The University Of Sheffield.

Swarm Robotic Systems with Minimal Information Processing

Melvin Gauci

Supervisors:

Dr Roderich Groß

Dr Tony J. Dodd

A Thesis Submitted for the Degree of Doctor of Philosophy

30th September 2014

Nature has a
great simplicity
and, therefore,
a great beauty.

Richard Feynman
The Character of the Physical Law

Abstract

This thesis is concerned with the design and analysis of behaviors in swarm robotic systems using minimal information acquisition and processing. The motivation for this work is to contribute in paving the way for the implementation of swarm robotic systems at physically small scales, which will open up new application domains for their operation. At these scales, the space and energy available for the integration of sensors and computational hardware within the individual robots is at a premium. As a result, trade-offs in performance can be justified if a task can be achieved in a more parsimonious way.

A framework is developed whereby meaningful collective behaviors in swarms of robots can be shown to emerge without the robots, in principle, possessing any run-time memory or performing any arithmetic computations. This is achieved by the robots having only discrete-valued sensors, and purely reactive controllers. Black-box search methods are used to automatically synthesize these controllers for desired collective behaviors.

This framework is successfully applied to two canonical tasks in swarm robotics: self-organized aggregation of robots, and self-organized clustering of objects by robots. In the case of aggregation, the robots are equipped with one binary sensor, which informs them whether or not there is another robot in their line of sight. This makes the structure of the robots' controller simple enough that its entire space can be systematically searched to locate the optimal controller (within a finite resolution).

In the case of object clustering, the robots' sensor is extended to have three states, distinguishing between robots, objects, and the background. This still requires no run-time memory or arithmetic computations on the part of the robots. It is statistically shown that the extension of the sensor to have three states leads to a better performance as compared to the cases where the sensor is binary, and cannot distinguish between robots and objects, or robots and the background.

Acknowledgments

I would like to start by thanking my supervisors, Dr Roderich Groß and Dr Tony J. Dodd for their constant support throughout my PhD studies. They have helped me to make the transition from a student to a researcher, by pushing me to be self-critical and providing me with the invaluable feedback. Roderich introduced me to the fields of swarm robotics and evolutionary robotics in 2009, and conveyed to me a sense of excitement about these fields, such that I had no hesitation in taking them up for my research. I owe him a debt of gratitude for all that I have picked up from him. Tony was always available to provide me with level-headed advice when I needed it, making sure I did not miss the forest for the trees.

Thank you to my lab mates. Jianing Chen was a great help in programming e-puck robots and carrying out physical experiments. Without his collaboration, my project would have taken much longer. Wei Li provided me with an opportunity to extend my research breadth by collaborating with him on his work on coevolutionary learning of agent behaviors. I also had the privilege of meeting and learning from Fernando Perez Diaz, Yuri Kasubowski Lopes, Christopher Parrott, Stefan Trenkwalder, and Gabriel Kapellman Zafra.

I would also like to thank everyone from outside my research group who helped to make my stay in Sheffield an enjoyable one. These people are far too many to list by name, but I will single out Celso Gomes and Bárbara Luísa da Silva were my house mates during the final — and most testing — year of my stay in Sheffield. Together we discovered some of the city's best restaurants, mixed many successful and a few less successful cocktails, mastered the art of espresso making, and watched FC Barcelona win and lose (but mostly win).

I close with my family. My parents have always supported me along my seemingly endless educational journey, and made many sacrifices for my sake. My grandmother, Katy, has been for me an unwavering reference point of resilience, optimism and above all, simplicity. To her I dedicate this work.

Contents

1. Introduction

Robots have been at the heart of industrial automation since the mid-20th century, proving extremely effective in a multitude of ways. They have, for the most part, replaced humans in tasks that are purely mechanical (i.e. not requiring cognition) and repetitive. In such tasks, robots can offer much higher production speeds and, as they suffer less from fatigue, much better degrees of consistency. They can also achieve significantly better degrees of accuracy than humans in high-precision tasks, such as tasks involving manipulation of minute objects, where hand tremors in humans become a limiting factor.

To date, most robots that can be found in industry take the form of a rigid structure consisting of a number of links connected together by prismatic and rotational joints, and an end effector at the tip of the structure. These robots tend to be re-programmable in their operation, but are normally only able to perform one specific task at a given time. Moreover, robots in this category tend to be fixed in place, and not very flexible in their structural (or morphological) reconfigurability.

As a result of these properties, most robots that are in wide service today, while certainly useful, are not without their significant limitations. They are normally only able to operate in well-structured environments, and restricted to performing tasks that are relatively simple and have been well-defined a priori, as they do not have the cognitive abilities to handle unexpected situations by applying what humans consider to be 'common sense'. These robots typically offer only a limited degree of flexibility in their ability to be reappropriated for different tasks, and such operational and/or structural reappropriations, require high degrees of human expertise. Such robots usually are also not designed to be very compliant with humans, and as such, the more powerful ones can constitute a safety hazard for humans who have not been properly trained to 'interact' with them.

Since the inception of artificial intelligence in the 1960s, researchers have been attempting to design a new generation of robotic systems that overcome the limitations of the previous generation. Early efforts gave birth to the field of *mobile robotics* [1], which involves the study of robots that can sense their environment and move within it. Typical problems studied in mobile robotics include simultaneous localization and mapping (SLAM) and navigation through a field of obstacles by the use of path planning. Research in the 1960s also initiated the study of *cognitive robotics* [2]. This is concerned with endowing robots with cognitive abilities, such forming concrete, step-by-step plans to achieve given goals, and revising these plans on the spot in the case of unexpected events.

More recently, starting around the 1990s, some researchers have started seeking to take more direct inspiration from biological systems in addressing the issue of building a new generation of robotic systems. This research effort is often described by the umbrella terms of *bio-inspired artificial intelligence* [3] (which is broader than just robotics) and more specifically, *bio-inspired robotics* [4]. The realization of such robotic systems is expected to contribute towards what some researchers have predicted will be a "new industrial revolution" [5, 6].

As this is a challenging problem, the associated research field has several aspects to it, most of which can be considered to be sub-fields in their own right. A few examples follow: *Developmental robotics* [7] aims to achieve cognition in robots by taking inspiration from the way in which humans acquire cognitive abilities by learning, especially in the early stages of their lives. *Soft robotics* [6, 8] refers to robots that eschew the traditional link-and-joints structure in favor of flexible structures, as inspired, for instance, by elephant trunks, octopus legs and snake bodies. This allows soft robots to operate in environments where rigid ones fail, such as in narrow spaces. *Legged robots* [9] are a class of mobile robots that achieve locomotion through the use of legs. This makes them appropriate for navigating rough terrains where traditional means of locomotion, such as wheels, are not viable.

The sub-field of bio-inspired robotics that this thesis is concerned with is *swarm robotics* [10, 11, 12, 13, 14]. This is inspired by biological systems that are composed of large numbers of individuals, such as colonies of social insects, flocks of birds, and schools of fish. These systems exhibit a number of fascinating characteristics. For instance, although the individuals are distinct from each other, they are often seen to behave as a single meta-entity, such as in the fluid-like motion of flocks of birds. Swarm systems can also be seen to transcend the capabilities of their constituent individuals, in that "the whole is larger than the sum of the parts". This is observed, for example, in the building of complex mound structures by small termites, which relative to their size, are the equivalent of human skyscrapers. Furthermore, swarm systems are self-organized, in the sense that their collective behaviors emerge as a result of local interactions between the individuals. Consequently, these systems are fairly robust with respect to the failures (e.g. deaths) of a few individuals.

Swarm robotics is closely related to *self-assembling robotics* [15] and *reconfigurable robotics* [16]. Swarm robotics is concerned with robots that, while acting in collaboration, are physically disjointed from each other. On the other hand, self-assembling and reconfigurable robotics are concerned with systems in which a number of modular entities can physically self-assemble into a single structure, and with how such structures can adapt their morphologies for different purposes. Note that these two sub-fields also take inspiration from biological systems; for example, ants have been observed to form physical ensembles in order to navigate rough terrains [17]. The overall aim of swarm, self-assembling and reconfigurable robotics is to design systems that convert the characteristics of natural swarm systems into engineering advantages, namely robustness, scalability, and flexibility [14].

1.1. Motivation

The motivation for this thesis comes from the possibility of opening up new application domains for swarm robotic systems at small scales. We will briefly discuss two such potential applications.

At the centimeter down to the the millimeter scale, swarms of insect-like robots may be a major contributing technology to the development of *precision agriculture* [18, 19]. They could be used to monitor the state of large crop areas at a detailed level. Inspired from the behavior of honeybees, micro aerial robots [20, 21] could be used for performing controlled pollination. In order to maximize yield, robots could seek out infected crops and provide them with individually-tailored treatment. Alternatively, if a crop is deemed unlikely to recover, the robots might weed it out and plant a new one in its place. Another interesting possibility in precision agriculture is to conducting harvesting as a continuous rather than a batch process. In this case, robots could monitor individual crops regularly, and only harvest the ones that have reached their optimal stage of maturity. This would minimize the waste of prematurely harvested crops that occurs in a batch process.

An even smaller-scale, potential future application of swarm robotic systems lies within the human body. In this domain, the robots would need to be at the milli-, micro-, or even the nano-scale [22, 23] (for comparison, a red blood cell is around $6 - 8\,\mu$m in diameter). Technology that operates within the human body is already available; for instance in wireless capsule endoscopy [24], the patient swallows a 'pill' that travels through the gastrointestinal tract to take images and relay them wirelessly to an external station. In the future, this technology could extend to active robots that have the ability to move within the body, sense their environment and act within it, and can therefore operate at least partly autonomously. These robots might be used to perform internal surgery, such as treating internal hemorrhages, or removing dangerous blood clots. They might deliver drugs to specific locations inside the body, or augment the natural immune system's capacity for fighting pathogens [25].

Distributed robotic systems (swarm robotic systems, along with self-assembling and reconfigurable robotic systems), offer promising potential for these application domains, for a number of reasons. To begin with, swarm robotic systems ease the requirements on the complexity of individual robots by allowing for complexity to emerge as a result of self-organized, collective behaviors. This facilitates the implementation of the resulting individual robots at small scales. Moreover, as the robots are individually relatively simple, these solutions could also be cost-effective in scenarios where a proportion of the robots may not be recoverable after a mission, and may therefore need to be disposable [23]. Another advantage of distributed robotic systems at small scales has to do with the associated navigation problems [16]. For example in a precision agriculture scenario, the robots may have to navigate in the tight spaces between crops. Within the human body, they may be required to navigate through narrow blood vessels which include sharp bends. Distributed robotic systems can offer a large degree of flexibility for such navigation.

1.2. Problem Definition

In general, mobile robotic entities (i.e. whether monolithic robots or single modules within a multi-robot system) consist of a structural chassis, a number of sensors and actuators, computational hardware and energy storage and/or harvesting devices. The problem of integrating all these components into a single entity is already challenging at the macro scale. In fact, the design processes of robots of even moderate complexity normally involve inter-disciplinary collaborations and require considerable development time.

At smaller scales, the challenge of integrating all the required components into a robot is amplified, as the available space is at a premium. While the robot is expected to act autonomously, the amount and type of information that it can gather from its environment is limited by the sensors that can be fit within it. Moreover, the complexity of the processing that can take place on that information, especially in real-time, is limited by the specifications of the computational hardware that is available, which in turn is also restricted by the available space. Even if a robot does possess the required computational hardware to perform complex information processing, doing so would place increasing stress on its power harvesting/storage subsystems, thereby restricting the time for which it is able to act autonomously.

As a result, it is envisioned that at such small scales, the information acquisition and processing requirements for solving a given task will be the bottleneck that determines whether an implementation is feasible. At larger scales, more sensors and more powerful computational hardware than strictly necessary are sometimes incorporated into robots for the sake of robustness, or to speed up the solution of a task. However, as the scale becomes smaller, this may quickly cease to be possible, and the information acquisition and processing complexity may have to be constrained to the absolute minimum possible, if the task is to be solved at all.

1.3. Aim and Objectives

In light of the above problem definition, the overall aim of this thesis is to advance on the state of the art in terms of the simplicity of solutions to canonical, swarm robotic tasks.

The specific objectives of this project are:

- To conduct a literature review of the history and the current state of swarm intelligence and swarm robotics, and identify canonical swarm robotic behaviors that can be addressed using less information acquisition and processing than has been demonstrated so far.

- To develop a framework for the design and implementation of swarm robotic systems with minimal information processing. This framework shall constitute of both the structure of the solutions, and a methodology for obtaining these solutions for different tasks.
- To evaluate the developed framework in-depth on one selected swarm robotic task. This task shall be relatively simple in order to facilitate a comprehensive and detailed analysis. The obtained solution shall also be implemented on a physical swarm robotic platform.
- To show that the developed framework can be successfully extended to other, more complex swarm robotic tasks. This shall comprise of a less detailed, but nevertheless complete application of the framework, from solution design to validation both in simulation and on a physical platform.
- To exploit the knowledge gained from applying the developed framework to different swarm robotic tasks for suggesting concrete, structured ideas for further analysis and development of the framework.

1.4. Preview of Contributions

- The simplest solution so far — in terms of both sensor and controller complexity — has been presented for the problem of self-organized aggregation in a swarm of robots. Each robot has one binary, line-of-sight sensor that lets it know whether or not there is another robot in front of it. This could in principle be implemented using a single-pixel, monochromatic camera. The solution requires no run-time memory or arithmetic computation on the part of the robots.
- A self-organized aggregation behavior with binary, line-of-sight sensors has been implemented and successfully demonstrated on a physical swarm robotic platform (consisting of 40 robots) for the first time. To the best of our knowledge, this is also the first instance of a solution for self-organized aggregation that was designed using an evolutionary robotics methodology and subsequently, directly ported onto physical robots.
- Mathematical proofs of correctness for the memory- and computation-free sensor/-controller solution to the self-organized aggregation problem have been developed

for the cases of (i) a moving robot and a static robot or circular cluster of robots; and (ii) two simultaneously-moving robots.

- For the first time, the performance landscape of the self-organized aggregation problem has been systematically evaluated over an *entire* controller space, within a finite resolution, by using a grid search (rather than, say, a stochastic search algorithm). Moreover, a method has been designed for producing a visualization of this landscape, and several such visualizations are provided for a number of different configurations of the self-organized aggregation problem.

- The effect of the field of view of the sensor on the performance of self-organized aggregation has been systematically investigated, within the context of a controller that is memory-free and performs no arithmetic computations.

- The performance of a memory- and arithmetic computation-free controller on the problem of self-organized aggregation has been systematically compared with that of a controller with memory. Both types of controllers were synthesized with an evolutionary robotics methodology under identical conditions. It has been found that the controller without memory scales better on systems with more robots than were used in the synthesis process.

- The framework of a discrete line-of-sight sensor combined with a memory- and arithmetic computation-free controller has been extended to the problem of self-organized clustering of objects by robots. This is the simplest solution that has been presented so far for this problem, in terms of both sensor and controller complexity. This solution has been demonstrated on a physical platform of robots and objects. To our knowledge, this is the first successful real-world transfer of an evolved controller for this problem.

- A systematic investigation is presented for the first time of how the performance of a swarm of robots in the self-organized object clustering problem is affected by the ability of robots to distinguish other robots from the background of the environment, and the objects that are being clustered. It has been statistically shown that the swarm robotic system can solve the problem faster if the robots can recognize each other.

1.5. Publications

This thesis represents the author's own work, and includes a number of original contributions to scientific knowledge. The work presented herein has led so far to three publications in academic journals and conferences:

1. **M. Gauci**, J. Chen, W. Li, T. J. Dodd, and R. Groß, "Self-organized aggregation without computation," *The International Journal of Robotics Research (IJRR)*, vol. 33, no. 8, pp. 1145–1161, 2014.
2. **M. Gauci**, J. Chen, T. J. Dodd, and R. Groß, "Evolving aggregation behaviors in multi-robot systems with binary sensors," in *Proceedings of the 2012 International Symposium on Distributed Autonomous Robotic Systems (DARS)*, ser. Springer Tracts in Advanced Robotics, vol. 104. Berlin & Heidelberg, Germany: Springer-Verlag, 2014, pp. 255-367.
3. **M. Gauci**, W. Li, J. Chen, W. Li, T. J. Dodd, and R. Groß, "Clustering objects with robots that do not compute," in *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. Richland, SC, USA: IFAAMAS, 2014, pp. 421–428.

Publications 2 and 3 were presented orally as full papers by the author himself at the respective conferences, held in Baltimore, MD, USA and Paris, France.

The material in Publications 1, 2 and 3 loosely corresponds to the contents of Chapters 3, 4 and 5 of this thesis, respectively. The introductory and related work material from all these three publications has contributed to the contents of Chapters 3 and 4.

During the course of his PhD studies, in addition to his main work, the author has also contributed to other projects that are not featured in this thesis. These have led to the following publications:

1. J. Chen, **M. Gauci**, W. Li, A. Kolling and R. Groß, "Occlusion-based cooperative transport with a swarm of miniature mobile robots." Under review for IEEE Transactions on Robotics (TRO).
2. W. Li, **M. Gauci** and R. Groß, "Coevolutionary learning of swarm behaviors without metrics," *Proceedings of 2014 Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press: New York NY, USA, 2014, pp. 201–208.

3. W. Li, **M. Gauci** and R. Groß, “A coevolutionary approach to learn animal behavior through controlled interaction,” *Proceedings of 2013 Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press: New York NY, USA, 2013, pp. 223–230.

4. J. Chen, **M. Gauci** and R. Groß, “A strategy for transporting tall objects with a swarm of miniature mobile robots,” in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Computer Society Press: Los Alamitos, CA, USA, 2013, pp. 863–869.

5. J. Chen, **M. Gauci**, M. J. Price and R. Groß, “Segregation in swarms of e-puck robots based on the Brazil nut effect,” in *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. IFAAMAS, Richland, SC, USA, 2012, pp. 163–170.

6. **M. Gauci**, T. J. Dodd and R. Groß, “Why ’GSA: a gravitational search algorithm’ is not genuinely based on the law of gravity,” *Natural Computing*, vol. 11, no. 4, pp. 719–720, 2012.

1.6. Thesis Outline

This thesis is structured as follows:

- Chapter 2 provides an overview of background material that places this thesis in context, and a survey of related work on the two case studies in swarm robotics that are addressed in the subsequent chapters of the thesis. The background material starts in Section 2.1 with a historical perspective of the integration of artificial intelligence (AI) into robotic systems. This outlines the progression from abstract, symbol-based reasoning towards a more bio-inspired, signal-based approach based on embodiment. Section 2.2 discusses swarm systems, starting from the characteristics of and the mechanisms behind natural swarm systems (2.2.1), and on to swarm-intelligence-based optimization (2.2.2), and swarm robotics (2.2.3). The evolutionary robotics methodology is used in designing swarm robotic behaviors in the framework developed in this thesis. Therefore, Section 2.3.2 presents a brief discussion of evolutionary computation (2.3) and its use in evolutionary robotics

(2.3.2). Section 2.4 discusses related work for the first case study, which is self-organized robot aggregation. It starts with a review of self-organized aggregation behaviors in nature (2.4.1) and moves on to discuss works from three design methodologies that have been used to address the problem, namely: hand-designed deterministic (2.4.3) and probabilistic (2.4.2) controllers, and evolutionary robotics approaches (2.4.4). Section 2.5 presents related work for the second case study, namely that of the self-organized clustering of objects by robots.

- Chapter 3 introduces the framework developed in this thesis for swarm robotic systems with minimal information processing, within the context of the problem of self-organized robot aggregation. Section 3.1 provides a recap of the context for the contributions to follow. Section 3.2 starts with a technical formulation of the problem being considered (3.2.1), including the robots' sensor and controller structures, and proceeds to a detailed description of the methodology used for its solution. This includes the performance metrics used for synthesizing the robots' controller and evaluating the resulting performance (3.2.2), and the robotic (3.2.3) and simulation (3.2.4) platforms used in the work. Section 3.3 presents a mathematical result that is a consequence of the problem formulation presented in Section 3.2.1. This result justifies the use of an unlimited sensing range in the simulations for synthesizing the controller. Section 3.4 discusses the process of synthesizing controllers using different numbers of robots ($n = 1 \ldots 10$). Section 3.5 analyzes the behaviors of the synthesized controllers, on both the individual robot (3.5.1) and the collective (3.5.2) levels. It is shown that the controller synthesized using $n = 10$ robots scales best in performance to larger numbers of robots. In Section 3.6, the behavior produced by this controller is mathematically analyzed. Proofs and a time upper-bounds are presented for the cases of a moving robot aggregating with a static robot or circular cluster (3.6.1), and of two simultaneously-moving robots (3.6.2). Section 3.7 presents a validation of the solution on a physical swarm robotic platform. It discusses how the sensor and the controller were implemented on the e-puck robotic platform (3.7.1), the experimental setup and procedure used (3.7.2), the results obtained (3.7.3), and the observations made from these results (3.7.4). Section 3.8 summarizes the chapter.

- Chapter 4 presents a number systematic investigations using computer simulations into the self-organized robot aggregation solution introduced in Chapter 3. Section 4.1 investigates how much performance is traded off by the minimalistic sensor/controller solution with respect to a solution that makes use of global knowledge of all the robots' positions. This includes a comparison of the aggregation dynamics (i.e. over time) of both solutions (4.1.1), and an analysis of how their performance scales up with increasing numbers of robots (4.1.2). Section 4.2 discusses the effects of sensor non-idealities that are likely to arise in practical implementations, namely the presence of noise (4.2.1) and the limitations in range (4.2.2). Section 4.3 presents an analysis of a sensor variation. Instead of being a line of sight, the sensor is now modified to have a finite field of view. We discuss how new controllers were synthesized (4.3.1) and post-evaluated (4.3.2) for various fields of view, and discuss the effects observed (4.3.3). Section 4.4 discusses a variation on the controller. Memory is introduced in the form of a neural network with recurrent connections (4.4.1). We discuss how new controllers were synthesized (4.4.3, 4.4.2) and post-evaluated (4.4.4) with and without memory, and discuss the effects observed (4.4.5). Section 4.5 summarizes the chapter.

- Chapter 5 extends the framework introduced in the previous two chapters to the task of self-organized clustering of objects by robots. Section 5.1 describes the methods used, including the problem definition (5.1.1), the characteristics of the objects and the robots (5.1.2), the performance metrics (5.1.1.3), and the evolutionary algorithm used to synthesize controllers (5.1.4). Section 5.2 explains how controllers were synthesized using the evolutionary algorithm for three different sensor configurations, and how post-evaluations were used to select the best of these controllers (in each configuration) for further investigation (5.2.2). Section 5.3 presents the results from a number of simulation studies about the emergent behaviors of the synthesized controllers (5.3.1), their scalability with respect to the number of robots in the environment (5.3.2), and their robustness with respect to sensory noise (5.3.3). Section 5.4 starts by describing how one of the sensor/controller configurations was ported onto a physical platform consisting of 5 e-puck robots and 20 objects (5.4.2). It then outlines the experimental setup procedure used for evaluating this implementation (5.4.2), and presents the results obtained (5.4.3). Section 5.5 summarizes the chapter.

- Chapter 6 concludes the thesis and discusses a number of potential directions for future work.

2. Background and Related Work

This chapter consists of two parts. The first part (Sections 2.1 to 2.3) contextualizes the work presented in this thesis, starting from a broad, historical perspective of artificial intelligence (AI) and robotics, and proceeding to overviews of the fields (i) of swarm intelligence and robotics (Section 2.2), which is the central theme of this thesis; and (ii) evolutionary computation and robotics (Section 2.3), which will be used as a methodology for synthesizing robotic controllers. The second part of this chapter provides a more focused literature review on the specific swarm robotic tasks that are addressed in this thesis, namely self-organized aggregation (Section 2.4) and self-organized clustering of objects by robots (Section 2.5).

2.1. A Historical Perspective of AI and Robotics

The field of swarm robotics, to which this work belongs, appeared around the last decade of the 20th century. Its object of study is the collective behaviors of large groups of relatively simple agents, and it is mainly inspired by biological systems at the scales of bacteria to insects. Historically, the research interest in replicating such 'low-level' biological systems can be understood against the backdrop of a movement that emerged in the late 1980s, emphasizing a bottom-up, behavior-based approach to AI and robotics. This movement, in turn, can be understood within the context of the first AI/robotics research field that appeared in the 1960s (corresponding with the popularization of electronic computers), which attempted to replicate biological intelligence in artificial machines based on the manipulation of high-level, symbolic representations.

In this section, we will take a brief, historical tour of these two AI/robotics movements:

the symbolic, and the emergent[1]. We begin in Section 2.1.1[2] by discussing the origins of symbolic AI and its integration into robotic agents, based on the physical symbol system hypothesis. We show how early successes of this method were followed by somewhat less fruitful attempts to scale the method up to higher levels of complexity and adaptability, leading to a divergence of the fields of AI and robotics: AI focusing on abstract reasoning; robotics on extracting information from the external world and acting within it. In Section 2.1.2, we discuss the emergent AI/robotics paradigm, which attempts to minimize the dichotomy between body and brain by looking at intelligence as an emergent property from the interactions within an agent, and between an agent and its environment (which could also include other agents). In this paradigm, symbolic representations are eschewed in favor of an interplay of sub-systems that operate directly on signals from/in the real world[3].

2.1.1. Classical AI and Robotics

2.1.1.1. The Birth of AI

The first half of the 20th century saw rapid developments in the theory of computation, epitomized by Turing's seminal 1936 paper "On Computable Numbers" [31]. This development was accompanied by the earliest realizations of digital computers, and in 1947 the development of the bipolar junction transistor allowed for more efficient realizations than before, signalling the dawn of the computing age [32].

