were then 'added' together so that every cell contained between zero and three colors (inclusive). This process is depicted in Fig. 4.2 (right two), where the color combinations are represented visually according to the standard RGB color model.

The collective's task now is to decide whether each of the red, green, and blue fill ratios is below or above 0.5. The three fill ratios were chosen so as to provide features of varying difficulties for the collective to decide on: red fill ratio $=$ 0.55 (hard); green fill ratio $=$ 0.8 (easy); and blue fill ratio $=$ 0.65 (intermediate).

We investigated six feature switching laws as discussed in the previous section, based on two possibilities for when the agents are allowed to switch between features, and three possibilities for which feature they switch to:

1. Agents may switch after deciding on the current feature to the least certain of the undecided features

2. Agents may switch after deciding on the current feature to the most certain of the undecided features

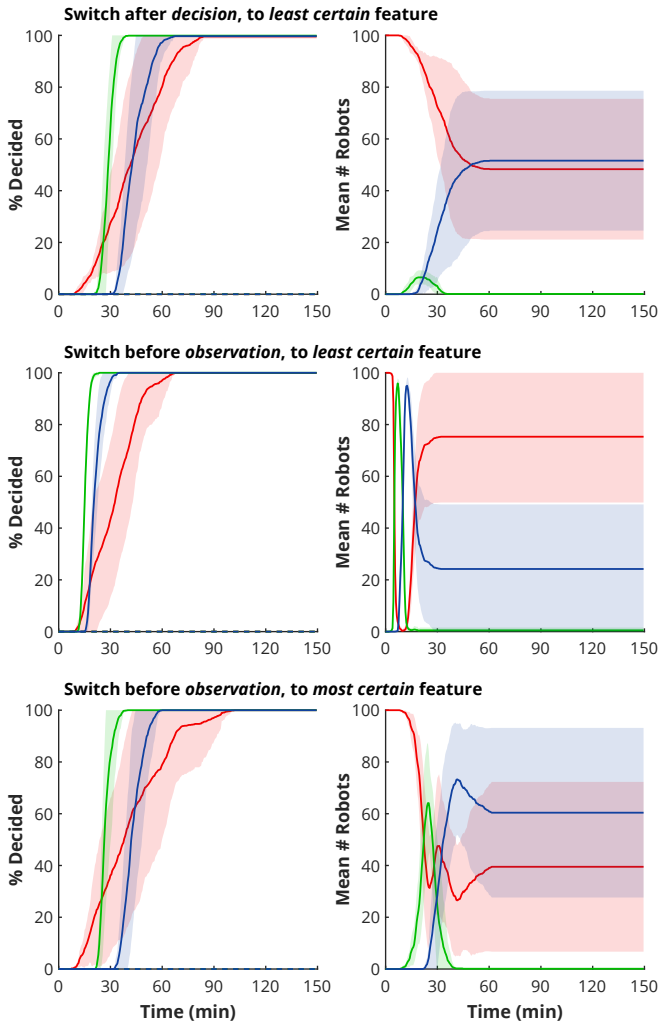


Figure 4.7: Decision-making progress for different feature switching conditions, with all agents initially allocated to red. (*Left:* Decision-making progress for all features. Shading represents standard deviation. *Right:* Allocation of agents between features.)
Top: Agents switch to their least certain undecided feature after making a decision about their current feature. A few agents decide for red and switch to green and blue, both of which are decided faster than red. Fewer than ten agents were allocated to green for all agents to make a decision.
Middle: Agents switch to their least certain undecided feature before every observation period. Agents are more quickly re-allocated to blue and green for a short period of time, resulting in quicker decisions than when feature switching only occurs after decisions.
Bottom: Agents switch to their most certain undecided feature before every observation period. More agents end up allocated to blue than when agents change the least certain feature, reducing the accuracy of beliefs about red and prolonging the feature's decision time.

3. Agents may switch after deciding on the current feature to a random undecided feature
4. Agents may switch before each observation period to the least certain of the undecided features
5. Agents may switch before each observation period to the most certain of the undecided features
6. Agents may switch before each observation period to a random undecided feature

For each switching law, we ran simulations with four initial agent-to-feature allocations: one with an equal allocation of agents to each feature, and three simulations with all the agents allocated to a single feature. For each switching law and initial allocation (16 combinations in total), we ran 10 simulation trials with 100 agents using randomly-generated feature distributions.

We compare the multi-feature decision-making time to a baseline of 100 agents deciding on each feature, a case of single-feature decision-making from Sec 4.3. When agents are initially distributed equally between the features, no feature-switching strategy shows a clear advantage over others. Notably, all remain close to the reference time to reach decisions.

However, when agents are not initially equally distributed, there is an advantage in decision-making time when agents switch to the least certain feature at each observation period. For the easier features (blue and green), this strategy produces quicker decisions when agents are not initially allocated to those features.

We investigate the reason for this advantage in Fig. 4.7, which demonstrates the changes in allocation and decisions over the simulations when agents are initially allocated to red. This is an expansion of Column 3 in Fig. 4.6.

Switching features for each observation period instead of after a decision results in faster decision-making because agents are more quickly reallocated to uncertain features. Looking at the agent allocation in the middle row of Fig. 4.7, we see that agents detect red for the first observation cycle,

then switch almost entirely to green (which, having not been previously observed, is the least certain feature with a concentration of 0.5). Most agents make a decision on green before their next observation period because of the large number of agents dedicated to the task. They then switch to blue (also unobserved and therefore with a concentration near 0.5) and make a fast decision before switching back to red. In contrast, in the top row of Fig. 4.7, where agents switch features only after decisions, changes in allocation are much slower. However, in this scenario there is no cost for switching features; if such a cost existed (for example, if robots had to move to a new location or replace their sensor for a different feature), the benefit of switching between observation periods could be negated.

Switching to the least certain feature produces an advantage because it prevents over-allocation of agents to easy features. This can be seen in the contrast between the bottom 2 rows of Fig. 4.7, where agents switch to the most or least certain feature for each observation period. Counterintuitively, agents that switch to the most certain feature remain on red until some agents have made a decision on the feature; belief updates will push the concentration above 0.5 and make it perceived as the most certain feature. This delays decision-making on other features until agents this perceived easy feature is decided by a few agents. However, after agents move to and quickly decide green, they disproportionately switch to blue instead of red (at approximately 30 min. in the bottom right of Fig. 4.7). If they decide red while observing blue, they will remain detecting blue. Compared to agents switching to the least certain feature, agents are over-allocated to blue. This reduces the accuracy of estimates (and by extension, beliefs) for red, prolonging the decision process.

4.5 Conclusions and Future Work

In this chapter, we investigated both the single-feature and multi-feature collective decision-making problems for inspection. Our algorithm uses only local communication, and is able to consistently

make a correct unanimous decision in finite time, even on features that are almost completely ambiguous. The algorithm can correctly classify a number of features simultaneously in a multi-feature environment. This holds even if the algorithm is presented with pathological initial agent-to-feature allocations, thanks to a dynamic task allocation mechanism. We examined different types of task switching rules, and identified the one that works best over various initial allocations.

This represents an important skill for multi-robot inspection. Even simple robots are able to correctly identify the state of multiple relevant environmental features through prudent communication and an algorithm for mapping observations to classifications. It also shows that multiple features can be inspected for by a single swarm in a single pass, which can also speed evaluation of a site.

In future work, we can implement multi-feature collective inspection where the features are fundamentally different from each other. For instance, the agents could be required to evaluate the color fill ratio, entropy, and curvature of the environment, as described in Sec. 4.1. These types of features can require different movement strategies, which is representative of different real-world inspectable features that might require evaluation. In the next chapter, we will also look at another approach to solving the same single-feature decision-making problem; this builds on the concept presented here of building a model representation that accounts for a robot's uncertainty about the world state, but instead of creating a multi-level, hierarchical model, it represents this knowledge with a Bayesian distribution.

This is what we mean when we talk about "robots." We're talking about cognitive abilities, not the fact that they're made of metal instead of flesh and powered by electricity instead of chicken nuggets.

Kevin Drum

5

Bayesian Site Inspection with a Minimal Swarm

For groups of robots to cooperate in complex scenarios, they must be able to collectively make choices at multiple decision points. In many cases, this takes the form of a "go/no-go" problem: each robot must select the best of two possible choices based on some incomplete information avail-

able to them. In swarms of robots, this challenge is compounded: cooperative behavior relies on all the robots quickly coming to the same decision.

This is a form of collective robotic inspection where the robots must perform a binary classification of their environment, which is prevalent in potential applications. For example, robots may first classifying whether an agriculture field contains pest, then eliminate the pests if present. Or, robots may inspect a potential site for a human habitat on Mars, then build the habitat if it is classified as suitable. Solving this problem may require a decentralized approach if the robots cannot rely on a central process for collecting information and taking decisions. It is challenging to have a large, decentralized group of robots quickly and accurately make collective decisions, especially in the binary go/no-go scenario.

In Chapter 4, we presented a bioinspired approach to solving this type of classification site inspection problem. However, while this algorithm successfully achieved collective decision-making, it was heuristically designed and lacked a mathematical grounding. This makes is difficult to intuitively understand parameter selections and the speed vs. accuracy tradeoff in decision-making. In contrast, Bayesian algorithms provide a statistically grounded approach to decision-making from multiple observation sources in multi-agent systems[95,100,101].

In this chapter, we present a novel Bayesian algorithm for a robot collective to achieve fast, accurate decisions about their environment. We abstractly model the go/no-go decision by tasking robots with classifying monochrome environments as filled with a majority of black or white, as in[41] and Chapter 4, which is shown in Fig. 5.1; this represents any scalar environmental feature that could be observed by robots. Each robot behaves as a Bayesian estimator, while exchanging and integrating observations from nearby robots. We show that collective decisions are possible even with few assumptions about the capabilities of the robots: a collective of 100 simulated Kilobot robots is able to achieve accurate decisions even when they are sparsely distributed and have locally-limited sensing and communication. We find that positive feedback improves both the speed and accu-

racy of decisions, and that each robot making fewer observations can improve decision accuracy by reducing their spatial correlations. In addition, a well-chosen regularizing prior allows for a lower decision-making threshold with a small accuracy cost. We also demonstrate that the algorithm's speed naturally adapts to the difficulty of the environment. Finally, we compare this approach to a fixed-time benchmark algorithm that provides theoretical accuracy guarantees even in worst-case environments.

This work was conducted in collaboration with Dr. Melvin Gauci, with theoretical work for the benchmark algorithm by Dr. Frederick Mallmann-Trenn. It was published at the 2020 International Conference on Robotics and Automation (ICRA)[158].

5.1 Methods

5.1.1 Problem Definition

We present a problem in which k robots complete a binary classification task. Robots are placed in a bounded black and white environment, where the proportion of white within the space is the environment's fill ratio f, as shown in Fig. 5.1. The goal is to collectively decide whether the majority of the environment is filled with black or white (i.e, is the fill ratio above or below 0.5). Because the problem is symmetric, we show results for environments where $f > 0.5$.

Classifying an environment results in a trade-off between the time for all robots to decide and the collective accuracy of the decision. This is particularly pronounced in the most challenging environments, where the fill ratio is close to 0.5; the small difference between black and white area makes it more difficult to distinguish than extreme fill ratios.

This formulation represents an abstraction of real-world problems. In an agricultural application, the white regions would be analogous to pest-damaged areas of a field, with the goal of determining if a field requires pest treatment. Alternatively, the color could represent mineral deposits

in a Mars exploration mission. In each case, the goal is to make a go/no-go decision about a single, spatially distributed feature.

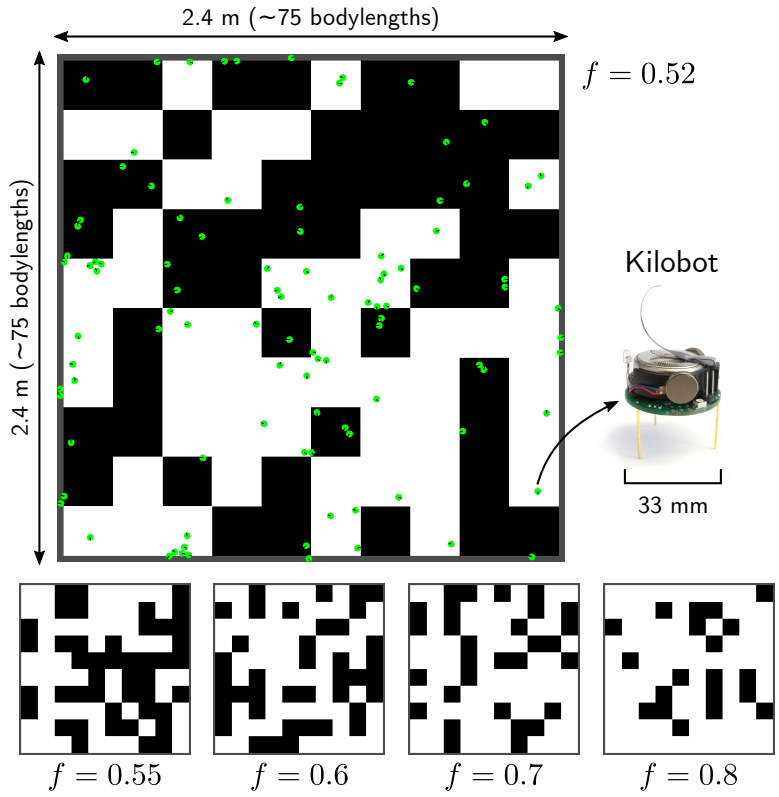


Figure 5.1: Examples of simulated environments with different fill ratios f. The goal is for robots to determine whether the environment is mostly white ($f > 0.5$) or mostly black ($f < 0.5$). **Top:** Image from a Kilosim simulation with $f = 0.52$, containing 100 robots, each able to communicate within a radius of 3 bodylengths. **Right:** Kilobot robot, which is the model for the simulated robots. **Bottom:** Example environments with different fill ratios.

5.1.2 Robot Model

We investigate this decision-making problem using Kilobots as our model robot platform [80], whose capabilities narrow the complexity of possible decision algorithms. Kilobots are able to sense their environment with an ambient light sensor located on the top of the robot, allowing them to distinguish black, white, and gray regions of an environment projected from above onto a bounded

2D arena. These small robots (33 mm diameter) lack complex bearing or localization capabilities. Therefore, we rely on pseudo-random walks. The robots also have limited communication bandwidth and range; they can broadcast 9-byte messages to robots within approximately 3 bodylengths.

In environmental classification problems, robots are typically sparse. 100 Kilobots cover only 1.5% of the 2.4×2.4 m arena ($\approx 75 \times 75$ bodylengths) available for the physical robots. Therefore, we cannot rely on assumptions required in many distributed algorithms, like maintaining a strongly connected network. However, we demonstrate that it is possible to design robust decision-making algorithms even for robots with limited capabilities.

5.2 Algorithms

We developed a Bayesian algorithm that allows simulated Kilobot robots to classify black and white environments. The goal was to classify the fill ratio f of the environment as mostly white (a decision of $d_f = 1$) or mostly black ($d_f = 0$). Each robot employs a Bayesian model of the fill ratio and makes decisions using credible intervals of the posterior distribution. We also compare to a benchmark algorithm that provides accuracy guarantees for even the worst-case scenarios, sacrificing speed for accuracy.

5.2.1 Bayesian Decision-Making Algorithm

The algorithm followed by each robot is shown in Alg. 5.1.

Robots make binary color observations C of their environment, which we model as draws from a Bernoulli distribution where the probability of observing white is the fill ratio:

$$C \sim \text{Bernoulli}(f) \tag{5.1}$$

Each robot models the unknown fill ratio f of the environment as a Beta distribution:

$$f \sim \text{Beta}(\alpha, \beta) \tag{5.2}$$

resulting in the posterior update for each observation:

$$f \mid C \sim \text{Beta}(\alpha + C, \beta + (1 - C)) \tag{5.3}$$

Initialization: Robots are placed uniformly in the arena with random orientation. Each robot's prior model is initialized with $\text{Beta}(\alpha, \beta)$, where both parameters are initialized as α_0, a parameter determining how regularizing the prior is. Each robot also sets its observation index $i = 0$.

Movement: For the duration of the trial, each robot performs a pseudo-random walk to cover the arena, defined by segments of movement in a straight line, followed by a random turn. The durations of the straight segments are drawn from an exponential distribution with mean of 240 s, while turns are drawn uniformly from $0 - 2\pi$. This parameterization was previously determined in Chapter 4. The edge of the bounded environment is defined by a gray region, as seen in Fig. 5.1. If a robot detects gray light, it turns continuously until it exits the border region.

Observation: Each robot makes an observation C every τ seconds: $C = 1$ if white, $C = 0$ if black, and ignoring gray observations. The posterior of the fill ratio is updated with the observation as in Eq. 5.3 and i increments by 1.

Communication: After a robot makes its first observation, it begins broadcasting its most recent observation index i and observed color C. While continuing to move, observe, and broadcast, all robots also listen for messages from neighboring robots. Upon receiving a new observation, the receiver updates its posterior as with its own observations.

Decision: Each robot checks whether its decision criterion is met after every posterior update.

Algorithm 5.1 Bayesian Decision-Making Algorithm
Input: Observational interval τ, credible threshold p_c, prior parameter α_0, positive feedback indicator u^+, robot UID id
Output: Binary classification of environment d_f

Init counter of white observations $\alpha = \alpha_0$
Init counter of black observations $\beta = \alpha_0$
Init observation index i
Init incomplete decision $d_f = -1$
Init dictionary of received observations $s = \{\text{ID} : (0, 0)\}$
for $t \in [1, T]$ **do**
    Perform pseudo-random walk
    if τ divides t **then**
        $C \leftarrow$ observed color $(0, 1)$
        $\alpha \leftarrow \alpha + C$
        $\beta \leftarrow \beta + (1 - C)$
        $i \leftarrow i + 1$
    Let $m = (id', i', C')$
    if $s(id') \neq m(id')$ **then**
        $\alpha \leftarrow \alpha + C'$
        $\beta \leftarrow \beta + (1 - C')$
    if $d_f = -1$ **then**
        Let p denote the cumulative distribution function of $\text{Beta}(\alpha + \alpha_0, \beta + \alpha_0)$ at 0.5.
        if $p > p_c$ **then**
            $d_f \leftarrow 0$
        else if $(1 - p) > p_c$ **then**
            $d_f \leftarrow 1$
    if $d \neq -1$ and u^+ **then**
        Broadcast message (id, i, d_f)
    else
        Broadcast message (id, i, C)

The credible threshold p_c defines the probability mass of the posterior that must lie on one side of 0.5 in order for a decision to be made. If the posterior's cumulative distribution p at 0.5 passes the criterion ($p \geq p_c$), a decision is made that the environment is black ($d_f = 0$), as most of the probability is below 0.5. Conversely, if $(1 - p) \geq p_c$ (i.e., most of the probability mass is above 0.5), the environment is classified as white ($d_f = 1$). *This is the robot's irreversible go/no-go decision.*

After a decision is made, a robot will broadcast its decision in place of its observation if positive feedback (u^+) is used. Otherwise, it will continue to transmit its observations.

This algorithm depends on four parameters:

- **Observation interval** τ (s) is the time between observations, where $\tau > 0$. Shorter observation intervals mean collecting observations quicker, but results in observations that are less spatially distributed. Longer intervals result in more independent observations.

- **Credible threshold** p_c is the minimum probability mass of the posterior that must lie on one side of 0.5 in order to make a decision. We assume $0.5 \leq p_c < 1$. Higher credible thresholds require more observations before enough probability amasses to make a decision.‘

- **Prior parameter** α_0 is a positive integer used for both shape parameters of each robot's prior distribution of f. Setting $\alpha_0 = 1$ forms a uniform prior, while $\alpha_0 > 1$ creates a symmetric prior peaked around 0.5. This regularizing prior indicates a lower prior belief that the fill ratio is near 0 or 1, analogous to having previously made $\alpha_0 - 1$ black and $\alpha_0 - 1$ white observations.

- **Positive feedback** u^+ is a boolean indicating whether robots will transmit their decision d_f in place of their most recent observation C after they make decisions. Positive feedback is used effectively for decision-making in insects and bacteria. This feedback reinforces decisions made by robots that decide early, but it may push the group to the wrong decision or split the group if early-deciding robots conflict.

While there is intuition behind the trends of these parameters individually, the interactions and optimal choices are unknown. We use a parameter sweep to investigate the effect of parameter values on speed and accuracy, as well as the interactions between the parameters.

5.2.2 Benchmark Decision-Making Algorithm

We now describe a fixed-time algorithm for which parameter settings can be derived that guarantee correct decision-making to an arbitrary accuracy in an arbitrary environment of known size, as shown in Alg. 5.2. Given a worst-case fill ratio that we wish to be able to detect, and a desired accuracy (i.e., tolerance for incorrect decisions), we can compute the number of weakly correlated samples S that a single robot requires to make a decision. If we instead have k robots, robots first independently collects samples (Phase 1), and then disseminate information among each other (Phase 2). The observation phase must be long enough for each robot to collect at least S/k samples; the second phase must be long enough for all pairs of robots to communicate, such that each robot has a total of at least S samples.

Phase 1: Sample Collection

We first select a worst-case fill ratio $\hat{f}$ (i.e., how close to 0.5) to be able to distinguish. To make a correct decision, we need enough samples that the sample mean is within $\varepsilon = 2 \cdot |f - 0.5|$ of the true fill ratio. For a given confidence level $1 - \delta/2$, we need a total of S uncorrelated* samples:

$$S \geq \frac{4\hat{f}(1-\hat{f})Z\left(1-\frac{\delta}{4}\right)^2}{\varepsilon^2} \tag{5.4}$$

*As mentioned above, there is a very weak correlation between samples. However, by fine-tuning the time between samples τ, we can make sure that the probability of sampling a white cell is within $f \pm \varepsilon/4$. This error is small enough to ensure our calculations hold.

using the Z-score of the standard normal distribution. This is derived from the two-tailed $1 - \delta/2$ confidence interval using a Gaussian approximation of the Binomial distribution [159].