Researchers soon became fascinated by the prospect of applying the newly-available technology to the development of intelligent machines, and promptly initiated "the quest for artificial intelligence" [33]. The field is generally considered to have been instituted at the 1968 Dartmouth Summer Research Project on Artificial Intelligence (now referred to

[1]A note on terminology. A wide variety of terms have been used to refer to these two 'flavors' of AI/robotics. For the original AI, one finds: "symbolic" [26], "good, old-fashioned AI (GOFAI)" [27], "good, old-fashioned AI and robotics" (GOFAIR) [28], and "cognitive AI" [29]. For the 'new AI' one finds, among others: "embodied AI" [30], "emergent AI" [29], and "subsymbolic AI" [26]

[2]The structure of this section (i.e. 2.1.1) is based on a talk by Prof. Alessandro Saffiotti, entitled "AASS Cognitive Robotic Systems Lab: 15 years integrating AI and robotics", and delivered at the First Örebro Winter School on Artificial Intelligence and Robotics, Örebro, Sweden, 2013.

[3]In this context, the term "world" is often prepended by "real" or "physical" to disambiguate it from other possible meanings, such as an agent's 'internal world'. Unless otherwise stated, the terms "environment" or "surroundings" also refer to the real world.

as the "Dartmouth Conference"), which was proposed and organized by McCarthy, Minsky, Rochester, and Shannon, and attended by several researchers who would eventually come to be considered as founders of the field.

The definition of what constitutes "intelligence" or "intelligent behavior" was (and still is), unsurprisingly, highly debated. Most characterizations that have emerged are definitions-by-proxy rather than exact formulations. In 1950, Alan Turing had proposed his famous "imitation game," now more popularly known as the "Turing test" [34]. Roughly speaking, it suggests that a machine's language-processing ability can be assessed by having a person agent blindly evaluate it against that of another human agent. In the spirit of Turing's idea, Minsky offered the following succinct formulation of AI's objective, shortly after the Dartmouth Conference [35, quoted on p. 20]:

> "Artificial Intelligence is the science of making machines do things that would require intelligence if done by men."[35]

It is remarkable that the early AI researchers aimed directly for human-level intelligence. Indeed, the introduction to the proposal for the original Dartmouth Conference will readily convey the optimism that permeated the field in its infancy: it stated the aim of the conference as being "to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves" [36], and put forward the conviction that "a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer" [36].

Because computers are at their fundamental level, manipulators of symbols, AI became also concerned at its core with building intelligent systems based on symbols. While it was recognized that the architecture of computers is vastly different than that of biological organisms, it was believed that this difference was irrelevant to AI's ambitions, because knowledge could be *represented* in the form of symbols and formal statements using those symbols. Newell and Simon made this belief explicit when they formulated the "physical symbol system hypothesis":

> "A physical symbol system has the necessary and sufficient means for general intelligent action. By 'necessary' we mean that any system that exhibits general intelligence will prove upon analysis to be a physical symbol system. By 'sufficient' we mean that any physical symbol system of sufficient size can be organized further to exhibit general intelligence. By 'general intelligent action' we wish to indicate the same scope of intelligence as we see in humian

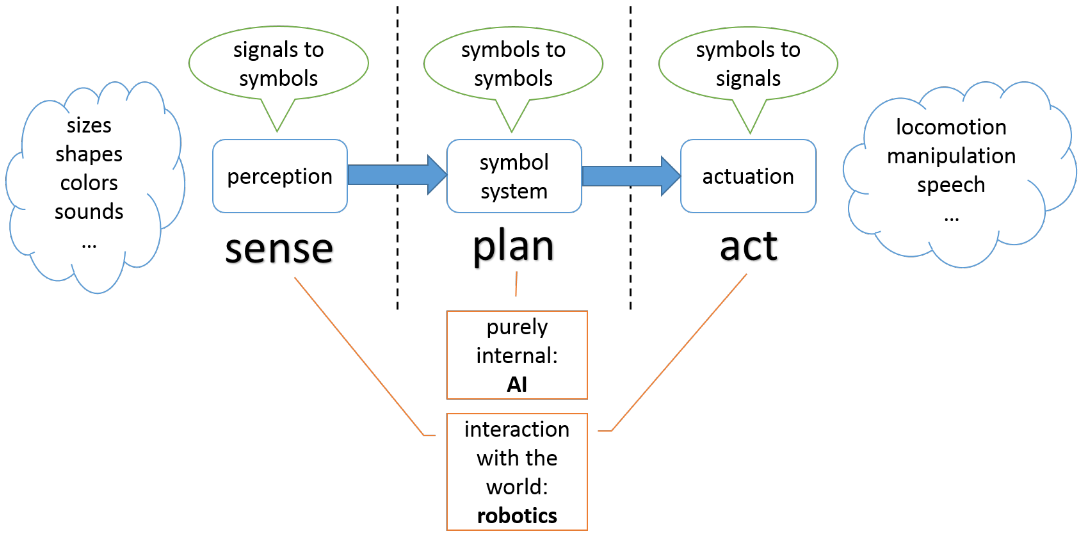


Figure 2.1.: The sense → plan → act architecture for embodying a physical symbol system within a robotic agent.

> [sic] action: that in any real situation behavior appropriate to the ends of the system and adaptive to the demands of the environment can occur, within some limits of speed and complexity.” [37]

2.1.1.2. AI and Robotics Combined

Encouraged by positive results from work in AI, it was a natural progression for researchers to aim to emulate not only ‘abstract’ thought, but also concrete, intelligent action in the physical world. As we have seen in the previous section AI had proceeded from its outset on the assumption symbol processing is sufficient for achieving intelligent behavior, but the physical world certainly does not consist of readily-available symbols. Consequently, an agent that is to act based on the physical symbol system hypothesis must have the means to extract meaningful symbols from the signals that it can pick up from its environment. Moreover, if the agent’s reasoning ‘engine’ acts on these input symbols to produce output symbols, then the agent must have the means of converting these output symbols onto actions in the environment. This idea led to the development of the “sense → plan → act” architecture [38, pp. 5–8] for embodying a physical symbol system within a robotic agent (Figure 2.1). This architecture consists of three modules:

- The *sensing module* consists of physical sensors, and of processing that must be performed on the sensory readings for arriving at meaningful symbols. For example, a camera is a vision sensor that returns a raw image, but by using vision processing algorithms on that image, it is possible to deduce that, say "there is a small red box in the corner of the room"[4]. Both the camera and the vision processing subsystem are part of the robot's sensing module.

- The *symbol system* acts on the symbols extracted by the sensing module and produces (a) *goals* that the robot should achieve and (b) a *plan of actions* for achieving those goals. Building on our example, let us say that the robot has been instructed that 'small red boxes should be deposited to Room 7.' If the robot is not in Room 7, the symbol system would deduce that the state of the world is inconsistent with the given rules, and produce a goal to remedy this situation: 'carry the small red box in the corner of this room to Room 7.' The robot may ne far away from the corner, and would therefore first need to move in that direction. It would then need to grab hold of the box, move to Room 7, and release the box there. The symbol system usually requires a *symbolic representation* of the robot's environment, such as a map, in order to compute its goals and plans.

- The *actuation module* is responsible for the low-level execution of the plan formulated by the symbol system. In our example, for instance, if the plan says "move three meters forward", velocity feedback from the robot's wheels might be used to ensure that the distance moved is indeed three meters.

The sense $\rightarrow$ plan $\rightarrow$ act architecture is a sequential architecture, in the sense that execution always flows from one module to the next (with actuation looping back to sensing), and each module only 'releases' execution once it has finished performing its responsibilities. In practice, it is common to have some primitive behaviors implemented 'below' the main architecture, with a common example being collision avoidance. If for any reason the robot is about to crash into an object, this behavior would override the main one to prevent damage to the robot and/or the environment.

The prototypical example of the sense $\rightarrow$ plan $\rightarrow$ act architecture is a robot called Shakey (Figure 2.2), developed and maintained at the Artificial Intelligence Center of the Stanford Research Institute (now SRI International) between 1966 and 1972 [39]. Shakey's sensing module consisted of a TV camera and a range finder, mounted on a

[4] "Small", "red", "box", "corner" and "room" are all potential symbols in our example. The representation must ensure that the meaning is properly conveyed; e.g. that "small" and "red" pertain to the box and not the room.

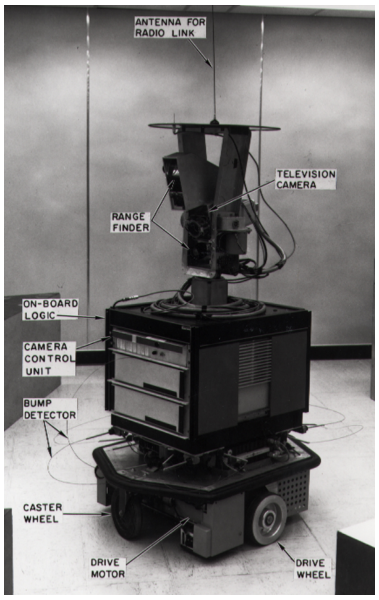


Figure 2.2.: A photograph of Shakey the robot, with callouts describing its various components. Shakey is the prototypical example of the sense → plan → act architecture. Image used under the Creative Commons License.

head that could move in two degrees of freedom: pan and tilt. It had the computational capabilities of storing simple maps of its environment, and of processing the information from its camera and range finder to recognize doors and distinguish between simple objects. In terms of locomotion, it could move straight forwards and backwards, and rotate on the spot, and it was equipped with a low-level collision avoidance behavior as described above. Shakey was capable of formulating goals and plans similar to the one in our example, as was demonstrated by a series of successful experiments during the course of its lifetime.

2.1.1.3. AI and Robotics Diverge

The Shakey project, and other ones similar to it, were an impressive success, especially when one considers the unsophisticated sensors and computational hardware available at the time. Nevertheless, the caveat was that the robots operated in very simplified environments, and under well-controlled conditions. For example, objects had 'perfect' geometric shapes, everything had uniform colors, the floor was flat so that the robot would not get stuck, and so on (see e.g. the technical note on Shakey [39]).

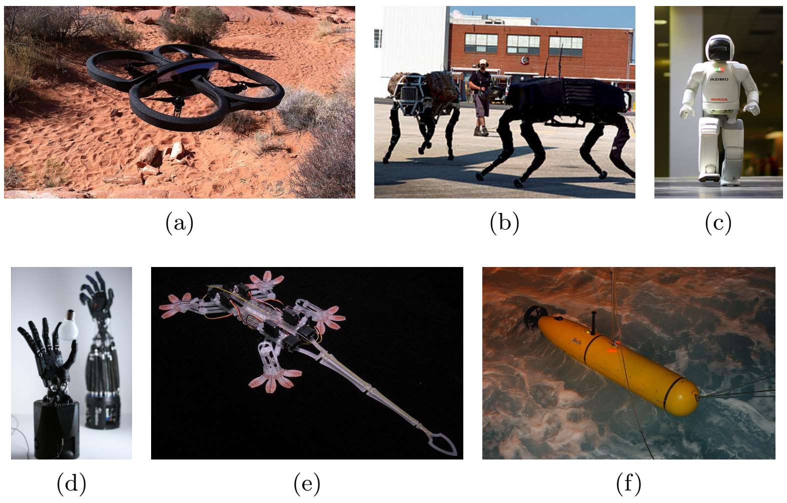


Figure 2.3.: Advances in robotic locomotion and manipulation. (a) AR.Drone 2.0, a quadcopter from Parrot. Quadcopters offer dynamically stable flight dynamics; (b) BigDog, a quadruped by Boston Dynamics, can navigate very rough terrains, such as rubble and snow/ice; (c) ASIMO, a bipedal humanoid from Honda; (d) Dexterous Hand from Shadow Robot Company that can hold a light bulb without breaking it; (e) StickyBot, a quadruped from Stanford University that can climb smooth vertical surfaces; (f) BPAUV, an autonomous underwater vehicle from Bluefin Robotics. Images used under the Creative Commons License.

The next goal was therefore to generalize AI/robotic systems like Shakey to operate in more realistic environments. This required both (a) work on the sensing and actuation modules to become more effective at extracting meaningful symbols from the world and acting within it, such as recognizing objects against textured backgrounds and navigating through rough terrain; and (b) work on the AI module to exhibit more intelligent behaviors, such as coping with uncertain knowledge and unexpected events.

Because these were both vastly challenging problems in themselves, the fields of AI and robotics somewhat diverged from each other over time. Since then, both fields have made significant progress individually. One of the well known crowning achievements of symbolic AI was when in 1997, a computer (Deep Blue) was able to defeat the world champion chess champion of the time, Garry Kasparov [40]. Symbolic AI also provided the underpinnings for pervasive contemporary technologies such as web search engines and "big data" methods [41]. On the other hand, robotics has produced systems that can operate in various challenging environments, by coping with sensing in real-world conditions, and navigating complex terrains (see Figure 2.3 for some examples).

2.1.1.4. Criticisims of Symbolic AI and Robotics

A quest as ambitious as AI was always likely to encounter its share of scepticism and criticism, especially in the light of the extremely optimistic projections put forward by its early proponents. We will here give two examples of philosophers who stood out in their criticism, but the doubt was widespread, and led to a fervent debate between scientists and philosophers.

The philosopher John Searle formulated his “Chinese room argument” in 1980 to illustrate that the “strong AI” position suffered from the symbol grounding problem [42]. By “strong AI”, he referred to the belief that symbolic AI could achieve real cognition, and the outline of his argument was that a human could learn how to manipulate a set of symbols without understanding their true meaning, that is, without having a grounding for the symbols. By manipulating the symbols in the right manner, the person could give an external observer the impression that they understand the language associated with the symbols; in other words, pass the Turing test.

Hubert Dreyfus was another philosopher who opposed the physical symbol system hypothesis[5]. In a 1965 report published by the RAND Corporation, Dreyfus compared symbolic AI’s quest for true intelligence to alchemy, the ancient quest to convert common metals into gold [44]. In his later book “What Computers Can’t Do” [45], he identified what he considered to be four flawed assumptions of symbolic AI: the biological, psychological, epistemological, and ontological assumptions. These flawed assumptions are essentially attacks from different angles on the physical symbol system hypothesis.

In the same book, Dreyfus suggested “alternatives to traditional assumptions”, based on “the role of the body in intelligent behavior” and the potential for “orderly behavior without recourse to rules”. These proposed alternatives were later to serve as the foundations of a new artificial intelligence movement, which is the subject of the next section.

[5]Dreyfus somewhat overshot his criticism of symbolic AI when he pointed out the ineptness of chess-playing computer programs, to the extent that a ten-year-old could beat the state-of-the-art efforts — as it would turn out, structured game playing would become one of the crowning achievements of symbolic AI. Dreyfus himself was eventually challenged to a game of computer chess and lost, and a headline was swiftly run in the Association for Computing Machinery bulletin that “A Ten Year Old Can Beat the Machine, but the Machine Can Beat Dreyfus” [43].

2.1.2. Emergent AI/Robotics

By the last decade of the 20th century, it had become quite clear that the philosophers' skepticism about the potential of symbolic AI was justified and that AI had more or less stalled in its progress towards its original goal:

> "Artificial intelligence started as a field whose goal was to replicate human level intelligence in a machine. Early hopes diminished as the magnitude and difficulty of that goal was appreciated. Slow progress was made over the next 25 years in demonstrating isolated aspects of intelligence." [46]

The above quotation is from what turned out to be one of the seminal papers of a new AI movement by Rodney Brooks, entitled "Intelligence Without Representation". In a closely-related paper published later in the same year, "Intelligence Without Reason" [47], Brooks goes on to say that "The last fifteen years have seen the discipline thundering along on inertia more than anything else."

In [46], Brooks argued that the reason for this lack of success was that the original AI movement had somewhat missed the point by aiming directly for human-level intelligence. He provided an analogy based on a fictional team of engineers from before the invention of artificial flight (AF), who have magically been transported into the future and spent some time in the cockpit of an aircraft. Back in their time, they now have a proof of existence for AF, but as they have been *inside* the aircraft, they set out on "designing pitched seats, double pane windows, and believe that if only they can figure out those weird 'plastics' they will have their grail within their grasp". They have, of course, failed to understand that the real crux of creating AF lies in the mastery of aerodynamics, and that all those other features are merely secondary embellishments.

To relate this back to the quest for AI, Brooks suggested that the real key to AI may lie in first mastering the seemingly less exciting, more primitive behaviors exhibited by creatures of lower cognitive levels than humans. Perhaps as humans, our more refined capabilities, such as symbol processing, cannot be viewed in isolation, but only within the context of our long evolutionary heritage of more primeval, but still intelligent, abilities. After all, "elephants don't play chess" [48]. A look at the timeline of the milestones in the history of evolution serves to reinforce this view:

> "It is instructive to reflect on the way in which earth-based biological evolution spent its time. Single-cell entities arose out of the primordial soup

> roughly 3.5 billion years ago. A billion years passed before photosynthetic plants appeared. After almost another billion and a half years, around 550 million years ago, the first fish and Vertebrates arrived, and then insects 450 million years ago. Then things started moving fast. Reptiles arrived 370 million years ago, followed by dinosaurs at 330 and mammals at 250 million years ago. The first primates appeared 120 million years ago and the immediate predecessors to the great apes a mere 18 million years ago. Man arrived in roughly his present form 2.5 million years ago. He invented agriculture a mere 10,000 years ago, writing less than 5000 years ago and "expert" knowledge only over the last few hundred years." [46]

The cry, therefore, was to, at least for the time being, forget about human-level intelligence, and adopt instead a bottom-up approach, mastering every level of complexity before moving on to the next. Symbols are primarily human constructs, and were therefore to be eschewed in favor of signals, which are more primitive, and exist directly in the real world. This is reflected in the titles of Brook's papers [46, 47], calling for intelligence without representation and reason.

In this new view, intelligence is therefore seen as the *emergence* of complexity from interacting with one's environment. This interaction, in turn, requires *situatedness and embodiment*. Finally, the bottom-up approach was to go hand in hand with building real systems at every step: the *synthetic methodology*. These three concepts are briefly discussed in the following sections.

Emergence

The concept of emergence is at the heart of non-symbolic AI. If symbols are refuted, then, complexity, which is closely associated with intelligence, must come around in some other manner. Vernon defines "emergence" as when:

> "global pattern emerges as something qualitatively different from the underlying assembly of components and, most significantly, is not simply a consequence of the superposition of the contributions of the individual components: they are not just "added together" to produce the result. Something else is involved in the process. It may be due to a particular form of non-linearity in the way the local components combine together to form the global pattern but it may also be due to a mutual influence between the systems local interactions and the global pattern."

Situatedness and Embodiment

These ideas led to both a "new view of intelligence" [30], and also "new approaches to robotics" [49]. These two new perspectives are closely related to each other through concepts that came to be known as *situatedness* (i.e. being in the world) and *embodiment*. In contrast to the classical, dualist standpoint [50] that the body is little more than a vessel for carrying the brain, situatedness and embodiment suggest that in fact, the body and the brain have a synergistic relationship. On the one hand, the brain is responsible for interpreting the signals received from the body and issuing signals back to the body. On the other hand, however — and equally importantly — "the body shapes the way we think" [30]. The following quotation from Margaret Boden offers some examples of the importance of situatedness and embodiment from human perspective:

> "In everyday life you usually remember your "place" largely because the external world is there to remind you what you have or haven't done. For instance, you can check up on whether you have already added the vanilla essence by sniffing or tasting the mixture, or perhaps by referring to the pencil and paper representation of the culinary task that you have drawn up for this mnemonic purpose. A computational system that solves its problems "in its head" rather than by perceiving and acting in the real world, or pencil and paper models of it, has to have all its memory aids in the form of internal representations." [51]

The Synthetic Methodology

The emergent AI/robotics approach places emphasis on the synthetic methodology, or "understanding by building". Understanding in this context refers to insights into (a) the function of biological systems (the biological motivation of emergent AI), (b) the underlying principles of intelligence in general (the cognitive motivation) and (c) the generation of knowledge about building more effective artificial systems (the engineering motivation). Pfeifer and Bongard offer the following description of and motivation for the synthetic methodology:

> "The synthetic methodology [...] can be characterized by the slogan understanding by building. If we are interested in how desert ants find their way back to their nest, or how humans walk or recognize a face in a crowd, we build a system — an artifact — that mimics certain aspects of the behavior we wish to study [...]. This way of proceeding has proved enormously

> powerful: because you have to build something that actually works in the real world, there is no way of glossing over details, which is possible when you formulate a theory abstractly." [30]

The synthetic methodology is often understood to refer strictly to building in the physical world, as opposed to in computer simulations. While simulations are often employed out of necessity, especially in the early stages of building, actual implementations are considered to be the ultimate demonstrations of progress. This is because "simulations are doomed to succeed," [52] meaning that researchers are often prone to (perhaps subconsciously) tweaking simulations until a desirable result is achieved. Consequently, when such a result is achieved, one cannot be certain whether it is still applicable to the physical world, no matter how sophisticated the simulations were deemed to be, until an actual implementation is attempted. It is only the physical world that entirely forbids one from "glossing over details" [30].

2.1.2.1. Behavior-Based Robotics and the Subsumption Architecture

Behavior-based robotics [53] is a general term used to describe a methodology for building robots inspired from a synergistic view of brain and body. The emphasis is placed on sensing and acting directly within the world, rather than relying on abstract representations and reasoning based on those representations.

As early as 1986, Braitenberg had proposed a series of thought experiments using extremely simple "vehicles", and showed that unexpectedly complex behaviors can emerge from atomic behaviors, as long as the vehicles exploit their interaction with the environment [54]. Braitenberg's vehicles had simple sensors that could sense, for example, light intensity and proximity to objects. The signals from these sensors were connected in a direct way to the vehicles' wheels, without abstract symbols entering the picture. In other words, instead of an architecture with deliberate planning, these vehicles employed a "sense $\rightarrow$ act $\rightarrow$ sense" architecture: a perpetual loop of tightly coupled sensing and actuation based on very simple rules. Yet, Braitenberg argued, although the vehicles were only following very simple rules, *to an external observer*, they could be made to exhibit traits such as "fear and aggression", "love" (i.e. attraction to other objects or vehicles, also called "taxis") and "foresight", to mention a few [54].

The question now was whether the 'results' from these interesting experiments could be scaled up to higher levels of complexity, that is whether the sense $\rightarrow$ act $\rightarrow$ sense architecture could be used for problems significantly more challenging than Braitenberg's 'toy' problems. The emergent AI/robotics community responded by coming up with the "subsumption architecture," for integrating basic behaviors like the ones from Braitenberg's vehicles together, and have a new level of complexity emerge from the interactions between these behaviors. This architecture was introduced by Brooks in [55, 46, 47], and a more detailed description, along with many examples, can be found in Arkin's book "Behavior-Based Robotics" [53].

The subsumption architecture consists of a number of atomic, simple behaviors that all act directly on signals from the real world in parallel. These behaviors are arranged in a hierarchy and the higher-level behaviors are able to assimilate, or "subsume", within them all of the lower-level behaviors. It has been shown that when using this architecture, complex behaviors can emerge from the atomic ones [46].

2.2. Swarm Intelligence and Robotics

The field of swarm robotics attempts to replicate swarm behaviors in nature using artificial agents. The motivations for this discipline are twofold. Firstly, such systems can benefit from the strengths of natural swarms. Secondly, in accordance with the synthetic methodology, from building such systems we can learn about the principles behind other systems, e.g. multi-cellular organisms can be seen as swarms of cells.

In this section, we begin by a brief overview of swarm behaviors in nature (2.2.1), and discuss some of the main principles that have been identified across these behaviors. We then take a brief excursion into swarm optimization, which historically came before swarm robotics (2.2.2). Finally, we discuss the field of swarm robotics itself (2.2.3).

2.2.1. Swarms in Nature

We will now briefly discuss the main mechanisms and properties that characterize natural swarms, and that therefore serve as inspiration for the construction of artificial swarm systems, both computational and physical. Each characteristic will be accompanied by illustrative examples, and this section thus also serves as an exposition of a range of well-researched instances of swarm behavior in nature.

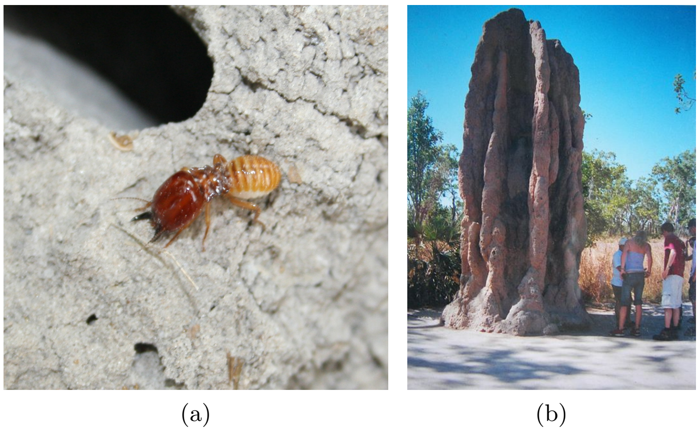


Figure 2.4.: (a) A termite from the *macrotermitinae* subfamily of termites; (b) A mound built by termites, with humans on the right, for scale. Relative to the size of the termites, the height of the structure is the equivalent of a human skyscraper. Images used under the Creative Commons License.

Self-Organization

Self-organization is the central feature of swarm systems. It refers to a bottom-up emergence of behaviors, or patterns, in the absence of top-down mechanisms, such as leadership or shared, high-level plans. It is self-organization that makes swarm behaviors scalable to very large numbers of individuals, allowing them to act in a cohesive manner, as if they were a single entity. Self-organization also plays a major role in making swarm systems extremely robust, in the sense that they are more or less insensitive to the failures or losses of a few individuals.

In order to understand how self-organization works, let us contrast the manner in which humans might go about erecting a new building, to the way in which termites build mounds that offer them protection from the climate and from predators. Note that, relative to the size of the termites, these mounds are equivalent in scale to modern human skyscraper structures (see Figure 2.4).

The building of a skyscraper might go along these lines. A small team of architects, under the lead of a senior, begin by studying the building's structural and functional requirements, and finally produce a design for the building in the form of a set of blueprints. This design is passed on to a small team of project managers — again under the lead of a senior — who study it and decide on an efficient project plan, dividing the work into sub-tasks that can be executed by small teams of builders, and deciding on the order in which these sub-tasks will be carried out. The building phase now starts, and the architects and the project managers will routinely carry out inspections to ensure that the design and the project plan are being adhered to. This whole process is top-down in structure, relying on a well-defined hierarchy of responsibilities, and a global plan that everyone involved in the project is at least aware of and has full knowledge of at least some parts of it.