The S/k samples each robot collects must be uncorrelated in order for Eq. 5.4 to hold. If nothing is known about the distribution of colors within the environment, we must design for the worst case, where samples are highly locally-correlated (i.e. a non-homogeneous environment). Then, each robot must move $\tau \geq t_{\text{mix}}$ between samples, where the mixing time t_{mix} is a property of the size and topology of the environment, and the nature of the random walk. Conversely, if the environment is homogeneous (i.e., if each cell is colored independently of its neighbors, as in 5.1), then the observation interval τ need only be long enough that a robot does not sample more than once from the same grid cell consecutively. Therefore, the Phase 1 duration is $S/k \cdot \tau$.

Phase 2: Sample Communication

Each robot now has S/k samples but needs S samples to make an accurate decision. We assume that the robots have IDs, can ignore repeated information, and have a communication radius r_{comm}. When robots A and B are within r_{comm} of each other, A collects and stores B's samples if it has not done so already, and vice-versa. We must now determine how long robots need to move such that each pair of robots has interacted, to some desired confidence level $1 - \delta/2$. This notion is captured by the meeting time, t_{meet}, which is defined as the worst-case expected time for two robots to meet, regardless of their starting location. Note that t_{meet} is a function of: (i) the environment size; (ii) the environment topology; (iii) the nature of the random walk; (iv) the communication radius. To guarantee with probability $1 - \delta/2$ that all pairs of robots have communicated, we require a Phase 2 duration of:

$$t_{\text{comm}} = 2 \log \left(\frac{k^2}{\delta} \right) t_{\text{meet}} \tag{5.5}$$

The probability that two random walks meet after $2t_{\text{meet}}$ steps is, by the Markov inequality, at

least $1/2$. Thus, the probability that any two given random walks do not meet after $\log(k^2/\delta)$ intervals of length $2t_{\text{meet}}$ is $\frac{1}{2^{\log(k^2/\delta)}} = \frac{\delta}{k^2}$. Taking the union bound over all $\binom{k}{2}$ pairs gives that the total probability of failure is at most $\frac{k(k-1)}{2}\frac{\delta}{k^2} \leq \frac{\delta}{2}$.[160]. Combining the $\delta/2$ failure risk from each phase, we can guarantee the decision will be correct with probability $1-\delta$.

Algorithm 5.2 Benchmark Decision-Making Algorithm
Input: Total communication time t_{comm}, observation interval τ, robot UID id, number of samples S/k
Output: Binary classification of environment d_f

Init counter of white observations $\alpha = 0$
Init counter of black observations $\beta = 0$
Init dictionary of received samples $s = \{\text{ID} : (0,0)\}$
for $t \in \left[1, \frac{S}{k}\tau\right]$ **do**
    Perform pseudo-random walk
    if τ divides t **then**
        $C \leftarrow$ observed color (0, 1)
        $\alpha \leftarrow \alpha + C$
        $\beta \leftarrow \beta + (1 - C)$
$s(id) = (\alpha, \beta)$
for $t \in \left[\frac{S}{k}\tau, \frac{S}{k}\tau + t_{\text{comm}}\right]$ **do**
    if new message (id', α', β') **then**
        $s(id') = (\alpha', \beta')$
    Broadcast message (id, α, β)
Let α_T denote the sum of the α values in s
Let β_T denote the sum of the β values in s
if $\beta_T > \alpha_T$ **then**
    $d_f = 0$
else
    $d_f = 1$

5.3 Experiments

We conducted experiments testing both algorithms in Kilosim, an open-source Kilobot simulator we developed that is able to run at over 700× real speed for 100 robots [124], allowing us to thoroughly investigate the parameter space. A demonstration video is available on YouTube [161]. All experiments were conducted with 100 robots in a 2.4 m × 2.4 m arena. To investigate the performance in settings of varying difficulty, we tested five different fill ratios f: 0.52, 0.55, 0.6, 0.7, 0.8. Fill patterns (as seen in Fig. 5.1) were generated for each trial by pseudo-randomly filling a 10 × 10 grid of squares with black or white to match the fill ratio. Trial duration was capped at 50,000 s (≈ 14 hours) each.

5.3.1 Bayesian Algorithm

We conducted a parameter sweep across the following values, running 100 trials for each of the resulting 7,280 parameter combinations.

- τ (s) : 1, 5, 10, 15, 20, 25, 50, 75, 100, 150, 200, 250, 300
- p_c : 0.9, 0.95, 0.98, 0.99
- α_0 : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 50
- u^+: True, False

5.3.2 Benchmark Algorithm

For the benchmark algorithm, we computed the required time parameters τ and t_{comm} to meet the guarantees of $\delta = 0.1$ (equivalent to the Bayesian $p_c = 0.9$) and $\varepsilon = 0.04$, which matches the most difficult environment ($f = 0.52$).

Because the robots' random walk is highly correlated relative to the grid cell size, an upper bound on τ is calculated from the expected time to cross a grid cell. Given a robot speed of 1 bodylength/s and grid cells of approximately 7×7 bodylengths, we selected $\tau = 10$ s, which is the time to cross a cell diagonally. From Eq. 5.4 have that $S = 2{,}398$ samples, or 24 per robot, resulting in a Phase 1 duration of 240 s.

We computed t_{meet} empirically, because it depends on environment- and robot-specific factors. To match the worst expected meeting time across all possible starting positions, we placed two robots in opposite corners of the arena with random orientation. Over 5,000 trials, we computed the mean time for the robots to first communicate as 1,151 s. From Eq. 5.5, we therefore set a Phase 2 duration of $t_{\text{comm}} = 26{,}503$ s. We conducted 100 trials with these parameters.

5.4 Results

We assess the success of the Bayesian decision-making algorithm by considering the speed vs. accuracy trade-off across our parameter sweep. We treat decision-making as a multi-objective optimization problem by comparing the accuracy of decision-making vs. the time for all robots complete decisions in each parameter condition. The optimal parameter selections are those that lie along the Pareto frontier of accuracy and decision time.

We first consider the impact of positive feedback (u^+) in the most difficult environment, where the fill ratio is the most ambiguous at $f = 0.52$. As shown in Fig. 5.2A, positive feedback is essential for pushing the group to decisions, dramatically improving both the decision accuracy and speed. In many conditions without positive feedback, the entire group was unable to reach decisions within the 14 hour simulation limit, while with positive feedback the worst decision time was under 5 hours. One might expect positive feedback to split the swarm into two groups, resulting in lower overall accuracy; however, this occurs more when positive feedback is *not* used, resulting in a collec-

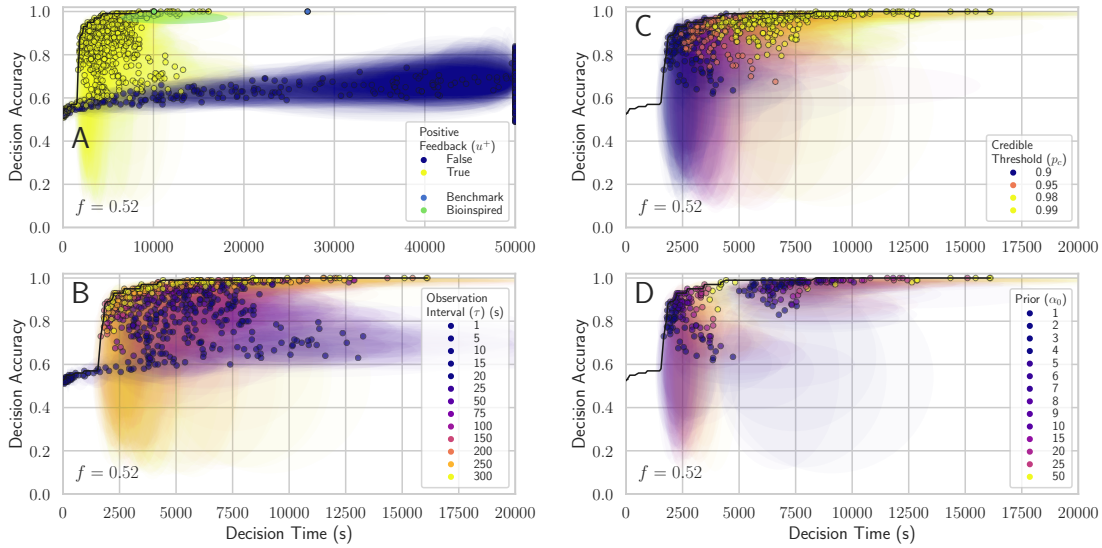


Figure 5.2: Speed and accuracy of Bayesian decision-making. Each point represents the median time for all robots to reach a decision and median accuracy of the resulting decisions, over all trials for a particular condition. Ellipses show the 25–75th percentile in each dimension. The black line shows the Pareto front of decision time vs. accuracy. Each successive figure shows a subset of the data from the preceding one. **A:** For a fill ratio of 0.52, decisions were faster and more accurate when positive feedback was used. Comparing to results from Chapter 4, certain parameter choices were faster while maintaining high accuracy. The benchmark algorithm exceeded its accuracy guarantees but was slower than comparatively accurate Bayesian parameter combinations. **B:** Longer intervals between observations counter-intuitively resulted in more optimal decisions (showing $u^+ = \text{True}$). **C:** Lower credible thresholds save time with minimal accuracy cost (showing $\tau \geq 15$). **D:** Lower credible thresholds are effective only if a regularizing prior prevents premature decisions (showing $p_c \in \{0.9, 0.99\}$ on left and right, respectively).

tive accuracy consistently below 70%. This is consistent with both previous robot results in [31,155], and the use of positive feedback in biological collectives. As a comparison, we look at the bi-inspired algorithm from Chapter 4 in the same environment. It lies on the Pareto front, but many parameter choices for the Bayesian algorithm achieve the same accuracy faster.

Focusing only on conditions where positive feedback is used, we investigate the impact of the interval between observations (τ), shown in Fig. 5.2B. Somewhat surprisingly, higher times between observations yields results closer to the Pareto optimal front. While a shorter observation interval yields more total observations, they are highly spatially correlated. A longer interval results in fewer observations but more mixing, therefore resulting in more representative samples and more accurate

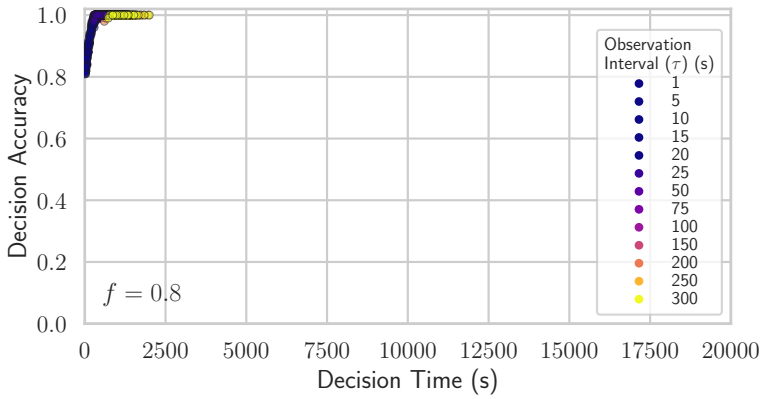


Figure 5.3: Analogous plot to Fig. 5.2B for a fill ratio of 0.8. On the same timescale, decisions were significantly faster and more accurate in this easier environment.

decisions in a shorter time.

Given the benefits of positive feedback and longer observation intervals, we now look at the effect of credible threshold (p_c), shown in Fig. 5.2C. An intuitive pattern emerges that higher credible thresholds produce higher accuracy, but choosing the highest threshold of 0.99 can incur a large time cost. In Fig. 5.2D, we see when this occurs by contrasting the decision results for $p_c = 0.9$ and $p_c = 0.99$. Here it becomes apparent that there is an interaction between the the choice of prior and the credible threshold. When a sufficiently large regularizing prior is used, the credible threshold can be low because the prior prevents premature decisions.

The benchmark algorithm consistently exceeded its 90% accuracy guarantee, as seen in Fig. 5.2A. The high ratio of time spent communicating vs. observing also underlines the previously-observed benefit of collecting fewer, less correlated samples. However, meeting this algorithm's worst case guarantees incurs a time cost in comparison to many Bayesian parameter configurations. It requires sufficient time for enough direct pairwise robot communications, rather than forming a multi-hop network by re-transmitting observations. However, this accurately reflects the constraints in the modeled robot system, whether bandwidth limitations and channel capacity prevent effective

re-communication. Fig. 5.3 also shows that the Bayesian algorithm is adaptively faster in an easier environment, with $f = 0.8$. The benchmark algorithm performed perfectly, but still took 26,743 s ($\approx$ 7.5 hours) for all robots to reach a decision, because its fixed time is determined by the most difficult fill ratio.

5.5 Conclusions and Future Work

We have introduced a decentralized Bayesian algorithm (Alg. 5.1) that allows simple, sparsely spaced robots to achieve accurate classifications of an environmental feature. This is representative of the types of tasks we expect robots to be able to achieve in inspection tasks. With well-selected parameters, the robots were able to achieve this go/no-go conclusion even when the difference between black and white fill was only 4%. When the feature distinction was greater, decision speed and accuracy significantly improved, while becoming less sensitive to parameter choice. This adaptability makes this approach suitable for applications where little *a priori* knowledge is available about the feature under consideration, in contrast with the benchmark algorithm, where providing guarantees for the worst case requires pre-selecting a longer decision time for all environments. However, robots using the Bayesian algorithm do not know whether others have made a decision, in contrast to the guarantee of decisions after the benchmark's fixed duration.

The Bayesian algorithm is also tunable; for example, if an expected fill ratio is known, an informed prior can be selected. If the accuracy requirements are lower in a particular case, the credible threshold could be lowered to speed up decisions. Positive feedback increasing decision speed and accuracy also demonstrates that bio-inspiration can be beneficial when used with statistically-grounded decision models, rather than as an alternative approach.

We also showed that it is possible to create an algorithm with accuracy guarantees for simple robots without knowing the environment's difficulty, but this incurred a trade-off in total decision

time, across all environment difficulties. The benchmark algorithm's guarantees may also be fragile in the real world; for example, robot failure after starting would violate the requirement of k robots and result in insufficient observations. In practice, we have shown high accuracy can be consistently achieved with the Bayesian algorithm, without the explicit guarantees of the benchmark.

In Chapter 4, we showed a different, bioinspired algorithm to solving the same collective classification problem; this Bayesian approach provides a number of benefits over this previous approach. The Bayes Bots algorithm provides a more streamlined, statistically grounded approach with fewer parameters. Unlike the bioinspired algorithm, its parameters also have a more interpretable and intuitive impact on the speed and accuracy of the robots' decision-making. The bioinspired approach also modeled each robot's knowledge of the world as a *point estimate:* the pseudo-concentration. In contrast, the Beta distribution in Bayes Bots maintains a *distribution* to model knowledge of the environment. This means that uncertainty in the robots' knowledge is directly built in, which allows us to drop the multi-tiered bioinspired approach of estimates, beliefs, and concentrations without losing information vital to accurate decision-making.

In addition to binary classification problems, this Bayesian approach is potentially applicable to a variety of inspection problems where observations of the target feature can be Bayesian-modeled. For example, determining the average density of items in an environment, such as pests in an agriculture field, could be modeled with a Poisson likelihood, with each robot maintaining a representation of the density with a Gamma distribution. The Bayesian model is also easily extendable to multiple features of the same type, which can be represented as a multi-dimensional distribution. We can extend our model to classify an environment with multiple color features, as Chapter 4, by using a Dirichlet distribution in place of our Beta posterior.

Personally, I'm not afraid of a robot uprising. The benefits far outweigh the threats.

Daniel H. Wilson

6

A Hybrid PSO Algorithm for Multi-robot Target Search and Decision Awareness

Robot collectives can work together to identify the locations of features in an unknown environment, such as determining the weakest points when inspecting a structure. When robots in a swarm identify such locations and share the information across the whole group, the group can

chain together actions into complex behavior. For example, a group of robots could be tasked with building a human habitat on Mars: first, they could collectively inspect a site to determine whether it is a suitable region, as in Chapters 4 and 5, then identify a specific location in the area with stable enough ground, then begin construction. While the preceding chapters focused on global classification tasks in inspection, in this chapter we shift our attention to the second class of inspection tasks: localizing a feature within an environment. This problem appears in many scenarios, such as identifying a high enough elevation to place a communications tower or a fault in a structure for repair. In each case, there is scalar feature that can be detected at all points in the environment, and robots can identify a location that is close enough to optimal (i.e., past a given threshold) to complete their search task.

This problem is difficult: the feature under investigation, such as elevation or structural integrity, often has a non-convex distribution or cannot be well-modeled. To guarantee locating the global minimum, the optimal solution is exhaustive coverage[43]. In many real-world situations, though, a location with a value past a threshold can be identified with non-coverage search algorithms. This threshold-based task is typical in inspection, where a fault is defined by crossing a safety threshold, or a target location is considered good enough if it means a pre-defined criteria. A robot collective can then speed target localization by adding sensors. However, without global communication, robots would need to independently find the target, wasting time and energy. Instead, robots can move to facilitate communication, which enables collective decision-awareness in the group and allows them to chain collective tasks such as habitat construction or informing a human operator of a fault requiring repair.

In this chapter, we consider a specific case of this collective search problem: simulated robots seek to (1) locate a position in an environment where a value is below a threshold and (2) disseminate that information to the entire group, completing their task when all robots are aware of the target and return to their deployment location. We model the environment as terrain generated by 2D Per-

lin noise to create varying difficulties, which is classically used for terrain generation in computer graphics. We present a hybrid algorithm designed to balance exploration of the environment, exploitation of previous observations, and communication with other robots in the swarm. We show that this algorithm operates successfully within realistic constraints of real robot inspection systems, such as limited run duration due to battery constraints, minimizing use of energy-intensive communication, and not requiring tight inter-individual coordination.

This work has been submitted to the International Conference on Intelligent Robots and Systems (IROS) for review and inclusion in the 2022 conference [162].

6.1 Methods

6.1.1 Problem Definition

We present a problem in which a group of simulated robots identify a location with a value below a threshold value v^*. Robots move in a bounded, monochrome arena composed of grid cells with values $v \in \mathbb{N}\,\{0..255\}$, as shown in Fig. 6.1. The goal is for each robot to identify a target grid location X where the value $v \leq v^*$; this knowledge obtained from its own observations or communicated by other robots.

Robots begin at a home position in the corner of the environment, equivalent to being deployed together, and the simulation is considered complete when all robots have returned to the home position. A robot will only return to its home position when it knows of a target location, and when it believes that all other robots also know it. This collective awareness is essential for any task where the robots must collectively change their action after a decision is made, including returning to a home position for collection. If robots return before all others make a decision, this leaves the remaining robot(s) to locate the source alone. Note that there can be more than one location with $v \leq v^*$. These discrepancies can be reconciled when robots collect themselves, because they return to the

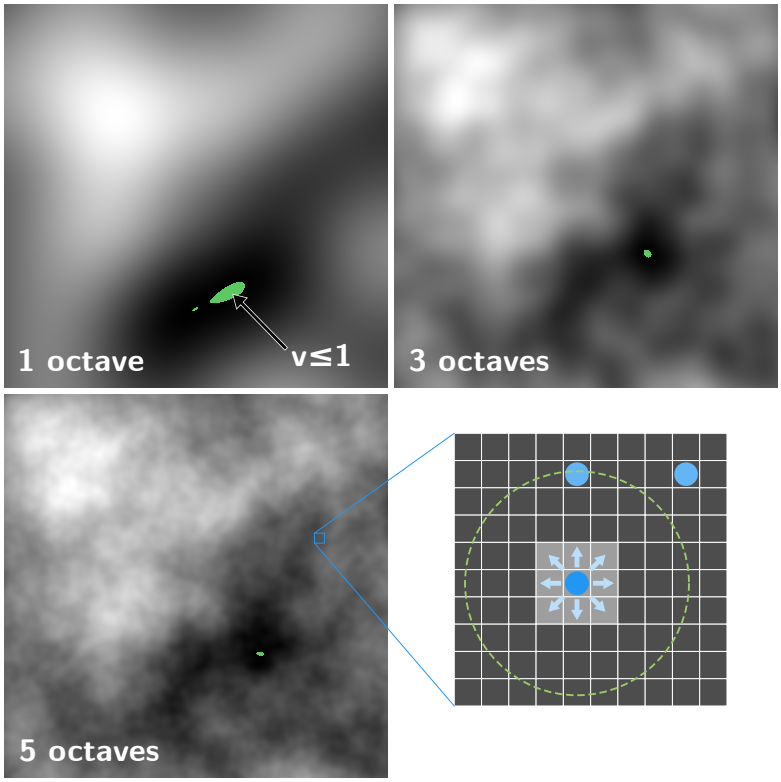


Figure 6.1: Examples of simulated environments generated by 1, 3, and 5 octaves of Perlin noise, with overlay of values $v \leq v^*$ for decision threshold $v^* = 1$. Robots must identify a location with a value below v^*. Bottom right: 10×10 cell segment of environment, showing robots (blue dots), movement allowed in a single step (blue arrows), and example communication range $d_c = 4$ (green circle).

same location.

6.1.2 Robot Model

We use an abstract robot model that can move and observe in the grid world. Each robot occupies a single cell, such that the grid discretization represents the sensing resolution of a robot; multiple robots may occupy the same cell. At each time step, a robot can move to any of the neighboring eight grid cells, while maintaining knowledge of its position. Accurate localization is a reasonable assumption, given the ubiquity of GPS outdoors, increasing accuracy of SLAM algorithms. While [34]

demonstrated that it is possible to perform PSO on robots without global positioning, its presence enables accurate memory and reporting of the target location. Each robot can sense the value in its current cell and the neighboring eight cells, enabling gradient estimates. Robots can also communicate with neighbors within a communication range d_c, which we vary in the experiments. This reflects a variety of communication with different ranges, from local line-of-sight light beaconing (as in Kilobots [80]) to Bluetooth to cellular communication.

6.1.3 Environment Model

Environments are generated by monochrome, multi-octave 2D Perlin noise [163], which is normalized to cover $[0, 256)$. This tunable procedure generates smooth, multi-scale textures and is used in computer graphics to model naturally occurring phenomena, such as terrain and smoke. We chose it as an abstract representation for many possible types of features that could be investigated using the algorithms presented in this chapter. The multi-scale nature of Perlin noise allows us to easily tune the environment complexity by adjusting the number of octaves (layers) to add smaller-scale noise in the environment.

6.2 Algorithms

We created a two-stage algorithm, where robots switch from target localization (pre-decision) to dissemination (post-decision). In addition, we compare performance to a benchmark in which robots conduct a pre-defined sweep over the environment.

6.2.1 Decision Algorithm

Movement: Each robot moves from the home corner to a random location in the environment and is then assigned a random virtual velocity V^t. Robots then switch to the movement algorithm

described in Section 6.2.2. As long as the robot has not yet found a suitable target location X with $v \leq v^*$, the robot continues executing the pre-decision movement algorithm.

After identifying a location X with value $v \leq v^*$, a robot switches to one of three possible movement algorithms to disseminate the decision, as described in Section 6.2.3: (1) Flocking behavior with other robots, (2) dispersing away from nearby robots, or (3) continuing the pre-decision movement.

As robots can only move one cell per tick, executed paths are generated from V^t by Bresenham's line algorithm [164].

Observation: Every tick, robots observe the value v at their current position X^t. If the observed value is lower than any previous observation the robot made, it updates its personal best value v^p, its position X^p, and the observation time t^p; if lower than any value that it knows, the robot updates its global best value v^g, position X^g, and time t^g.

Communication: Throughout each trial, robots continuously communicate with any robots within the communication range d_c. Each robot maintains a table of the information received, containing the following for each transmitting robot: its ID, best value $v^{g\prime}$, value's location $X^{g\prime}$, tick the entry was added t_{add}, and the position X^t and velocity V^t of the transmitting robot when the message was received. Entries in the table expire and are removed after a duration t_{rx}, increases the robustness of the collective decision; if one or more robots prematurely returns to the origin or entered a failure state, the remaining robots would still be able to complete their collective decision without needing to hear from that robot again.

Robots send messages with the values described above, as well as their own neighbor table. When receiving a message, a robot adds or updates its table entry for the sending robot and incorporates the neighbor's table by updating with any newer values. Note that t_{add} does not change for entries in the received table, as this represents the time when the observation was first transmitted. For any new values, the receiving robot will update v^g, X^g, and t^g if a lower value was received.

Ending Conditions: There are two different experimental conditions determining when robots complete their run. In the collective awareness condition, robots conclude their run and return to the collection point when they believe that all robots know a target location. This is done by checking whether all robots in their current neighbor table know a location with a value below v^*. When all robots have returned, the trial is finished. In the second case, the robots have a fixed maximum duration $t_{\max}$. They may return to the collection point early if the collective awareness condition is met, but they will always return to the collection point by $t_{\max}$, regardless of collective awareness.

6.2.2 Pre-decision Movement

Robots balance exploration and exploitation with a combination of PSO and gradient descent, with a variable velocity update interval. For each update interval Δt, a robot generates a new intermediate velocity $V^{(*)}$:

$$V^{t+\Delta t(*)} = \omega V^t + c_{\mathrm{p}} r_{\mathrm{p}}^t \left(X^{\mathrm{p}} - X^t\right) + c_{\mathrm{g}} r_{\mathrm{g}}^t \left(X^{\mathrm{g}} - X^t\right) + c_{\mathrm{GD}} r_{\mathrm{GD}} \nabla f(X^t) \tag{6.1}$$

The first term, with inertia coefficient ω, limits the change in velocity that can occur in each update. The second term moves a robot toward its personally observed best location X^{p} from its current location X^t, while the third term does the same for the best location known to the robot X^{g}, either from its own observations or communication. These two terms have random coefficients $r_{\mathrm{p}}, r_{\mathrm{g}} \sim U(0,1)$, which make this a random walk biased toward the best known locations. If the fixed coefficients $c_{\mathrm{p}}, c_{\mathrm{g}}$ are too large, this walk will be biased too strongly toward a local minimum; if too small, the observed optima play a negligible role in the movement. The final gradient term allows for exploitation of local observations to move toward a minimum. For these grid-based robots, the gradient is the direction of the neighboring cell with the smallest value. This term also employs a random coefficient $r_{\mathrm{GD}} \sim U(0,1)$.

To generate the virtual velocity used for future updates, the intermediate velocity $V^{(*)}$ is normalized to a maximum speed $V_{\max}$:

$$V^{t+\Delta t} = \min\left(V^{t+\Delta t(*)}, V_{\max}\frac{V^{t+\Delta t(*)}}{\left\|V^{t+\Delta t(*)}\right\|}\right) \tag{6.2}$$

Each robot updates its virtual velocity after Δt ticks. Preliminary experiments showed that traditional PSO frequently failed by getting stuck in local minima. We therefore added this variable update interval Δt to escape local minima.

$$\Delta t = \min\left(128, t^{\mathrm{p}}\right) \tag{6.3}$$

If v^{p} was observed recently, this generates a local search around where the value was observed. To prevent capture in local minima, Δt increases when no better values are observed, which increases exploration. The maximum Δt of 128 ticks was selected from pilot experiments.

6.2.3 Post-decision Movement

During the initial pre-decision search, robots spread out to explore different regions. We investigate whether a specialized movement strategy following an individual decision improves dissemination of this knowledge through the group. We propose two options: (1) flocking, which creates a loosely connected network and (2) dispersion, which causes mixing.

Flocking

Robots update their velocity according to Boids flocking strategy [123], which was created to simulate bird flocking. It has since been used extensively to create flocking behavior in robot swarms [165,166,167] because robots can prioritize maintaining a connected network without strict enforcement by a cen-

tral controller. The Boids model updates the agents' velocities based on the combination of alignment toward, cohesion with, and separation from neighboring agents. Here, neighbors include all robots communicated with since the last velocity update. We define alignment as matching the sum of the neighbors' velocity vectors V_k, and model cohesion and separation by the Lennard-Jones force F_{LJ} [168]. A robot i updates its velocity based on the positions and velocities of its k neighbors:

$$V_i^{t+1(*)} = 2V_i^t + \frac{1}{N}\sum_{k=1}^{N} V_k^t + F_{i\mathrm{LJ}} \tag{6.4}$$

and apply the normalization in Eq. 6.2.

$$F_{i\mathrm{LJ}} = \frac{1}{N}\sum_{k=1}^{N}\left(-\left[a\left(\frac{d_{\mathrm{t}}}{|r_{ik}|}\right)^{a} - 2b\left(\frac{d_{\mathrm{t}}}{|r_{ik}|}\right)^{b}\right]\right)\hat{r}_{ik} \tag{6.5}$$

where r_{ik} is the vector between robots i and k, and d_{t} is the target distance between robots. We fix d_{t} at 75% of the communication range to maintain a communication network. a and b are constants that determine the intensity of the forces; we used the standard values of $a = 12$ and $b = 6$.

Dispersion

To disperse robots, we use only the separation/cohesion term of the velocity update in Eq. 6.4, but set the target distance d_t to $10 \cdot d_{\mathrm{c}}$. Robots become close enough to communicate for a single tick and exchange messages, then separate again.

6.2.4 Benchmark Movement: Coverage Sweep

In addition to the algorithms described above, we also implemented a benchmark in which robots perform a collective lawnmower sweep over the environment, as shown in Fig. 6.2. Robots begin in the corner, then spread into a line separated by 2 cells to maximize coverage without overlapping

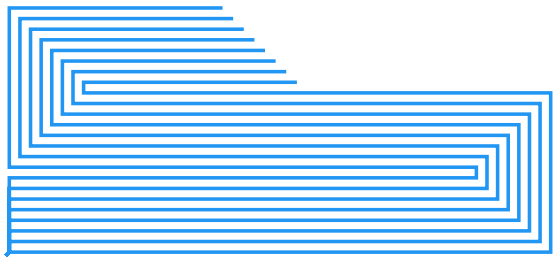


Figure 6.2: Example paths of robots in benchmark coverage sweep. Robots sweep until locating where $v \leq v^*$ and all robots know the location.

sensing ranges. They then sweep over the whole environment until a target location is identified. If all robots are in a connected network (which occurs for all cases with $d_c > 4$), this information is disseminated to the group within a few ticks, and all return to the origin. If not in communication, the robots must sweep the entire environment before returning.

6.3 Experiments

We conducted experiments in Kilosim, an open-source simulator we developed for high-throughput robot swarm simulations [124]. Additional code for this chapter is available on GitHub [169]. All simulations were run in a 384×384 cell arena. This is large enough to allow a sparse density of robots, with large-scale features and local noise. The precise dimensions were chosen as a multiple of the robot group size to easily generate benchmark sweep paths.

In all simulations sets, we varied the following, which allow us to understand the effect of the swarm and environment on the algorithm:

- Number of robots n: $\{8, 16, 32\}$
- Communication range d_c: $\{4, 8, 16, 32, \text{global}\}$
- Environment octaves: $\{1, 3, 5\}$

The varied octaves used to generate the environments correspond to three different difficulties constructed from Perlin noise, as shown in Fig. 6.1, where each cell is a pixel in the generated image. The parameters of the texture generation were a frequency (scale) of 100, lacunarity (change of scale per octave) of 2.1, and persistence (change of intensity per octave) of 0.5. These were selected from pilot experiments to provide a variety of feature scales that influenced robot behavior. We generated 50 images per difficulty, to be used with the corresponding trial. In all conditions, we used a fixed threshold of $v^* = 1$ to simplify experiments.

6.3.1 Pre-decision Simulations

We first conducted a parameter sweep to choose parameter values for the pre-decision movement (Eq. 6.1). The goal was to identify values that minimized the time for a first robot to locate where $v \leq v^*$. In addition to the variables described above, our parameter sweep covered the following:

- PSO inertia ω: $\{0, 0.5, 0.75, 1.0, 10\}$
- PSO weights c_p, c_g: $\{0, 0.01, 0.025, 0.05, 0.1, 1\}$
- Gradient weight c_{GD}: $\{0, 4, 8, 16\}$
- Maximum virtual speed V_{max}: $\{2, 25\}$

Parameter ranges were selected from pilot experiments. Note that the personal and collective PSO weights are paired, to constrain the size of the parameter sweep. We conducted 50 trials for each of the resulting 10,800 parameter combinations. Each trial was capped at 5,000 ticks; if a source was not found in that time, we considered it a failure.

6.3.2 Post-decision Simulations

After selecting the parameters for a single robot to locate the target, we conducted a parameter sweep for the post-decision strategy. We varied the movement strategy, as described in Section 6.2.3. The neighbor table timeout t_{rx} was fixed at 512 ticks to balance hearing from neighbors while avoiding unnecessary delays. For the collective awareness ending condition, trials were capped at 20,000 ticks. For the time-based ending condition, we set the maximum duration $t_{max} = 8000$ ticks.

6.4 Results

6.4.1 Pre-decision

We first look at the effect of parameters on time for a single robot to locate a target, seen in Fig. 6.3. Across all conditions, the performance of a 32 robot collective was hardly impacted, likely due to the density of robots; regardless of the parameter selection, at least one robot was close enough to a target location to quickly identify it. This demonstrates that the algorithm is scalable. Increasing the number of robots is therefore the best way to improve performance, creating group that is robust to parameter selection.

The trends in parameter effects held across all environments, so here we present the results for environments with five octaves of Perlin noise, which is the most challenging, with small-scale noise and local minima. The parameter effects also become more pronounced for smaller groups. We found that inertia reduced performance (Fig. 6.3A), likely by minimizing the responsiveness of robots to locally observed information. In physical robots, inertia is often inevitable, and we see that plausible real-world inertia of $\omega = 1$ had a small impact on localization time. A higher maximum speed V_{max} (Fig. 6.3B) allows more variation in velocities, particularly when inertia exists while preventing runaway values that occur if velocity is unbounded.

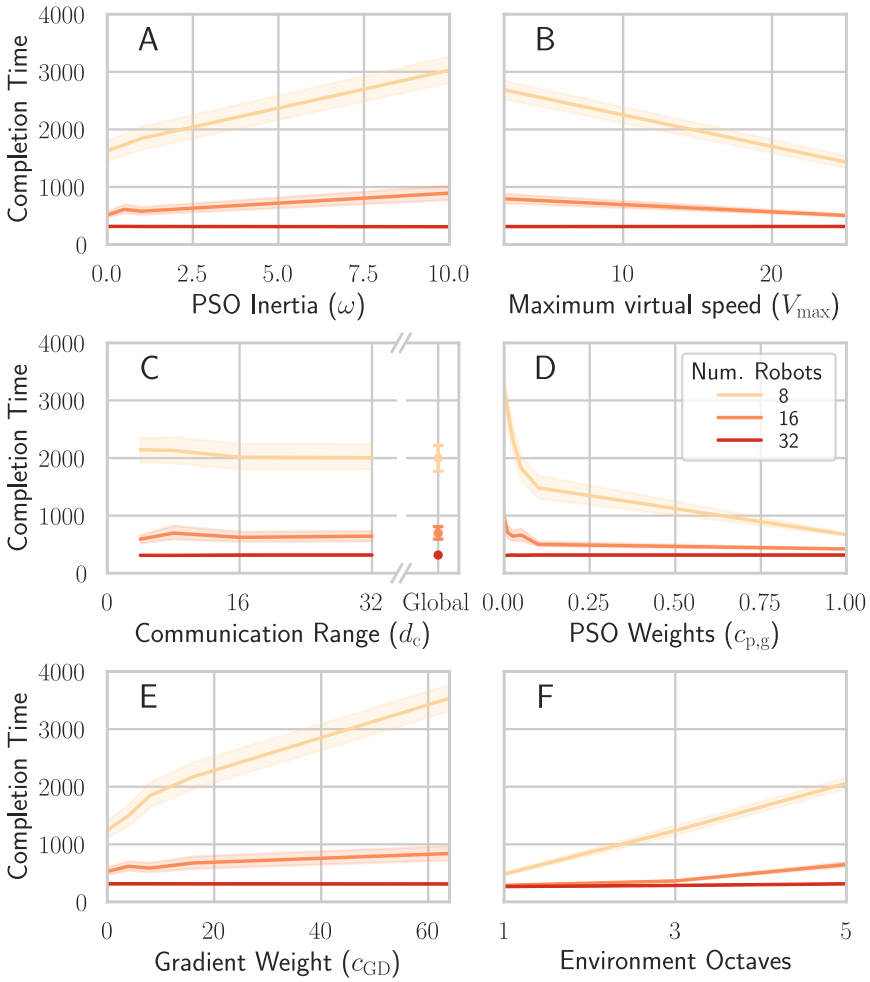


Figure 6.3: Parameter effects on time for first robot to locate target, for 5 octave environment. Lines and shading show median and 25th/75th percentiles, respectively.

Surprisingly, communication did not aid the search (Fig. 6.3C). As robots are exploring different regions, receiving information from robots exploring elsewhere can interfere with the local search. In fact, it could be advantageous to reduce or turn off communication before a robot makes a decision, as long-range communication is energy-intensive. This differs from traditional PSO, where global communication is assumed. Here, robots are not trying to congregate at the global maximum, but only identifying it. More significantly, robots have limited speed (unlike abstract particles), so physically distant information cannot be acted upon without a significant time delay to move to that location.

We see the most significant parameter benefit from increasing the PSO weights $c_{\mathrm{p,g}}$ (Fig. 6.3D). Given that long-range communication was not beneficial, this shows that robots benefit most from acting on local observations. However, if robots only moved toward their best observed position, they could become trapped in local minima. We show below how our algorithm prevents this complication.

We also see that increasing the octaves of Perlin noise in the environment increased the difficulty of the task (Fig. 6.3F), likely due to more spatial noise (therefore increasing the number of local minima) and often fewer positions below the decision threshold, seen in Fig. 6.1. However, in all environments, we found that employing the gradient in the search strategy did not improve search times (Fig. 6.3E). We hypothesize that this reactive component did not provide additional benefit beyond PSO; when paired with the variable update interval, PSO already allowed robots to react quickly to local information. This also demonstrates that robots do not need the more-advanced ability to detect or estimate gradients to complete this type of search task.

In Fig. 6.4, we also see the benefits of our approach by comparing to conventional PSO, where the velocity update interval is fixed. We ran a subset of our experiments ($n = 8, d_{\mathrm{c}} = 32$) with a fixed update interval and no gradient, and selected the parameters with the lowest median time to first target localization ($c_{\mathrm{p,g}} = 1, \omega = 1, V_{\max} = 25, \Delta t = 1$). Our approach allowed robots to

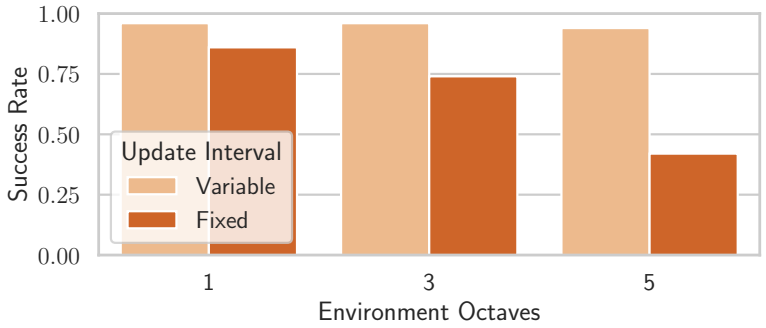


Figure 6.4: Success rate of locating the target within 5,000 ticks, for our algorithm (variable update interval) and traditional PSO (fixed updated interval). The fixed update interval did not allow robots to escape local minima in the higher noise (higher octave) environments.

successfully identify the target in noisier environments by allowing them to escape local minima. For traditional PSO, the short update interval trapped robots in these minima, while longer update intervals created overshooting instead of local investigation.

6.4.2 Post-decision

From the pre-decision results, we selected the best set of parameters to use for the post-decision simulations: $c_{p,g} = 1, c_{GD} = 0, \omega = 0, V_{max} = 25$.

In Fig. 6.5A-C, we can compare the success of different post-decision strategies. While initially locating a target did not require large-scale communication, we see that the small groups with limited communication failed to consistently disseminate target information within the 20,000 tick time limit. Overall, communicating target information while continuing to perform the search algorithm yielded the worst performance; communication does not factor into this movement approach. Flocking performs better, adding a communication component that allows robots to maintain a loose network once they meet. This means that any information obtained by one robot in the flock will be known to the whole group. Fig. 6.6 shows that flocking maintained the highest count of neighbors heard from each tick because of the network created. However, this results in

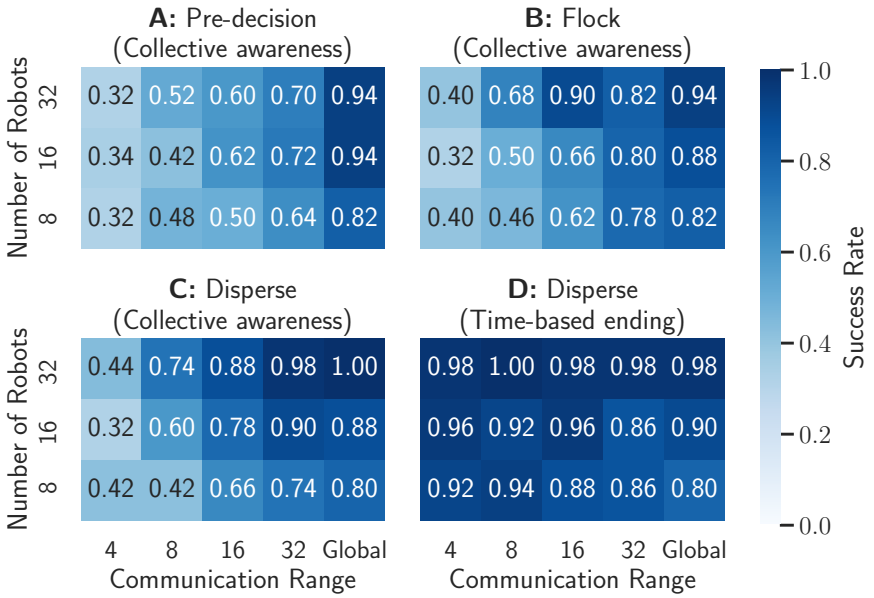


Figure 6.5: A-C: Success rate by post-decision movement type. Dispersion resulted in higher success by allowing robots to communicate with more robots. **D:** Success rate for time-based ending condition using dispersion, with $t_{\max} = 8000$ ticks. Allowing time-based termination improved success, especially for low-communication regimes.

the robots covering a smaller area of the environment, meaning they are less likely to encounter individuals unaware of the target. While a pre-existing flock will allow information to be transmitted within the group, the limitation in this scenario is that the robots must *form* a flock, which is non-trivial for sparse robots.

Dispersion had the highest success rate, as the algorithm does not prioritize *maintaining* communication with the group, but communicating with as many individuals as possible. This can be inferred from the higher success rate; communicating with more individuals resulted in spreading target knowledge to more unique individuals, thus completing the trial within the time limit.