In stark contrast, there is ample evidence that when building their mounds, termites do not make use either of leadership mechanisms, of global plans [56, 57]. Instead, each termite follows simple rules, based on cues from the environment and other termites. For instance, in building the “royal chamber” of the mound, the queen termite lays down, and the working termites use her pheromone scents as a template for the size and the shape of the chamber (in this scenario, the queen acts as a guide, but not as a leader, in the sense that she is not issuing instructions). Each termite proceeds to roll small balls of mud from near the queen, carry them to the edge of the chamber, and form them into a column-like structure (this constitutes a *positive feedback* mechanism, which will be explained shortly). However, when a column has reached a certain height threshold, the termite will switch its rule; instead of continuing to increase the height of the column, it will start to produce small plates which serve to roof the structure. In this process, the structure can therefore be considered to be an emergent property from the individual behaviors of the termites, due to self-organization.

Another vivid illustration of self-organization involves the synchronization of the flashing of firefly beetles. Some species of fireflies are capable of emitting a small light, and in the mating ‘ritual’, one of the genders (depending on the species) uses this light to attract mates of the opposite gender, and the pair then use their lights to communicate with each other, guiding the flying individual towards the stationary one [58, 59]. It has been observed on several occasions that sometimes, the female fireflies in a tree or a cluster of trees will synchronize the flashing of their lights, leading to a spectacular sight in which it appears as thought the trees themselves are flashing. So impressive is this display, that some early observers refused to acknowledge that it happens altogether; for example in 1917, Laurent [60] wrote:

> “I could hardly believe my eyes, for such a thing to occur among insects is certainly contrary to all natural laws. However, I soon solved the enigma. The apparent phenomenon was caused by the twitching or sudden lowering and raising of my eyelids. The insects had nothing whatsoever to do with

it." [60]

Subsequent research was to prove this view wrong. The synchronization has indeed everything to do with the insects, and nothing to do with the observer's eyelids. The self-organization mechanism by which it happens is now well-understood, and is based on the theory of coupled oscillators. Each firefly's flashing is controlled by an oscillatory center in its brain, which has an associated "excitation" level. If the firefly is in isolation (i.e. not near to other fireflies), this exictation level increases steadily over time ('charges') until it reaches a threshold. At this point, a signal is triggered from the firefly's brain that causes a flash in the firefly's bioluminescent cells. At the same time, the excitation level quickly decreases back ('discharges') to its base value, and the process starts to repeat. This mechanism causes the firefly to flash at a steady rate.

However, when fireflies are located in proximity to each other, their oscillatory centers interact with each other through a mechanism based on the fireflies' flashing. When a firefly perceives a neighbor's flash with a sufficiently high intensity, this causes the excitation level in the firefly's oscillatory center to reset almost instantly to its base value, no matter where it currently is in the cycle. Thus, if the stimulus is received when the excitation level is in the charging portion of the cycle, the flash to follow will be delayed by the time elapsed between the start of the charging and the reception of the stimulus. On the other hand, if the stimulus is received in the discharging portion of the cycle, the current flash will not be affected, as the flashing signal has already been issued by the brain; however, the stimulus causes the discharge process to be sped up, and hence the subsequent flash will happen slightly earlier that it otherwise would have.

Whenever two fireflies that are sufficiently close to each other interact with each other in this manner, their flashing will eventually become synchronized because the cycles of the excitation levels in their oscillatory centers become synchronized. In the context of a large swarm of fireflies, each firefly is only coupled to its immediate neighbors as defined by some distance threshold; however, assuming that there is an unbroken chain of neighbors between any two fireflies (formally, this constitutes a "connected graph"), the coupling effect between the fireflies' oscillatory spreads out throughout the swarm, causing all the fireflies to eventually synchronize their flashing.

The flashing of fireflies is only one example of self-organized synchronization in nature. In fact, this phenomenon manifests itself in many different forms and scenarios, but it is usually based on the same underlying principle of coupled oscillators. A well-known example in humans is the synchronization of clapping which tends to happen naturally during prolonged applause. In this case, it can be argued a mechanism other than coupled oscillators may also be at play, namely, that each person may be more or less consciously synchronizing their clapping with their neighbors by anticipating the neighbors' claps based on visual perception. Then, this phenomenon would not appear to be as "contrary to all natural laws" [60] as the synchronization of firefly flashing, as it would rely on the higher cognitive abilities of humans. However, it has been shown in systematic experiments that the synchronization of clapping in humans can also occur when all the participants are blindfolded [61]. In this case, there is no way for the individuals to directly (i.e. visually) anticipate their neighbors' claps, and the emergence of the synchronization must therefore rely on a similar mechanism to the one in fireflies. For a comprehensive overview of self-organized synchronization behavior in nature, including a treatment of its underlying theory and several examples of its manifestation, see [62].

It is interesting to inquire how and why self-organizing mechanisms came to be evolved. How did termites 'learn' how to collectively build such impressive structures? What caused fireflies to start synchronizing their flashes when they are part of a swarm? In answering these and similar questions, the concept of the *extended phenotype* is particularly useful [63, 64, 65]. In evolutionary biology, the term *genotype* refers to an individual's genetic code (or a part of it), while the term *phenotype* refers to the manifestation of this code [65, 66]. Thus, an individual may have a gene (i.e. part of its genotype) that causes it to have blue eyes, which is one aspect of its phenotype. In contrast to a phenotype, an extended phenotype refers to the manifestation of a genotype that is beyond the confines of the individual itself. In the case of the termites, the mound can be viewed as an extended phenotype, and its quality puts a selective pressure on the termites' genotypes (e.g. termites that are better protected from the environment have a better chance of survival). In the firefly scenario, the quality of the synchronized flashing is the extended phenotype. In this case, the evolutionary advantages associated with the quality of this behavior are still debated [67], but one often-cited reason, at least for some particular species, is that by flashing synchronously, swarms of fireflies become more effective at attracting their sexual counterparts than if they had to flash in a random fashion [68].

Positive and Negative Feedback

In broad terms, feedback is a mechanism within a system whereby the system's output has an effect on its own input, forming a recursive closed loop. In the case where the feedback loop reinforces the system's output (i.e. causes it to grow, in some sense), this is known as *positive feedback*. On the other hand, *negative feedback* is when the feedback loop inhibits the system's output, thus having a self-regulatory effect.

The use of negative feedback is the underlying principle of the discipline of control theory and the associated practice of control engineering [69]. Here, the goal is to force the output of a dynamic system either to remain at a fixed value or to track a desired output. Towards this purpose, negative feedback has proven to be a powerful tool for dealing with deficiencies or inaccuracies in the available system's model as well as with non-deterministic disturbances to the system. On the other hand, the use of positive feedback is normally avoided in the design of control systems, because it "has a destabilizing effect and is usually accompanied by saturation effects that limit the growth of the quantity [under control]" [69]. A well-known, everyday example of this phenomenon is the high-pitched sound that occurs when a microphone is placed too close to a speaker that is amplifying its input.

However, positive feedback is often present in natural systems. In most cases, it is found in combination with negative feedback, which acts to keep it under control. As an illustration of how such an interplay of the two types of feedback might work, consider a child who is building a sand pile on the beach by following the simple rule "*keep adding more sand*". Initially, the pile will grow and grow, but after it has reached a certain slope, the physics at play will cause the addition of any more sand to trigger an avalanche effect, bringing the slope of the pile back to roughly its original value [70, 71].

Both positive and negative feedback have been shown to play a role in several biological systems, both within organisms components, and in interactions among the organisms in a group (e.g. swarm systems) [72, 73]. Examples of feedback mechanisms playing a role within organisms include thermoregulation, cell differentiation [74] and division [75], the organization of lymphoid follicles [76], and the growth and patterning of limbs [77].

On an inter-organism level, positive feedback is usually involved in one form or another wherever social behavior is observed. In flocking [78], for instance, the individuals follow a rule of the form "*align your direction with that of your neighbors*". Consequently, a random change in direction by one individual will often trigger a chain reaction causing all the other individuals to also change their direction, thereby enabling the flock to remain moving as a cohesive unit. At the same time, the individuals will adjust their direction to avoid collisions with neighbors that are too close too them. This acts as a negative feedback mechanism that ensures that each individual has the necessary space to move and execute changes in direction.

Large Numbers of Quasi-Identical and Simple Individuals

Natural swarms are characterized by large numbers of individuals. This distinguishes them from other group behaviors in nature. For instance, wolf packs typically consist of no more than 10 members; in contrast, a single bee hive may consist of up to $50,000$ bees, and ants have been shown to form “supercolonies” (i.e. groups of separate but cooperating colonies) that total in the range of millions — and in some cases even billions — of individuals [79]. The distinction, however, goes beyond the simple measure of numbers, and has a fundamental effect on how the group functions. In small groups, like packs, the members may have distinguishable identities and possibly play individual roles, such as is the case in coordinated hunting [80]. On the other hand, in swarms the number of individuals preclude the possibility of them interacting with each other on a ‘personal’ level. In fact, Gerardo Beni, who coined the term *swarm intelligence* and was later instrumental in founding the field of swarm robotics, made the following observation about the characteristics of natural swarms:

> “It [the artificial system] had some special characteristics, which in fact are found in swarms of insects, i.e., decentralized control, lack of synchronicity[6], simple and (quasi) identical members. Important was also the size, i.e., the number of units. It was not as large as to be dealt with statistical averages, not as small as to be dealt with as a few-body problem. The number of units was thought to be realistically of the order of 10^2 - 10^{23}. So “swarm” was not just a buzz word but a term quite appropriate to distinguish that type of group.” [11]

Beni also mentions that the individuals in a swarm are quasi-identical. This means that the individuals share many of their characteristics with each other, although they may also differ in some, as we shall see shortly in the case of specialization and division of labor. This quasi-identical property arises from the facts that the individuals belong to the same species, and moreover, that the species belong towards the ‘simpler’ end of the biological spectrum, where the individuals do not differ too much from each other in

[6]In this context “lack of synchronicity” refers to the lack of centralized time-keeping. This does not preclude the *emergence* of synchronicity through self-organization, as we have seen in the case of the flashing of fireflies.

terms of ‘personalities’. Even in the case where specialization and/or division of labor are present, the attribute of quasi-identical-ness still holds because the different groups will not be too structurally or functionally different from each other, and moreover, because each group will itself consist of a large number of quasi-identical members. On the whole, therefore, swarms can be described as being *relatively homogeneous*.

Simplicity of the constituent individuals is another characteristic of swarms identified by Beni. This simplicity has two aspects to it. On the one hand, the species involved in swarm behaviors are, as we have just observed, relatively simple when viewed in the context of the entire evolutionary tree. The individuals of these species certainly do not possess the cognitive abilities associated with more refined behaviors, such as symbol processing. On the other hand, and perhaps more importantly, simplicity in this context should also be understood within the context of the global behavior. Termites are relatively simple in relation to the complexity of the mounds that they build; honey bees are relatively simple in relation to the structures of their bee hives and the intricacies of their collective decision making (to be discussed shortly). The word “relatively” here implies that

> “Models based on SO [self-organization] do not preclude individual complexity: they show that *at some level of description* it is possible to explain complex collective behavior by assuming that insects are relatively simple interacting entities. For example, flying from a food source back to the nest involves a set of complex mechanisms, but when one is interested in collective food source selection in honey bees, the primitives that one may use in a model need not go into the detail of the implementation of flight, and flying back to the nest will be considered a simple behavior, although at the neurobiological level of description, flying back to the nest is certainly not simple.” [10]

Together with the principle of self-organization, the characteristics of natural swarms that we have discussed in this section, namely large numbers of individuals, and relative homogeneity and simplicity, render swarms with their impressive flexibility and robustness. Swarms are generally almost completely insensitive to the failure (e.g. death) of single individuals, and in many cases, still adaptable even in cases of failures of larger numbers of individuals (e.g. the invasion of a colony by predators). This quality of swarms is what makes them so interesting from an engineering perspective, which has led to the research effort into computation based on swarm intelligence, and swarm robotics, as we shall discuss in Sections 2.2.2 and 2.2.3.

Simple Communication and Stigmergy

We have seen how collective decision making in swarms of honeybees relies on relatively simple, direct communication mechanisms between the individuals, namely the waggle dance and the stop signal. Similar explicit communication mechanisms can also be found in other instances of swarm behaviors, but they are relatively simple, and are always signal- rather than symbol-based. Moreover, the communication is always local, in the sense that each individuals only exchanges information with other individuals that are within some limited radius [12]. Information can then permeate throughout the swarm via a relay mechanism from individual to individual, as we have seen in the case of the synchronized flashing of fireflies (in that case, the phase of the oscillations can be regarded as the 'information' that is being spread out).

In contrast to the direct transmission of information between individuals, another communication mechanism within swarms is *stigmergy* [10, 73]. This refers to an implicit form of communication that is affected via changes to the environment. The most important properties of stigmergy is that it allows for communication to transcend space and time, in the sense that it allows for individuals to pass information to each other beyond the spatial and temporal constraints that apply to explicit communication.

A well-known example mechanism for stigmergy in insect societies that has been identified and studied for a considerable time is based on pheromones [81, 82], which are chemicals that some insects are able to biosynthesize, secrete and detect [83]. One such application of pheromones to stigmergy is found in the manner in which some species of ants collectively forage for sources of food [84, 85]. In this activity, the ants start by exploring the environment in a random manner. When an ant locates a food source, it carries some of the food back to the nest, leaving a pheromone trail along the way. When other ants that are exploring the environment encounter this trail, they will, with some probability, choose to become recruited to that food source. They will follow the trail, pick up some more food, and carry it back to the nest. In doing so, they will themselves secrete pheromones along the trail, thereby reinforcing it such that subsequent ants are even more likely to become recruited to that food source. This behavior constitutes a positive feedback mechanism, and a negative feedback mechanism is introduced by the fact pheromone trails that are not reinforced for some time will gradually evaporate away. Through this mechanism, the ants are effectively communicating to each other the existence, location and quality of food sources, without necessarily ever being at the same place at the same time. It has been shown that in this manner, the ants can choose the richest among a number of food sources in the environment [86], or the food source with the shortest path to the nest [87, 88].

Collective Decision Making

In 1907, Galton [89] offered a concrete illustration of how the decisions of groups can be better than those of individuals. In a livestock fair, a competition was held whereby visitors would pay a small amount of money to guess the weight of an ox that was on display. Those that were closest to the mark would then be rewarded with prizes. Galton observed that this setup would make for a good case study into collective decision making, because the entry fee and the lure of prizes encouraged the participants to do their best, and therefore analyzed the statistical distribution of the guesses. It turned out that the 'collective guess' (i.e. the median) was significantly more accurate than a randomly-chosen guess, and Galton thus concluded that "the result is, I think, more creditable to the trustworthiness of a democratic judgment than might have been expected" [89].

With collective decision making being such a powerful mechanism, it comes as no surprise that it has evolved as a common phenomenon within swarm systems. Perhaps the most poignant examples can be found in the so-called social insect colonies [90], so much that the collective behavior of honeybees has been termed "honeybee democracy" by Seeley [91]. Honeybees have been observed to employ elaborate group mechanisms when deciding on nectar sources [92]. It has been shown that each bee evaluates the quality of a nectar source independently and objectively, and subsequently indicates this quality to the members of its hive based on the frequency with which it returns to the source, and also via a communication mechanism known as the *waggle dance*. The waggle dance consists of the bee vibrating its lower body, and conveys information about the direction, the distance and the quality of a nectar source. The direction is conveyed by the bee's orientation; the distance by the duration of the waggle dance; and the quality by the amplitude and frequency of the vibrations. Other bees then decide on whether to be recruited to the same nectar source, with the probability of recruitment being inversely correlated with distance of the source, and directly correlated with its quality. The interplay of the assessment of nectar sources by individual bees, and the recruitment of other bees based on the sources' qualities, forms a positive feedback mechanism.

On some occasions, bees are attacked by predators on their way to or from a nectar source, in which case the source becomes a threat to the hive's well-being, irrespective of its quality. It has been shown that the bees deal with this situation by issuing *stop signals* to each other, thus creating a negative feedback mechanism [93, 94]. Bees that have been attacked while visiting a nectar source, upon returning to the hive, issue the stop signal to bees that are performing the waggle dance for the same source. The stop signal takes the form of the issuer bee head-butting the receiver bee with a particular frequency for a short duration [95]. A bee does not always terminate its waggle dance immediately upon receiving a stop signal; rather, it only does so in a probabilistic manner, with the likelihood increasing as it receives more stop signals from multiple bees. Recently, it has been shown that, apart from protecting the bees from danger, the stop signal mechanism also plays a role in avoiding dead-lock situations in the case of multiple options having similar qualities [96].

2.2.2. Swarm Optimization

In this section, we will take a brief detour into swarm optimization before we come to discuss the subject of swarm robotics. The motivation for this detour is that, historically, it can be argued that it was the success of swarm optimization methods that led to the serious research effort into swarm robotics, and also helped to inspire the fundamental principles that shaped the field. Two main swarm optimization techniques, namely Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) appeared around the same time in the 1990s, and quickly attracted the attention of a significant body of researchers. However, it was 'only' in 2001 that the first large-scale project on swarm robotics — the Swarm-Bots project [97] — was started, and it completed successfully in 2006. Here, we will briefly review ACO and PSO in their basic, original forms, as applied to the optimization of real-valued and combinatorial problems, respectively. Since their inception, both ACO and PSO have evolved into research sub-fields in their own right, and many variants of them have appeared, some of which cross into each others' problem domains (e.g. PSO has be applied to combinatorial problems [98], and ACO to real-valued problems [99]). Moreover a number of other swarm-inspired optimization methods have also appeared since the appearance of ACO and PSO. For a broad overview of swarm-based computational intelligence, see e.g. [100].

2.2.2.1. Ant Colony Optimization

In 1991, Dorigo et al. [101] introduced a search algorithm inspired by the behavior of ants foraging for food sources, and applied it to the so-called Traveling Salesman Problem (TSP). Later, this algorithm became known as Ant Colony Optimization (ACO) [102, 103].

The TSP is a well-known combinatorial optimization, that, in its most basic version, goes as follows. A salesman has to conduct a tour of a number of cities, visiting each city once and only once. The problem is to find the tour that minimizes the total distance that the salesman must travel in his assignment. The difficulty in locating this optimal tour is that, for n cities, there are $n!$ possible tours that the salesman has to choose from. This number makes it prohibitively large to search over all possible tours unless n is considerably small ("considerably" depends on the available computational hardware).

In ACO, a number of virtual ants are released at random among the nodes (cities). At first, each ants starts to move from one node to another at random, with the constraint that it does not visit the same node twice. Once it has completed a tour, it calculates the total distance of that tour, and makes its way back along the same route, depositing a pheromone scent as it goes. The strength of this pheromone scent is inversely proportional to the total distance of the tour that the ant had taken. When pheromones have been deposited along the routes, ants start incorporating its presence when choosing which node to visit next. An ant will choose with the highest probability as its next node the one with the strongest link (in terms of pheromone strength) to its current node, and so on. This creates a positive feedback effect, by which the knowledge gained by the ants in their individual tours propagates throughout the system. In order to prevent the algorithm from converging prematurely to sub-optimal tours, the pheromone strengths are also set to evaporate slowly unless reinforced by ants. This effectively acts as a negative feedback effect in the system.

For reviews on advances in the theory and practice of ACO and the current state-of-the-art, see [104, 105].

2.2.2.2. Particle Swarm Optimization

Particle Swarm Optimization (PSO) was introduced by Kennedy and Eberhart [106] in 1995. Its operational principle is based on a swarm of agents ("particles") that move within an n-dimensional landscape that has a fitness value associated with its every point (e.g. $f : \mathbb{R}^n \to \mathbb{R}$). Each agent moves independently within the landscape, but there is an interplay between the agents as they communicate with each other about the fitness of their current and past locations.

In the original form of PSO [106], the change in velocity (magnitude and direction) of each agent i is computed based on a randomized, weighted sum of (i) its own, current velocity; (ii) the best location that it (i.e. agent i) has found so far; and (iii) the best location that the whole swarm has found so far. Formally:

$$\mathbf{v}_i^{(t+1)} = c_0\mathbf{v}_i^{(t)} + c_1 r_1^{(t)}(\mathbf{p}_{i,\text{best}} - \mathbf{p}_i^{(t)}) + c_2 r_2^{(t)}(\mathbf{g}_{\text{best}} - \mathbf{p}_i^{(t)}), \tag{2.1}$$

$$\mathbf{p}_i^{(t+1)} = \mathbf{p}_i^{(t)} + \mathbf{v}_i^{(t+1)}, \tag{2.2}$$

where t is the time index; $\mathbf{p}_i$ and $\mathbf{v}_i$ are the position and the velocity of agent i, respectively; $\mathbf{p}_{i,\text{best}}$ and $\mathbf{g}_{\text{best}}$ are the best locations found by agent i and the whole swarm so far, respectively; r_1 and r_2 are numbers generated uniformly randomly in $[0, 1]$; and $c_0, c_1, c_2 \in \mathbb{R}$ are fixed parameters of the algorithm.

The agents are initially ($t = 0$) dispersed throughout the landscape, usually with a uniform distribution, unless some a priori information about the landscape is available to inform the decision otherwise. Equations 2.1 and 2.2 are then applied to each agent in the swarm. This process is repeated iteratively, until some termination criterion is met, such as a satisfactory solution has been found or a predetermined maximum number of iterations have been computed.

Variants of PSO have been applied successfully to many different types of problems, and much theoretical and empirical research has been carried out into the original algorithm and numerous variations of it. An broad review of this research can be found in [107, 108].

2.2.3. Swarm Robotics

In the previous two sections (2.2.1 and 2.2.2) we have discussed the main characteristics and mechanisms of swarms, and how these inspired successful optimization algorithms starting around the 1990s. Because of the unique and impressive properties of swarms, it was a natural progression for researchers to attempt to emulate these in artificial, robotic systems [11]. The field of *swarm robotics* can therefore be defined as:

> [...] an approach to collective robotics that takes inspiration from the self-organized behaviors of social animals. Through simple rules and local interactions, swarm robotics aims at designing robust, scalable, and flexible collective behaviors for the coordination of large numbers of robots." [14]

Swarm robotics is closely related to reconfigurable robotics [15], but the two are distinct fields. Reconfigurable robotics is concerned with relatively simple robotic entities that can physically attach to each other to self-assemble into meta-structures that can change their structure and function. On the other hand, swarm robotics is concerned with robots that act (physically) independently of each other, as agents. Nevertheless, the two fields can complement each other, in systems where the constituent entities can, as required at different points in time, self-assemble and dissemble to act as as meta-structures or as individual agents.

In the following, we will begin in Section 2.2.3.1 by elaborating on the motivations of swarm robotics as identified in the quote above, namely robustness, scalability and flexibility. In Section 2.2.3.2 also discuss how the inspiration of how these properties arise in natural swarm systems, helped to shape a set of design principles for swarm robotic systems. Finally, in Section 2.2.3.3 we will outline different design methodologies that have arisen to realize the design principles of swarm robotics in practice.

2.2.3.1. Motivations

Swarm robotic systems can be seen as a special case of multi-robot systems [109], in the sense that they are certainly systems that are composed of many robots, but they also adhere to further principles that give them their special identity, as we shall see in the this section and the next one. Parker and Tang [110] argued that, in general, multi-robot systems offer a benefit over monolithic robotic systems when:

> "1) the task complexity is too high for a single robot to accomplish; 2) the task is inherently distributed; 3) building several resource-bounded robots is much easier than having a single powerful robot; 4) multiple robots can solve problems faster using parallelism ; and 5) the introduction of multiple robots increases robustness through redundancy." [110]

Swarm robotic systems, by definition, share all of these motivations, but they place a special emphasis on robustness, scalability, and flexibility, as we have seen in Section 2.2.3

in the definition provided by Brambilla et al. [14]. We will now discuss what these three properties means within the relevant context, and how swarm robotic systems strive to achieve them.

Robustness

Robustness, in general, refers to a system's sturdiness with respect to disturbances to its structural integrity. In a swarm robotics context, disturbances to the system's structural integrity are generally understood to be in the form of the failure of some of the constituent robots. Failures can take multiple forms. For instance:

- A robot may stop performing, or start to execute a different behavior than the one there were intended to; common causes for these failures are battery depletion, and lose contacts and component drifts in a robot's electronic systems.
- As the robots in a swarm robotic system normally have limited perception and communication ranges, a robot can lose contact with the rest of the swarm and become lost in the environment. This might happen, for example, when performing a random walk to explore the environment.
- A robot may get stuck in place, either due to rough terrain, or because it collides with another robot or object in the environment. The latter case may be mitigated by incorporating a collision-avoidance behavior in the robots' algorithms; however, the robots that make up a swarm system may sometimes lack the capabilities for this (e.g. at very small scales).

We have seen how in natural swarm systems, the property of robustness emerges from the mechanism of self-organization, and the characteristics of large numbers of individuals and relative simplicity and homogeneity. In other words, robustness results from each agent performing only a relatively small task, and there being other agents to readily replace it if it fails (dies). In engineering terminology, the robustness property of swarm robotic systems is normally described as there being no *single point of failure*. Thus, for example, a multi-robot system that relies on a leader robot would not constitute a true, robust swarm robotic system, unless it has the property of being able to autonomously appoint a new leader if the current one fails.

Some studies have directly investigated robustness in swarm robotic systems: an overview can be found in [111]. Bjerknes and Winfield [112] warned that robustness does not come for free or by default in swarm robotic systems. Winfield and Nembrini [113] proposed failure mode and effects analysis (FMEA) as a tool for modeling the robustness of such systems, and applied this to a case study of the problem of physical containment and encapsulation. Christensen et al. [114] presented a system in which the robots are able to detect the failure of neighboring robots, based on a flashing behavior inspired by fireflies, and deal with this situation.