In Fig. 6.5D, we see that adding the additional constraint of a time limit to the post-decision dispersion improved the success. If one robot knows of the target location, it can be disseminated when all the robots are collected at the origin. This prevents cases where the task fails to complete because a small number of robots are not communicated with before the 20,000 tick time limit,

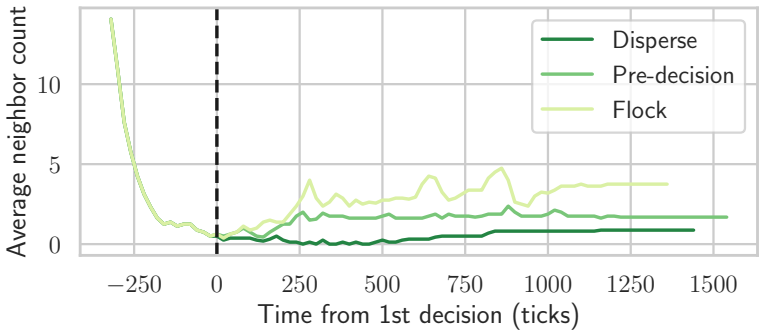


Figure 6.6: Example average number of neighbors for different post-decision dissemination movement strategies. Dispersion had the lowest average neighbor count, and the highest success rate.

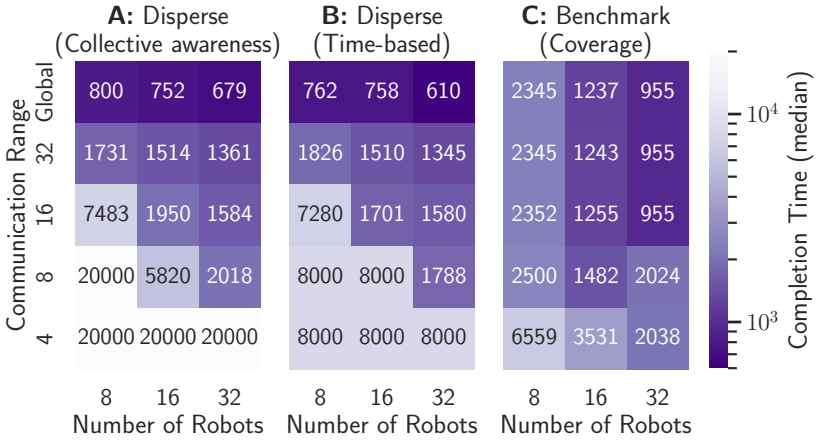


Figure 6.7: Comparison of median task completion time for 3-octave environments, on a logarithmic scale. Collective awareness and time-base ending conditions both used dispersion post-decision, but adding a time limit to the search reduced average search time.

which was most likely to occur in settings with small communication ranges. This time-based ending condition also represents a realistic system constraint, as robots typically have a limited mission duration due to battery constraints. In contrast to employing *only* a time-based ending condition 8,000 ticks, in Fig. 6.7B we see that allowing completion with the collective awareness condition allows for rapid decisions when communication is better — comparable to the collective awareness ending condition seen in Fig. 6.7A — but creates a backstop to prevent failures where a subset of robots do not learn of the target in the field.

In Fig. 6.7C, we see that the benchmark sweep is typically fastest, though for larger groups of robots, the hybrid algorithm is competitive. Because the benchmark is a coverage algorithm, it will also always locate a target. By maintaining a formation, any case with $d_c > 4$ maintains a connected communication network; if one robot finds the target, all robots will quickly learn it and the task can be terminated. However, even with small communication ranges, we find that the sweep completes the task faster; the robots only need to cover the environment, rather than continuing to wander to communicate. With global communication, robots in Fig. 6.7A and B will complete the task as soon as a single robot locates the target, meaning that they only utilize the variable-update-interval PSO stage of the algorithm. Here they are faster than the benchmark because they more quickly explore different areas of the environment, while robots in the sweep are always in the same region. Despite the apparent success of the benchmark, it assumes perfect, synchronous motion of the robots, which is difficult to achieve in groups of physical robots. In contrast, our hybrid algorithm does not require synchronized movement, and unlike flocking, the post-decision dispersion requires no coordinated movement.

6.5 Conclusion

We have shown an algorithm for the multi-stage task of robot target-searching with a continuous cue: locating the target, communicating this information to the rest of the group, and concluding the task by returning to their deployment position. This demonstrates the potential of creating complex behavior by thoughtfully combining variations on existing algorithms. In turn, this ability opens the possibility to employ simple robot collectives in autonomous inspection tasks: robots in this scenario were able to complete the task without centralized control, global communication, or inter-robot motion coordination.

We found that a form of PSO with variable update intervals allowed robots to locate a target without large-scale communication. To disseminate this, dispersion proved best at spreading information, rather than forming a flock to create and maintain a communication network. This counter-intuitive result stems from two challenges: forming a flock is challenging for physically distributed robots, and maintaining network limits the ability to spread information across in a large environment. When we also included a realistic time constraint representative of battery limitations, we were able to nearly double the success rate for low-communication regimes, while maintaining fast decision-making for larger groups with larger communication ranges. For these groups, our algorithm was competitive with the benchmark sweep algorithm, but without tight constraints on coordination.

While this algorithm was demonstrated in simulation with an abstract environment model, we expect the results will hold on physical robots because it does not require complex coordination, movement, or sensing by robots. In future work, we plan to implement this algorithm on physical robots and extend it to more complex environments and cues beyond our Perlin-based terrain model. [170] has demonstrated that PSO can be conducted with obstacles and inconsistent signal sensing. We intend to apply our algorithm to fault inspection tasks containing similar challenges, with

the goal of creating a system that can be used to inspect infrastructure such as bridges or space stations without requiring full environment coverage.

Anything humans can do in space, robots can do better.

Trevor Paglen

7

Decision-making Applied to Space Station Fault Inspection

In the preceding three chapters, I have presented algorithms to solve abstractions of inspection problems. At the start of this dissertation, however, I presented a more concrete application of inspection: identifying structural faults on the exterior of a space station. This provides an oppor-

Figure 7.1: Artist rendering of proposed NASA Lunar Gateway space station. (Image courtesy of NASA Johnson Space Center[173])

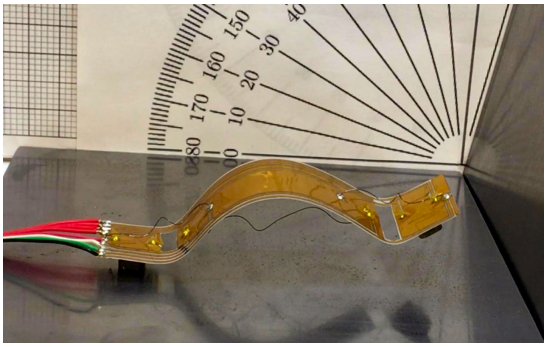
tunity to validate many aspects of the more abstract representations used so far. Inspecting space station structure is already a topic of concern for space agencies. In 2019, slow air loss was detected in the International Space Station (ISS). However, it took astronauts over a year to identify the location of the leak, as the rate of air loss increased, and apply a patch[171]. While this particular leak posed a minimal risk to the astronauts aboard, it demonstrates the inherent dangers in space station living, especially as they age. Robotic inspection for space stations has the potential to identify faults early, increasing safety and prolonging the operational lifespan of space stations.

These inspections are especially prescient as NASA develops the Lunar Gateway space station this decade. An image of the proposed design can be seen in Fig. 7.1. The Gateway will serve as a long-term hub in lunar orbit for human missions to the moon, which will require years of service, possibly including unmanned periods where the space station will need to operate fully autonomously[172].

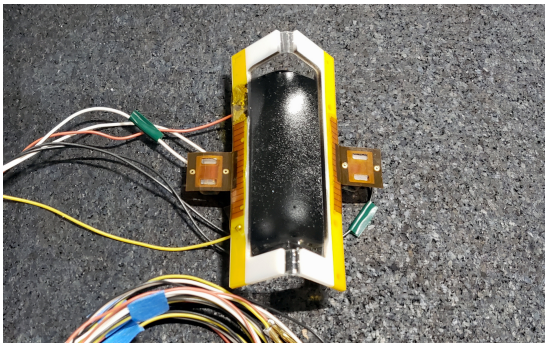
Space station inspection presents particular domain-specific challenges. One could deploy fixed sensor nodes on the surface of the space station, but these statically deployed sensors would be subject to failure from the same debris and cosmic rays that damage the space station itself. To provide sufficient coverage of the surface, they would also need to be placed at high density, significantly increasing cost and flight payload. In contrast, mobile robots can provide the same coverage with fewer sensors, while providing robustness to failure of individual agents [174]. They can also be kept safely inside the space station when not deployed.

We must also have robots capable of traversing the potentially complex exterior surface of a space station. This surface can be classed as 2.5D: a curved surface with obstacles. Space stations are also in microgravity, so we cannot use gravity to maintain the robots' connection to the surface. This can prove challenging for traditional wheeled robots. Therefore, we investigate the possibility of robot inspection using a soft-bodied robot using an inchworm-like locomotion. This design is simple to control, and its gait can overcome obstacles and variations in the surface. Our model system is the Ferrobot robot [175], seen in Fig. 7.2. In different variations of this robot, its gait is actuated by a shape memory alloy (SMA) (Fig. 7.2a) or the faster dielectric actuator (DEA) (Fig. 7.2b). In both cases, it attaches to the surface using switchable electropermanent magnets (EPMs) in the feet at each end of the robot's body. This allows the robot to attach to any ferromagnetic surface, regardless of the presence of gravity.

In addition to movement, robots must be capable of detecting damage to the structure. We selected vibration as our inspection modality, because it is well-studied for structural inspection applications [176]. Vibration sensing and analysis methods are based on detecting changes in vibration response on a known structure or surface [177,178]. Vibration-based inspection is nondestructive and noninvasive, making it safe for both healthy and damaged surfaces. Traditionally, however, vibration sensing has been used with fixed sensor networks [179] or a single mobile robot [180], rather than leveraging the power of a group of robots as a mobile sensor network.



(a) First ferrobot robot prototype, with EPM feet and SMA-actuated soft body. (Image courtesy of Jeremy Wanner.)



(b) Ferrobot robot with DEA-actuated body and EPM feet. (Image courtesy of Bahar Haghighat.)

Figure 7.2: Ferrobot robot models used for space station fault inspection studies. Both feature a bending soft body and two EPM feet for attaching to ferromagnetic surfaces.

In this chapter, I present work toward developing a collective of autonomous robots for inspecting the exterior surface of a space station such as the Lunar Gateway. First, I present applied physics-based simulations representing vibration sensing for a multi-robot, multi-source inspection task on a 2.5D surface. Second, I present hardware validation that the proposed sensing and movement are achievable in microgravity.

This work was conducted in collaboration with the Space Exploration Initiative at the Massachusetts Institute of Technology Media Lab. Particular contributions were made by Bahar Haghighat Johannes Boghaert, through his master thesis [109], and Fangzheng Liu. Portions of this work have been published 5th International Symposium on Swarm Behavior and Bio-Inspired Robotics (SWARM5) [181]. It was funded by a NASA Space Technology Research, Development, Demonstration, and Infusion (REDDI) grant.

7.1 Multi-source Fault Localization on a Simulated Space Station Surface

Our goal is to demonstrate that multi-robot fault inspection is possible on a simulated space station surface, with realistic assumptions of the fault propagation and robot behavior. We particularly consider the case where there are multiple possible faults on a surface with or without obstacles. Robots must identify the locations of all faults, but the number of faults is not known *a priori*. We demonstrate this using a collective of eight Ferrobot robots on 2.5D surface, where faults are represented by propagating vibration sources. We solve this problem with a multi-part algorithm built on particle swarm optimization (PSO), combined with niche formation for source confirmation, and a coverage target to guarantee that all faults have been discovered.

7.1.1 Problem Statement

We can formally define this inspection task as repeated localization of any number of failure sources on a 2.5D surface in microgravity, using a collective of robots that use vibration signals as a cue, until they meet a completion condition.

We define a failure source as a featured that disturbs normal functioning of a system. Detecting a failure source requires knowing the functional state of the system to compare against. In a system such as a space station, we expect that the system can be initially well-modeled and classified in its functional state, creating a baseline against which to compare. We expect that failure sources such as cracks and fissures on the surface will result in the creation of specific vibration profiles that are detectable in, and classifiably distinct from, the presence of the endemic vibration of the system. In our model of failure sources, we further simplify the points of mechanical failure as sources of induced vibration. We consider this equivalent to sensing a signal that has already been filtered for the distinct vibration signal of a fault. We model the vibration source as an external force applied to

the surface, following a sinusoidal pattern at a frequency of 1 Hz and an amplitude of 1 N. This falls within the mid-frequency range of the vibratory regime of the ISS [182]. The amplitude was chosen such that the resulting acceleration values are within the ISS acceleration spectrum, ranging from below microgravity to 10 milli-g [182].

The cue used by the robots is the magnitude of the acceleration signal measured during the search. For inspecting space station surfaces, we hypothesize that the cue can be either an endemic or induced vibration signal.

7.1.2 Simulation Framework

The simulation framework serves as a virtual environment in which to deploy and study our inspecting robots. We use two main software components: ANSYS, which we use to create realistic vibration signals propagating on a surface that models a floating orbital structure; and the Webots robotic simulator [133], which we use for simulating the robot locomotion, sensors, actuators, and the environment surface. The pipeline and simulation setup is shown in Fig. 7.3.

Within Webots, we created three main components:

1. A model of the bioinspired Ferrobot. This is comprised of a multi-segment piece-wise approximation of the SMA-actuated soft body shown in Fig. 7.2a. The body is approximately $20 \times 100 \times 7$mm, and it uses an inchworming locomotion pattern.

2. Target surfaces that the robots move on and inspect. Examples of these can be seen in Fig. 7.6 and Fig. 7.7.

3. Fupervisor controller code that is responsible for passing the vibration data to the robots as the move over the surface. depending on their location at each simulation step. The supervisor is a black box that emulates an acceleration sensor, along with a processing unit that returns to the robots the maximum observed accelerated amplitude at their current location.

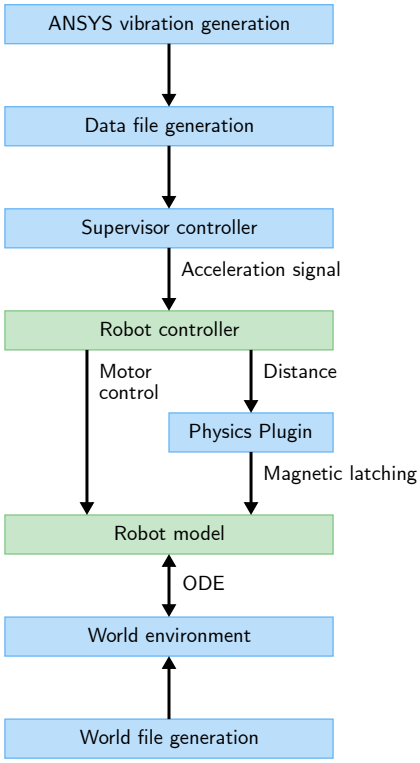
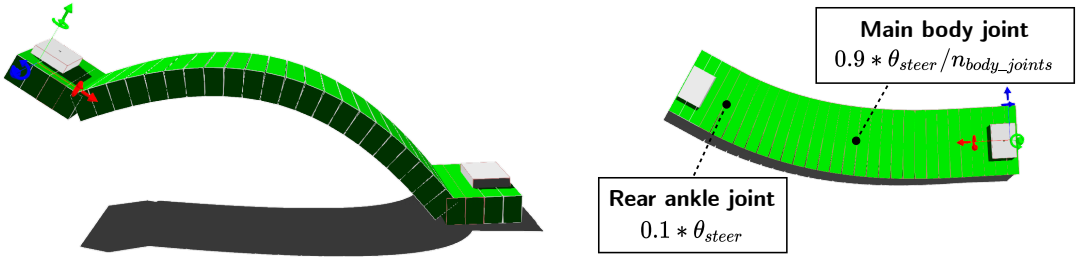


Figure 7.3: Simulation setup, with environment modeling components in blue and and Webots simulation code in green.



(a) Ferrobot extended, with front foot detached.

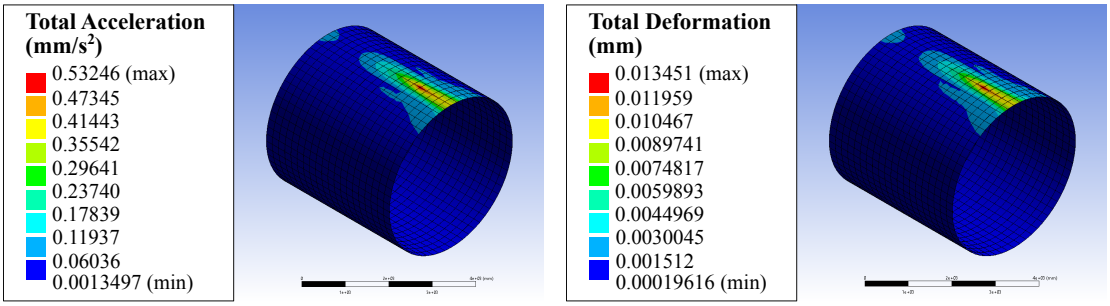
(b) Steering angle for body and ankle joints.

Figure 7.4: Simulated ferrobot robot model in Webots.

Of these pieces, the robot model is the most complex component. For the sake of brevity, and since this is not the focus of the current work, we will focus on the two pieces of the robot most relevant to our validation: (1) the inchworming locomotion, and (2) the sensing and communication capabilities of the simulated robots.

The robot model, as shown in Fig. 7.4, can move on flat and curved surfaces. The simulated robot has two connectors on its feet that emulate the switchable EPM connectors on the physical Ferrobot robot. The main body of the robot is a simulated flexible structure that bends along two axes of the body section, allowing it to contract and extend for forward movement, as well as steer left and right. It also bends at the two ankle joints to allow lifting the main body and adjusting the position of the front foot relative to the landing surface. The inchworming locomotion pattern allows the robot to achieve a step size of about half of its body size, or roughly 5 cm. The robots are assumed to have knowledge of the map of the environment, including the location of walls and obstacles, as well as their own location using a global positioning sensor. A loss-free global communication channel is assumed between the robots. The robots use this communication to share knowledge of faults, as well as their location on the map, which is then used to perform collision avoidance with other robots.

Within ANSYS, we use Transient Analysis to subject our surface model to a transient sinusoidal load case of 1 N at 1 Hz, which represents the vibration source. To represent the placement of the



(a) Acceleration **(b)** Deformation

Figure 7.5: Cylindrical surface models with elastic support boundary conditions in ANSYS.

surface in orbit, we use an elastic support boundary condition that involves the notion of foundation stiffness, express in $^{N}/_{mm^3}$. This is typically used to model soil-supported or submersed structures. We empirically set the foundation stiffness parameter to $0.0001^{N}/_{mm^3}$ by running a series of simulations and qualitatively evaluating the results in discussion with a human expert. The deformation amplitude for the applied load case is 0.013mm. Fig. 7.5 shows the surface models. In order to simplify the data processing and export, we create text files that approximate the acceleration signal observed in ANSYS with 2D Gaussian distributions. This data is then retrieved by the supervisor controller code in Webots and passed on to the robots depending on their location on the surface at each simulation step, as shown in the block diagram structure in Fig. 7.3.

7.1.3 Algorithm

The space inspection task, as we formally defined in Section 7.1.1, can be interpreted as a generalization of the source or target localization problem that has been discussed throughout this dissertation, particularly in Chapter 6, and in the literature [105,102,183,184]. The source localization problem can be divided into three sub-problems: (1) finding a cue, (2) tracing the cue to localize the source, and (3) confirming the source location. Our formal definition of the multi-source inspection task involves repeating these steps, as well as a termination condition based on a predefined coverage

threshold.

Algorithm 7.1 Inspection Algorithm Overview

```
1:  run Lévy Random Walk (RW)                                  ▷ Initialize
2:  while coverage < 80% do
3:      if robot in collision then                             ▷ Collision avoidance
4:          run Collision Avoidance (CA)
5:      else if cue picked up then                             ▷ Local search
6:          run Particle Swarm Optimization (PSO)
7:          if cue is a source then                            ▷ Source confirmation
8:              declare source
9:              run Directed Walk (DW)                         ▷ Re-initialization
10:             return to Lévy Random Walk (RW)
11:     else
12:         run Lévy Random Walk (RW)                          ▷ Global search
13:     update coverage                                        ▷ Update coverage
```

Source confirmation and reaching the coverage threshold require local and global search behaviors, respectively. We therefore define two goals for our inspection algorithm: (1) finding all sources present in the environment as fast as possible, and (2) reaching the global coverage threshold for task completion as fast as possible. Below, we describe our algorithm for the local search (i.e., the behavior that results in localizing a source in the environment) and the global search (i.e, the behavior that results in exploring the environment and reaching the coverage threshold for termination).

The algorithm structure is shown in Alg. 7.1. The algorithm contains four main control states, which we describe briefly here and in more detail below. In the absence of any prior cue sensing, a robot starts in the Random Walk (RW) state, performing an unbiased Lévy random walk around the environment until they sense a cue. Upon sensing a cue, the robot will begin a biased random walk toward the source. in the Particle Swarm Optimization (PSO) state. Once a robot has localized a source, it starts the Directed Walk (DW) start and moves to an unexplored area of the environment. At any point, if the robot is closer than a threshold distance to an environment obstacle or

another robot, it will switch to the Collision Avoidance (CA) state.

We use a multi-model variation of PSO, combined with a niche formation behavior, as our heuristic local search component. The PSO velocity update for each individual in each dimension j is as follows:

$$v_j^{t+1} = \omega v_j^t + c_\mathrm{p} r_\mathrm{p}^t \left(X_j^\mathrm{p} - X_j^t \right) + c_\mathrm{g} r_\mathrm{g}^t \left(X_j^\mathrm{g} - X_j^t \right) \tag{7.1}$$

where X^t is the robot's current position, and X^p and X^g are the positions of the best values observed by the individual and the group, respectively. The inertia term ω and c_p, c_g are weights to balance exploration and exploitation in order to find the maximum value of the cue — i.e., to locate the source. We use parameter values $\omega = 0.15$, $c_\mathrm{p} = 0.35$, $c_\mathrm{g} = 0.5$, selected from pilot experiments. To account for the constraints of the robot model and its locomotion, we also introduce restrictions on the velocity and the steering angle, which are passed to the robot controller.

Niche formation is part of the local search behavior, and it allows robots to confirm an identified source location. Once a robot is in the vicinity of a source and moves to the PSO state, it forms a two-robot niche by recruiting its nearest neighbor. The recruited robot then starts moving toward the location of the identified source. Once it also perceives a cue, it performs a PSO walk toward the source.

After localizing a source, a robot switches to a directed walk, moving toward an unexplored area of the environment. This is achieved with a sliding window over the environment, to identify the regions with the lowest coverage. The robot then performs a weighted sampling to select the target, where the likelihood of selecting a goal decreases quadratically with the coverage percentage.

Collision avoidance is based on Artificial Potential Fields (APF)[185]. The obstacles to avoid are created (1) from a map of the environment in which the boundaries of the area and the obstacles are marked, and (2) by communicating with other robots to obtain their location on the map. Each obstacle contributes a repulsion term to update the robot's velocity. For a repulsive term i in dimen-

sion j, we have:

$$r_{i,j} = c_{\text{weight}} \times \left[\frac{1}{d_i} - \frac{1}{\text{threshold}_i}\right] \times \left[\frac{X_j - \text{point}_{i,j}}{d_i^3}\right] \tag{7.2}$$

where d_i is the distance from the robot to obstacle i, X_j is the robot's position in dimension j and $\text{point}_{i,j}$ is the closest point on obstacle i in dimension j. The threshold is the distance to the obstacle below which the robot will engage in collision avoidance. Beyond this range, no collision avoidance is performed with the obstacle. Threshold and weight values can differ depending on the type of obstacle.

We require a coverage guarantee to ensure that, given enough time, all the sources that are present in the environment will be found. To achieve this, we deploy a Lévy random walk for the global search component. The Lévy walk assigns a random orientation to the robot and a random step length, following a Lévy distribution. This exploratory random walk guarantees asymptotic full coverage of the search space. We terminate the inspection earlier, based on a predefined coverage threshold calculated from the coverage map.

All robots have access to a shared coverage map, which is used to compute the total coverage. In a physical system, this could be achieved in a distributed system by robots sharing coverage over their communication channel, and updating their internal maps accordingly. The coverage map is represented as a grid of 10×10 cm cells. As the robots move through the environment, they update this shared map. It is updated based on a simplified sensor model, which represents sensing with a 2D Gaussian probability density function (PDF). To update the coverage map based on a robot's observation, the sensor model PDF is superimposed on the coverage map, center on the robot's position.

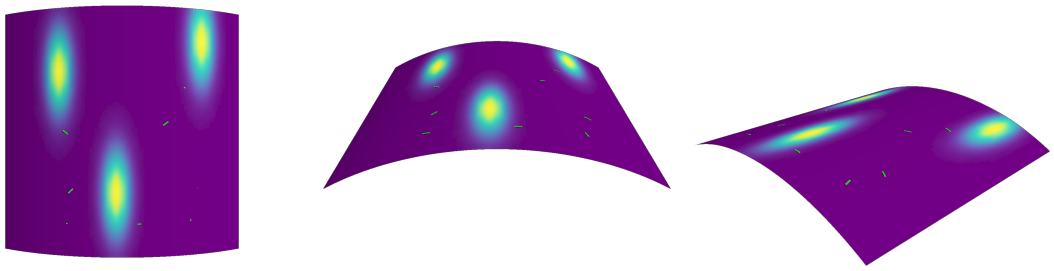


Figure 7.6: Webots simulation environments for Scenario 1.

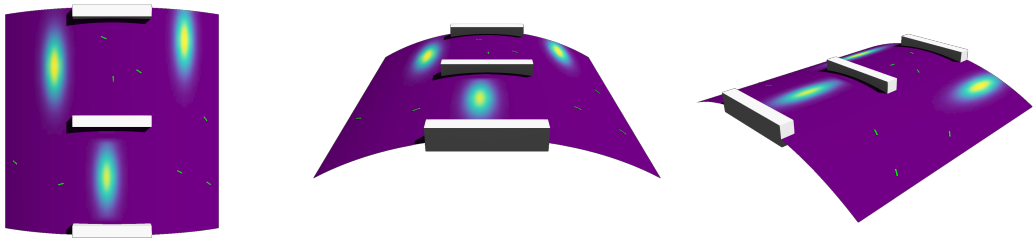


Figure 7.7: Webots simulation environments for Scenario 2.

7.1.4 Simulation Experiments

This section presents our experimental objectives and the two simulation scenarios used in our simulation experiments.

Experimental Objectives

We consider three primary objectives when evaluating the performance of the robot collective's inspection. These are related to but distinct from the performance metrics described in the next paragraph. In each experimental scenario, we would like to see whether the robot swarm succeeds at (1) localizing all sources (localization success), (2) reaching the coverage threshold for task completion (termination success), and (3) navigating the environment to follow cues without becoming trapped or stuck (maneuverability success).

To quantify the performance on these three goals, we consider three performance metrics. These

are based on similar metrics used in previous work on source localization and target search.[20,186]. In each experimental scenario, we quantify (1) the source localization accuracy (i.e., the distance of a robot-determined fault location from the fault's ground truth location), (2) the time to find each source in the environment, and (3) the time to reach the coverage threshold. To better understand the collective's dynamics, we also consider the time that robots spend in each of the algorithm's four primary control states: Collision Avoidance (CA), Particle Swarm Optimization (PSO), Random Walk (RW), and Directed Walk (DW).

Experimental Scenarios

We present two experimental scenarios to investigate a robot collective's ability to inspect a multi-source environment. This provides a tractable task that simplifies the complex real-world inspection task we aim to emulate. In each case, we deploy a group of $N = 8$ robots.

In Scenario 1, robots inspect a curved 2.5D cylindrical surface with a projected flat area of 4×4 m, shown in Fig. 7.6. The ANSYS simulation involves propagating vibrations through a full cylinder with a surface thickness of 2 mm, 4 m radius, and 6 m axial length. The vibration sources are located at (2m, 3m), (1m, 1m), and (3.5m, 0.5m) on the projected surface reference frame. The entire cylinder is subject to a foundation stiffness of $1 \times 10^{-4} \mathrm{N/mm^3}$, and the mesh is sized uniformly with nodes of 100×100mm. At the location of the vibration source, we apply a load case with a sinusoidal of amplitude 1 N and frequency 1 Hz. The peak amplitude at steady state at each location (i.e., after roughly 9.75s), is then considered for constructing the 2D Gaussian signal used in Webots (see Section 7.1.2). The ANSYS simulations revealed that the vibration propagation on a cylindrical surface is strongly biased along its length.

In Scenario 2, we extend Scenario 1 by increasing the geometric complexity with the addition of three rectangular obstacles to the environment. These represent features that might be present on the exterior surface of a space station, such as ridges or add-on modules. This environment can be

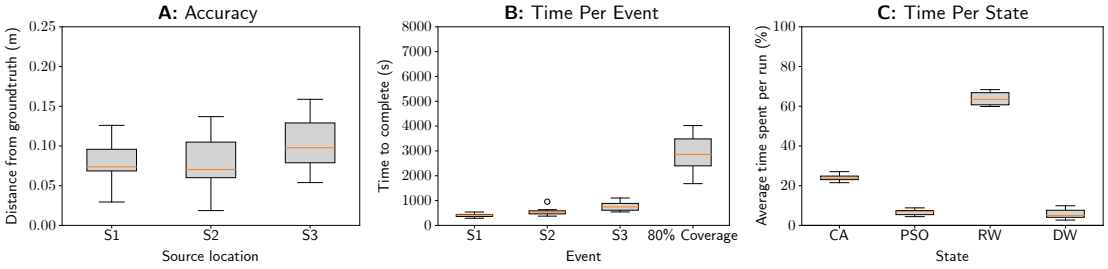


Figure 7.8: Performance results for Scenario 1.

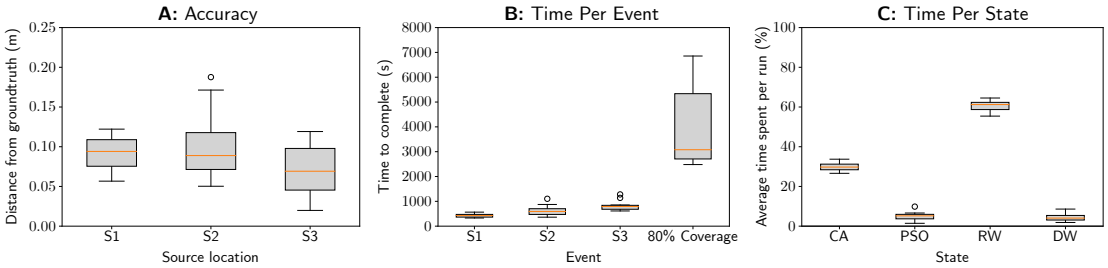


Figure 7.9: Performance results for Scenario 2.

seen in 7.7. We expect the obstacles to affect the propagation of the vibration signal on the surface, as well as the movement of the robots.

7.1.5 Results

We conducted 10 trials in each of the experimental scenarios described above. In each trial, the random seed for each robot is fixed, but the starting positions are randomized. We conducted a larger set of experiments with different robot group sizes to observe the effect of robot density on inspection performance, and selected $N = 8$ robots to present. This provides a high enough density of robots to leverage multi-robot behavior, while being low enough density to avoid excessive time spent in collision avoidance with other robots. Looking at the overall performance of our algorithm between the two experimental scenarios, we note how the swarm performance changes as the level of complexity in the search space increases.

The results are shown in Fig. 7.8 and Fig. 7.9. We consider the three metrics described in Section 7.1.4: source localization accuracy, time to locate sources and reach the decision threshold, and time spent in each control state. As the environment complexity increases from Scenario 1 to Scenario 2, we see that this impacts the time to locate each source and the inspection completion time, as defined by the time to reach the 80% coverage threshold, shown in Fig. 7.8B and Fig. 7.9B. The also corresponds to the time spend in the RW state between the two scenarios, seen in Fig. 7.8C and Fig. 7.9C.

In Fig. 7.8A and Fig. 7.9A, we see variation in the source localization accuracy, which which explain with three main considerations. First, when robots spend more time in PSO and less time in CA, they have a higher chance of achieving more accurate source localization. Second, the weight parameters used for PSO (Eq. 7.1) will affect the way that robots utilize their own observations and those of their neighbors to locate the source. However, these parameters have not yet been optimized, and we hypothesize that the best parameters will depend on the overall geometry of the environment. Finally, there is an interplay between the interval between observations (determined by the robot's step length), the shape and spread of the vibration cue, and the location of the sources in the environment. These likely play a role in how accurately a source can be localized. Why hypothesize that by optimizing the PSO parameters and the robot step size for a given search space, we can improve source localization accuracy.

7.2 Hardware Validation for Space Station Inspection

Through our simulation work, we have demonstrated an algorithm that can locate fault sources, but we still need to validate the assumptions that we made about the physical system. In particular, we must demonstrate that (1) the Ferrobot robots are capable of locomoting in a microgravity environment, and (2) they can detect the relevant vibrations in the surface. We achieved this by con-

ducting locomotion and vibration-sensing studies in parabolic flights and ground-based control experiments.

7.2.1 Experimental Setup

We created an 18 $\times$ 36 inch experimental box containing all experimental setups, which is shown in Fig. 7.10 and Fig. 7.12. This was designed to meet the safety and containment requirements for Zero-G flights. The enclosed upper portion of the box contains five ferromagnetic steel sheets of varying widths, secured to the box frame at each end. Each sheet is dedicated to an individual experimental setup; we will discuss the two used for Ferrobot locomotion and sensing.

Locomotion

As seen in the nearest plate in Fig. 7.10, a Ferrobot robot is attached to the metal sheet by its EPM feet. An interface layer of Kapton tape is applied to sheet to prevent short circuits of the active DEA and provide a low-friction surface for moving the EPM. The robot is attached to by a tether to a box placed beneath the frame. This provides power and control for the robot. While we endeavored to reduce the weight of the tether, it does create additional weight that the robot must pull when gravity is present. However, in microgravity conditions, the tether has no weight to affect locomotion.

The control box provides three gait options for the robot: move forward, move in reverse, and stop. The necessary control for the gait is shown in Fig. 7.11. The forward and reverse gaits are identical, except to switch which EPM is considered the front and which is the rear. Stopping the robot turns on both EPMs to ensure that the robot is attached to the surface.

Movement consists of four stages. The robot begins with both EPM feet on, and therefore attached to the surface. Then the front foot is turned off and detaches, while the DEA is turned on to extend the robot body. With the body still extended, the front foot is reattached. The DEA is then

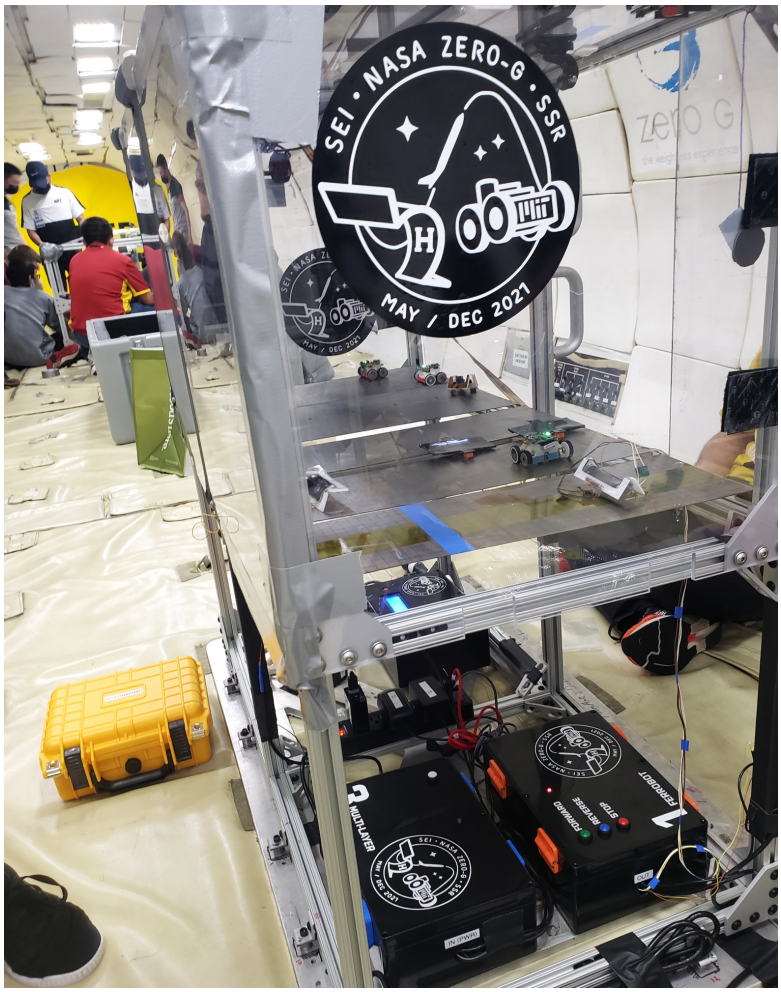


Figure 7.10: Experimental box for Zero-G flight testing. The walking Ferrobot is located on the nearest plate and tethered to control boxes placed underneath the frame. (Image courtesy of Bahar Haghighat.)

turned off, causing the body to contract, while rear foot is detached. Finally, the rear foot re-attaches to complete the gait cycle. The duration of each of these phases, as well as the offset of EPM and DEA switching, are shown in Fig. 7.11; these were determined from ground-based pilot testing.

Each EPM is turned on and off by applying a set of short, high current pulses through a MOSFET. As such, each is controlled by turning on a pair of pins that set both sides of the MOSFET, indicated in Fig. 7.11 as H and L. This figure represents the EPM switching as a single pulse, but it is actually composed of eight 50μs pulses separated by 20 ms. Multiple pulses are used to ensure that the EPM is fully switched, and the 20 ms gap between pulses allows the high current capacitor to recharge.

During parabolic flights, we tested the robot in three conditions: microgravity (approximately 0g), high gravity between microgravity parabolas (approximately 2g), and level flight (1g).

Sensing

Vibration sensing was conducted on a 20 cm wide steel plate within the experimental box, as shown in Fig. 7.12. We conducted both ground-based and flight-based tests of the Ferrobot's ability to sense induced vibrations. At one end of the surface, a 1 Hz pounding is induced by a pounding bar on a Rovable robot[187]. The pounding provides a vibration source comparable to that used in the simulation portion of this chapter. By inducing this with another robot, this opens the door to future extensions of coordinating the Ferrobot behavior as part of a more complex, heterogeneous robotic inspection collective.

While the pounding Rovable was kept at a fixed position on the plate, a Ferrobot robot body was placed at different locations along the sheet to investigate the range over which the pounding is detectable. This robot body was not autonomously actuated, but was manually moved between positions. An Adafruit BNO055 inertial measurement unit (IMU)[188] was attached to the robot body to detect the vibrations, and connected to a Raspberry Pi 3B+ collecting data at approximately

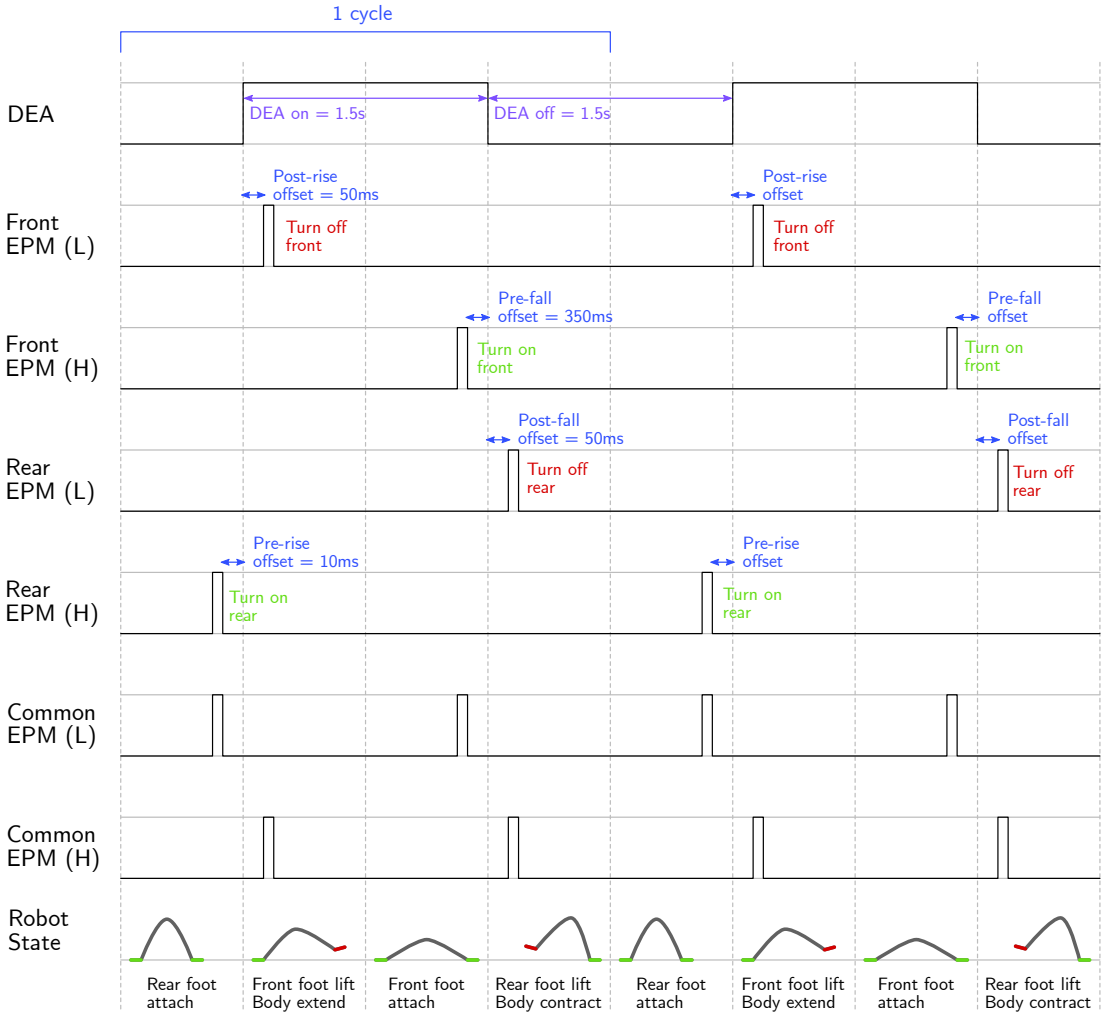


Figure 7.11: Schematic of the four stages of Ferrobot gait. The DEA state is set by a single control pin, while each EPM is switched by a series of pulses on a pair of pins controlling a MOSFET. Times are not to scale.