Scalability

Scalability, as a general term, refers to the capability of a system to maintain its level of performance as some variable is increased. In a swarm robotics context, this variable is normally taken to mean the number of robots in the system:

> "Scalability requires that a swarm robotic system should be able to operate under a wide range of group sizes. That is, the coordination mechanisms that ensure the operation of the swarm should be relatively undisturbed by changes in the group sizes." [12]

However, it is important to keep in mind that this is not an end in itself, but rather a means to achieve some useful goal(s). For example, increasing the number of robots in a swarm robotic systems could allow the system to cover a larger area, or speed up its performance in cases where "multiple robots can solve problems faster using parallelism" [110]. An illustration of the latter case is found in [115], who investigated the performance of a swarm robotic system that transports an object as the size of the object and of the swarm are increased. For a particular size of object, a 'swarm' of 8 robots was almost completely unsuccessful in transporting the object to the goal location within a set time; 16 robots were around 20% successful, 32 robots around 80% and 32 robots achieved a success rate close to 100%. Moreover, as the size of the swarm was increased further than necessary, to 64 robots, the system's performance did not degrade, but rather increased slightly.

The long-term vision of swarm and reconfigurable robotic systems is to make them scalable to arbitrary numbers of robots. Perhaps, much like the supercolonies of ants that have been observed in nature [79], in the future it will also be possible to build artificial systems that consist of consist of millions of robots [116, 117]. This would open a range of new applications for swarm and reconfigurable robotics.

Flexibility

Flexibility refers to the capability of a system to perform a variety of tasks of a different nature. This property is well-demonstrated in several natural swarm systems. For instance, honeybees are able to construct structurally intricate hives, make 'optimal' decisions collectively, forage for food in a social manner, fend off invaders of their hive, and so on [96].

Although the property of flexibility is in principle achievable in swarm robotic systems, it does not seem from the literature to have received much research attention so far. Most works in swarm robotics at the moment focus on solving one of a set of established problems more efficiently, or in a way that would be particularly appropriate for some specific scenario (e.g. very restricted energy availability). The limited consideration that has been given to the subject of flexibility up until now includes a proposed metric for measuring flexibility [118] and a system that can distinguish between behavioral modes of a swarm that are known a-priori [118], by measuring the movements of a limited number of agents Brown and Goodrich [119].

The integration of several different tasks within the same system, with the ability for a human operator to switch between them [120], or the system to decide on this switching autonomously, is an exciting challenge for the future of swarm robotics. When such flexibility is convincingly demonstrated, especially on physical systems, it will be one of the factors that help swarm robotic systems make the transition from the laboratory to the real world.

2.2.3.2. Design Principles

Based on the mechanisms and characteristics of natural swarm systems (Section 2.2.1), and the motivations discussed in the previous section, a set of design principles have now become more or less established for swarm robotic systems. This is not to say that a system must adhere to each one of these in order to be a swarm robotic system; on the contrary, from an engineering perspective, it is important to keep in mind that biological systems should provide inspiration, not constraints, and it is therefore acceptable for a system to 'violate' some bio-inspired design principle if this clearly benefits the system's performance for a given specification. However, swarm robotic system should exhibit a reasonable number of the established characteristics. Here, we will discuss the list proposed by [14]; slightly different viewpoints can be found in [11, 12, 121]:

- "*Robots are autonomous*": This means that each robot has the ability to get by on its own in all the aspects that it is required to within a given scenario (task or set of tasks). It includes autonomy in terms of energy, sensing, computation, locomotion, and any other actuation, such as manipulation.

- "*Robots are situated in the environment and can act to modify it*": This is a re-iteration of the concepts of situatedness and embodiment, which we have discussed within a broader context in Section 2.1.2. This characteristic distinguishes swarm robotic systems from sensor networks [122]; in the latter, the agents are passive to the state of the environment, and do not have the capability to change it.

- "*Robots sensing and communication capabilities are local*": Here, local is to be understood in a spatial sense: each robot can only sense the environment within some limited radius, and can only communicate with other robots within some limited — but perhaps different — radius. The motivation behind this principle is that when using local sensing and communication, the robots can only learn about their environment gradually, and as [46] argues, "when intelligence is approached in an incremental manner, [...] reliance on representation disappears."

- "*Robots do not have access to centralized control and/or to global knowledge*": This means that each robot perceives the environment individually, and decides on its own actions. There is no centralized entity that monitors the state of the environment and communicates it back to the robots, or decides itself on the actions of the robots and communicate these to them. This property is crucial for robustness, as centralization introduces a single point of failure.

- "*Robots cooperate to tackle a given task*": This means that, while the robots are autonomous and do not share centralized control, they do interact with each other in some meaningful manner, rather than act completely independently of each other.

2.2.3.3. Behavior Design Methodologies

In the previous section, we outlined the design principles of swarm robotic systems. The question now is how these principles are designed into a system in practice. In other words, given some desired collective behavior by the swarm (i.e. at the global level), how does one go about decomposing this behavior into rules for the individuals (i.e. at the local level) that adhere to the aforementioned design principles? This is perhaps the biggest challenge in swarm robotics at the moment, because:

> "Unfortunately, in swarm robotics there are still no formal or precise ways to design individual level behaviors that produce the desired collective behavior. The intuition of the human designer is still the main ingredient in the development of swarm robotics systems." [14]

Because of this, one finds a multitude of different design methods in the literature on swarm robotics. In the following, we shall briefly describe three of the paradigms that seem to appear frequently.

Finite State Machines

Finite state machines are a popular paradigm for designing behaviors in swarm robotic systems. They can be either deterministic, or probabilistic. The former means that for the same set of inputs, a current state will always be mapped onto the same next state. In a probabilistic state machine, on the other hand, this mapping has a stochastic element to it. This methodology has been used, for example, to design aggregation behaviors in swarms of robots, as we shall discuss in detail in Section 2.4.2.

Sense-Compute-Act

This refers to a paradigm where the robots perform a repetitive cycle involving three, discrete steps: gathering information from their environment, computing some action based on that information, and then performing that action. We call this paradigm "sense-compute-act" , because while it is similar in spirit to the "sense-plan-act" paradigm of classical AI/robotics, we want to emphasize that the middle step does not involve elaborate processing based on symbolic representations of the environment, but rather simple computations from readily-available signals.

As an illustration, consider the algorithm for the segregation of robots into annular structures based on the Brazil nut effect, proposed by Groß et al. [123]. In this behavior, the robots emulate physical particles of different sizes that are being shaken in the presence of a gravitational field. At every time step, each robot first measures the relative positions of nearby robots and of a source that is common to the swarm. The robot then computes three vector components based on this information: (i) a repulsion from neighboring robots, emulating gravity; (ii) an attraction to the common source, emulating gravity; and (iii) a random vector, emulating shaking. The robot then computes a weighted sum of these three vector components, and finally moves in the direction of this vector for a distance that is proportional on the vector's magnitude. This behavior was also ported onto a swarm of physical, e-puck robots [124]. In this case, the robots had to perform a slightly more complicated "sense" step. As they did not possess omnidirectional cameras, they had to rotate in steps to take eight separate pictures. The "compute" step then collated these pictures into a panorama before computing the motion vector.

Evolutionary Approaches

This design paradigm attempts to bypass the decomposition of the desired global behavior of the swarm robotic systems into local rules for the individual robots, and of these local rules into sensorimotor maps. This is done by applying the technique of evolutionary computation (to which the next major section will be devoted).

Dorigo et al. [125] presented two of applications of this paradigm to a swarm of *s-bot* robots, evolving aggregation behaviors and coordinated movements. Trianni [126] presented further investigations, also with *s-bot* robots, including synchronization inspired by fireflies.

2.3. Evolutionary Computation and Robotics

Evolutionary robotics [3, 127] refers to the use of black-box metaheuristic methods in the design of robotic structures (i.e. morphologies) and/or behaviors (i.e. controllers). In the next sections, we will begin by discussing black-box metaheuristics in Section 2.3.1.1, and proceeding to evolutionary algorithms, because these are the most widely-used metaheuristics in evolutionary robotics 2.3.1.2. We will then discuss evolutionary robotics in Section 2.3.2.

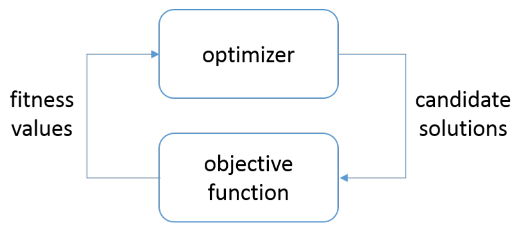


Figure 2.5.: Black box optimization.

2.3.1. Evolutionary Computation

2.3.1.1. Black Box Optimization and Search

Evolutionary computation methods belong to the class of black-box optimization and search methods. Black-box optimization refers to the situation where the optimizer only makes use of calls to the objective function to evaluate specific candidate solutions, but has no explicit knowledge of the function's internal structure (see Figure 2.5). These methods are useful when the function not amenable to being analytically optimized based on its derivatives. For instance, the function may not be differentiable everywhere, or it may not be defined mathematically in closed form: a common example of the latter case is when the fitness function is calculated from the output of a computer simulation.

Real-life fitness functions are often of a very complex nature, being, for example:

- *high-dimensional.* The dimensionality, n, of the fitness function is the number of non-separable (see the next item) objective parameters on which it depends. This corresponds to the (hyper-)volume in which the optimizer must operate: the search space. As the dimension increases, there is normally a combinatorial explosion in the size of the search space, a phenomenon that was termed the "curse of dimensionality" by Bellman [128]. For instance, take a 30-dimensional fitness function where each objective parameter can take 10 possible values; then an exhaustive search will require 10^{30} fitness evaluations. At a rate of 1 μs per fitness evaluation, this would work out to more than the estimated age of the universe [129].

- *non-separable*: the different objective parameters are correlated in a non-trivial manner, and cannot therefore be treated individually. For example, in the case of a single-valued function defined on a real-valued vector space, $f : \mathbb{R}^n \to \mathbb{R}$, (multiplicative) non-separability can be defined by the condition that $f(x_1, x_2, \dots, x_n)$

cannot be expressed in the form $f_1(x_1)f_2(x_2)\dots f_n(x_n)$. Note that if a fitness function is separable, then it can be optimized as n different, one-dimensional problems, which is much easier.

- *highly multi-modal*: multi-modal functions, also known as convex, are ones that have more than one optimum. The one/s with the best fitness value/s is/are known as the global optimum/optima, while the others are called local optima. Multiple optima can arise either (i) due to multiple encodings of the same solution in a given representation, which is a common phenomenon in the optimization of artificial neural networks), or (ii) because there are substantially different solutions to the problem that lie in different regions of the search space.

- *misleading in structure*: this means that the structure of the fitness landscape might not 'naturally' lead to the optima. For instance, the global optimum might be surrounded by a region of low fitness; the bigger this region, the harder it will be to locate the global optimum. Or there could be a cluster of local optima that are close to each other in the search space, and a global optimum that lies in an entirely different location: in this case, the algorithm may be more prone to converge towards the local optima than the global one.

- *non-trivial constraints*: optimization and search problems often involve constraints in one form or another, which is to say that not all the solutions in the domain on which the fitness function is defined can be implemented in practice. The search space can therefore be thought of as being divided into feasible and unfeasible regions. Some constraints can be easily handled, for instance, ones that only apply to a single objective parameter. On the other hand, problems often have more complex constraints that depend on multiple objective parameters; furthermore, there may be a large number of constraints that must be satisfied simultaneously.

- *multiple objectives*: sometimes problems cannot be fully characterized by a single objective, but have rather multiple, possibly competing objectives. For instance, in designing a car engine, it might be desirable to maximize the performance while at the same time minimizing the fuel consumption. It is possible to combine multiple objectives into a single-valued fitness functions; however, the solution/s found will then depend on how this combination is affected (e.g. how the different objectives are weighted). An alternative is to treat the different objectives separately, and much research has been carried out into "multi-objective optimization" [130].

When dealing with black box optimization, we can think in terms of a spectrum of algorithms ranging from pure, uninformed exploration of the fitness landscape, to pure exploitation of the information that is already available, that is, the fitnesses of previously-evaluated candidate solutions:

- On the exploration extreme, we find random search algorithms, which are naïve black box optimizers that simply formulate candidate solutions at random, and keep track of the best solution found that has been so far. Note that if it is additionally ensured that the solutions are not repeatedly evaluated, or if there exists some enumeration of the solutions that can be evaluated systematically, then this becomes an exhaustive search. But as we have seen, the size of the search space is normally prohibitively large for this to be a viable option.

- The exploitation extreme corresponds to so-called "greedy algorithms". These can take different forms, but the underlying idea is that the algorithm always makes the decision that maximizes the short-term return. For example, it may start with a random candidate solution, sample some other solutions in its neighborhood[7], and then choose the best one out of these as the starting point for the next iteration. This is known as "hill climbing". An alternative form of a greedy algorithm is one that starts from a partial solution and builds it up by maximizing the return at each point. For example in the Traveling Salesman problem (Section 2.2.2.1), the algorithm could start from a random city, and then proceed to always choose the next city that is closest to the current one, until the tour is completed. Unsurprisingly, greedy algorithms usually suffer from a high sensitivity to the choice of the starting point, and from premature convergence.

Because of the complexity of real-life problems, and the problems associated with both extremes of the exploration-exploitation spectrum, most useful black box optimization algorithms implement some kind of trade-off between these two aspects, and therefore fall towards the middle of the spectrum. As the internal structure of the fitness function is (at least relatively) unknown, this trade-off has to be implemented in the form of 'soft' (in the sense that they cannot be rigorously justified), but reasonable rules, called "heuristics". For this reason, the field of black box optimization is also sometimes called "meta-heuristics".

[7]In this case, the meaning of "neighborhood" is defined in the problem domain, or in evolutionary computation parlance, the genotypic domain. If optimizing on $\mathbb{N}$, for example, the neighborhood $\mathbf{x}'$ of a candidate solution $\mathbf{x}$ could be defined in terms of the Euclidean norm as $\{\mathbf{x}' : ||\mathbf{x}' - \mathbf{x}|| < r\}$ for some $r \in \mathbb{R}^+$.

2.3.1.2. Evolutionary Algorithms

The distinguishing feature of evolutionary algorithms as black box optimization/search methods is that they take inspiration from the natural evolution metaphor. In an evolutionary algorithm, a number of candidate solutions, individually called genomes and collectively called a population, compete with each other for survival over a number of generations. In each generation, the genomes are evaluated, and a selective pressure is applied so that the fittest among them are more likely to contribute to the next generation.

Another key aspect of evolutionary algorithms is variation. From one generation to the next, new genomes are created via recombination and/or mutation based on the previous genomes. Recombination is inspired from sexual reproduction in nature, refers to a process where material from two or more genomes is somehow mixed together into the new genome. On the other hand, mutation refers to random changes in a genome, which is also a phenomenon that is observed in natural evolution.

It is not the scope of this discussion to enter into the merits of specific classes of evolutionary algorithms, but it is worth mentioning that several exist. Among the most popular are genetic algorithms, evolutionary programming, evolution strategies and differential evolution. A review of these algorithms and more can be found in [131].

2.3.2. Evolutionary Robotics

2.3.2.1. Strengths of Evolutionary Robotics

There is a substantial body of knowledge about hand designing controllers for autonomous mobile robots. For instance, the behavior-based robotics methodology is based on combined basic behaviors, that are straightforward to implement, in a manner so as to achieve more complex behaviors. The work by Braitenberg [132] is considered a landmark in this area; interestingly, however, Braitenberg himself suggests how useful it can be to "give chance a chance".

In designing controllers for autonomous robots, designers often encounter the problem of how to translate the desired behavior onto a set of sensory-motor mappings. Solving this problem for a particular desired behavior can be a long and arduous task, involving a build-up of human 'expertise' into the behavior. For this reason,

> "[a]utomatic robot controller development methods that do not require hand coding or in-depth human knowledge are potentially of great value because it may be possible to apply them to domains in which humans have insufficiency knowledge to develop adequate controllers directly." [133]

Evolutionary robotics is one such automatic development method. When using an evolutionary robotics methodology, the design problem is bypassed, in that the mapping from the behavioral to the sensorimotor level is delegated to the evolutionary process. The task of the designer is transformed into identifying the relevant inputs to the controller, deciding upon a sensible controller structure, setting up an adequate evolutionary algorithm, and finding an effective way of describing the desired behavior by means of a performance evaluation function.

Another advantage of an evolutionary process is that, given a particular selection criterion (i.e. performance evaluation method), the process is solely performance- driven. This is in contrast to humans, who often enter the design process with a set of preconceptions and idiosyncrasies. Evolutionary selection is, as Braitenberg calls it, "the impersonal engineer" [132].

2.3.2.2. Caveats of Evolutionary Robotics

While it is true that an evolutionary robotics methodology can bypass the design problem, setting up the evolutionary robotics process itself is not always a trivial task. As Mataric and Cliff point out,

> "[...] for a real reduction in human effort, the effort expended in designing or configuring the evolutionary system should be less than that required to manually design or configure the robot controllers that it pro- duces." [134]

Nelson et. al admit that evolutionary robotics "is still in its infancy". We know that the methodology has been successfully applied to evolve robotic controllers leading to rudimentary behaviors; the question now is

> "[...] whether the methods used so far to obtain the moderately complex proof-of- concept results reported over the last decade and a half can be generalized to produce more sophisticated autonomous robot control systems." [133]

This, of course, remains to be seen. It is worth pointing out, however, that evolutionary and hand design methods do not necessarily have to exclude each other; in fact, they can be used in conjunction. Baldassare et al. [135] propose one interesting method of doing this. In their approach, the evolutionary methodology is essentially used as a starting point in the design process. Therefore, it has the potential of discovering novel solutions, as discussed in Section 2.3.2.1. When the evolutionary step is finished, the resulting feedforward neural network is statistically mapped onto a set of parametric equations. This allows the designer to better understand *how* the evolved controller works, and the designer can also tune the controller as desired.

Aside from the effort involved in setting up an evolutionary robotics process, there is another cost involved in using the evolutionary robotics methodology: the computing power required to run the evolution process. However, parallel computing techniques can be helpful in this respect, since evolutionary algorithms lend themselves very naturally to a parallel implementation. Nowadays, parallel computing can even be performed on a desktop computer equipped with a graphics processing unit (GPU). Fok et al. [136] describe how evolutionary algorithms can be implemented on such GPUs.

Another important caveat of evolutionary robotics is that the designer is still responsible for setting up the selection criterion for the evolutionary process. Therefore, any preconceptions that find their way into this criterion will naturally reflect themselves in the evolutionary process and, ultimately, in the final result. For this reason, in evolutionary robotics there is a growing interest in designing open-ended evolutionary processes, where the fitness functions are as implicit as possible, and ideally introduce no bias between different solutions that are equally well-performing.

The biggest challenge of evolutionary robotics research is perhaps the so-called reality-gap. In many cases, it is not feasible to evolve a controller on physical robots, because of (i) the time it takes, (ii) potential damage to the robots involved, and (iii) the need for constant human intervention [137]. Therefore, the evolutionary process is normally run on a computer, with the performance evaluations done via simulation. The question is: how well will the evolved controller perform when it is ported onto the physical robot(s)? Several methods have been suggested to address this issue; Floreano et al. [137] list four: (i) introducing noise on the sensors and actuators, (ii) minimal simulations, (iii) using co-evolution to evolve both the controller and relevant simulation parameters and (iv) evolving learning rules that can eventually adjust the controller parameters on the physical robot(s).

The method of minimal simulations is related to the approach taken here, and hence merits some discussion. This method was first proposed by Jakobi [138]. The fundamental idea is that no matter how detailed a simulation is, it can never capture all the properties of the real world. Therefore, making a simulation more and more detailed yields diminishing returns in terms of controller portability. For this reason, the methodology of minimal simulations suggests the counter-intuitive measure of going in the opposite direction. Instead of attempting to capture all of the robot-world interactions, the simulation only captures the essential interactions, which are known as the "base set". Moreover, these interactions are varied randomly to some extent from one performance evaluation to another, so that the evolved controller does not over-adapt to the base set interactions as represented by the simulation. Random variations are also applied to the simulation parameters, so that the evolved controller does not exploit the particular simulation details. Jakobi goes as far as stating that

> "As long as the right amount of variation is included according to the methodology [of minimal simulations], controllers that evolve to be reliably fit will almost certainly transfer into reality." [138]

While "almost certainly" may be somewhat too optimistic, the method of minimal simulations does indeed provide a good framework for evolving controllers that have a good chance of performing well in practice.

2.3.2.3. Performance Evaluation in Evolutionary Robotics

Selection is at the heart of any evolutionary process. In an evolutionary robotics process, the selective pressure is applied according to the performance of every controller and/or morphology. In many cases, the designer specifies a fitness function that is intended to achieve some desired behavior. This function operates upon the behavior of the robot(s) in a simulation and returns a value that reflects the fitness of that particular solution. According to Nelson et al.,

> "[t]he current focus of ER [evolutionary robotics] is on developing methods for evolving controllers capable of performing more difficult and complex tasks, rather than optimizing the evolution process for tasks that have already been achieved" [133].

In other words, the crux of evolutionary robotics is designing fitness functions in such a way that the evolution process returns the desired solution.

Nolfi and Floreano [139] propose a 3-dimensional 'space' in which to view fitness functions. The axis of this space are (i) behavioral-functional, (ii) internal-external and (iii) implicit-explicit. The first dimension measures whether the fitness function is concerned only with the behavior of the robot, or whether it also specifies *how* this is to be achieved (e.g. specifying desired sensorimotor mappings). The internal-external dimension measures whether the fitness function includes only quantities that the robots themselves can measure (internal), or quantities that are only available to an external observer. Finally, the implicit-explicit dimension measures whether the fitness function specifies the goal explicitly or whether it uses an indirect means of doing so.

The implicit-explicit dimension is particularly interesting, because the more implicit a fitness function is, the more free the evolution process is to explore the whole design space. This is why there is an interest within evolutionary robotics in open-ended evolution, which can be seen as the ultimate implicit performance evaluation. In such evolutionary setups, selection is performed, for example, according to how long a robot can survive in its environment. To cite one example, Bianco and Nolfi [140] describe how they set up an evolutionary process aimed towards evolving self-assembly and self-reproduction in a swarm of robots that approached an open-ended process. In their system, it turned out that

> "[s]urvival is typically accomplished by moving as straight and as fast as possible, in order to minimize the risk of being caught by other robots, and by quickly avoiding walls. Reproduction is maximized by the ability to catch other robots by chasing them and by trying to anticipate their movements." [140]

Thus, the selection method by no means rewarded wall-following or straight and fast motion in an explicit manner. Rather, this was evolved because it allowed the robots to survive for a longer time. [8]

[8] There seems to be a lack of consensus regarding the meaning of the term 'open-ended'. Jakobi [138] uses the term with regards to the structure of the neural network being evolved, which has nothing to do with the *aim* of the evolutionary process.

2.3.2.4. Controllers and Algorithms

The fundamental principle of evolutionary robotics is that variation is performed at the controller/morphological level, but selection is performed at the behavior level.

The controller of an autonomous robot is, on a basic level, a map from the sensor readings to the robot's actuators. This map can be implemented in a number of ways, each having their own strengths and weaknesses. Naturally, the choice of algorithm used in the evolutionary process depends to some extent on the controller representation used.

It is possible to evolve a robotic controller as a piece of code. For example, Lazarus and Hu used Genetic Programming [141] to evolve such a controller to perform a rudimentary wall-following behavior [142]. The controller could also incorporate some form of fuzzy logic; for example, Hoffmann evolved, using an evolution strategy [143], a wall-following behavior using a fuzzy controller, as well as an autopilot for a model helicopter [144].

However, "the majority of experiments in evolutionary robotics employ some type of neural controller [139]". Neural networks belong to the connectionist school of thought:

> Connectionism is an attempt to get closer to the physical basis of mind by viewing representations as brain states. [145]

2.4. Case Study: Aggregation

This thesis uses the task of self-organized aggregation as a first case study in order to illustrate the proposed framework for swarm robotic systems with minimal information processing. Trianni [126] argues that in distributed robotic systems, "aggregation is of particular interest since it stands as a prerequisite for other forms of cooperation."

This section discusses how aggregation occurs in nature, and how it has been implemented on distributed robotic systems so far.

2.4.1. Aggregation in Nature

Self-organized aggregation is a widely-observed phenomenon in nature. It occurs in a range of organisms, including bacteria, arthropods, fish and mammals [73, 146]. In some cases, self-organized aggregation is aided by environmental heterogeneities, such as areas providing shelter or thermal energy [147, 148, 149]. However, aggregation can also occur in homogeneous environments, purely as a result of interactions between the individuals. For example, Deneubourg et al. [150] observed that bark beetle larvae aggregate in homogeneous Petri dishes by using a mechanism based on pheromone secretion and detection.

2.4.2. Probabilistic Algorithms

Jeanson et al. [151] investigated aggregation in cockroach larvae and developed a model of their behavior. The cockroaches were reported to join and leave clusters with probabilities that depend on the sizes of the clusters. The larger a cluster is, the more likely a cockroach is to join it, and the less likely it is to leave it, and vice versa. Several works have implemented algorithms based on this model for self-organized robot aggregation [152, 153, 154]. Perhaps most notably, the work of Garnier et al. [152] demonstrated aggregation with 20 physical Alice robots in a homogeneous environment.

Probabilistic aggregation algorithms have the advantage that, as long as the environment is bounded, it is not required to assume that the robots initially form a connected graph in terms of their sensing and/or communication ranges. Nevertheless, by analyzing a model similar to the one in [151] and [152], Correll and Martinoli [155] showed that the "robots [still] need a minimum combination of communication range and locomotion speed in order to aggregate into a single cluster when using probabilistic aggregation rules".

The algorithms in this category require the agents to obtain estimates of cluster sizes or robot densities. In order to meet this requirement, Garnier et al. [152] used local infrared communication to estimate the number of nearby robots. Bayindir and Şahin [153] showed how the cluster size can be estimated from its area using circle packing theory. As an alternative to explicitly estimating cluster sizes, Soysal and Şahin [154] proposed a variation on the probabilistic finite state machine method of aggregation, where each robot emits a sound, and a robot, when in the "approach" state, moves in the direction of the maximum sound it perceives. Kernbach et al. [156] proposed another method of bypassing the direct measurement of cluster sizes, based on a weakly correlated random walk in a heterogeneous environment, whereby a robot, upon meeting another robot, decides on a time to wait before moving on, based on an activation function.

The probabilistic algorithms discussed in this section all assume arithmetic computations on the part of the robots, in order to calculate the probabilities governing their joining and leaving of clusters. A particularly challenging aspect of implementing them on physical robotic systems lies in the estimation of cluster sizes and/or robot densities.

2.4.3. Deterministic Algorithms

Ando et al. [157] introduced an algorithm for achieving aggregation in a group of mobile robots with a limited sensing range. Cortés et al. [158] generalized this algorithm and showed that it can be used to achieve aggregation in arbitrarily high dimensions. These algorithms require that the robots initially form a connected visibility graph, and are based on geometrically ensuring that this graph is maintained in every time step. The robots are required to measure the relative positions (distances and angles) of all their neighbors. Gordon et al. [159] relaxed this requirement, such that the robots need only to measure the angles to their neighbors, and not the distances. Although the algorithm was theoretically proven to work, simulation results revealed that "the merging process [was] generally agonizingly slow" [159]. In order to address this problem, in follow up work, Gordon et al. [160] introduced an additional crude distance sensing capability to the algorithm in [159], where the robots could discriminate between closer and further neighbors.

Gennaro and Jadbabie [161] developed an aggregation algorithm based on every robot computing the Laplacian matrix of its neighbors. Gasparri et al. [162] developed an algorithm for aggregation based on an attractive/repulsive force model that was introduced by Gazy and Passino [163], and generalized further by Li [164]. This algorithm was further developed to handle actuator saturation by Gasparri et al. [165], and to cope with obstacles by Leccese et al. [166].

The algorithms presented in this section so far all require the robots to perform arithmetic computations, based on the positions and/or orientations of perceived robots. To our knowledge, none of these algorithms have been evaluated on physical robots, with the exception of [162, 165, 166]. These algorithms were implemented on 5 SAETTA robots, but the sensing was performed in a centralized fashion, with an external camera being used to measure the distances between the robots.

Yu et al. [167] proposed a connectivity-based algorithm for Dubins-type vehicles that, like this paper, is in the spirit of minimalism. Their algorithm assumes relatively little sensing and control capabilities on the part of the agents; however, for the algorithm to work, each agent has to either be able to track a specific agent that it is assigned to, or at least choose the closest one from a number of agents that are within its “wind shield”. Furthermore, the algorithm requires the robots to synchronise their control when they are in proximity, and it has only been validated in simulation with point-robots.

2.4.4. Evolutionary Approaches

Dorigo et al. [125] addressed the problem of aggregation by using an evolutionary robotics approach [127] in order to design a deterministic control algorithm. In their system, the robots can emit a sound and can sense each other using proximity sensors and directional microphones. A controller, in the form of a feedforward neural network, was evolved and validated in simulation with up to 40 robots. Bahceci and Şahin [168] used a similar setup and investigated the effects of several parameters, such as the number of robots, arena size, and run time.

These controllers all rely on memory and/or arithmetic computation. They have all been evolved and evaluated with simulated embodied robots. The feasibility of implementing them on physical robotic systems remains an open question.

2.5. Case Study: Object Clustering

The task of self-organized object clustering by robots, while having received considerable attention in the literature, has not been as widely investigated as the self-organized robot aggregation problem. Consequently, rather than organizing this literature survey by the type of algorithm used, we will present an account of a more historical nature. We will briefly describe the individual algorithms used in each case along the way, and indicate whether they are of a probabilistic or deterministic nature (it turns out that in some cases, this distinction is not clear-cut). To our knowledge, the problem of self-organized object clustering had not, prior to our work, been approached using an evolutionary robotics methodology.

An early work by Deneubourg et al. [169] (1991) appears to be the first to propose a self-organized, rather than a hierarchical solution to the clustering problem. Their work was inspired from the manner in which ants sort their brood (i.e. larvae). They noted that research performed with ants suggested that ants do not have the ability to explicitly communicate with each other. Moreover, their behavior does not suggest any form of leadership structure. Thus, Deneubourg et al. used these principles to devise a decentralized algorithm for the clustering of objects by robots, and evaluated it using computer simulations. This algorithm was essentially a probabilistic finite state machine, and operated as follows. The (simulated) robots roamed at random in a discretized (i.e. grid) environment, independently of each other. When a robot encountered an object, it decided whether to pick it up or not in a probabilistic fashion. The more isolated the object was from other objects, the more likely the robot was to pick it up. A robot that was carrying an object continued to move at random, and decided whether release the object also in a probabilistic manner. The more (other) objects there were in the neighborhood, the more likely the robot was to release the object that it was carrying. This algorithm was successfully shown to achieve clustering of objects in the environment, although the objects did not generally end up within a single cluster.

The first instance of self-organized object clustering being demonstrated with physical robots seems to be in the work by Beckers et al. [170] (1994). The robots used wheels for locomotion, and were equipped with a relatively large gripper in which a number of objects (called "pucks") could fit. In terms of sensing, the robots had two infra-red proximity sensors, as well as a micro-switch within the gripper that activated when the number of objects being pushed was above a threshold. The robots' controller had the form of a deterministic finite state machine with three states. "Deterministic" here referes to the fact that that the state transitions were not probabilistic. However, the behavior, which is quoted below, did include a random component:

> "When no sensor is activated, a robot executes the default behaviour of moving in a straight line until an obstacle is detected or until the microswitch is activated (pucks are not detected as obstacles). On detecting an obstacle, the robot executes the obstacle avoidance behaviour of turning on the spot away from the obstacle and through a random angle; the default behaviour then takes over again, and the robot moves in a straight line in the new direction. If the robot is pushing pucks when it encounters the obstacle, the pucks will be retained by the gripper throughout the turn. When the gripper pushes three or more pucks, the microswitch is activated; this triggers the puck-dropping behaviour, which consists of backing up by reversing both motors for 1 second (releasing the pucks from the gripper), and then executing a turn through a random angle, after which the robot returns to

> its default behaviour and moves forwards in a straight line. The obstacle avoidance behaviour has priority over the puck-dropping behavior." [170]

This relatively simple behavior was demonstrated to achieve a convincing degree of object clustering using one to five robots. In each case, around 80% of 100 objects ended up in one cluster. However, the use of more robots led to this clustering being achieved substantially faster.

The work of Beckers et al. [170] suffered from the following phenomenon. It was observed that the robots often pushed objects to the boundary of the environment, and it was subsequently unlikely for these objects to be recovered, hence degrading the clustering performance. Holland and Melhuish [171] addressed this problem in a setup similar to that of Beckers et al. [170], with the robots using infra-red proximity sensors and grippers with force sensors. They modified the original deterministic controller of Beckers et al. [170] in the following manner. When a robot was carrying objects and encountered an obstacle (including the boundary), it would not always avoid that obstacle while maintaining the objects it was carrying. Rather, with some probability, it would release its objects and make a random turn. This modification was shown to effectively improve the clustering performance with respect to the boundary phenomenon.

Song et al. [172] built further on the work of Beckers et al. [170] and Holland and Melhuish [171]. They also used robots with force sensors and infra-red proximity sensors; however, they now considered the problem of clustering square objects, rather than circular ones as in the preceding two works. This is a more challenging problem, especially with regards to the boundary phenomenon, as square objects that are stacked along the boundary are harder to retrieve. [172] managed to tackle this problem by exploiting the mechanics of square objects. They used a combination of two basic behaviors, termed "twisting" and "digging", each of which were implemented as deterministic finite state machines (deterministic in the sense of state transitions; the robots sometimes executed random turns).

Maris and te Boekhorst [173] used a slightly different approach from the above works for tackling the self-organized object clustering problem. Their robots used four infra-red proximity sensors — two on each side — but no micro-switch to detect the presence of objects being pushed. The two front-most sensors were sufficiently far apart that an object could fit in between them without activating either one. The readings from a robot's sensors were directly used to drive its two wheels in a deterministic manner, using a simple, feedforward neural network without hidden neurons. The weights of this neural network were set such that, if a robot detected an obstacle on one of its sides, it turned towards the other side, thus avoiding the obstacles. If no sensors were activated, a robot moved straight. This meant that if a robot encountered an object in a head-on manner, it would push it directly forward until one of the robot's sensors became activated for some reason (e.g. the presence of another object). This system was demonstrated to achieve a substantial degree of clustering. However, no clustering behavior was observed with only one robot, and furthermore, the presence of more than four robots was found to introduce a destructive interference that caused the clustering performance to degrade quickly (i.e. the system was not particularly scalable).

Martinoli et al. [174], like Maris and te Boekhorst [173] also addressed the self-organized object clustering problem by using only infra-red proximity sensors and no force sensors, but their method was substantially different. Their robots had six infra-red proximity sensors, and were equipped with a behavior for distinguishing single objects from obstacles based on these sensors (in this context, "obstacles" referred to the environment boundaries, other robots, or clusters of objects). The robots were equipped with grippers for picking up objects, and executed the following simple behavior. Each roamed randomly in the environment, and upon receiving an activation on its sensors, it executed the distinguishing behavior to determine whether the activation was caused by a single object or an obstacle. If the sensor activation was caused by an obstacle, the robot would avoid the obstacle. On the other hand, in the case that the robot had found a single object, (i) if the robot was not already carrying another object in its gripper, it picked up this object; (ii) if it was already carrying another object, it dropped it off next to the one it had found. The robot then resumed roaming the environment. This behavior led to the robots 'clustering' the objects into lines, rather than into compact clusters. In the same work, Martinoli et al. [174] developed a probabilistic model of this behavior, and successfully evaluated it against experiments both in simulation and on a physical platform.

Kazadi et al. [175] developed a model based on an "aphysical system" for self-organized object clustering. By an "aphysical system", they referred to an abstract formulation where the robots were not restricted to any particular embodiment (i.e. sensing and actuation capabilities). The robots interacted with the clusters according to abstractly-stated rules (e.g. pick up an object from one cluster, drop it in another). The model was based on a rate-of-change analysis using differential equations, and it led to a formulation of necessary and sufficient conditions for clustering to occur. Kazadi et al. [175] went on to apply this model to the physical systems of Beckers et al. [170] and Martinoli et al. [174], and achieved reasonable predictions in both cases.

3. Aggregation without Computation

3.1. Introduction

In the previous chapter (Section 2.4), we identified self-organized aggregation as one of the canonical problems in swarm robotics. We discussed a number of works that have proposed solutions to this problem, and organized them into probabilistic and deterministic approaches. Most of these solutions require a significant amount of information processing (sensing and control) on the part of the robots. Probabilistic approaches rely on the robots being able to estimate, explicitly or implicitly, the sizes of clusters of other robots. On the other hand, most deterministic approaches require the robots to estimate the distances — and sometimes the bearings — to other robots. A significant number of the solutions that have been proposed (and most importantly, the one with the least information processing requirements [167]) have only been validated using simulations with point robots, and it is therefore uncertain whether they would be able to directly cope with robot collisions. As such, their effectiveness may rely on additional collision avoidance and/or motion synchronization behaviors being executed by the robots, adding to their information processing requirements.

The relative complexity of these solutions does not make them ideally-suited for implementations at small scales. In light of this, there is a need for a solution to the self-organized aggregation problem which, albeit possibly at some performance cost (such as speed), is extremely lightweight in terms of its requirements on the robots' information processing capabilities. In order to be truly parsimonious, this solution needs to be able to deal with the robots' embodiment implicitly; that is, it should be effective in the presence of collisions among the robots, which will therefore not be required to execute any additional behaviors. Moreover, as we are interested in swarm robotic systems, the simplicity of the solution must not come at the cost of scalability, meaning that, under reasonable conditions, any number of robots should be able to aggregate into a single cluster (at least in their majority). Finally, because “simulations are doomed to succeed” [52], the solution must be capable of coping with real-world conditions, such as sensory uncertainty, and unit failures. Therefore, it should to be demonstrated to work on a system consisting of a substantial number of physical robots.

This chapter begins to address the above challenge. It studies the self-organized aggregation problem in a scenario where the robots are equipped with only one sensor that merely allows them to know whether or not there is another robot in their line of sight. In principle, this sensor can be implemented as a single-pixel camera. Moreover, the robots are constrained to have have no (run-time) memory. As a result of the discrete nature of the robots’ sensors, and the reactive structure of their controllers, it turns out that the controllers do not benefit from being able to perform on-line arithmetic computations (e.g. addition or multiplication). In spite of these severe limitations, it turns out that the problem can be solved effectively, meeting the aforementioned desiderata.

The structure of this chapter is as follows. Section 3.2 presents the methodology used, including a formal definition of the problem (3.2.1), the performance metrics (3.2.2), and the robotic (3.2.3) and simulation (3.2.4) platforms. Section 3.3 proves that the sensor needs to have a theoretically-unlimited range; otherwise, aggregation cannot be guaranteed, irrespective of the controller used. Section 3.4 shows how different optimal controllers were obtained for $n = 2, 3, \ldots, 10$ robots, while Section 3.5 analyzes the behavior of these controllers (3.5.1) and their scalability to $n = 100$ robots (3.5.2). It turns out that only the controller obtained with $n = 10$ robots is scalable, and therefore, this controller is analyzed in greater detail in Section 3.6. This section provides proofs of correctness for this controller for the case with one moving robot and one static robot or circular cluster (3.6.1), and the case with two simultaneously-moving robots (3.6.2). It also provides time upper-bounds for aggregation to occur in each of these two cases. Section 3.7 starts by describing how one of the sensor/controller configurations was ported onto a physical platform consisting of 40 e-puck robots (3.7.1). It then outlines the experimental setup procedure used for evaluating this implementation (5.4.2), presents the results obtained (3.7.3), and discusses the observations made from these results (3.7.4).

3.2. Methodology

In this section, we will begin by formally defining the problem in consideration, and specifying the sensing and control capabilities and restrictions of the robots. Based on this problem formulation, we will then identify suitable metrics for quantifying the performance of a given solution. These metrics are required for two reasons: (i) to be able to synthesize controllers using an evolutionary robotics methodology; and (ii) to evaluate the resulting controllers under various conditions. At the end of the section, we will describe the specific robotic and simulation platforms that will be used to carry out the investigations in the remainder of this chapter (and thesis).

3.2.1. Problem Formulation

We have seen in Section 2.4 that the self-organized robot aggregation problem has been formulated in different ways throughout the literature. One important distinction is whether the environment is homogeneous or heterogeneous; in the latter case, the robots may use environmental cues to help them achieve aggregation [149, 176, 177, 178]. Another differentiation can be made on whether the environment is bounded or unbounded; probabilistic algorithms rely on a bounded environment to ensure that the robots do not drift away while performing a random walk, while deterministic algorithms have no such reliance. Finally, while most works have considered 2 dimensional, obstacle-free environments algorithms have been proposed that scale up to 3 dimensions [158] or take obstacles into consideration [166].

In this work, we consider a homogeneous, 2-dimensional environment with no obstacles. This environment contains a number, n, of robots that are embodied, meaning that they cannot overlap with each other. The robots have a circular body, and move via a differential wheel drive [1]. Each robot is equipped with one binary, line-of-sight sensor at its front, which merely allows a robot to know whether or not there is another robot in its direct line of sight. The robots are memoryless (in the sense that they are not able to remember previous sensor or actuator values), and are homogeneous in their control.

Initially, the robots are placed in arbitrary positions, facing in arbitrary directions. The objective is to aggregate the robots as quickly as possible via decentralized control, into one cluster that is as compact as possible.

The following two sections will provide more details about the robots' sensors and controllers. This will serve to highlight the place of this work in the context of other solutions that have been proposed for the problem at hand, and will concretely illustrate why the solution proposed here is particularly well-suited for implementations at small scales.

3.2.1.1. Sensor

Recall from Sections 2.4 and 3.1 that different studies have used various types and amounts of sensory information to solve the self-organized robot aggregation problem without environmental cues. In the extreme case, each robot may know about all the other members of the group [179], but the reliance on such a large amount of information can be relaxed in two ways: (i) Instead of measuring the relative positions of other robots, the robots estimate the sizes of clusters in their immediate proximity [152, 154, 153] (ii) The robots measure the relative positions of other robots, but only in some limited way [158, 157, 159, 160].

The main advantage of the first approach is that it allows for the sensing to be restricted to a relatively short range. However, extracting the required information may not be straightforward, both in a morphological and in a computational sense. We therefore choose the second approach as a more appropriate candidate for implementing at small scales. One recent study based on this approach stands out with regards to its minimalism: Yu et al. [167] presented a system wherein the robots can only detect the presence of other robots within a visual "wind shield" having a specific field of view.

Taking inspiration from this approach, and furthering its simplicity, we formulate a *binary light-of-sight sensor*, which merely allows a robot to know whether or not there is another robot in the direct line of sight of the sensor (this is equivalent to the "wind shield" sensor in [167] with an infinitesimal field of view). We say that the sensor gives a reading of $I = 1$, if there is a robot in the line of sight, and a reading of $I = 0$ otherwise. Note that this sensor does not (explicitly) provide the robot with information about the distance to perceived robots, or about the number of robots in the sensor's line of sight. Each robot in our system is equipped with one such binary line-of-sight sensor at its front, and the robots do *not* have access to any other form of sensory information (e.g. proximity or tactile sensing).

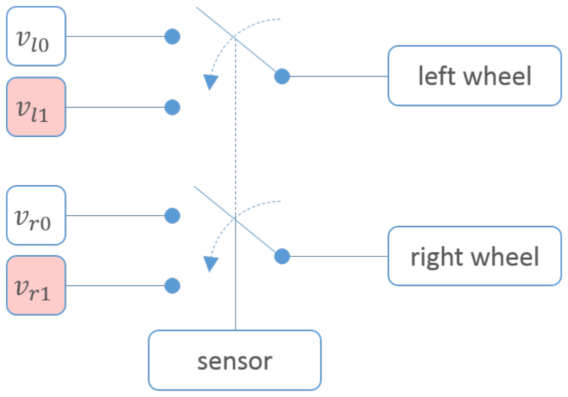


Figure 3.1.: A conceptual schematic of the robot's controller, which is simply a mapping from each possible sensor reading onto a pair of speeds for the robot's wheels. This structure lends itself naturally to a hard-wired implementation.

3.2.1.2. Controller

We have formulated that the robots are equipped with one binary line-of-sight sensor, which can give a reading of $I = 0$ or $I = 1$. Recall that the robots are differential wheeled, meaning that their motion is governed by the speeds of their two, independently-controlled wheels. Let $\bar{v}_l, \bar{v}_r \in [-1, 1]$ represent the normalized left and right wheel velocities, respectively, where -1 (1) corresponds to the wheel rotating backwards (forwards) with maximum speed.

In the interest of simplicity, we impose the constraint that the robots are not equipped with run-time memory, that is, they are not able to store previous sensory information or actuator outputs (such robots are sometimes termed "oblivious" [180]). In other words, at any time step t, $\bar{v}_l, \bar{v}_r \in [-1, 1]$ must depend solely on $I^{(t)}$.

As a consequence of the discrete nature of the sensor (i.e. of I), combined with the constraint that the robots have no run-time memory, it results that the robots' controller can only assume one structure: a mapping from each possible value of I onto a pair of wheel speeds for the robot. Note that any other structure (e.g. a feed-forward neural network) can be distilled down to such a mapping. Formally, we can write this controller as a quadruple:

$$\mathbf{x} = (\bar{v}_{\ell 0}, \bar{v}_{r0}, \bar{v}_{\ell 1}, \bar{v}_{r1}) \,, \; \mathbf{x} \in [-1, 1]^4 \,, \tag{3.1}$$

where $\bar{v}_{\ell 0}$ refers to the normalized velocity of the left wheel when $I = 0$, and so on. Figure 3.1 shows a conceptual schematic of the controller's structure.

Note that the controller's structure would not have been so simple if the sensor did not have a discrete nature, and/or if the robots had run-time memory. If the sensor were continuous, the mapping from I onto $(\bar{v}_l, \bar{v}_r)$ would be able to take the form of any piecewise continuous function. On the other hand, if run-time memory were available, the controller could in general assume more complex structures, such as an artificial neural network with recurrent connections, or an auto-regressive moving average (ARMAX) structure using tapped delay lines.

The most salient feature of the controller given by Equation 3.1 is that it does not call for any arithmetic computations (e.g. addition or multiplication). The practical implication of this is that the robots are not required to possess an on-board microprocessor. In fact, Figure 3.1 suggests a straightforward hard-wired implementation of this controller consisting of, for example, resistors and operational amplifiers[1].

3.2.2. Performance Metrics

In Section 3.2.1, we stated that our objective to aggregate a number of initially dispersed robots as quickly as possible, into one cluster that is as compact as possible. In this section, we will describe the performance metrics that we will use to (i) synthesize controllers for achieving this objective using an evolutionary robotics methodology, and (ii) evaluate the performance of these controllers in different scenarios.

We will describe two metrics: the first one measures the dispersion of the robots in space (which we want to minimize), and will be the one used for synthesizing controllers. The second metric measures how many robots are in the largest cluster. This metric will be used to supplement the dispersion metric when analyzing the performance of synthesized controllers. Note that it is not suitable for synthesizing controllers, as it does not provide any information regarding the robots' compactness.

Dispersion Metric

Different metrics can be used to measure the dispersion of a number of objects (here, disk-shaped robots) in space. One may consider using some measure based on the convex hull of the robots, such as its perimeter or area. However, this presents two

[1]Naturally, more complex controllers that make use of memory could be hard-wired as well. However, they would require more components than resistors and operational amplifiers, at which point the benefits of a hard-wired implementation over a microprocessor-based one may diminish.

problems for our scenario. Firstly, it would be overly sensitive to outlier robots—for instance, a configuration with $n-1$ compactly-configured robots and a single outlier robot may be deemed as being of a lower quality than the same n robots arranged in a less compact configuration, but with no severe outliers. The second problem has to do with future extensibility. In 3-dimensions, such a metric would suffer from the "sausage catastrophe" [181]: the optimal configuration would correspond to the robots arranged in a straight line rather than what we would intuitively consider to be a compact cluster.

Graham and Sloane [182] proposed the second moment of disks as a measure of their dispersion, and we adopt this metric to measure the dispersion of our robots. Let r represent the radius of one robot. Let $\mathbf{p}_i^{(t)}$ represent the position of robot i at time t, and let $\bar{\mathbf{p}}^{(t)} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i^{(t)}$ represent the centroid of the positions of the robots. Then, the second moment of the robots is given by:

$$u^{(t)} = \frac{1}{4r^2} \sum_{i=1}^{n} ||\mathbf{p}_i^{(t)} - \bar{\mathbf{p}}^{(t)}||^2. \tag{3.2}$$

The $4r^2$ in the denominator serves to normalize $u^{(t)}$ such that it becomes independent of r for geometrically similar configurations. $u^{(t)}$ does not have an upper bound, because the robots can be arbitrarily dispersed. It does, however, have a (positive) lower bound, because of the physical constraint that the robots cannot overlap with each other, i.e. $||\mathbf{p}_j^{(t)} - \mathbf{p}_i^{(t)}|| \geq 2r$, $i \neq j$. Graham and Sloane [182] report lower bounds of $u^{(t)}$ for several values of n up to $n = 499$. Except for $n = 212$, the packings corresponding to the lower bounds of $u^{(t)}$ are optimal among hexagonal packings [183].

Other works on robot aggregation have used similar metrics to the second moment, such as the average distance to the centroid [125]. This metric corresponds to Equation 3.2 without the square being applied to the norm, and without the normalizing factor. Our motivation for choosing the second moment was to be able to rely on the lower bound values of $u^{(t)}$ supplied in [182] as baselines when evaluating the performances of our controllers.

Cluster Metric

In order to aid our understanding when analyzing the performance of controllers, we will use a somewhat more intuitive metric to supplement the dispersion metric, as follows. Let a cluster of robots be defined as a maximal connected subgraph of the graph defined by the robots' positions, where two robots are considered to be adjacent if another robot

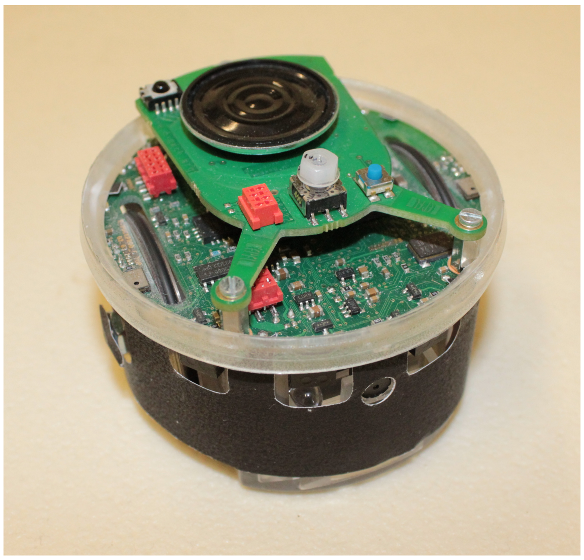


Figure 3.2.: The e-puck robot fitted with a black ‘skirt’. The e-puck’s diameter and height are approximately 7.4 cm and 5.5 cm, respectively.

cannot fit in between them (in other words, robots i and j are adjacent if $||\mathbf{p_j}^{(t)} - \mathbf{p_i}^{(t)}|| < 4r$). The sizes of the clusters can be found by using a depth-first search algorithm [184]. As the objective of this work is to bring all the robots into a single cluster, we define the cluster metric as the proportion of robots in the largest cluster:

$$c^{(t)} = \frac{1}{n} (\text{number of robots in the largest cluster at time } t) . \tag{3.3}$$

Note that this metric provides no information about the configuration of the largest cluster: the same number of robots may be arranged in a compact cluster (minimzing the dispersion metric), or—to cite an extreme case—in a straight line, and would have the same value of $c^{(t)}$. For this reason, the cluster metric will only be used to supplement the dispersion metric when analyzing the performance of controllers, and not for synthesizing controllers.