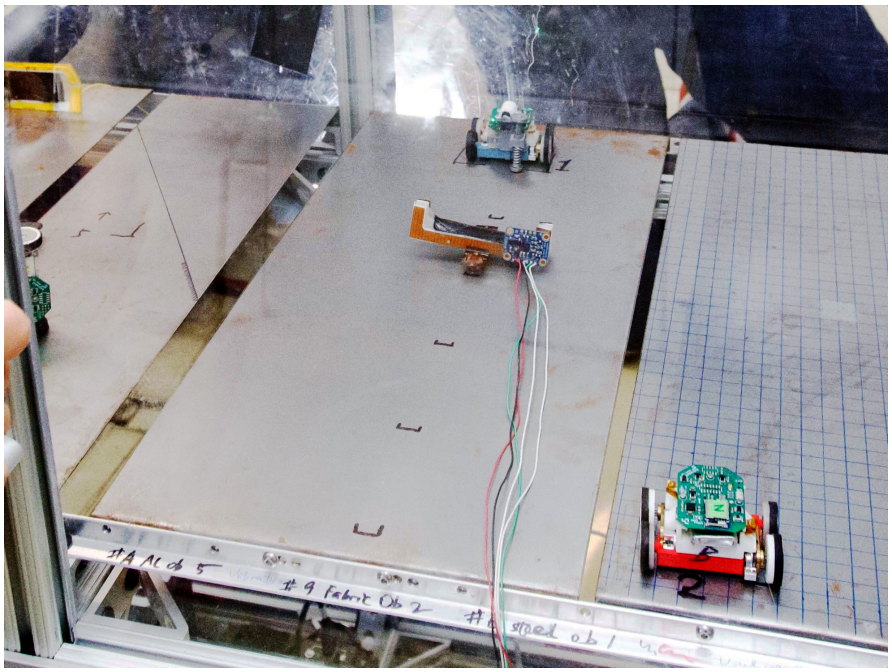


Figure 7.12: Experimental setup for testing vibration sensing on Zero-G flights and ground testing. Ferrobot with attached IMU is placed on a suspended metal sheet. It is moved to marked 7 cm intervals from the Rovable with a pounding bar at the far end of the sheet. (Image courtesy of Stephen Boxall/ZERO-G.)

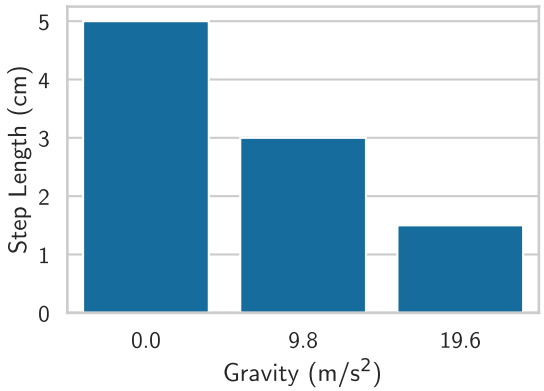


Figure 7.13: Approximate Ferrobot step lengths in different gravity conditions.

50 Hz.

7.2.2 Microgravity Locomotion Results

We evaluated the performance of the Ferrobot walking by measuring its step length under different gravitational loads, seen in Fig. 7.13. The step lengths were determined by visual inspection of video footage on the flight, and estimated using a 1 cm square grid on the plate surface. Under 1g conditions, we see that the robot achieves 3 cm long steps, but under a 2g load during the "pull" of the parabolic flights, this is halved to 1.5 cm. This expected behavior is likely the result of additional frictional forces of the feet on the plate surface, as well as the additional weight of the tether that provides power and control to the robot. In microgravity conditions, however, we see that the step length increases to 5 cm. Without the weight of the robot body or tether constraining its movement, the step length reached approximately 70% of the robot's 7 cm bodylength. This is particularly important, because microgravity is the intended use setting for this robot design. These results validates that a soft robot with an inchworming locomotion pattern can successfully move over a ferromagnetic surface in a variety of gravitational regimes.

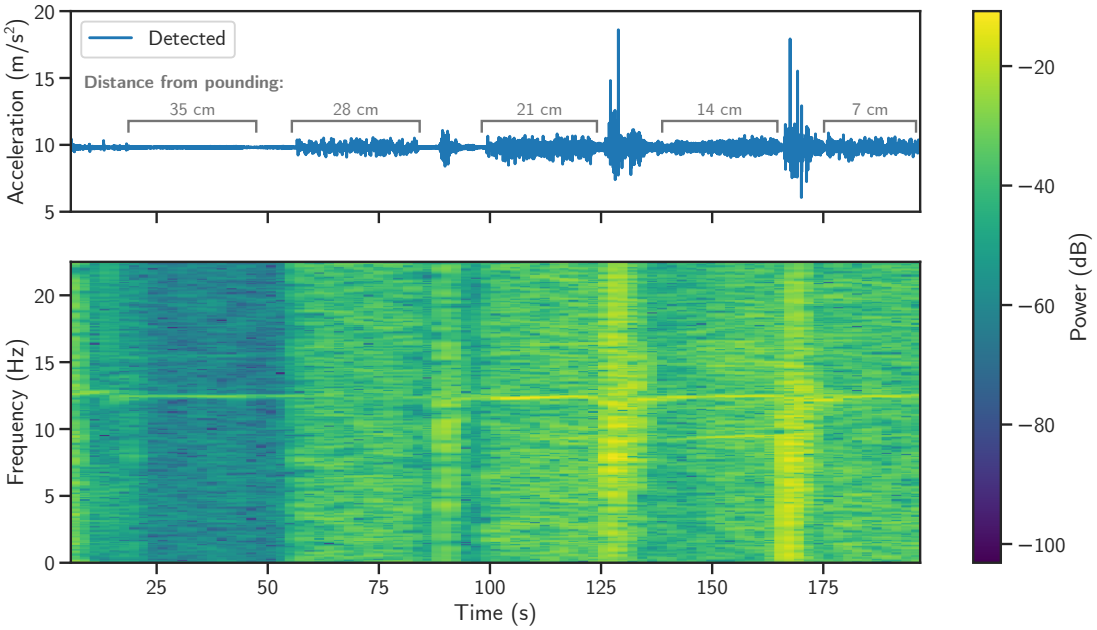


Figure 7.14: IMU-detected vibrations in ground-based testing. The Ferrobot was placed at 7cm intervals from a 1 Hz pounding source. **Top:** Total magnitude of acceleration. **Bottom:** Spectrogram of detected frequencies. 12 Hz signal is detected at all but one position.

7.2.3 Vibration Detection and Signal Processing Results

We tested the detected vibration response in both ground testing, and for the full duration of a Zero-G flight.

In Fig. 7.14, we can see the acceleration detected by the ferrobot IMU at different distances from the pounding Rovable, as it is moved closer over time. In the top subfigure, we can see that there is an increased magnitude of acceleration as the ferrobot as it is moved closer to the pounding. In the bottom subfigure, we see that the base frequency of 1 Hz was not directly detected, possibly due to damping in the metal plate or the compliant body of the robot. However, even at 35 cm from the pounding, there is a clear signal at 12 Hz. This is possibly a harmonic of the 1 Hz pounding, or a resonant frequency of the metal plate. Notably, this while the magnitude of this signal changes over

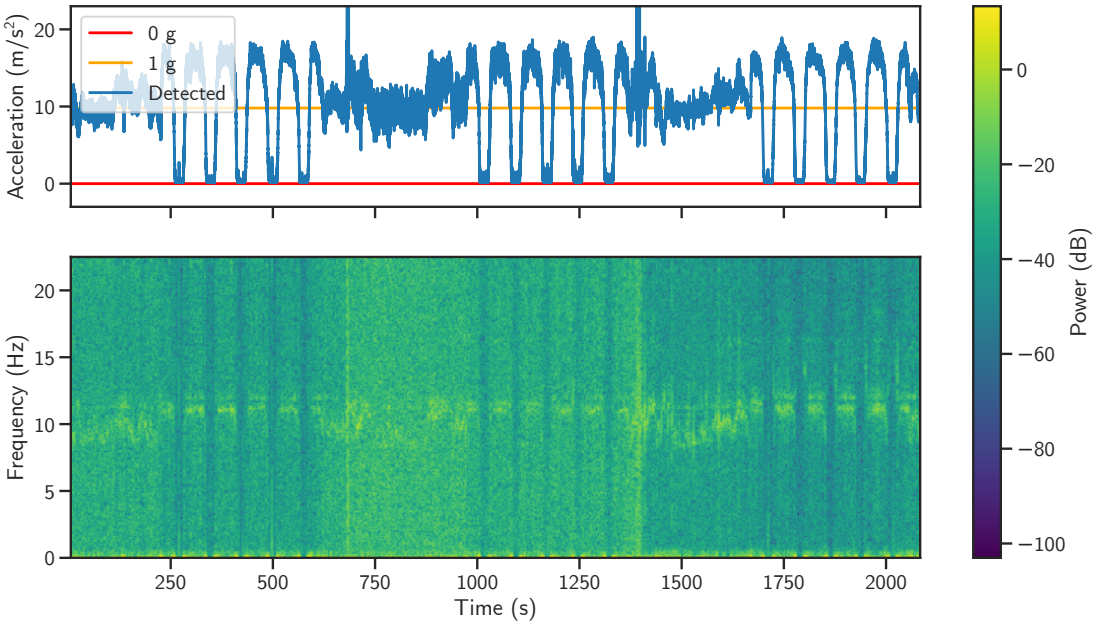


Figure 7.15: IMU-detected vibrations in parabolic flight testing. **Top:** Total acceleration detected by Ferrobot IMU. Parabolas dominate the sensing. **Bottom:** Spectrogram of frequencies detected during flight. 12 Hz signal is visible whenever plane engines are on, but absent whenever this background excitation is absent.

distance, the frequency response remains consistent.

In Fig. 7.15, we can see the acceleration detected during three sets of five parabolas on a Zero-G flight. Due to an error in data collection, we cannot match this acceleration data to which parabolas contained Rovable pounding, or the Ferrobot's distance from the Rovable. However, we do not see any clear signal in the spectrogram data that suggests a correlation between distance, pounding, and the detected frequencies. Throughout the flight, the spectrogram shows the same 12 Hz signal visible in ground tests. We see that the 12 Hz signal power drops off during the microgravity parabolas, which corresponds to the plane's engine shutoff, and the 12 Hz signal is strongest in the 2g portions of the flight. This suggests that the 12 Hz signal is not induced only by Rovable pounding (as was used in the ground testing), but by the plane's background vibration exciting the plate, which supports the hypothesis that the signal is a resonant frequency of the plate.

The consistent vibratory response of the plate to vibration means that we can use it to characterize the metal plate. In this case, we could consider the 12 Hz signal to mean that the plate is healthy. In contrast, we reasonably expect that a different configuration of the plate (such as adding a hole or a crack) would result in a different resonant response to vibration, allowing us to classify this variation as "unhealthy". We see that sufficient consistent background vibration, such as that of the plane engines, could be used to classify the quality of the structure. McPherson et al. [182] showed that background vibration is also present on the ISS, which could be used for similar classification in space stations.

Together, these ground and flight experiments demonstrate that it is possible to classify the health of a structure by its vibratory response to induced vibrations. We found that both Rovable pounding and background plane vibrations generated a sufficient signal to allow the ferrobot IMU to consistently detect a signature vibratory response of the metal surface, and this can be detected at least 35 cm away from the vibration source.

7.3 Conclusion

The particular real world scenario that underlies our developments is the need for inspection of the outer surface of long-term deployed spacecrafts. The goal of the inspection is to identify mechanical failures such as fissures and cracks resulting from natural wear and tear of the structures. In this chapter, we have demonstrated an algorithm to allow robots to detect multiple mechanical failure sources in a physics-based simulation of a space station surface. With physical Ferrobot robots, we have further demonstrated that these robots are capable of meeting the assumptions made in the simulation: they are able to effectively perform inchworming locomotion in microgravity, and they can utilize ambient excitation to identify resonant frequencies of a metal surface. Together, these demonstrate the potential to use a small soft-bodied robot collective to evaluate the structural in-

tegrity of a space station in orbit.

Our algorithmic work here focused on development and presentation of a framework for surface inspection using a collective of robots. We deployed our proposed algorithm on a simulated swarm of vibration sensing surface-crawling robots that use an inchworming gait for locomotion. Within the simulated world where the robots perform the inspection task, we modeled the points of mechanical failure on the surface under inspection as sources of vibration. The robots then used the signal that propagates through the surface as a cue for localizing the sources of vibration. We simulated realistic vibration signal propagation in ANSYS, then simplified data transfer by fitting 2D Gaussian functions to the simulation results. In the Webots robotic simulator, we investigated the performance of the swarm within two experimental scenarios comprising three sources on a 2.5D cylindrical surface and three sources on a 2.5D cylindrical surface with additional obstacles on the surface. Our algorithm succeeded at finding all the sources present in the search space in each of the two experimental scenarios. Additionally, we showed that we are able to reach a predefined coverage threshold as a termination criterion for the inspection task. Our results provide evidence supporting the viability of robot swarms for inspection of 2.5D surfaces based on sensing vibration cues on the surface.

Future work will involve extending our modeling and algorithmic framework in several ways, as well as improving the hardware validation. First, we plan to develop a fully automated simulation pipeline to facilitate randomized studies of a variety of environments. In particular, we plan to automate the currently manual process of simulating the vibration signal within ANSYS and transferring the corresponding data to a file format that is accessible by the simulated robots within Webots. Second, we plan to implement realistic constraints in the communication range and bandwidth used by the simulated robots within Webots. Third, given a specific search environment, we plan to leverage the automated simulation framework developed in this work to perform a parameter optimization in order to find the set of parameters (i.e., number of robots, local search behavior

parameters, global search behavior parameters, obstacle avoidance behavior parameters, etc.) that result in the best desired performance metrics.

For future hardware validation, we can confirm our hypothesis that different surfaces will result in different resonant responses that can be detected by the Ferrobot's IMU. Additionally, we can improve the locomotion of the Ferrobot robot. Currently, the simulation uses a fully soft-bodied robot model, based on the SMA-actuated Ferrobot, while the hardware validation was conducted with the DEA-actuated robot because of its faster walking speed. However, the current DEA-based Ferrobot has only a single actuator and therefore cannot turn, as is required in the simulation.

Finally, a future algorithmic extension is extending this inspection work to a heterogeneous robot swarm. As alluded to earlier, the combination of Ferrobots and Rovables could result in a more capable swarm than either individual. First, they have different locomotion (inchworming compared to magnetic wheels), which would make each suited to different regions of the surface; the Ferrobot's inchworming could allow it to cross small gaps in the surface, while the Rovable can travel faster to more distant areas. In addition, the pounding investigation demonstrates a potential for synergistic collaborations, where one robot could produce a signal which the other robot could detect the propagation of.

Divergence is interesting, but convergence is beautiful.

Johannes Boghaert

8

Conclusions and Future Work

At the outset of this dissertation, I presented an inspection challenge representative of the dirty, dull, and dangerous task of inspection: How can we employ robots to find cracks in a space station, such that we can maintain its structural integrity and prolong its operational lifespan? This is a challenge of matching the task to the right robot collective and the right algorithm to solve the task. We can achieve this by developing algorithms that allow us to coordinate the movement of large groups

of robots, share the right information, and integrate that information into collective decisions. We have not yet deployed inspecting robots to space structures, but in this dissertation I have moved toward this goal of creating algorithms for a broad variety of inspection tasks, including space station inspection.

8.1 Contributions

Traditionally, the case for automation and robot applications is tasks that are "dirty, dull, and dangerous," which includes many inspection tasks. However, to efficiently deploy robots for these inspection tasks, we must first fundamentally understand the behavior and trade-offs involved in using different types of robots and decision-making algorithms for inspection. This thesis helps lay the groundwork for an abstracted understanding of swarm behavior in inspection tasks. I have demonstrated algorithms for solving two different types of inspection tasks that represent the broad variety of applications within inspection: global site classification and target search. We achieved this using groups of simple robots with decentralized computation and communication. This results in algorithms that are robust to individual failure, which can be deployed on cheap and simple hardware for a variety of applications.

Historically, research in robotic inspection and swarm robotics have had limited intersection[82]. However, the identifying features of robotic inspection — sensing, classification, and mobility — map well onto the goals that researchers in swarm robotics are also trying to solve. In this dissertation, I have helped to close that gap by creating inspection-oriented algorithms built on techniques and behaviors within swarm robotics.

We are able to complete both types of tasks with fewer assumptions than previous work in multi-robot inspection. Our robots required limited communication range and bandwidth, limited on-board computation, and for global classification, they were not even capable of localization. All

computation was distributed across the group, with no centralized controller required. These principle also applied across a large range of group sizes, from 8 robots in Chapters 6 and 7 up to 100 robots in Chapters 4 and 5.

We were able to succeed at these tasks without relying on coverage, which meant that we did not need to maintain close coordination between robots. Global site classification could be conducted without localization by relying on random walks and integration of information from many individuals. While target search required robots with the ability to localize themselves in the environment, in many cases we were able to achieve performance comparable to coverage without relying on visiting every location in the environment — a strategy that can fail if robots cannot conduct a coordinated search or if individuals fail.

8.2 Future Work

There remains much work to do before robot swarms are inspecting our infrastructure. One of the most significant requirements is moving from simulation to reality. Chapter 4 presented validation of abstract simulation work on Kilobot robots, and Chapter 7 showed the beginnings of real world space station inspection. However, there remain significant gaps that the simulations do not address, such as the uncertainty of localization and noisy sensing. There are also many failure modes that we cannot model or predict, but would only present themselves on physical hardware in a true deployment setting. In addition, most of the simulations presented are abstract, representing the world as a monochrome 2D environment. In reality, the nature of the cue will vary depending on the inspection task, which can impact algorithmic choices. Future simulations can aim to more closely match features of real-world inspection tasks, which will also help to close the simulation to reality gap.

In both simulation and hardware implementations, we see potential for improving the perfor-

mance of our algorithms through improved optimization. The primary tool used for performance tuning in this thesis was parameter sweeps. However, when tuning multiple parameters simultaneously to understand their interactions, the number of parameter combinations scales poorly. This becomes intractable for algorithms with many parameters, and limits testing to faster-than-realtime simulation, rather than on physical hardware. In the future, we can employ advanced techniques for parameter tuning, such as evolutionary algorithms [189], reinforcement learning [190], and even the meta-solution of employing established PSO techniques for high-dimensional parameter tuning [191]. These techniques can also be incorporated into a combination of simulation and hardware experiments to maintain a close connection between the two.

In the future, we also hope to develop a more systematic approach to selecting the optimal movement and decision-making algorithms for certain inspection situations. As more swarm robotic inspection tasks are investigated, we can develop a more sophisticated taxonomy of inspection tasks than this dissertation's binary division of global classification and target search. For example, this could incorporate the nature of the cue (e.g., is it continuously or intermittently sensed, and can it be modeled). Combined with a taxonomy of swarm behaviors, as discussed in Chapter 2, we can develop a mapping between inspection tasks and algorithms that will allow for faster development of algorithms for particular robot capabilities, group sizes, and features of the task itself. This simplification will aid the adoption of swarms for inspection tasks in the real world.

Thus far, this dissertation has presented inspection with homogeneous swarms, where all robots in the group are the same. Employing robots with different capabilities can increase the capabilities of the group, but introduces additional challenges in algorithmic design. At the end of Chapter 7, we suggested a particular heterogeneous swarm application, where Ferrobots would team with wheeled Rovable robots to inspect a space station surface. This introduces differences in locomotion — a Rovable can move quickly, but a Ferrobot can overcome more obstacles — as well as potential different sensing payloads. It is also possible to introduce different computation and commu-

nication capabilities to a swarm, creating a semi-centralized group to take advantage of the benefits of both centralization and decentralization, such as reduced computational complexity and robustness, respectively. In general, however, the challenges of reasoning about heterogeneous groups have made the topic under-studied within both inspection and swarm robotics literature [192,29]. The case of Ferrobot and Rovable collaboration presents a clear use case and division of labor that will serve as a valuable test bed for further investigating the potential of heterogeneous swarms in inspection.

8.3 Complex Swarm Tasks from Fundamental Behaviors

Real world applications of robot collectives are typically complex tasks. As this study of robotic inspection has demonstrated, many fundamental behaviors must be combined to complete the task. Many other collective multi-robot tasks are similarly complex, like a robotic construction task that can require locating materials, collectively transporting them, and assembling a variety of complex sub-components like electrical wiring, structural framing, and facing. In fact, inspection itself can even become a sub-behavior of robotic construction to validate the resulting structure.

The swarm behavior taxonomy discussed in Chapter 2 shows a current understanding of these fundamental behaviors built from existing literature within the swarm robotics research community. However, this is only the start. By building stronger connections between swarm robotics researchers and application domain researchers, we can develop a stronger understanding of both the complex application tasks and the fundamental behaviors that can be used to compose them. We can begin to construct a hierarchical understanding of swarm behavior: Application tasks are composed of fundamental behaviors, which in turn can be achieved by different algorithms depending on the nature of the robots, group size, and environment.

As we move into a world where robots are not operating in isolation, and where these groups of robots will be completing increasingly varied tasks, it is important not to develop collective algo-

rithms with a myopic focus on a particular task, but rather maintain a perspective on the broader algorithmic context. This will allow algorithms to be developed faster for a broader variety of applications.