3.2.3. Robotic Platform

In Section 3.2.1, we specified that the robots considered here shall have a circular body, and are differential wheeled. These characteristics are fulfilled by the e-puck robot [185], shown in Figure 3.2, which is a miniature mobile robot developed at the École Polytechnique Fédérale de Lausanne, that is particularly well-suited for swarm robotics research. The e-puck's diameter and height are approximately 7.4 cm and 5.5 cm, respectively, and its inter-wheel distance is 5.1 cm. The e-puck weighs around 150 g.

The e-puck has a considerable range of communication and sensory capabilities, including Bluetooth and infra-red transceivers, a speaker, three microphones, eight infra-red sensors, a 3-dimensional accelerometer, and a directional CMOS RGB camera. Out of these, we only make use of the e-puck's camera, which is located at its front, to realize the binary sensor in the physical implementation[2] (see Section 3.7)

The e-puck's processor is a Microchip dsPIC micro-controller with 8 KB of RAM and 144 KB of flash memory as secondary storage. This amount of memory is not enough to store a single whole image from the camera, which comes to illustrate the importance of keeping information processing to a minimum on small-scale robotic systems.

3.2.4. Simulation Platform

The simulations were performed using the open-source Enki library [186], which is used by Webots^{TM} [187] in 2-D mode. Enki is capable of modeling the kinematics and the dynamics of rigid bodies in two dimensions, and has a built-in model of the e-puck. In Enki, the body of an e-puck is modeled as a disk of diameter 7.4 cm and mass 152 g. The inter-wheel distance is 5.1 cm. The velocities of the left and right wheels along the ground[3] can be set independently in $[-12.8, 12.8]$ cm/s.

We did not rely on Enki's simulation of the e-puck's camera for implementing the binary line-of-sight sensor. Rather, we extended Enki's model of the e-puck with an additional sensor class, which works by projecting a line (ray) from the robot's front and checking whether it intersects with another robot's body.

In all the simulations performed in this thesis, unless otherwise stated, the length of the control cycle was set to 0.1 s, and the physics was updated at a rate of 10 times per control cycle.

[2]As we will explain in Section 3.7.1, we only use an area of the image that is a single pixel wide. Although this inevitably makes the sensor have a narrow field of view (rather than being strictly a line of sight), the results obtained show that this does not noticeably undermine the system's performance.

[3]This refers to the instantaneous linear velocity of the robot at the point at which the wheel makes contact with the ground. It is equal to the wheel's angular velocity multiplied by the wheel's radius.

3.3. The (Theoretical) Need for an Unlimited Sensing Range

In Chapter 2, we saw that different solutions to the self-organized aggregation problem that have been proposed in the literature impose different requirements on the sensing range of the robots. With probabilistic controllers, the sensors can have a relatively short range, but at the cost of the requirement of a bounded environment. On the other hand, deterministic controllers require that the robots initially form a connected visibility graph.

In this section, we investigate the range requirements of the binary line-of-sight sensor that we formulated in Section 3.2.1. It turns out that using this extremely simple sensor comes at a theoretical cost: aggregation of the robots cannot be guaranteed, even if the robots initially form a connected visibility graph. This is irrespective of the controller used.

In practice, as we shall see (Section 3.7), this theoretical result does not have any serious ramifications: it simply means that the range of the sensor has to be sufficiently long with respect to the initial distribution of the robots. Nevertheless, in this section, we will formally state and prove the above impossibility result, which justifies not limiting the range of the sensors when synthesizing controllers and analyzing their performance in simulation in subsequent sections.

Theorem 1. *Let the range of the sensor be limited to some value δ, and let the robots initially form a connected visibility graph*[4]*. Under the problem formulation of Section 3.2.1, it is impossible to guarantee that the robots will always aggregate into one cluster (as defined in Section 3.2.2), irrespective of the controller used.*

Proof. For the purposes of this proof, we neglect the inertia of the robots, and assume that a robot's velocity changes instantly when its actuation parameters change. For a fixed pair of wheel velocities, in general, a differential wheeled robot moves along a circular trajectory with a center of curvature $\mathbf{c}$. Two special cases arise: the radius of curvature can be zero, in which case the robot rotates on the spot, and it can be infinite, in which case the robot moves in a straight line. Let N denote that a robot does not move at all; S denote that it rotates on the spot (in either direction), and F and B that it moves forwards or backwards along a circular trajectory (in either direction; possibly

[4]In this graph, two robots are defined as being connected if the distance between them does not exceed their sensing ranges.

with an infinite radius of curvature). The controller (see Section 3.2.1.2) maps each of the two possible sensor readings onto one of these four behaviors, meaning that the same sensor reading always results in the same relative displacement (unless the robot collides with other robots). Let us use a tuple $(*_0, *_1) \in \{N, S, F, B\}^2$, to denote what the robot does when $I = 0$ and $I = 1$. The proof now reduces to finding, for each of the $4 \times 4 = 16$ possible tuples, at least one initially-connected configuration of robots with a limited sensing range δ, that does not lead to aggregation.

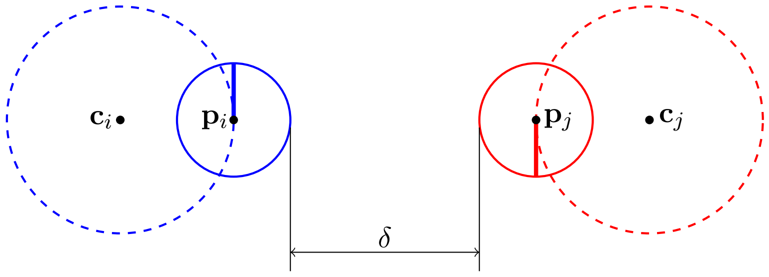
The seven cases where the tuple contains at least one N are trivial, as is the case (S, S). Pathological initial configurations for the cases $(B, *)$ and $(F, *)$ are easily achieved with $n = 2$, where the robots start off at a distance δ apart[5], not seeing each other, and it is ensured that they will never see each other as they move (Figure 3.3a). The case (S, B) is also easily eliminated with $n = 2$ robots that are initialized a distance δ apart: once one of these robots sees the other, it moves backwards and the connectivity is broken and never restored (Figure 3.3b). The remaining case is (S, F). This case cannot be eliminated with $n = 2$; in particular, in the special case where the robots move forward in a straight line when $I = 1$, this controller can easily be shown to always aggregate $n = 2$ robots. We therefore consider $n = 3$ robots: i starts off at a distance δ from j, which in turn starts off at a distance δ from k. With the right set of initial orientations, j sees k first, and moves towards it, while i rotates on the spot. i is now disconnected from j and k, and rotates on the spot indefinitely, while j and k move towards each other (Figure 3.3c). $\square$

3.4. Controller Synthesis

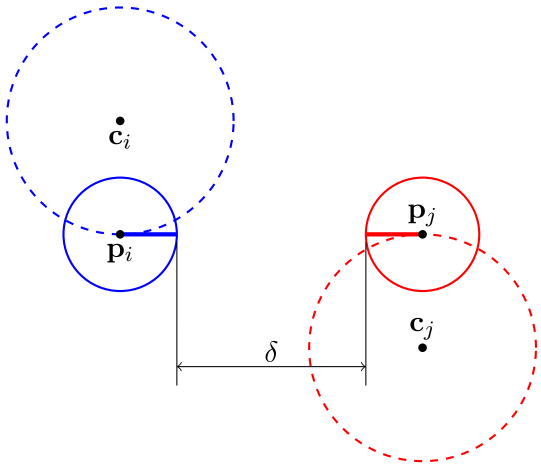
In Section 3.2.1, we specified the structure of the robots' sensors and controllers. The problem now reduces to finding suitable parameters for the controller (Equation 3.1) in order to achieve the stated objective, that is, to aggregate the robots as quickly as possible into one cluster that is as compact as possible.

In order to achieve this, we will use an evolutionary robotics approach based on the dispersion metric specified in Section 3.2.2. As we have seen in Chaper 2, evolutionary robotics methods generally employ stochastic algorithms to search the vast space spanned by the objective parameters. However, in our case, we have only four, bounded

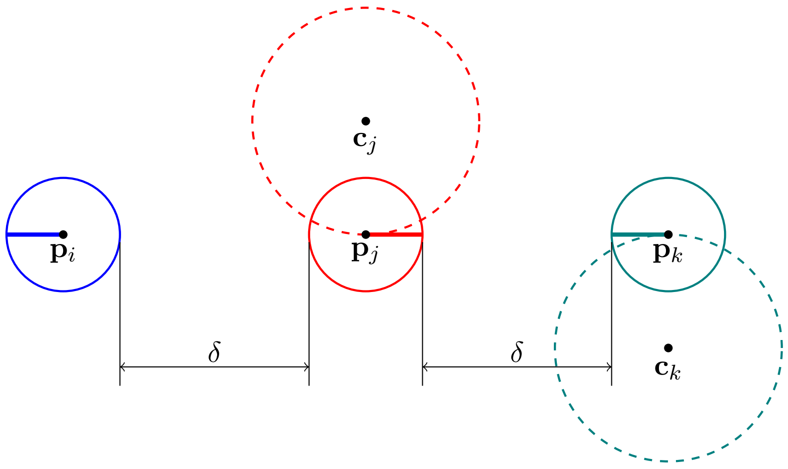
[5] More precisely, the distance δ refers to the length of the sensors. If this is measured between the robots' peripheries, then the distance between their centers is actually $\delta + 2r$, where r is their radius.



(a) A pathological initial configuration for the six cases $(B, *)$ and $(F, *)$.



(b) A pathological initial configuration for the case (S, B).



(c) A pathological initial configuration for the case (S, F).

Figure 3.3.: Pathological initial configurations for Theorem 1. $\mathbf{c}$ and $\mathbf{p}$ denote the center of curvature and the position of a robot, respectively. δ denotes the sensor's range.

objective parameters, namely the four wheel speeds in Equation 3.1. For this reason, rather than relying on a stochastic algorithm, we may perform a grid search over the entire space of controllers, to within some finite resolution. This has two significant benefits:

- A grid search is guaranteed to find the global optimum of the performance landscape (within a finite resolution). This is in contrast to stochastic algorithms which, while normally very effective, offer no such guarantees.
- More importantly, performing a grid search provides us with information about the entire performance landscape (again, to within a finite resolution). We can use this information to render a visualization of this landscape which, in turn, will allow us to gain insights into the problem at hand.

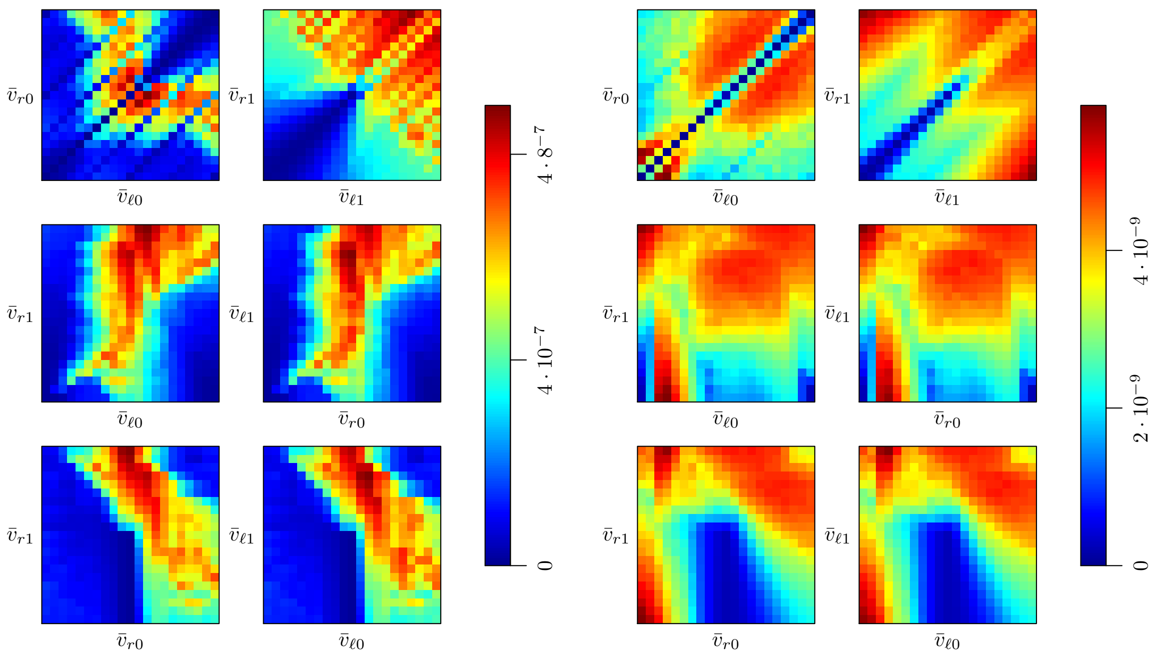
We performed 9 grid searches with each value of $n = 2, 3, \ldots, 10$ robots. We used a resolution of 21 settings per parameter, which means that $21^4 = 194481$ controllers were evaluated for each value of n. Each parameter took values in:

$$\{-1, -0.9, \ldots, -0.1, 0, 0.1, \ldots, 0.9, 1\}.$$

In order to evaluate each controller, 100 simulations employing the controller were run for $T = 1800$ time steps (corresponding to 180 s) with different initial configurations of the robots (but the set of initial configurations was identical for all the controllers). In each simulation, the robots were initialized with a uniform random distribution in a (virtual) square such that the area per robot was, on average, $10000\,\text{cm}^2$ (i.e. the square had sides of length $(10000n)^{\frac{1}{2}}$ cm). The performance of the controller, $\mathbf{x}$, in each simulation $k \in \{1, 2, \ldots, 100\}$, was evaluated by using a metric based on the second moment of the robots, as defined in Equation 3.2:

$$U(\mathbf{x}) = \sum_{t=0}^{T-1} t\, u^{(t)}. \tag{3.4}$$

Equation 3.4 is designed to reward both a low dispersion at the end of the time interval, as well as the speed with which the robots aggregate. It penalizes large values of $u^{(t)}$ at every time instant (except for the very first one), but gives increasing importance to small values of $u^{(t)}$ for increasing t. The overall performance of the controller was calculated as the average of these values, i.e. $\bar{U}(\mathbf{x}) = \sum_{k=1}^{100} U_k(\mathbf{x})/100$.



(a) $n = 2$ (b) $n = 10$

Figure 3.4.: Visualizations of the controller performance landscapes for (a) $n = 2$ robots and (b) $n = 10$ robots. Each box shows the landscape over two out of the four controller parameters. Each point represents the maximum performance that is obtainable over the space of the two omitted parameters. The performance is shown as the reciprocal of the value of Equation 3.4, meaning that higher (red) is better. See the text of Section 3.4 for more details.

The performance landscapes are 5-dimensional (4 controller parameters plus the performance measure, $\bar{U}$), and cannot be visualized directly. Therefore, we considered each combination of two controller parameters ($\binom{4}{2} = 6$ combinations) as a sub-space, and for each point in this sub-space, we calculated the performance measure as the best (i.e. the minimum) that is achievable with the remaining two parameters as degrees of freedom. For instance, on the sub-space $(\bar{v}_{\ell 0}, \bar{v}_{r0})$, the performance measure $\bar{U}^* (\bar{v}_{\ell 0}, \bar{v}_{r0})$ was calculated as:

$$\bar{U}^* (\bar{v}_{\ell 0}, \bar{v}_{r0}) = \min_{\bar{v}_{\ell 1}, \bar{v}_{r1}} \bar{U} (\bar{v}_{\ell 0}, \bar{v}_{r0}, \bar{v}_{\ell 1}, \bar{v}_{r1}) \, .$$

Figure 3.4 shows the performance landscapes of the *reciprocal* of $\bar{U}^*$ over the 6 sub-spaces corresponding to the grid searches with $n = 2$ and $n = 10$. The reciprocal was taken in order to improve the plot's visuals, because otherwise, extreme values of U, caused by the robots diverging due to a badly-performing controller, dominate the landscape and render the other points indistinguishable from each other. The landscapes corresponding to the grid searches with $n = 3, 4, \ldots, 9$ are provided in Appendix B.

As the robots used here are symmetrical, if their left and right wheel velocities are inverted, their behavior is unaffected, other than that it is 'inverted' in space. In other words, if we take any controller and swap $\bar{v}_{\ell 0}$ with $\bar{v}_{r0}$, and $\bar{v}_{\ell 1}$ with $\bar{v}_{r1}$, the resulting controller will be isomorphic to the original one. As a consequence of this fact, we expect (i) the landscapes on $(\bar{v}_{\ell 0}, \bar{v}_{r0})$ and $(\bar{v}_{\ell 1}, \bar{v}_{r1})$ to be symmetrical along the diagonals $\bar{v}_{\ell 0} = \bar{v}_{r0}$ and $\bar{v}_{\ell 1} = \bar{v}_{r1}$, respectively, (ii) the landscape on $(\bar{v}_{\ell 0}, \bar{v}_{r1})$ to be identical to the one on $(\bar{v}_{r0}, \bar{v}_{\ell 1})$, and (iii) the landscape on $(\bar{v}_{r0}, \bar{v}_{r1})$ to be identical to the one on $(\bar{v}_{\ell 0}, \bar{v}_{\ell 1})$. The landscapes shown in Figure 3.4 agree with these expectations. The small discrepancies that are present arise from the fact that the controllers are only evaluated a finite (100) number of times.

We now analyze the performance landscapes. The landscape obtained with $n = 2$ robots appears to be unimodal. The landscape on the sub-space $(\bar{v}_{\ell 0}, \bar{v}_{r0})$ (top left plot in Figure 3.4) shows a marked 'valley' along the diagonal defined by $\bar{v}_{\ell 0} = \bar{v}_{r0}$, implying that having the robots move in a straight line when they do not perceive another robot leads to a poor performance. The landscape on the sub-space $(\bar{v}_{\ell 1}, \bar{v}_{r1})$ (top right plot in Figure 3.4) also shows a 'valley' along the diagonal defined by $\bar{v}_{\ell 1} = \bar{v}_{r1}$, implying that it is sub-optimal for the robots to move straight towards perceived robots.

As n increases, the region of high performance in the landscape obtained with $n = 2$ robots start to become less marked, and a new region of good performance starts to emerge (first clearly visible with $n = 7$, see Appendix B) (observe the bottom-left corners of the lower two rows of subplots). With $n = 10$, although the landscape is still multi-modal, this region has a higher performance than the region that had a high performance with $n = 2$.

3.5. Controller Behaviors and Scalability

We are now in possession of the optimal controllers corresponding to $n = 2, 3, \ldots, 10$. In this section, we will first analyze the structural similarities and differences of these controllers themselves—in evolutionary robotics terms, these are the "genotypes"—and infer the behavior that they lead an individual robot to execute. As we are ultimately

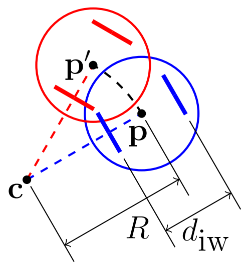


Figure 3.5.: Differential wheeled robot kinematics. Over a short time interval, the robot moves along a circular trajectory from position $\mathbf{p}$ (blue) to position $\mathbf{p}'$ (red). R and $\mathbf{c}$ denote the radius and the center of curvature, respectively, and d_{iw} denotes the robot's inter-wheel distance.

interested in the behavior of the swarm, we will then investigate the emergent behaviors that can be observed when these controllers are scaled up to a large number of robots (relative to the numbers that were used in synthesizing them). In evolutionary terms, these emergent behaviors can be considered as the "phenotypes" corresponding to the controllers found.

3.5.1. Individual Robot Behaviors

In this section, we will analyze the behavior of an individual robot under each of the 9 controllers returned by the grid searches with $n = 2, 3, \ldots, 10$ in the previous section. We will start with a brief overview of the kinematics of differential wheeled robots.

Recall from Section 3.3 that for a fixed pair of wheel velocities, a differential wheeled robot moves, in general, along a circular trajectory of radius R, with an angular speed ω. As shown in Figure 3.5, let $\mathbf{p}$ represent the position of a differential wheeled robot, $\mathbf{c}$ represent its center of curvature, d_{iw} represent its inter-wheel distance, and v_ℓ and v_r represent its left and right wheel velocities along the ground, respectively. Then, it can be shown [1] that R and ω are related to v_ℓ and v_r by[6]: .

$$R = \frac{d_{\text{iw}}}{2}\left(\frac{v_r + v_\ell}{v_r - v_\ell}\right), \qquad \omega = \frac{1}{d_{\text{iw}}}\left(v_r - v_\ell\right). \tag{3.5}$$

Table 3.1 shows the parameters $(\bar{v}_{\ell 0}, \bar{v}_{r0}, \bar{v}_{\ell 1}, \bar{v}_{r1})$ of the optimal controllers for $n = 2, 3, \ldots, 10$ robots, obtained from the grid searches in Section 3.4. For each controller,

[6]Note that if $v_\ell = v_r$, R becomes infinite and ω becomes 0. This means that the robot moves along a straight line.

n	$\bar{v}_{\ell 0}$	$\bar{v}_{r0}$	$\bar{v}_{\ell 1}$	$\bar{v}_{r1}$	R_0	ω_0	R_1	ω_1
					cm	rad/s	cm	rad/s
2	0.2	0	0.7	1	−2.55	−0.50	14.45	0.75
3	0.3	0	0.7	1	−2.55	−0.75	14.55	0.75
4	0	0.3	1	0.7	2.55	0.75	−14.55	−0.75
5	0.5	0.1	0.6	1	−3.83	−1.00	10.2	1.00
6	0.1	0.5	1	0.6	3.83	1.00	−10.2	−1.00
7	0.2	0.6	1	0.6	5.10	1.00	−10.2	−1.00
8	0.2	0.6	1	0.6	5.10	1.00	−10.2	−1.00
9	0.8	0.4	0.6	1	−7.65	−1.00	10.2	1.00
10	−0.7	−1	1	−1	14.45	−0.75	0	−5.02

Table 3.1.: The parameters $(\bar{v}_{\ell 0}, \bar{v}_{r0}, \bar{v}_{\ell 1}, \bar{v}_{r1})$ of the optimal controllers for $n = 2, 3, \ldots, 10$ robots, obtained from the grid searches. Also shown with each controller are the corresponding radii of curvature and angular velocities of the robot for the two possible sensor readings $I = 0$ (R_0, ω_0) and $I = 1$ (R_1, ω_1).

the table also shows the corresponding radii of curvature and angular velocities of the robot for the two possible sensor readings $I = 0$ (R_0, ω_0) and $I = 1$ (R_1, ω_1), calculated from Equation 3.5. Recall from Section 3.4 that two controllers are isomorphic if each one is given by swapping $\bar{v}_{\ell 0}$ with $\bar{v}_{r0}$, and $\bar{v}_{\ell 1}$ with $\bar{v}_{r1}$ in the other.

Let us start by analyzing the behavior of the controller obtained with $n = 2$. With this controller, a robot moves forward along a clockwise/counter-clockwise trajectory with a very small radius when $I = 0$, and moves forward along a counter-clockwise/clockwise trajectory of a larger radius when $I = 1$. The reason why this happens to be the optimal controller for 2 robots is not hard to understand intuitively. Consider the assumptions that (i) the robots' sensing is continuous and (ii) their changes in velocity are instantaneous (i.e. no inertia). Under these conditions, the controller $(\bar{v}_{\ell 0}, \bar{v}_{r0}, \bar{v}_{\ell 1}, \bar{v}_{r1}) = (-1, 1, 1, 1)$ or $(1, -1, 1, 1)$ trivially guarantees aggregation, and is probably the fastest way of achieving it. This conjecture is supported by the fact that a grid search that was re-run with $n = 2$, but with the sampling time reduced from 0.1 s to 0.01 s, returned precisely this controller. Returning to our original setup with a finite sampling rate, it turns out that the high angular velocity when $I = 0$ becomes problematic: the faster the robots turn, the more likely they are to miss each other for a large number of successive revolutions. This explains why, in the actual optimal controller obtained from the grid search, while the robots do rotate almost on the spot when $I = 0$, they do so with a low angular velocity. The fact that the robots do not move perfectly straight when seeing another robot also seems to be an artifact of having a finite sampling rate.

The controllers obtained with $n = 3$ and $n = 4$ are isomorphic to each other. Moreover, they are behaviourally similar to the one obtained with $n = 2$. The sole difference is in one of the parameters corresponding to $I = 0$, which only implies a slightly higher angular velocity. The controllers obtained with $n = 5$ and $n = 6$ are also isomorphic to each other, and once again, behaviourally similar to the preceding controllers. This time, however, there is slightly less of a difference between the radii of curvature when $I = 0$ and $I = 1$. The controllers obtained with $n = 7$ and $n = 8$ are identical. The only difference between them and the ones obtained with $n = 5$ and $n = 6$ is a slightly larger radius of curvature for $I = 0$. The controller obtained with $n = 9$ once again only differs from the preceding controller in a larger radius of curvature for $I = 0$.

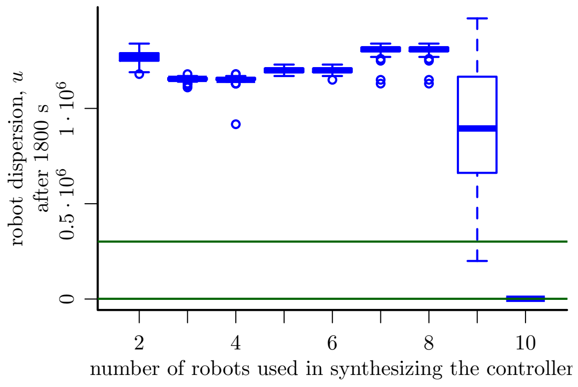
The controller obtained with $n = 10$ is strikingly different from all the previous controllers. With this controller, a robot moves backwards along a clockwise (or counter-clockwise for the isomorphic controller) circular trajectory when $I = 0$. The radius of this trajectory is much larger than any of those encountered in the previous controllers. When $I = 1$, the robot rotates clockwise/counter-clockwise on the spot with the maximum possible angular velocity.