References

[1] A. B. Alhassan, X. Zhang, H. Shen, and H. Xu, "Power transmission line inspection robots: A review, trends and challenges for future research," *International Journal of Electrical Power & Energy Systems*, vol. 118, p. 105862, Jun. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0142061519324524

[2] A. Pagnano, M. Höpf, and R. Teti, "A Roadmap for Automated Power Line Inspection. Maintenance and Repair," *Procedia CIRP*, vol. 12, pp. 234–239, 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2212827113006823

[3] H. Ahmed, H. M. La, and N. Gucunski, "Review of Non-Destructive Civil Infrastructure Evaluation for Bridges: State-of-the-Art Robotic Platforms, Sensors and Algorithms," *Sensors*, vol. 20, no. 14, p. 3954, Jul. 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/14/3954

[4] B. Zhang, B. Gu, G. Tian, J. Zhou, J. Huang, and Y. Xiong, "Challenges and solutions of optical-based nondestructive quality inspection for robotic fruit and vegetable grading systems: A technical review," *Trends in Food Science & Technology*, vol. 81, pp. 213–231, Nov. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0924224417307380

[5] L. Yu, E. Yang, P. Ren, C. Luo, G. Dobie, D. Gu, and X. Yan, "Inspection Robots in Oil and Gas Industry: a Review of Current Solutions and Future Trends," in *2019 25th International Conference on Automation and Computing (ICAC)*. Lancaster, United Kingdom: IEEE, Sep. 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8895089/

[6] D. Lattanzi and G. Miller, "Review of Robotic Infrastructure Inspection Systems," *Journal of Infrastructure Systems*, vol. 23, no. 3, p. 04017004, Sep. 2017. [Online]. Available: http://ascelibrary.org/doi/10.1061/%28ASCE%29IS.1943-555X.0000353

[7] Perseverence, "Perseverence Mars Rover and Ingenuity Helicopter," NASA/JPL-Caltech/ASU/MSSS, Mars, Tech. Rep., 2021. [Online]. Available: https://i.imgur.com/WX8TT3t.jpeg

[8] W. S. D. of Transportation, "SR 433, Lewis and Clark Bridge Inspection," Mar. 2015. [Online]. Available: https://www.flickr.com/photos/wsdot/16905614502

[9] J. Sawada, K. Kusumoto, Y. Maikawa, T. Munakata, and Y. Ishikawa, "A mobile robot for inspection of power transmission lines," *IEEE Transactions on Power Delivery*, vol. 6, no. 1, pp. 309–315, Jan. 1991. [Online]. Available: http://ieeexplore.ieee.org/document/103753/

[10] S.-y. Jiang, Y. Hu, Y. Wang, H. Jiao, and L. Ren, "Development of Hanging-Arm Inspection Robot for High-Voltage Transmission Line," in *Intelligent Robotics and Applications*, C. Xiong, Y. Huang, Y. Xiong, and H. Liu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1089–1098.

[11] C. Carbone, O. Garibaldi, and Z. Kurt, "Swarm Robotics as a Solution to Crops Inspection for Precision Agriculture," in *6th Engineering, Science and Technology Conference (2017)*. KnE Engineering, Feb. 2018, pp. 552–562. [Online]. Available: https://knepublishing.com/index.php/KnE-Engineering/article/view/1459

[12] M. Edmonds and J. Yi, "Efficient Multi-Robot Inspection of Row Crops via Kernel Estimation and Region-Based Task Allocation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 8919–8926. [Online]. Available: https://ieeexplore.ieee.org/document/9560826/

[13] M. S. A. Mahmud, M. S. Z. Abidin, and Z. Mohamed, "Development of an autonomous crop inspection mobile robot system," in *2015 IEEE Student Conference on Research and Development (SCOReD)*. Kuala Lumpur, Malaysia: IEEE, Dec. 2015, pp. 105–110. [Online]. Available: http://ieeexplore.ieee.org/document/7449304/

[14] C. Papachristos, K. Alexis, L. R. G. Carrillo, and A. Tzes, "Distributed infrastructure inspection path planning for aerial robotics subject to time constraints," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. Arlington, VA, USA: IEEE, Jun. 2016, pp. 406–412. [Online]. Available: http://ieeexplore.ieee.org/document/7502523/

[15] R. Almadhoun, T. Taha, L. Seneviratne, and Y. Zweiri, "Multi-Robot Hybrid Coverage Path Planning for 3D Reconstruction of Large Structures," *IEEE Access*, vol. 10, pp. 2037–2050, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9664539/

[16] R. Almadhoun, T. Taha, L. Seneviratne, J. Dias, and G. Cai, "A survey on inspecting structures using robotic systems," *International Journal of Advanced Robotic Systems*, vol. 13, no. 6, p. 1729881416663664, Dec. 2016. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1729881416663664

[17] K. Easton and J. Burdick, "A Coverage Algorithm for Multi-robot Boundary Inspection," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. Barcelona, Spain: IEEE, 2005, pp. 727–734. [Online]. Available: http://ieeexplore.ieee.org/document/1570204/

[18] A. Nejadfard, H. Moradi, and M. N. Ahmadabadi, "A multi-robot system for dome inspection and maintenance: Concept and stability analysis," in *2011 IEEE International Conference on Robotics and Biomimetics*. Karon Beach, Thailand: IEEE, Dec. 2011, pp. 853–858. [Online]. Available: http://ieeexplore.ieee.org/document/6181394/

[19] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Distributed Odor Source Localization," *IEEE Sensors*, no. June, pp. 1–13, 2002.

[20] U. Jain, R. Tiwari, and W. W. Godfrey, "Multiple odor source localization using diverse-PSO and group-based strategies in an unknown environment," *Journal of Computational Science*, vol. 34, pp. 33–47, May 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877750318308457

[21] T. Lochmatter, "Bio-Inspired and Probabilistic Algorithms for Distributed Odor Source Localization using Mobile Robots," PhD Dissertation, École polytechnique fédérale de Lausanne (EPFL), Lausanne, Switzerland, 2010.

[22] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S092188901300167X

[23] H. Choset, "Coverage for robotics – A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.

[24] R. Almadhoun, T. Taha, L. Seneviratne, and Y. Zweiri, "A survey on multi-robot coverage path planning for model reconstruction and mapping," *SN Applied Sciences*, vol. 1, no. 8, p. 847, Aug. 2019. [Online]. Available: http://link.springer.com/10.1007/s42452-019-0872-y

[25] B. P. Gerkey and M. J. Matarić, "Multi-robot task allocation: analyzing the complexity and optimality of key architectures," *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 3, pp. 3862–3868, 2003, iSBN: 0-7803-7736-2.

[26] ——, "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004, iSBN: 0278-3649.

[27] A. Prorok, M. A. Hsieh, and V. Kumar, "Fast Redistribution of a Swarm of Heterogeneous Robots," in *International Conference on Bio-inspired Information and Communications Technologies*, 2015.

[28] B. Khaldi and F. Cherif, "An Overview of Swarm Robotics: Swarm Intelligence Applied to Multi-robotics," *International Journal of Computer Applications*, vol. 126, no. 2, pp. 31–37, Sep. 2015. [Online]. Available: http://www.ijcaonline.org/research/volume126/number2/khaldi-2015-ijca-906000.pdf

[29] J. C. Barca and Y. A. Sekercioglu, "Swarm robotics reviewed," *Robotica*, vol. 31, no. 3, pp. 345–359, May 2013. [Online]. Available: https://www.cambridge.org/core/product/identifier/S026357471200032X/type/journal_article

[30] T. D. Seeley and P. K. Visscher, "Quorum sensing during nest-site selection by honeybee swarms," *Behavioral Ecology and Sociobiology*, vol. 56, no. 6, pp. 594–601, 2004, publisher: Springer.

[31] G. Valentini, E. Ferrante, H. Hamann, and M. Dorigo, "Collective decision with 100 Kilobots: speed versus accuracy in binary discrimination problems," in *Autonomous Agents and Multi-Agent Systems*, vol. 30. Springer New York LLC, May 2016, pp. 553–580, issue: 3 ISSN: 15737454.

[32] G. Valentini, H. Hamann, and M. Dorigo, "Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 2, 2015, pp. 1305–1314, iSSN: 15582914.

[33] H. Hamann, B. Meyer, T. Schmickl, and K. Crailsheim, "A Model of Symmetry Breaking in Collective Decision-Making," in *Proceedings of the 11th International Conference on Simulation of Adaptive Behavior*. Springer, 2010, pp. 639–648.

[34] J. Pugh and A. Martinoli, "Inspiring and Modeling Multi-Robot Search with Particle Swarm Optimization," in *2007 IEEE Swarm Intelligence Symposium*. Honolulu, HI, USA: IEEE, Apr. 2007, pp. 332–339. [Online]. Available: http://ieeexplore.ieee.org/document/4223193/

[35] Y. Meng, O. Kazeem, and J. C. Muller, "A Hybrid ACO/PSO Control Algorithm for Distributed Swarm Robots," in *2007 IEEE Swarm Intelligence Symposium*. Honolulu, HI, USA: IEEE, Apr. 2007, pp. 273–280. [Online]. Available: http://ieeexplore.ieee.org/document/4223185/

[36] H. Tang, W. Sun, H. Yu, A. Lin, M. Xue, and Y. Song, "A novel hybrid algorithm based on PSO and FOA for target searching in unknown environments," *Applied Intelligence*, vol. 49, no. 7, pp. 2603–2622, Jul. 2019. [Online]. Available: http://link.springer.com/10.1007/s10489-018-1390-0

[37] L. Bayindir and E. Şahin, "A Review of Studies in Swarm Robotics," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 15, no. 2, pp. 115–147, 2007.

[38] M. K. Heinrich, M. D. Soorati, T. K. Kaiser, M. Wahby, and H. Hamann, "Swarm robotics: Robustness, scalability, and self-X features in industrial applications," *it - Information Technology*, vol. 61, no. 4, pp. 159–167, Aug. 2019. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/itit-2019-0003/html

[39] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007, arXiv: 1009.6050 ISBN: 0018-9219.

[40] A. Shirsat, K. Elamvazhuthi, and S. Berman, "Multi-Robot Target Search using Probabilistic Consensus on Discrete Markov Chains," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. Abu Dhabi, United Arab Emirates: IEEE, Nov. 2020, pp. 108–115. [Online]. Available: https://ieeexplore.ieee.org/document/9292589/

[41] G. Valentini, D. Brambilla, H. Hamann, and M. Dorigo, "Collective Perception of Environmental Features in a Robot Swarm," in *Swarm Intelligence – Proceedings of ANTS 2016 – Tenth International Conference*, 2016, pp. 65–76.

[42] Seuss, *One Fish, Two Fish, Red Fish, Blue Fish.* New York: Beginner Books, 1960.

[43] L. D. Stone, *Theory of optimal search*. New York, NY, USA: Academic Press, 1976, vol. 118.

[44] D. Abbott, D. Briony, J. Dunlop, T. Farmer, M. Hedley, J. Herrmann, G. James, M. Johnson, B. Joshi, G. Poulton, D. Price, M. Prokopenko, T. Reda, D. Rees, P. Valencia, D. Ward, and J. Winter, "Development and Evaluation of Sensor Concepts for Ageless Aerospace Vehicles: Development of Concepts for an Intelligent Sensing System," Commonwealth Scientific Industrial Research Organisation, National Aeronautics and Space Administration, Tech. Rep. NASA/CR-2002-211773, Jul. 2002.

[45] Y. Shi, C. Zhang, R. Li, M. Cai, and G. Jia, "Theory and Application of Magnetic Flux Leakage Pipeline Detection," *Sensors*, vol. 15, no. 12, pp. 31 036–31 055, Dec. 2015. [Online]. Available: http://www.mdpi.com/1424-8220/15/12/29845

[46] D. R. Huston, J. Miller, and B. Esser, "Adaptive, robotic, and mobile sensor systems for structural assessment," in *Proceedings of SPIE*, S.-C. Liu, Ed., San Diego, CA, Jul. 2004, p. 189. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.546606

[47] F. Xu, X. Wang, and H. Wu, "Inspection method of cable-stayed bridge using magnetic flux leakage detection: principle, sensor design, and signal processing," *Journal of Mechanical Science and Technology*, vol. 26, no. 3, pp. 661–669, Mar. 2012. [Online]. Available: http://link.springer.com/10.1007/s12206-011-1234-x

[48] S.-N. Yu, J.-H. Jang, and C.-S. Han, "Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel," *Automation in Construction*, vol. 16, no. 3, pp. 255–261, May 2007. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0926580506000197

[49] D. Roca, S. Lagüela, L. Díaz-Vilariño, J. Armesto, and P. Arias, "Low-cost aerial unit for outdoor inspection of building façades," *Automation in Construction*, vol. 36, pp. 128–135, Dec. 2013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0926580513001477

[50] J. H. Lee, J. M. Lee, H. J. Kim, and Y. S. Moon, "Machine Vision System for Automatic Inspection of Bridges," in *2008 Congress on Image and Signal Processing*. Sanya, China: IEEE, 2008, pp. 363–366. [Online]. Available: http://ieeexplore.ieee.org/document/4566507/

[51] K. Reichling, M. Raupach, M. Stoppel, J. Kurz, and G. Dobmann, "BETOSCAN - Robot controlled non-destructive diagnosis of reinforced concrete decks," in *Proceedings of the NDTCE '09*, vol. 7/3/2009, Nantes, France, 2009, p. 8.

[52] F. Inoue, S. Doi, T. Ishizaki, Y. Ikeda, and Y. Ohta, "Study on automated inspection robot and quantitative detection of outer tile wall exfoliation by wavelet analysis," in *ICCAS 2010*. Gyeonggi-do: IEEE, Oct. 2010, pp. 994–999. [Online]. Available: https://ieeexplore.ieee.org/document/5669653/

[53] S. Ravikumar, K. Ramachandran, and V. Sugumaran, "Machine learning approach for automated visual inspection of machine components," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3260–3266, Apr. 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S095741741000919X

[54] J.-K. Park, B.-K. Kwon, J.-H. Park, and D.-J. Kang, "Machine learning-based imaging system for surface defect inspection," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 3, no. 3, pp. 303–310, Jul. 2016. [Online]. Available: http://link.springer.com/10.1007/s40684-016-0039-x

[55] C. Yang, Y. Sun, C. Ladubec, and Y. Liu, "Developing Machine Learning-Based Models for Railway Inspection," *Applied Sciences*, vol. 11, no. 1, p. 13, Dec. 2020. [Online]. Available: https://www.mdpi.com/2076-3417/11/1/13

[56] Y. Kawaguchi, I. Yoshida, H. Kurumatani, T. Kikuta, and Y. Yamada, "Internal pipe inspection robot," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 1. Nagoya, Japan: IEEE, 1995, pp. 857–862. [Online]. Available: http://ieeexplore.ieee.org/document/525390/

[57] V. Ramasamy, "Mobile Wireless Sensor Networks: An Overview," in *Wireless Sensor Networks - Insights and Innovations*, P. Sallis, Ed. InTech, Oct. 2017. [Online]. Available: http://www.intechopen.com/books/wireless-sensor-networks-insights-and-innovations/mobile-wireless-sensor-networks-an-overview

[58] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, pp. 25–50, 2000.

[59] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse Decompositions for Coverage Tasks," *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 331–344, Apr. 2002. [Online]. Available: http://journals.sagepub.com/doi/10.1177/027836402320556359

[60] H. Choset and P. Pignon, "Coverage Path Planning: The Boustrophedon Cellular Decomposition," in *Field and Service Robotics*, A. Zelinsky, Ed. London: Springer London, 1998, pp. 203–209. [Online]. Available: http://link.springer.com/10.1007/978-1-4471-1273-0_32

[61] T.-K. Lee, S.-H. Baek, Y.-H. Choi, and S.-Y. Oh, "Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 801–812, Oct. 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889011000996

[62] S. Wong and B. MacDonald, "A topological coverage algorithm for mobile robots," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 2. Las Vegas, NV, USA: IEEE, 2003, pp. 1685–1690. [Online]. Available: http://ieeexplore.ieee.org/document/1248886/

[63] S. Wong, "Qualitative Topological Coverage of Unknown Environments by Mobile Robots," PhD Dissertation, The University of Auckland, Aukland, New Zealand, Feb. 2006.

[64] E. Galceran, S. Nagappa, M. Carreras, P. Ridao, and A. Palomer, "Uncertainty-driven survey path planning for bathymetric mapping," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo: IEEE, Nov. 2013, pp. 6006–6012. [Online]. Available: http://ieeexplore.ieee.org/document/6697228/

[65] M. Mazo Jr and K. H. Johansson, "Robust Area Coverage using Hybrid Control," in *TELEC*, Santiago de Cuba, Cuba, 2004, pp. 1–8.

[66] Z. Shaogang and L. Ming, "Path Planning of Inspection Robot Based on Ant Colony Optimization Algorithm," in *2010 International Conference on Electrical and Control Engineering*. Wuhan, China: IEEE, Jun. 2010, pp. 1474–1477. [Online]. Available: http://ieeexplore.ieee.org/document/5630470/

[67] X. Lai, J. Li, and J. Chambers, "Enhanced Center Constraint Weighted A* Algorithm for Path Planning of Petrochemical Inspection Robot," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 4, p. 78, Aug. 2021. [Online]. Available: https://link.springer.com/10.1007/s10846-021-01437-8

[68] F. Kuang and G. Haquin Gerade, "Key Aspects for the Definition of On-site Inspection Challenging Environments," pico, other, Mar. 2020. [Online]. Available: https://meetingorganizer.copernicus.org/EGU2020/EGU2020-21601.html

[69] B. V. Barnes, K. S. Pregitzer, T. A. Spies, and V. H. Spooner, "Ecological Forest Site Classification," *Journal of Forestry*, vol. 80, no. 8, pp. 493–498, 1982.

[70] R. Dobry, R. D. Borcherdt, C. B. Crouse, I. M. Idriss, W. B. Joyner, G. R. Martin, M. S. Power, E. E. Rinne, and R. B. Seed, "New Site Coefficients and Site Classification System Used in Recent Building Seismic Code Provisions," *Earthquake Spectra*, vol. 16, no. 1, pp. 41–67, Feb. 2000. [Online]. Available: http://journals.sagepub.com/doi/10.1193/1.1586082

[71] K. M. Passino and T. D. Seeley, "Modeling and analysis of nest-site selection by honeybee swarms: the speed and accuracy trade-off," *Behavioral Ecology and Sociobiology*, vol. 59, no. 3, pp. 427–442, 2006, publisher: Springer.

[72] T. D. Seeley and S. C. Buhrman, "Nest-site selection in honey bees: how well do swarms implement the "best-of-N" decision rule?" *Behavioral Ecology and Sociobiology*, vol. 49, no. 5, pp. 416–427, Apr. 2001. [Online]. Available: https://doi.org/10.1007/s002650000299

[73] C. A. Parker and H. Zhang, "Cooperative decision-making in decentralized multiple-robot systems: The best-of-N problem," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 2, pp. 240–251, 2009.

[74] J. J. Roldán, J. d. Cerro, D. Garzón-Ramos, P. Garcia-Aunon, M. Garzón, J. d. León, and A. Barrientos, "Robots in Agriculture: State of Art and Practical Experiences," in *Service Robots*, A. J. R. Neves, Ed. InTech, Jan. 2018. [Online]. Available: http://www.intechopen.com/books/service-robots/robots-in-agriculture-state-of-art-and-practical-experiences

[75] D. Albani, J. IJsselmuiden, R. Haken, and V. Trianni, "Monitoring and mapping with robot swarms for agricultural applications," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. Lecce, Italy: IEEE, Aug. 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8078478/

[76] E. Camci, D. R. Kripalani, L. Ma, E. Kayacan, and M. A. Khanesar, "An aerial robot for rice farm quality inspection with type-2 fuzzy neural networks tuned by particle swarm optimization-sliding mode control hybrid algorithm," *Swarm and Evolutionary Computation*, vol. 41, pp. 1–8, Aug. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2210650217302407

[77] J. Moonrinta, S. Chaivivatrakul, M. N. Dailey, and M. Ekpanyapong, "Fruit detection, tracking, and 3D reconstruction for crop mapping and yield estimation," in *2010 11th International Conference on Control Automation Robotics & Vision*. Singapore, Singapore: IEEE, Dec. 2010, pp. 1181–1186. [Online]. Available: http://ieeexplore.ieee.org/document/5707436/

[78] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, Apr. 2016. [Online]. Available: http://link.springer.com/10.1007/s10514-015-9491-7

[79] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a Robot Designed for Education in Engineering," in *Proceedings of the 9th conference on autonomous robot systems and competitions*. Castelo Branco, Portugal: IPCB: Instituto Polit{\'e}cnico de Castelo Branco, May 2009, pp. 59–65.

[80] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., 2012, pp. 3293–3298, iSSN: 10504729.

[81] M. Schranz, M. Umlauft, M. Sende, and W. Elmenreich, "Swarm Robotic Behaviors and Current Applications," *Frontiers in Robotics and AI*, vol. 7, p. 36, Apr. 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2020.00036/full

[82] N. Correll, "Coordination Schemes for distributed boundary coverage with a swarm of miniature robots: Synthesis, analysis and experimental validation," Ph.D. dissertation, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2007.

[83] T. D. Seeley and P. K. Visscher, "Group decision making in nest-site selection by honey bees," *Apidologie*, vol. 35, no. 2, pp. 101–116, 2004, publisher: EDP Sciences.

[84] S. C. Pratt, E. B. Mallon, D. J. T. Sumpter, and N. R. Franks, "Quorum sensing, recruitment, and collective decision-making during colony emigration by the ant Leptothorax albipennis," *Behavioral Ecology and Sociobiology*, vol. 52, no. 2, pp. 117–127, Jul. 2002.

[85] T. D. Seeley and S. C. Buhrman, "Group decision making in swarms of honey bees," *Behavioral Ecology and Sociobiology*, vol. 45, no. 1, pp. 19–31, 1999, publisher: Springer Verlag.

[86] T. D. Seeley, P. K. Visscher, T. Schlegel, P. M. Hogan, N. R. Franks, and J. A. Marshall, "Stop signals provide cross inhibition in collective decision-making by honeybee swarms," *Science*, vol. 335, no. 6064, pp. 108–111, 2012, publisher: American Association for the Advancement of Science.

[87] A. Sendova-Franks and N. R. Franks, "Task allocation in ant colonies within variable environments (A study of temporal polyethism: Experimental)," *Bulletin of Mathematical Biology*, vol. 55, no. 1, pp. 75–96, 1993, iSBN: 0092-8240.

[88] N. R. Franks, E. B. Mallon, H. E. Bray, M. J. Hamilton, and T. C. Mischler, "Strategies for choosing between alternatives with different attributes: exemplified by house-hunting ants," *Animal Behaviour*, vol. 65, no. 1, pp. 215–223, 2003.

[89] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[90] D. M. Gordon, "The organization of work in social insect colonies," *Nature*, vol. 380, no. 6570, pp. 121–124, 1996, publisher: Springer.

[91] D. M. Gordon and N. J. Mehdiabadi, "Encounter rate and task allocation in harvester ants," *Behavioral Ecology and Sociobiology*, vol. 45, no. 5, pp. 370–377, 1999.