3.5.2. Emergent Behaviors and Scalability

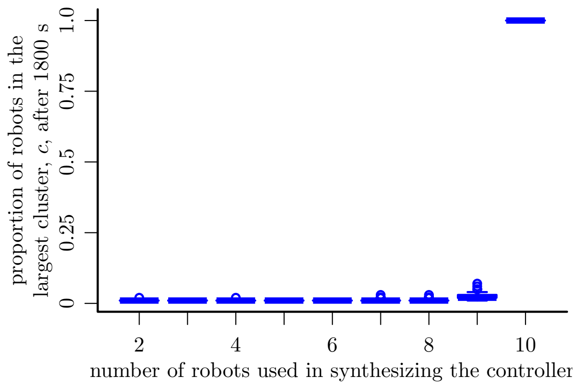
In this section, we investigate the emergent behaviors and the scalability of the controllers obtained from the grid searches with $n = 2, 3, \ldots, 10$ when employed on $n = 100$ robots.

The box plot[7] in Figure 3.6 (see the caption for details) shows that the performances of the controllers obtained with $n = 2, 3, \ldots, 8$ robots perform very badly on $n = 100$ robots. In fact, the dispersion of the robots after 1800 s is much larger than their

[7]The box plots presented in this thesis are all as follows. The line inside the box represents the median of the data. The edges of the box represent the lower and the upper quartiles (25-th and 75-th percentiles) of the data, and the whiskers represent the lowest and the highest data points that are within 1.5 times the inter-quartile range from the lower and the upper quartiles, respectively. Circles represent outliers.



(a)



(b)

Figure 3.6.: Performances of the controllers obtained from the grid searches when implemented on $n = 100$ robots. In both plots, the horizontal axis represents the number of robots, n, used in the grid search leading to that controller. In (a), the vertical axis represents the dispersion of the robots after 1800 s (Equation 3.2, lower is better), and in (b), it represents the proportion of the robots in the largest cluster after 1800 s (Equation 3.3, higher is better). In both plots, each box represents values obtained from 100 simulations with different initial configurations of robots. In (a), the lower and the upper green lines show, respectively, the minimum possible dispersion for 100 robots, and their expected dispersion at the point of initialization.

initial dispersion. The controller obtained with $n = 9$ exhibits a broader range of performances across 100 runs with different initial configurations of the robots; however, on average (and in the absolute majority of the cases), the dispersion of the robots still increases from its original value. It is not surprising that the controllers obtained with $n = 2, 3, \ldots, 9$ all perform similarly, given the similarity in their structures as discussed in the previous section.

In stark contrast, the controller obtained with $n = 10$ robots scales extremely well on $n = 100$ robots. In each of the 100 runs, the 100 robots ended up in a single cluster after 1800 s. Moreover, at this point, their dispersion was, on average, only 8.92% larger than the theoretical lower bound (lower green line in Figure 3.6 (a)).

We now provide visual explanation of the above results. Figure 3.7 shows a sample configuration of 100 robots after 1800 s with each of the controllers obtained from the grid searches with $n = 2, 3, \ldots, 10$. For up to $n = 8$, the controllers lead to the robots forming nearly-perfect circular configurations. For $n = 9$. the configuration is not a perfect circle, but still exhibits a circular pattern. In these configurations, most of the spaces between each two robots is larger than the diameter of one robot, which explains the low cluster values in Figure 3.6 (b). On the other hand, for $n = 10$, the robots form a compact cluster.

It was visually ascertained that all these configurations were consistent across the 100 runs with different configurations of the robots: the controllers obtained with up to $n = 9$ always lead to circular configurations, and the controller obtained with $n = 10$ always leads to a compact cluster. Therefore, it is important to stress that the different emergent behaviors that can be seen in Figure 3.6 are the result of *different controllers* that were obtained by searching over the performance landscape with different numbers of robots. In other words, this is not a case of the same controller leading to different behaviors in a swarm—a phenomenon that has been observed in other systems [120].

Having observed the emergent behaviors that the controllers lead to, we can now explain why such badly-scaling controllers were returned by the grid searches for up to $n = 9$. Recall that the performance metric used in the grid search (Equation 3.4) is designed to reward both a low dispersion at the end of the time interval, as well as the speed with which the robots aggregate. Now, while the second moment of the robots in a compact cluster is always lower than that of a circular configuration, for small values of n, this difference is not too large. Therefore, a controller that achieves a circular configuration fast enough will receive a better performance metric than one that achieves a compact cluster, but slowly. As n increases, however, this “loophole” disappears, as the second moment of a circular configuration becomes much larger than the one of a compact cluster.

Note that this effect corresponds to the emergence of a new region of high performance in the performance landscapes as n increases, which was observed in Section 3.4. In order to reinforce our understanding of this effect, we ran one more grid search with

$n = 25$ robots (however, the computational resource only allowed for 10 evaluations per controller, rather than the usual 100). In the corresponding performance landscape (see Appendix B), the region of high performance that had been present for $n = 2$ has completely disappeared, and the one that had started to emerge as n increased has become the only high-performance region. The optimal controller in this case is given by $(\bar{v}_{\ell 0}, \bar{v}_{r0}, \bar{v}_{\ell 1}, \bar{v}_{r1}) = (-0.8, 1, 1, 1)$, which only differs from the controller obtained with $n = 10$ robots by 0.1 in the leftmost parameter.

We chose to perform the remainder of the studies in this chapter with the controller obtained from the grid search with $n = 10$ robots, as it is the first one that is scalable to larger numbers of robots.

3.6. Detailed Analysis of the Chosen Controller

In this section, we show how the optimal controller that was found by the grid search with $n = 10$ in Section 3.4 exploits the geometry and the physics that are brought about by the robots' embodiment, in order to bypass the need for computation. In Section 3.6.1, we prove that a robot that is executing the controller always aggregates with another, static robot, or a circular, static cluster of already-aggregated robots. We also calculate an upper bound on the aggregation time. In Section 3.6.2, we generalize the analysis of Section 3.6.1 to the case of two robots that are simultaneously executing the controller. We prove that these always aggregate, and derive equations that can be used to numerically calculate an upper bound on the aggregation time.

Recall that the controller is given by $(\bar{v}_{\ell 0}, \bar{v}_{r0}, \bar{v}_{\ell 1}, \bar{v}_{r1}) = (-0.7, 1, 1, 1)$. These parameters correspond to $R_0 \approx 14.45\,\text{cm}$, $\omega_0 \approx -0.75\,\text{rad/s}$; $R_1 = 0\,\text{cm}$, $\omega_1 \approx -5.02\,\text{rad/s}$. This means that when $I = 0$, the robot moves backwards along a circular trajectory in a clockwise fashion. When $I = 1$, it rotates clockwise on the spot, with the maximum possible angular velocity.

In the remainder of this section, we assume that the robots react instantaneously when I changes from 0 to 1 or vice versa (i.e. the sensing is continuous). As $R_1 = 0$ and will not feature in the analyses, we henceforth drop the "0" subscript from R_0, with R representing this quantity unless otherwise stated.

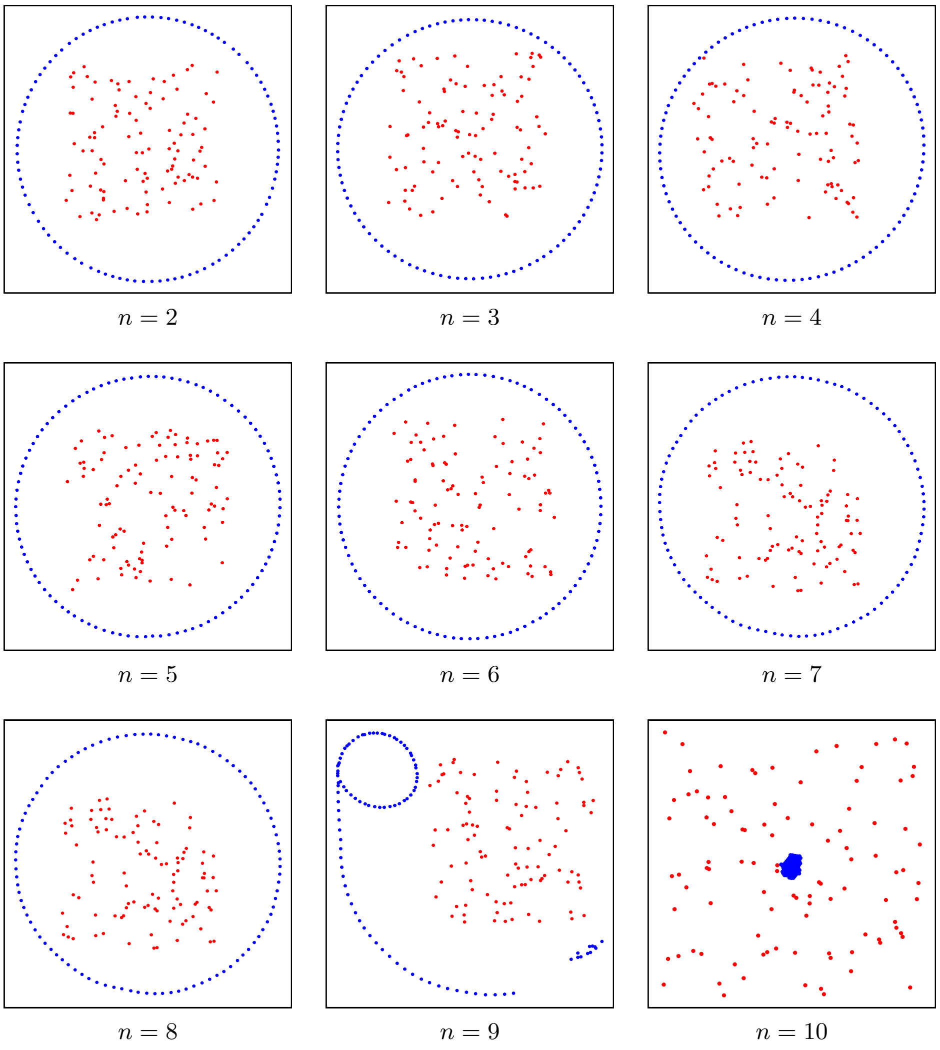


Figure 3.7.: The final configurations of 100 robots after 1800 s, resulting from the controllers obtained from the grid searches with $n = 2, 3, \ldots, 10$ robots. In each case, the configuration comes from the run that had the median dispersion after 1800 s across 100 simulations with different initial configurations of the robots.

3.6.1. One Moving Robot and One Static Robot or Circular Cluster

Let us analyze the situation where the environment contains one robot that is executing the controller, and one circular, static object. The latter can represent either a static robot, or a cluster of already-aggregated robots that is quasi-circular and quasi-static. Figure 3.8 shows the moving robot, i, of radius r_i, (small blue circle) and the static

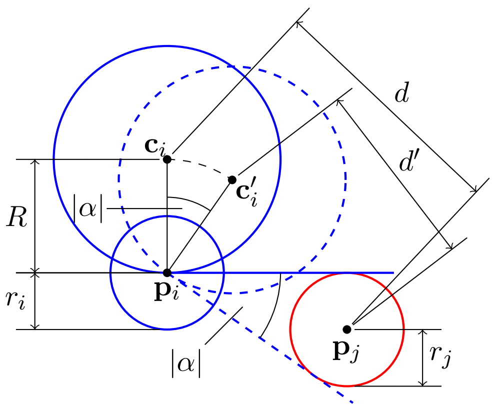


Figure 3.8.: Scenario with a moving robot i (small blue circle) and a static robot (or circular cluster) j (red circle). Robot i moves backwards along its circular trajectory centered at c_i (large solid blue circle), until it sees robot j (solid blue line). It then rotates clockwise on the spot through an angle of size $|\alpha|$, until it no longer sees robot j (dashed blue line). As it rotates, robot i's center of curvature moves from c_i to c'_i. Robot i then moves backwards along its new circular trajectory (dashed blue circle).

robot (or circular cluster), j, of radius r_j (red circle). Let us define $d = ||\mathbf{p}_j - \mathbf{c}_i||$. Note that the two robots cannot collide with each other as long as $d > R + r_i + r_j$, and we therefore consider them as being aggregated when this condition no longer holds.

Theorem 2. *One moving robot will always aggregate with another static robot or circular cluster.*

Proof. Consider the scenario shown in Figure 3.8. We want to obtain an expression for $d' = ||\mathbf{p}_j - \mathbf{c}'_i||$ in terms of $d = ||\mathbf{p}_j - \mathbf{c}_i||$. Without loss of generality, let $\mathbf{c}_i = [0, 0]^T$, and let the sensor of robot i point to the right along the horizontal at the moment when it starts seeing robot j. It follows that:

$$\mathbf{p}_j = \begin{bmatrix} \delta \\ -(R + r_j) \end{bmatrix}, \tag{3.6}$$

where δ is the length from the center of robot i to the point at which it touches the circumference of robot j, at the instant when robot i starts seeing robot j. δ is given by:

$$\delta = \sqrt{d^2 - (R + r_j)^2}. \tag{3.7}$$

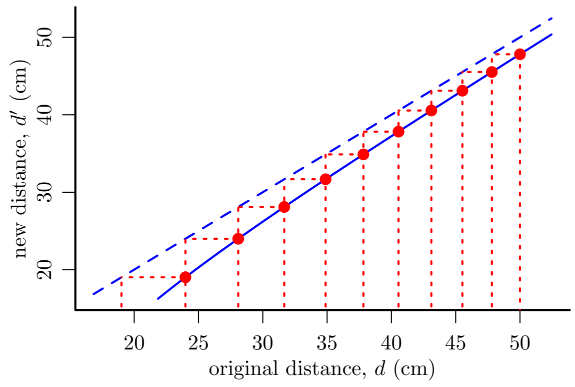


Figure 3.9.: Mapping from the original distance d to the new distance d' between the center of curvature of a moving robot, i, to a static robot, j. The red trajectory illustrates how, starting some value $d_0 > R + r_i + r_j$, d decreases stepwise until it becomes smaller than or equal to $R+r_i+r_j$. For the values used here, $R + r_i + r_j$ corresponds to 21.85 cm. The value of d_0 used in this figure is 50 cm.

From the geometry of Figure 3.8, the coordinates of $\mathbf{c}'_i$ are given by:

$$\mathbf{c}'_i = \begin{bmatrix} R\sin|\alpha| \\ R\,(\cos|\alpha| - 1) \end{bmatrix}, \tag{3.8}$$

where $|\alpha| \in (0, \pi)$. Moreover, we can write the equation $\tan(|\alpha|/2) = r_j/\delta$, from which we obtain[8]:

$$\sin|\alpha| = \frac{2\delta r_j}{\delta^2 + r_j^2}, \quad \cos|\alpha| = \frac{\delta^2 - r_j^2}{\delta^2 + r_j^2}. \tag{3.9}$$

Finally, by substituting Equations 3.6–3.9 into $d' = ||\mathbf{p}_j - \mathbf{c}'_i||$, and simplifying the resulting expression[9], we obtain:

$$d' = \sqrt{d^2 - 4Rr_j}, \qquad d > R + r_i + r_j. \tag{3.10}$$

Figure 3.9 shows the mapping from d onto d' according to Equation 3.10, for the values of R and r_j that are used here. The red trajectory illustrates how, starting from some value $d_0 > R + r_i + r_j$, d decreases stepwise until it becomes smaller than or equal to

[8]These relationships can be derived by using standard trigonometric identities.
[9]This simplification is not straightforward to work out by hand. We obtained it using the mathematical software Sage (http://www.sagemath.org/).

$R + r_i + r_j$, at which point the robots are considered to be aggregated. □

Theorem 3. *The time, in s, that it takes for a moving robot to aggregate with a static robot or circular cluster of radius r_j is bounded by $2\pi m/|\omega_0|\,\mathrm{s}$, where*

$$m = \left\lceil \frac{d_0^2 - (R + r_i + r_j)^2}{4Rr_j} \right\rceil, \tag{3.11}$$

and d_0 is the initial value of $||\mathbf{p}_j - \mathbf{c}_i||$ (see Figure 3.8).

Proof. We begin by calculating the number of $d \to d'$ updates that must occur until $d \leq R + r_i + r_j$. Let us write Equation 3.10 as a recurrence relation:

$$d_{m+1} = \sqrt{d_m^2 - 4Rr_j}, \qquad d_m > R + r_i + r_j. \tag{3.12}$$

Now, let $\hat{d}_m = d_m^2$, such that we can re-write Equation 3.12 as $\hat{d}_{m+1} = \hat{d}_m - 4Rr_j$, the solution to which is given by $\hat{d}_m = \hat{d}_0 - 4Rr_j m$. Therefore, the solution to Equation 3.12 is given by:

$$d_m = \sqrt{d_0^2 - 4Rr_j m}, \qquad d_m > R + r_i + r_j. \tag{3.13}$$

The number of $d \to d'$ that are required until $d \leq R + r_i + r_j$ is now given by setting $d_m = R + r_i + r_j$ in Equation 3.13, solving for m, and taking the ceiling of this value, which leads to:

$$m = \left\lceil \frac{d_0^2 - (R + r_i + r_j)^2}{4Rr_j} \right\rceil. \tag{3.14}$$

Given m, we can now calculate an upper bound on the time that it takes until $d \leq R + r_i + r_j$. From the time that robot i stops seeing robot j, it takes robot i slightly less than one cycle along its circular trajectory until it once again starts seeing robot j. The time that it takes for robot i to rotate on the spot while it sees robot j is negligible, because $|\omega_1|$ is much larger than $|\omega_0|$. At most one $d \to d'$ update occurs per cycle of robot i along its circular trajectory. The aggregation time is thus bounded by $2\pi m/|\omega_0|\,\mathrm{s}$. □

3.6.2. Two Simultaneously Moving Robots

We now analyze the situation with two robots that are simultaneously executing the controller. In this scenario, it is sensible to define $d = ||\mathbf{c}_j - \mathbf{c}_i||$. Note that the two

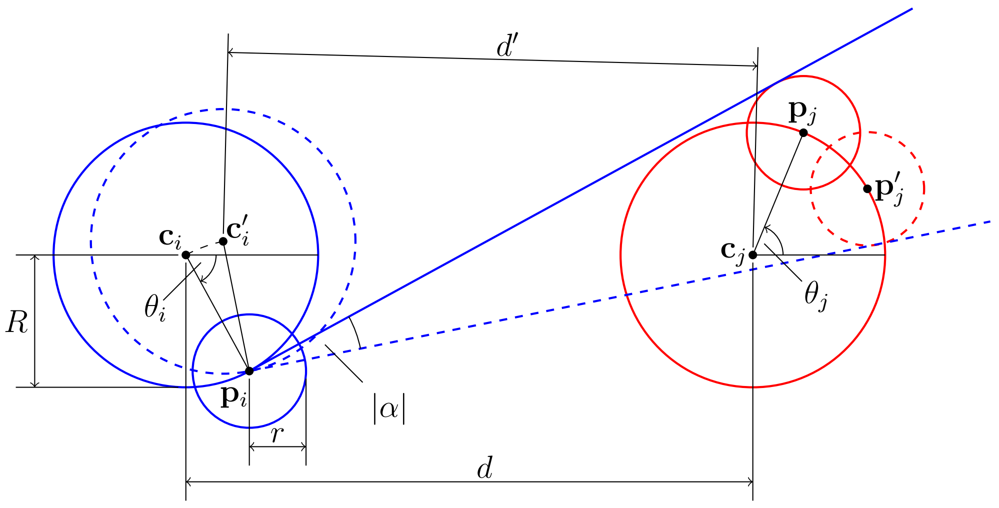


Figure 3.10.: Scenario with two simultaneously moving robots, i and j (small blue and red circles, respectively). Robot i moves backwards along its circular trajectory centered at c_i (large solid blue circle), until it sees robot j (solid blue line). Robot i then rotates clockwise on the spot through an angle of size $|\alpha|$, while robot j moves backwards along its circular trajectory. As it rotates, i's center of curvature moves from c_i to c'_i. Robot i stops rotating when it stops seeing robot j (dashed blue line), and it then moves backwards along its new circular trajectory (dashed blue circle).

robots cannot collide with each other as long as $d > 2(R+r)$, and we therefore consider them as being aggregated when this condition no longer holds.

Theorem 4. *Two simultaneously moving robots will always aggregate, and an upper bound on the time that this takes is given by solving Equations 3.15 – 3.18.*

Proof. Consider the arrangement shown in Figure 3.10. Without loss of generality, let $\mathbf{c}_i = [0,0]^T$, and $\mathbf{c}_j = [d,0]^T$. As in the case of the previous section, we want to calculate $d' = ||\mathbf{c}_j - \mathbf{c}'_i||$; d' now depends not only on d, but also on the position of robot j along its circular trajectory at the instant when robot i starts seeing it. The latter can be parametrized by the angle that robot j makes with the horizontal[10], $\theta_i \in \left(-\frac{\pi}{2}, 0\right)$. From the geometry of Figure 3.10:

$$\mathbf{c}'_i = \mathbf{p}_i + R \begin{bmatrix} \cos(\pi + \theta_i - |\alpha|) \\ \sin(\pi + \theta_i - |\alpha|) \end{bmatrix}, \tag{3.15}$$

[10]We use the convention of counter clockwise angles being positive, and clockwise angles being negative.

where $\mathbf{p}_i = \mathbf{c}_i + R\left[\cos\theta_i, \sin\theta_i\right]^T$. We now need expressions for θ_i and $|\alpha|$ in terms of d and θ_j.

The expression for θ_i can be obtained from the geometry of Figure 3.10 as:

$$\theta_i = \arcsin\left(\frac{R\sin\theta_j}{||\mathbf{p}_j - \mathbf{c}_i||}\right) + \arcsin\left(\frac{R+r}{||\mathbf{p}_j - \mathbf{c}_i||}\right) - \frac{\pi}{2}, \tag{3.16}$$

where $\mathbf{p}_j = \mathbf{c}_j + R\left[\cos\theta_j, \sin\theta_j\right]^T$.

We now turn to calculating $|\alpha|$. In Figure 3.10, robot j moves along its circular trajectory while robot i rotates on the spot[11]. Let t_{rot} denote the time that robot i spends rotating on the spot while seeing robot j, from which it follows that $|\alpha| = |\omega_1| t_{\text{rot}}$. Now, we can express $\mathbf{p}'_j$ in terms of t_{rot}, as $\mathbf{p}'_j = \mathbf{c}_j + R\left[\cos\left(\theta_j - |\omega_0| t_{\text{rot}}\right), \sin\left(\theta_j - |\omega_0| t_{\text{rot}}\right)\right]^T$. Using this, in turn, we can express $|\alpha|$ in terms of d, θ_j, and t_{rot} by noting from the geometry of Figure 3.10 that:

$$\begin{aligned} |\alpha|\left(d, \theta_j, t_{\text{rot}}\right) = {} & \angle\left(\mathbf{p}_j - \mathbf{p}_i\right) - \angle\left(\mathbf{p}'_j - \mathbf{p}_i\right) \\ & + \arcsin\left(\frac{r}{||\mathbf{p}_j - \mathbf{p}_i||}\right) \\ & + \arcsin\left(\frac{r}{||\mathbf{p}'_j - \mathbf{p}_i||}\right), \end{aligned} \tag{3.17}$$

where $\angle\left(\cdot\right)$ denotes the angle that the vector makes with the horizontal axis. The value of t_{rot}, and hence that of $|\alpha|$, is given when:

$$|\alpha|\left(d, \theta_j, t_{\text{rot}}\right) = |\omega_1| t_{\text{rot}}. \tag{3.18}$$

We are now in possession of expressions for θ_i in terms of d and θ_j (Equation 3.16) and an equation from which we can calculate $|\alpha|$ numerically given these quantities (Equation 3.18). By substituting the values of θ_i and $|\alpha|$ in Equation 3.15, we obtain $\mathbf{c}'_i$, and from this, in turn, we obtain $d' = ||\mathbf{c}'_i - \mathbf{c}_j||$. Figure 3.11 shows a plot of $d' - d$ against θ_j for one $d \to d'$ update. The lowermost curve corresponds to $d = 2\left(R + r\right)$, and subsequent curves correspond to d increasing in steps of 10 cm. As one expects, as

[11] It should be noted that for a small range of θ_j, robot j will simultaneously be seeing robot i, and will therefore be rotating clockwise on the spot. This range, however, is very small, and its effect can therefore be neglected by treating robot j as if it were still moving along its circular trajectory. This assumption does not violate the worst-case scenario.

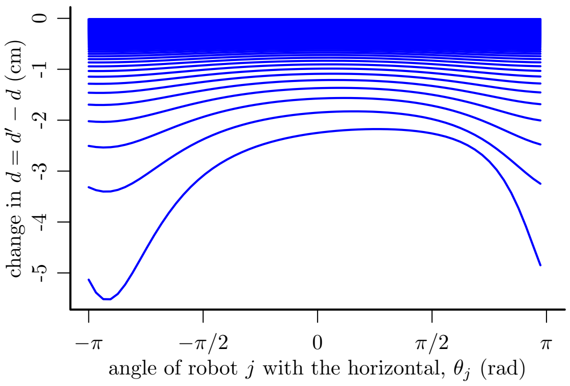


Figure 3.11.: $d' - d$ against θ_j for one $d \to d'$ update in the case of two simultaneously moving robots. The lowermost curve corresponds to $d = 2\,(R + r)$, and subsequent curves correspond to d increasing in steps of 10 cm.

d increases, the curves become flatter (meaning that the effect of θ_j on d' diminishes), and $d' - d$ tends towards zero.