[92] E. J. Robinson, O. Feinerman, and N. R. Franks, "Flexible task allocation and the organization of work in ants," *Proceedings of the Royal Society of London B: Biological Sciences*, 2009, publisher: The Royal Society.

[93] M. B. Miller and B. L. Bassler, "Quorum Sensing in Bacteria," *Annual Review of Microbiology*, vol. 55, pp. 165–199, 2001, arXiv: 1011.1669v3 ISBN: 0066-4227 (Print)\r0066-4227 (Linking).

[94] C. M. Waters and B. L. Bassler, "Quorum sensing: cell-to-cell communication in bacteria," *The Annual Review of Cell and Developmental Biology*, vol. 21, pp. 319–346, 2005.

[95] I. Y. Hoballah and P. K. Varshney, "Distributed Bayesian Signal Detection," *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 995–1000, 1989.

[96] S. Bandyopadhyay and S. J. Chung, "Distributed estimation using Bayesian consensus filtering," in *Proceedings of the American Control Conference*. Institute of Electrical and Electronics Engineers Inc., 2014, pp. 634–641, iSSN: 07431619.

[97] T. Zhao and A. Nehorai, "Distributed sequential Bayesian estimation of a diffusive source in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 55, no. 4, pp. 1511–1524, Apr. 2007.

[98] D. Fontanella, M. Nicoli, and L. Vandendorpe, "Bayesian localization in sensor networks: Distributed algorithm and fundamental limits," in *IEEE International Conference on Communications*, 2010, iSSN: 05361486.

[99] M. Alanyali, S. Venkatesh, O. Savas, and S. Aeron, "Distributed Bayesian hypothesis testing in sensor networks," in *Proceedings of the 2004 American Control Conference*. IEEE, 2004, pp. 5369–5374 vol.6.

[100] A. Makarenko and H. Durrant-Whyte, "Decentralized Bayesian algorithms for active sensor networks," *Information Fusion*, vol. 7, pp. 418–433, 2006.

[101] P. Landgren, V. Srivastava, and N. E. Leonard, "Distributed cooperative decision-making in multiarmed bandits: Frequentist and Bayesian algorithms," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 167–172.

[102] A. Lilienthal, A. Loutfi, and T. Duckett, "Airborne Chemical Sensing with Mobile Robots," *Sensors*, vol. 6, no. 11, pp. 1616–1678, Nov. 2006. [Online]. Available: http://www.mdpi.com/1424-8220/6/11/1616

[103] P. P. Neumann, V. Hernandez Bennetts, A. J. Lilienthal, M. Bartholmai, and J. H. Schiller, "Gas source localization with a micro-drone using bio-inspired and particle filter-based algorithms," *Advanced Robotics*, vol. 27, no. 9, pp. 725–738, Jun. 2013. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01691864.2013.779052

[104] E. Persson and D. A. Anisi, "A Comparative study of robotic gas source localization algorithms in industrial environments," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 899–904, Jan. 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1474667016437237

[105] G. Kowadlo and R. A. Russell, "Robot Odor Localization: A Taxonomy and Survey," *The International Journal of Robotics Research*, vol. 27, no. 8, pp. 869–894, Aug. 2008. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364908095118

[106] T. Jing, Q. Meng, and H. Ishida, "Recent Progress and Trend of Robot Odor Source Localization," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 16, no. 7, pp. 938–953, Jul. 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/tee.23364

[107] N. Voges, A. Chaffiol, P. Lucas, and D. Martinez, "Reactive Searching and Infotaxis in Odor Source Localization," *PLoS Computational Biology*, vol. 10, no. 10, p. e1003861, Oct. 2014. [Online]. Available: https://dx.plos.org/10.1371/journal.pcbi.1003861

[108] A. Lilienthal, D. Reimann, and A. Zell, "Gas Source Tracing with a Mobile Robot Using an Adapted Moth Strategy," in *Autonome Mobile Systeme 2003*. Berlin, Heidelberg: Springer Berlin/Heidelberg, 2003, pp. 150–160, series Title: Informatik aktuell. [Online]. Available: http://link.springer.com/10.1007/978-3-642-18986-9_16

[109] J. Boghaert, "Decentralized collective decision-making algorithms in simulated soft-bodied robot swarms for 3D surface inspection in space," Master's thesis, Swiss Federal Institute of Technology (ETH) Zurich, Oct. 2021.

[110] G. Sandini, G. Lucarini, and M. Varoli, "Gradient driven self-organizing systems," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, vol. 1. Yokohama, Japan: IEEE, 1993, pp. 429–432. [Online]. Available: http://ieeexplore.ieee.org/document/583132/

[111] G. Drews, "Contributions of Theodor Wilhelm Engelmann on phototaxis, chemotaxis, and photosynthesis," *Photosynthesis Research*, vol. 83, no. 1, pp. 25–34, Jan. 2005. [Online]. Available: http://link.springer.com/10.1007/s11120-004-6313-8

[112] V. Braitenberg, *Vehicles, experiments in synthetic psychology*. Cambridge, Mass: MIT Press, 1984.

[113] M. Vergassola, E. Villermaux, and B. I. Shraiman, "'Infotaxis' as a strategy for searching without gradients." *Nature*, vol. 445, no. 7126, pp. 406–409, 2007, iSBN: 1476-4687 (Electronic)\r0028-0836 (Linking).

[114] C. Song, Y. He, B. Ristic, and X. Lei, "Collaborative infotaxis: Searching for a signal-emitting source based on particle filter and Gaussian fitting," *Robotics and Autonomous Systems*, vol. 125, p. 103414, Mar. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889019306451

[115] J. S. Kennedy, "Zigzagging and casting as a programmed response to wind-borne odour: a review," *Physiological Entomology*, vol. 8, no. 2, pp. 109–120, Jun. 1983. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/j.1365-3032.1983.tb00340.x

[116] H. Hajieghrary, M. A. Hsieh, and I. B. Schwartz, "Multi-agent search for source localization in a turbulent medium," *Physics Letters A*, vol. 380, no. 20, pp. 1698–1705, Apr. 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0375960116002425

[117] C. Song, Y. He, B. Ristic, L. Li, and X. Lei, "Multi-agent collaborative infotaxis search based on cognition difference," *Journal of Physics A: Mathematical and Theoretical*, vol. 52, no. 48, p. 485202, Nov. 2019. [Online]. Available: https://iopscience.iop.org/article/10.1088/1751-8121/ab5088

[118] B. Ristic, C. Gilliam, W. Moran, and J. L. Palmer, "Decentralised multi-platform search for a hazardous source in a turbulent flow," *Information Fusion*, vol. 58, pp. 13–23, Jun. 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1566253518303592

[119] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[120] J. Hereford, "A Distributed Particle Swarm Optimization Algorithm for Swarm Robotic Applications," in *2006 IEEE International Conference on Evolutionary Computation*. Vancouver, BC, Canada: IEEE, 2006, pp. 1678–1685. [Online]. Available: http://ieeexplore.ieee.org/document/1688510/

[121] A. Cornejo, F. Kuhn, R. Ley-Wild, and N. Lynch, "Keeping Mobile Robot Swarms Connected," in *Distributed Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, vol. 5805, pp. 496–511, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-642-04355-0_50

[122] W. Luo and K. Sycara, "Minimum k-Connectivity Maintenance for Robust Multi-Robot Systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Macau, China: IEEE, Nov. 2019, pp. 7370–7377. [Online]. Available: https://ieeexplore.ieee.org/document/8968058/

[123] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987, arXiv: cond-mat/0208573 ISBN: 0897912276. [Online]. Available: http://portal.acm.org/citation.cfm?doid=37402.37406

[124] J. Ebert and R. Barnes, "Kilosim," Sep. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3406864

[125] B. Browning and E. Tryzelaar, "ÜberSim: A Multi-Robot Simulator for Robot Soccer," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, Jul. 2003, pp. 948–949. [Online]. Available: https://doi.org/10.1145/860575.860739

[126] T. Tosik, J. Schwinghammer, M. J. Feldvos, J. P. Jonte, A. Brech, and E. Maehle, "MARS: A simulation environment for marine swarm robotics and environmental monitoring," in *OCEANS 2016 - Shanghai*. Shanghai: IEEE, Apr. 2016, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/7485333/

[127] F. Berlinger, "Blueswarm: 3D Self-organization in a Fish-inspired Robot Swarm," PhD Dissertation, Harvard University, Cambridge, MA, 2021.

[128] F. Jansson, M. Hartley, M. Hinsch, I. Slavkov, N. Carranza, T. S. G. Olsson, R. M. Dries, J. H. Grönqvist, A. F. M. Marée, J. Sharpe, J. A. Kaandorp, and V. A. Grieneisen, "Kilombo: a Kilobot simulator to enable effective research in swarm robotics," *arXiv:1511.04285 [cs]*, May 2016, arXiv: 1511.04285. [Online]. Available: http://arxiv.org/abs/1511.04285

[129] G. H. W. Gebhardt and G. Neumann, "The Kilobot Gym," in *ICRA 2018 Workshop*, Brisbane, Australia, 2018, p. 3.

[130] E. Coumans, "Bullet Physics," 2021. [Online]. Available: https://github.com/bulletphysics/bullet3

[131] P. Katara, M. Khanna, H. Nagar, and A. Panaiyappan, "Open Source Simulator for Unmanned Underwater Vehicles using ROS and Unity3D," in *2019 IEEE Underwater Technology (UT)*. Kaohsiung, Taiwan: IEEE, Apr. 2019, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/document/8734309/

[132] "Unity," 2022. [Online]. Available: https://unity.com/

[133] O. Michel, "Cyberbotics Ltd. Webots: Professional Mobile Robot Simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, Mar. 2004. [Online]. Available: http://journals.sagepub.com/doi/10.5772/5618

[134] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3. Sendai, Japan: IEEE, 2004, pp. 2149–2154. [Online]. Available: http://ieeexplore.ieee.org/document/1389727/

[135] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo: IEEE, Nov. 2013, pp. 1321–1326. [Online]. Available: http://ieeexplore.ieee.org/document/6696520/

[136] "Isaac Sim," 2022. [Online]. Available: https://developer.nvidia.com/isaac-sim

[137] L. Pitonakova, M. Giuliani, A. Pipe, and A. Winfield, "Feature and Performance Comparison of the V-REP, Gazebo and ARGoS Robot Simulators," in *Towards Autonomous Robotic Systems*, M. Giuliani, T. Assaf, and M. E. Giannaccini, Eds. Cham: Springer International Publishing, 2018, vol. 10965, pp. 357–368, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-96728-8_30

[138] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, Dec. 2012. [Online]. Available: http://link.springer.com/10.1007/s11721-012-0072-5

[139] S. Magnenat, "Enki." [Online]. Available: https://github.com/enki-community/enki

[140] L. Gomila, "SFML - Simple and Fast Multimedia Library," 2018.

[141] M. Folk, G. Heber, Q. Koziol, E. Pourmal, and D. Robinson, "An Overview of the HDF5 Technology Suite and Its Applications," in *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, ser. AD '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 36–47, event-place: Uppsala, Sweden. [Online]. Available: https://doi.org/10.1145/1966895.1966900

[142] O. A. R. Board, "OpenMP Application Program Interface," Nov. 2015. [Online]. Available: https://www.openmp.org/wp-content/uploads/openmp-4.5.pdf

[143] A. Cornejo, "Kilolib." [Online]. Available: https://kilobotics.com/docs/

[144] A. M. Turing, "The chemical basis of morphogenesis," *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, vol. 237, pp. 37–72, Aug. 1952.

[145] I. Slavkov, D. Carrillo-Zapata, N. Carranza, X. Diego, F. Jansson, J. Kaandorp, S. Hauert, and J. Sharpe, "Morphogenesis in robot swarms," *Science Robotics*, vol. 3, no. 25, p. eaau9178, Dec. 2018. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.aau9178

[146] G. Seymour, "Morphogenesis in Kilosim," 2019. [Online]. Available: https://github.com/SeymourGabe/MorphSim

[147] J. Ebert, "Kilosim Gridbots," 2021. [Online]. Available: https://github.com/jtebert/kilosim-gridbots

[148] ——, "Gridsim: Grid-world robot simulator," 2020. [Online]. Available: https://github.com/jtebert/gridsim

[149] G. Espinosa and M. Rubenstein, "Using Hardware Specialization and Hierarchy to Simplify Robotic Swarms," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation*, 2018.

[150] H. Wang and M. Rubenstein, "Shape Formation in Homogeneous Swarms Using Local Task Swapping," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 597–612, Jun. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9000788/

[151] J. T. Ebert, "jtebert/kilosim-coachbot: v0.1," Jan. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5823430

[152] S. Camazine, *Self-organization in biological systems*. Princeton University Press, 2003.

[153] R. Beckers, J.-L. Deneubourg, S. Goss, and J. M. Pasteels, "Collective decision making through food recruitment," *Insectes sociaux*, vol. 37, no. 3, pp. 258–267, 1990.

[154] G. Theraulaz and E. Bonabeau, "A brief history of stigmergy," *Artificial Life*, vol. 5, no. 2, pp. 97–116, 1999, publisher: MIT Press.

[155] J. T. Ebert, M. Gauci, and R. Nagpal, "Multi-feature collective decision making in robot swarms," in *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018)*. IFAAMAS, 2018, pp. 1711–1719, iSSN: 15582914.

[156] T. D. Seeley, *The wisdom of the hive: the social physiology of honey bee colonies*. Harvard University Press, 1995.

[157] J. T. Ebert and M. Gauci, "Bioinspired decision-making with Kilobot robots [video]," 2017. [Online]. Available: https://www.dropbox.com/sh/9ievpc800luiehx/AABgNFcFJugt0PJVueoNFZNfa?dl=0

[158] J. T. Ebert, M. Gauci, F. Mallmann-Trenn, and R. Nagpal, "Bayes Bots: Collective Bayesian Decision-Making in Decentralized Robot Swarms," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2020.

[159] S. Wallis, "Binomial confidence intervals and contingency tests: Mathematical fundamentals and the evaluation of alternative methods," *Journal of Quantitative Linguistics*, vol. 20, no. 3, pp. 178–208, Aug. 2013.

[160] D. Aldous and J. A. Fill, "Reversible Markov Chains and Random Walks on Graphs," 2002.

[161] J. Ebert, "Simulation Video: Bayes Bots," 2020. [Online]. Available: https://youtu.be/O9C3wAPh1IA

[162] J. T. Ebert, F. Berlinger, B. Haghighat, and R. Nagpal, "A Hybrid PSO Algorithm for Multi-robot Target Search and Decision Awareness," in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2022)*. IEEE, 2022.

[163] K. Perlin, "An image synthesizer," *ACM SIGGRAPH Computer Graphics*, vol. 19, no. 3, pp. 287–296, Jul. 1985. [Online]. Available: https://dl.acm.org/doi/10.1145/325165.325247

[164] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965. [Online]. Available: http://ieeexplore.ieee.org/document/5388473/

[165] F. Berlinger, M. Gauci, and R. Nagpal, "Implicit coordination for 3D underwater collective behaviors in a fish-inspired robot swarm," *Science Robotics*, vol. 6, no. 50, p. eabd8668, Jan. 2021. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.abd8668

[166] A. E. Turgut, H. Çelikkanat, F. Gökçe, and E. Şahin, "Self-organized flocking in mobile robot swarms," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 97–120, Dec. 2008. [Online]. Available: http://link.springer.com/10.1007/s11721-008-0016-2

[167] G. Vasarhelyi, C. Viragh, G. Somorjai, N. Tarcai, T. Szorenyi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Chicago, IL, USA: IEEE, Sep. 2014, pp. 3866–3873. [Online]. Available: http://ieeexplore.ieee.org/document/6943105/

[168] J. E. Jones, "On the determination of molecular fields.—I. From the variation of the viscosity of a gas with temperature," *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, vol. 106, no. 738, pp. 441–462, Oct. 1924. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rspa.1924.0081

[169] J. Ebert, "Hybrid PSO Multi-robot Target Search with Kilosim Gridbots," Feb. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.6311164

[170] B. P. Duisterhof, S. Li, J. Burgues, V. J. Reddi, and G. C. H. E. de Croon, "Sniffy Bug: A Fully Autonomous Swarm of Gas-Seeking Nano Quadcopters in Cluttered Environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, Sep. 2021, pp. 9099–9106. [Online]. Available: https://ieeexplore.ieee.org/document/9636217/

[171] S. Neilson, "Space-station crew members just found an elusive air leak by watching tea leaves float in microgravity," *Business Insider*, Oct. 2020. [Online]. Available: https://www.businessinsider.com/astronauts-cosmonauts-found-space-station-leak-using-tea-leaves-2020-10

[172] P. Sloss, "NASA updates Lunar Gateway plans," Sep. 2018. [Online]. Available: https://www.nasaspaceflight.com/2018/09/nasa-lunar-gateway-plans/

[173] A. Bertolin, "A view of the full Gateway configuration with Orion approaching Gateway," Nov. 2021. [Online]. Available: https://www.flickr.com/photos/nasa2explore/51670686650/

[174] B. Bayat, N. Crasta, A. Crespi, A. M. Pascoal, and A. Ijspeert, "Environmental monitoring using autonomous vehicles: a survey of recent searching techniques," *Current Opinion in Biotechnology*, vol. 45, pp. 76–84, Jun. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0958166916302312

[175] J. Wanner, "Self-assembling ferro-elastomeric soft-bodied robotic modules inspired by the Army ants self-assembling swarm," Master's thesis, École polytechnique fédérale de Lausanne (EPFL), Mar. 2022.

[176] J. Park, "Special feature vibration-based structural health monitoring," *Applied Sciences*, vol. 10, no. 15, Jul. 2020, issue: 15 Pages: 5139 Publication Title: Applied Sciences Volume: 10. [Online]. Available: https://www.mdpi.com/2076-3417/10/15/5139

[177] S. W. Doebling, C. R. Farrar, M. B. Prime, and D. W. Shevitz, "Damage identification and health monitoring of structural and mechanical systems from changes in their vibration characteristics: a literature review," Los Alamos National Laboratory (LANL), Los Alamos, NM, Tech. Rep. LA-13070-MS, 1996, publisher: Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[178] O. Avci, O. Abdeljaber, S. Kiranyaz, M. Hussein, M. Gabbouj, and D. J. Inman, "A review of vibration-based damage detection in civil structures: From traditional methods to Machine Learning and Deep Learning applications," *Mechanical Systems and Signal Processing*, vol. 147, p. 107077, Jan. 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0888327020304635

[179] V. Ganesan, T. Das, N. Rahnavard, and J. L. Kauffman, "Vibration-based monitoring and diagnostics using compressive sensing," *Journal of Sound and Vibration*, vol. 394, pp. 612–630, Apr. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0022460X17300962

[180] "Gecko Robotics NDT Inspection Services." [Online]. Available: https://www.geckorobotics.com/ndt-inspection-services

[181] B. Haghighat, J. T. Ebert, J. Boghaert, A. Ekblaw, and R. Nagpal, "A Swarm Robotic Approach to Inspection of 2.5D Surfaces in Orbit," in *5th International Symposium on Swarm Behavior and Bio-Inspired Robotics (SWARM5)*, Beppu, Japan, Jan. 2022, p. 8.

[182] K. McPherson, K. Hrovat, E. Kelly, and J. Keller, "ISS Researcher's Guide: Acceleration Environment," National Aeronautics and Space Administration, Tech. Rep. NP-2015-11-040-JSC, Nov. 2015.

[183] X.-x. Chen and J. Huang, "Odor source localization algorithms on mobile robots: A review and future outlook," *Robotics and Autonomous Systems*, vol. 112, pp. 123–136, Feb. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889018303014

[184] A. T. Hayes, A. Martinoli, and R. M. Goodman, "Swarm robotic odor localization," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 2, 2001, pp. 1073–1078, iSSN: 02635747.

[185] E. Rimon, "Exact robot navigation using artificial potential functions," PhD Dissertation, Yale University, New Haven, CT, 1990. [Online]. Available: http://search.proquest.com.ezp-prod1.hul.harvard.edu/dissertations-theses/exact-robot-navigation-using-artificial-potential/docview/303842174/se-2

[186] J. Zhang, D. Gong, and Y. Zhang, "A niching PSO-based multi-robot cooperation method for localizing odor sources," *Neurocomputing*, vol. 123, pp. 308–317, Jan. 2014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925231213007698

[187] A. Dementyev, H.-L. C. Kao, I. Choi, D. Ajilo, M. Xu, J. A. Paradiso, C. Schmandt, and S. Follmer, "Rovables: Miniature On-Body Robots as Mobile Wearables," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. Tokyo Japan: ACM, Oct. 2016, pp. 111–120. [Online]. Available: https://dl.acm.org/doi/10.1145/2984511.2984531

[188] Bosch, "BNO055 Intelligent 9-axis absolute orientation sensor," Bosch, Tech. Rep., Nov. 2014. [Online]. Available: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf

[189] A. Eiben and S. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31, Mar. 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2210650211000022

[190] C. Shen, Y. Gonzalez, L. Chen, S. B. Jiang, and X. Jia, "Intelligent Parameter Tuning in Optimization-Based Iterative CT Reconstruction via Deep Reinforcement Learning," *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1430–1439, Jun. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8331966/

[191] Kuan Yew Wong and Komarudin, "Parameter tuning for ant colony optimization: A review," in *2008 International Conference on Computer and Communication Engineering*. Kuala Lumpur, Malaysia: IEEE, May 2008, pp. 542–545. [Online]. Available: http://ieeexplore.ieee.org/document/4580662/

[192] M. Dorigo, G. Theraulaz, and V. Trianni, "Reflections on the future of swarm robotics," *Science Robotics*, vol. 5, no. 49, p. eabe4385, Dec. 2020. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.abe4385