We can now calculate, starting from some initial $d = d_0 > 2\,(R + r)$, an upper bound on the number m of $d \to d'$ updates that need to occur until $d \leq 2\,(R + r)$. We do this by assuming the worst case scenario—that in each update, $d' - d$ assumes the closest value to zero that is possible (these values correspond to the maxima of the curves in Figure 3.11). As we do not have a closed-form expression for $|\alpha|$, we find m by numerically iterating from d onto d' using Equations 3.15–3.18, and counting the number of iterations that are required until $d' \leq 2\,(R + r)$. Once we have m, we obtain the time that elapses until $d \leq 2\,(R + r)$, as we did for the case of one moving robot in Section 3.6.1. In this case, at most two $d \to d'$ updates can occur per $2\pi/|\omega_0|$ s, because the two robots are moving simultaneously. Therefore, the aggregation time is bounded by $2\pi m/2|\omega_0| = \pi m/|\omega_0|$ s.

Figure 3.12 shows a plot of the upper bound aggregation time, starting from an initial value $d = d_0$ (red curve). Overlain on the plot are 1000 data points obtained from simulations with two simultaneously moving robots (blue crosses). By re-plotting Figure 3.12 using log-scales for both axes, it turns out that the relationship between log (time) and log (d_0) is sublinear for small values of d_0, and becomes asymptotically linear as d_0 increases. A least squares regression fit reveals that the slope of the linear part is almost exactly 2, which means that the relationship between d_0 and the time required until $d \leq 2\,(R + r)$ is asymptotically quadratic (compare this to the quadratic relationship in the case of one moving robot in Section 3.6.1, given by Equation 3.14). $\square$

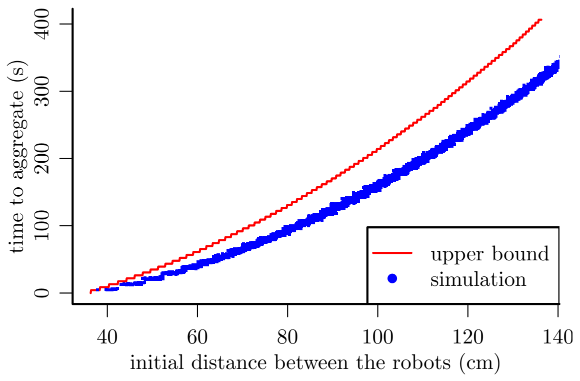


Figure 3.12.: The time required for two simultaneously moving robots to aggregate.

3.6.3. More Than Two Robots

The scenario with n robots does not appear be amenable to a mathematical analysis in the manner of the preceding two sections, because in this case, aggregation results from complex collective movements of clusters of physically-interacting robots, as can be seen in the videos on the online supplementary material [188]. It may be possible to approach a proof that ignores the robots' embodiment using dynamical systems theory. However, while this would be an interesting exercise, its practical value would be somewhat limited, because as we have argued before, one of the main features of our solution is that it is able to function in the presence of inter-robot collisions. This attempt shall therefore be deferred to future work.

3.7. Validation on Physical Robots

While the literature on the self-organized robot aggregation problem is considerably vast, only a small subset of works have demonstrated their solutions on a physical robotic system. The (related) deterministic controllers reported in [166, 162, 165] were implemented on ve SAETTA robots, but the sensing was performed in a centralized fashion, with an external camera being used to measure the distances between the robots. The probabilistic solution proposed in [152] was evaluated on 20 Alice robots, this time in a truly decentralized manner. To our knowledge, no solution that was obtained using an evolutionary robotics methodology has been demonstrated on physical robots.

In this section, we will start by discussing how the sensor/controller solution described in the previous sections was implemented on the e-puck robotic platform. We will then discuss the setup of and the results from a systematic experiment involving 30 trials with 40 e-pucks, and finish by presenting some visual observations from these experiments.

3.7.1. Sensor and Controller Implementation

The sensor was implemented using the e-puck's directional camera. The robots were fitted with black 'skirts' in order to make them distinguishable against the white walls of the arena (see Figure 3.2). The e-puck's camera is a CMOS RGB color camera with a resolution of 640 (horizontal) by 480 (vertical) pixels, with corresponding viewing angles of 56° and 42°, respectively.

In principle, using one pixel of the camera is sufficient for implementing the binary line-of-sight sensor. However, in order to account for vertical misalignments between the orientations of the robots' cameras, we used the middle column of pixels from the camera to implement the sensor. This column of pixels is sub-sampled in order to obtain 15 equally-spaced pixels from its bottom to its top, resulting in an area of width 1 pixel and height 15 pixels. The gray values of these 15 pixels are compared against a threshold, which was empirically set to 2/3 of the maximum possible value (pure white). If one or more of the pixels have a gray value below this threshold, the sensor gives a reading of $I = 1$. Otherwise it gives a reading of $I = 0$. The implemented sensor has been found to provide reliable readings for up to a range of around 150 cm.

The controller was implemented exactly as in simulation, that is, as a single *if-then-else* statement, with the optimal parameters found from the grid search. The complete, commented source code showing the sensor and controller implementation is given in Appendix C.

3.7.2. Experimental Setup and Procedure

The arena used for the experiments was a rectangle of size 400 cm × 225 cm. It had a light gray floor, and was surrounded by white walls 50 cm in height. Its floor was marked with a grid of $15 \times 8 = 120$ points, spaced 25 cm from each other and from the walls. For each trial, 40 of these points were chosen randomly to serve as the initial positions of the robots. With the robots positioned on these points, an infra-red signal was issued to instruct each robot to turn on the spot through a randomly generated portion of a revolution, such that robots now faced in random directions. Another infra-red signal was then issued to instruct the robots to start executing the controller. The robots were programmed to stop automatically after 900 s. If a robot stopped moving during a trial, the start signal was reissued in an attempt to restart it (moving robots were unaffected by this signal). If the robot failed to restart, it was left in the arena until the end of the trial. We performed 30 trials with $n = 40$ robots. Each trial was recorded by an overhead camera. All the 30 videos are available in the online supplementary material [188]

3.7.3. Results

Figure 3.13 shows a series of snapshots taken at different times from a single trial. After around 180 s, the robots had formed two clusters, and these clusters eventually approached each other, made contact after around 360 s, and amalgamated into one ‘body’ after around 540 s. The dynamics of this trial are representative of those of most of the other trials. Figure 3.15 shows a plot of the performance metrics over time. For $n = 40$ robots, the lower bound of $u^{(t)}$, as reported in [182], is 220.33. The final value of $u^{(t)}$ (i.e. after 900 s) was 49.08% larger than this theoretical lower bound (averaged across the 30 trials).

Figure 3.14 shows the final configurations of the robots in all the 30 trials. In 27 of the trials, all the 40 robots were aggregated into one cluster within 900 s. In 2 of the trials, the largest cluster contained 39 robots. In one trial, the robots were divided into two clusters of 25 and 15 robots after 900 s[12]. Therefore, on average, across the 30 trials, the proportion of robots in the largest cluster after 900 s was equal to 98.6%.

3.7.4. Observations

We observed that, after starting to execute the controller, most of the robots quickly form small clusters. These clusters then move collectively, and merge into one another. The collective movement of a cluster occurs by an effect that is similar to convection. Robots that are on the periphery of the cluster, and which do not see other clusters as they rotate on the spot, often displace themselves, and end up on another part of the

[12]After the trial was completed, the robots were restarted, and the two clusters eventually merged into one after around an additional 180 s.

cluster's periphery, from which other clusters can be seen. As this happens repeatedly, the cluster moves as a whole towards other clusters. This can be seen in the videos in the online supplementary material [188]. Note that this behavior seems to be rather inefficient in terms of the speed of the aggregation process.

We therefore conjecture that the severe limitations imposed on the amount of sensing and information processing lead to a performance trade-off as compared to deterministic algorithms that make use of more sensory information and/or computation. This will be investigated in the next chapter.

3.8. Summary

This chapter has presented the simplest solution so far to the problem of self-organized robot aggregation in a homogeneous environment. In this solution, each robot has only one sensor that provides it with one bit of information about its environment: whether or not there is another robot in its line of sight. As the sensor is binary and the robots are memoryless, computation by the controller is futile, and any controller can be reduced to a mapping of each of the two possible sensor readings onto a set of pre-defined actuation parameters.

The optimal controller was found by a grid search over the entire space of possible controllers, using 10 robots that are initialized uniformly randomly within a square region. Proofs of aggregation, as well as time upper bounds, have been provided for the cases of one moving robot and one static robot or circular cluster, and two simultaneously moving robots. Simulations have shown that the robots can form very dense packings (on average, only 8.92% worse than the optimum) when employing the sensor/controller solution. Such dense packings are desirable, for example, if after aggregating, the robots are required to perform tasks as a physically connected robotic ensemble [189, 15]. The solution was ported onto a system of 40 physical e-puck robots. On average, across 30 experiments, 98.6% of the robots aggregated into one cluster within 900 s. From these experiments, we observed that the aggregation depends on the complex collective movement of clusters of physically-interactive robots, which happens by an effect that is similar to convection.

We believe that the solution provided here is not restricted to the e-puck robotic platform on which it has been validated so far. Rather, it is employable on any system of robots that have the abilities to (i) tell whether there is another robot in their line of sight, and (ii) move along circular trajectories of different curvatures. We anticipate that the ease of porting the synthesized controller onto a system of physical robots that was demonstrated here will also carry forward to other robotic platforms, because of the extreme simplicity of the sensor and the controller. However, we suspect that the shape and the material properties of the robots will have an impact on the aggregation performance and the structures formed (see, e.g., [190]), which is yet to be investigated.

A current limitation of this work is that the robots are unable to stop searching for other clusters, even when none are present. This can lead to a continual energy consumption. However, the robots could be instructed to sleep when they have been rotating on the spot for some time, and wake up periodically in order to check whether this is still the case. In this way, robots that are inside a cluster would consume less energy.

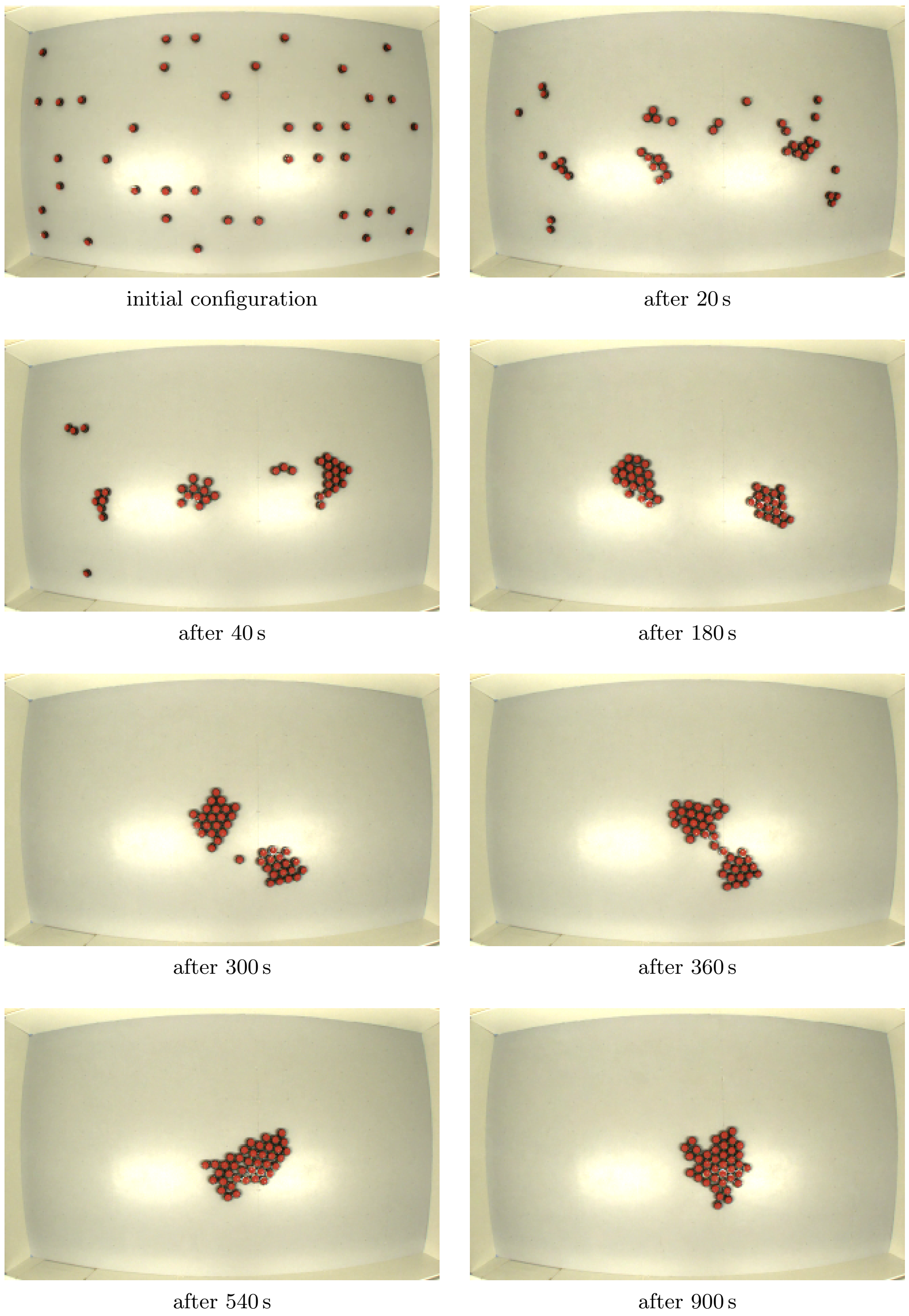


Figure 3.13.: A sequence of snapshots taken during one of the experiments with 40 physical e-puck robots. The dimensions of the arena are 400 cm × 225 cm. The bright patches that can be seen in the arena are reflections from the overhead lighting, and did not affect the behavior of the robots.

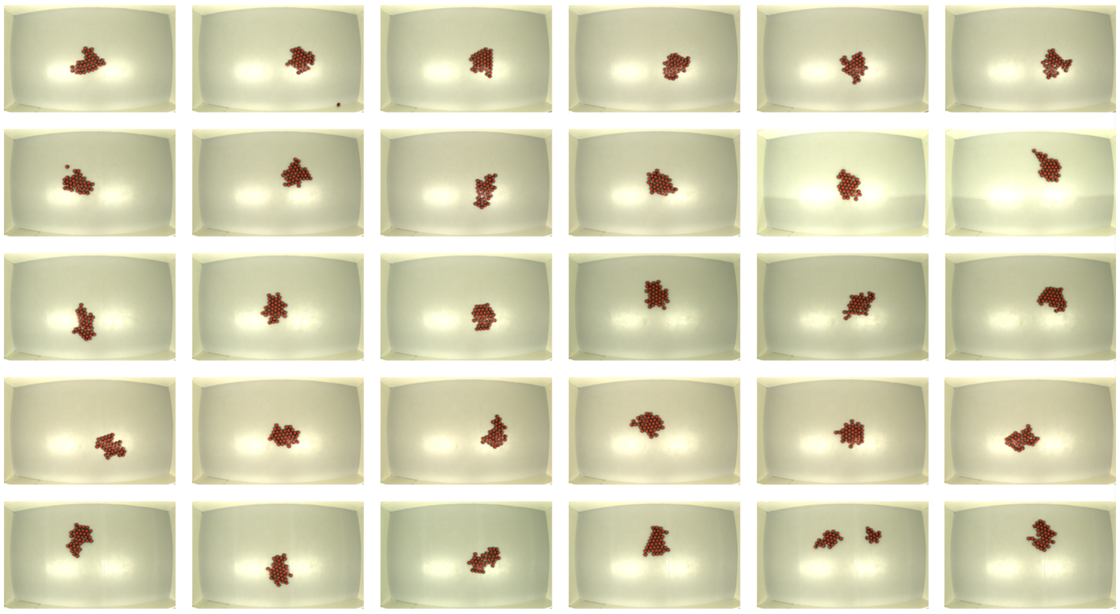
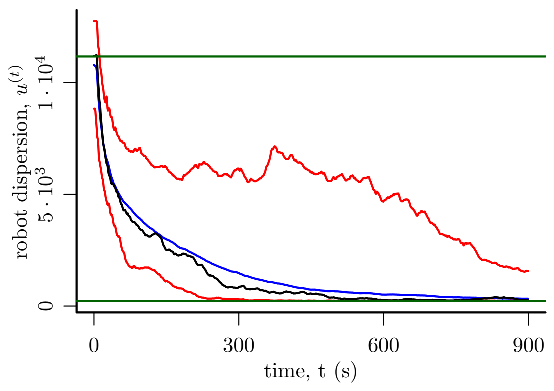
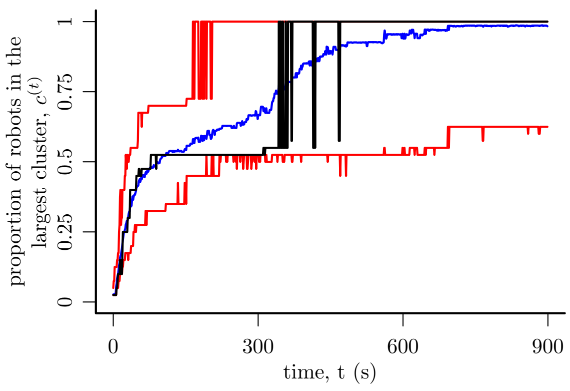


Figure 3.14.: Overhead snapshots of the final configurations of the 40 robots in the 30 experiments. The dimensions of the arena are 400 cm × 225 cm. The bright patches that can be seen in the arena are reflections from the overhead lighting, and did not affect the behavior of the robots.



(a)



(b)

Figure 3.15.: These plots show the aggregation dynamics with $n = 40$ physical e-puck robots. In both plots, the horizontal axis represents the time, t. In (a), the vertical axis represents the dispersion of the robots (Equation 3.2, lower is better), and in (b), it represents the proportion of the robots in the largest cluster (Equation 3.3, higher is better). The blue curves show the mean measure across the 30 trials, whereas the red curves show the range (minimum, maximum) of the measure across the 30 trials. The black curves show the measure in the trial shown in Figure 3.13. In (a), the lower and the upper green lines show, respectively, the minimum possible dispersion for 40 robots, and their expected dispersion at the point of initialization.

4. Systematic Investigations of Aggregation

The previous chapter introduced a swarm-robotic system in which self-organized aggregation was effectively achieved despite the robots being only equipped one binary, line-of-sight sensor, and employing a memoryless, computation-free controller. In synthesizing controllers for the robots, and analyzing these controllers, idealized conditions were assumed on the part of the sensor; namely, that it provided noise-free readings, and could detect other robots within an unlimited range. An implementation of the sensor/controller solution on a system of 40 physical e-puck robots was subsequently used to demonstrate that the solution is still effective under less ideal, 'real-world' (yet laboratory) conditions.

Nevertheless, in the previous chapter, we did not ask at what performance price the extreme sensor/controller simplicity comes. Nor did we *systematically* investigate the scalability of the solution with respect to the number of robots, and the effects of sensory non-idealities, such as noise and range limitations. Moreover, we only investigated one 'standard', minimalist system, without considering whether some additional complexity in the sensor and/or controller structures could benefit the performance of the system, and should therefore be considered in the design process, space and energy constraints allowing.

This chapter addresses the above questions by presenting systematic investigations into various aspects of the system introduced in Chapter 3. In Section 4.2, we compare the performance of our minimalistic solution with that of a sensor/controller combination that makes use of global knowledge of the environment. This includes a comparison of the aggregation dynamics (i.e. over time) of both solutions (4.1.1), and an analysis of how their performance scales up with increasing numbers of robots (4.1.2). Section 4.2 discusses the effects of sensor non-idealities that are likely to arise in practical implementations, namely the presence of noise (4.2.1) and the limitations in range (4.2.2). Section 4.3 presents an analysis of a sensor variation. Instead of being a line of sight, the sensor is now modified to have a finite field of view. We discuss how new controllers were synthesized (4.3.1) and post-evaluated (4.3.2) for various fields of view, and discuss the effects observed (4.3.3). Section 4.4 discusses a variation on the controller. Memory is introduced in the form of a neural network with recurrent connections (4.4.1). We discuss how new controllers were synthesized (4.4.3, 4.4.2) and post-evaluated (4.4.4) with and without memory, and discuss the effects observed (4.4.5).

A Note on the Simulations

Throughout this chapter, unless otherwise stated, the simulation platform is identical to the one described in Section 3.2.4. Moreover, each simulation with n robots is assumed to run for $T = 180n$ time steps, corresponding to $18n\,\mathrm{s}$. At the start of a simulation, the robots are initialized with a uniform random distribution in a (virtual) square such that the area per robot is, on average, $10000\,\mathrm{cm}^2$ (i.e. the square has sides of length $(10000n)^{\frac{1}{2}}$ cm).

4.1. Performance Trade-Off with Line-Of-Sight Sensors

In the previous chapter, we observed in the physical experiments that the aggregation process, while effective, was rather “inefficient”, as aggregation results from the collective movements of pre-formed clusters of robots. We therefore hypothesized that the minimalistic sensor/controller solution comes at a performance cost (more specifically, speed) in comparison with solutions that assume more information on the part of the robots. In this section, we will investigate this hypothesis, and quantify how much performance we are trading off when opting for a minimalistic solution.

Towards this end, we will formulate a sensor/controller solution that is on the other end of the spectrum as our minimalistic one. This solution assumes complete information; that is, each robot is assumed to know the relative positions of *all* the other robots in the swarm. Then, each robot can compute the centroid of these positions in its local coordinate frame (but this centroid will be common for all the robots in the global coordinate frame), and move towards it. A visual representation of this solution is shown in Figure 4.1.

Because our robots are differential wheeled, they cannot, in general, move instantaneously in an arbitrary direction. Rather, they can only move instantaneously along

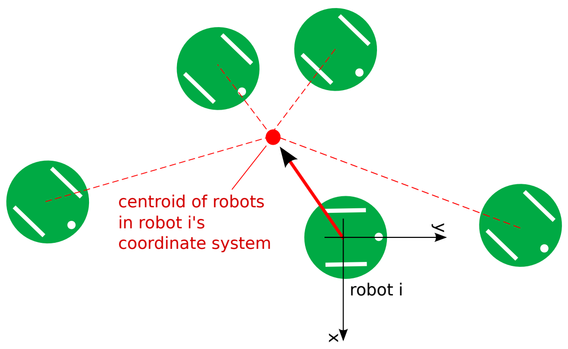


Figure 4.1.: The behavior of the global information controller. Each robot knows the relative positions of all the other robots in the environment, and moves towards the centroid of these positions.

their axis of orientation. For this reason, we require a method to translate the desired direction of motion onto an instantaneous motion, that is, wheel speeds. Groß et al. [123] provided one such way in a work in which they also used e-puck robots, and we therefore chose this method in the knowledge that it has been successfully applied to other behaviors.

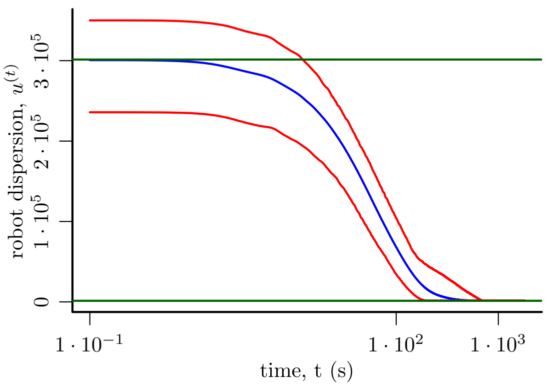
Let $\mathbf{p}_{cx}$ and $\mathbf{p}_{cy}$ represent the x and y components of the vector pointing towards the centroid in robot i's coordinate system, as shown in Figure 4.1. Then, the normalized velocities of the left and right wheels of the robots are computed as:

$$\begin{pmatrix} \bar{v}_\ell \\ \bar{v}_r \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{p}_{cx} \\ \mathbf{p}_{cy} \end{pmatrix}. \tag{4.1}$$

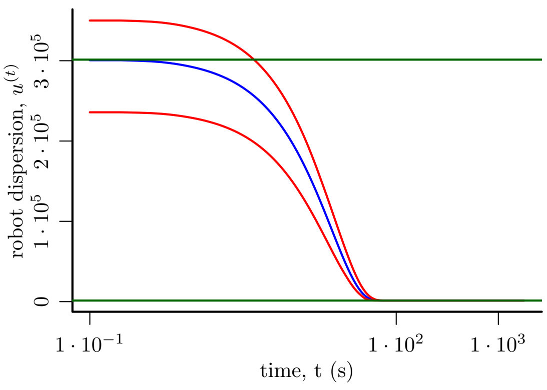
This completes our formulation of the global information control law for each robot. In the next sections, we will first look at the dispersion metric over time in simulations with $n = 100$ robots employing both controllers, and then at the scalability of these controllers to up to $n = 1000$ robots.

4.1.1. Dynamics

We performed 100 simulations with $n = 100$ robots with both the minimalistic and the global information controllers. Figures 4.2a and 4.2b show how the dispersion of the robots varies with time with the minimalistic and the global information solutions, respectively. With the former, on average (blue curve), the dynamics reach steady state somewhere between $t = 100$ and $1000\,\text{s}$, with the worst case being just before $t1000\,\text{s}$.



(a)



(b)

Figure 4.2.: These plots show the cluster dynamics with $n = 100$ robots with (a) the minimalistic and (b) the global information solutions. In both plots, the horizontal axis represents the time, t, and the vertical axis represents the proportion of the robots in the largest cluster (Equation 3.3, higher is better). The blue curves show the mean measure across 100 simulations with different initial configurations, whereas the red curves show the range (minimum, maximum) of the measure across the 100 simulations.

With the global information solution, on the other hand, the dynamics reach steady state before the $t = 100\,\mathrm{s}$, for all cases.

With regards to the steady-state performance, the minimum dispersion achievable for $n = 100$ robots is 1375.40 [182]. The minimalistic and the global information solutions achieve values that are 8.16% and 0.85% worse than the optimum, respectively. Figure 4.3 shows the configuration with the minimum possible dispersion for 100 robots, and final configurations with the minimalistic and the global information solutions (the trials chosen for this figure were the ones that led to the median robot dispersion in the

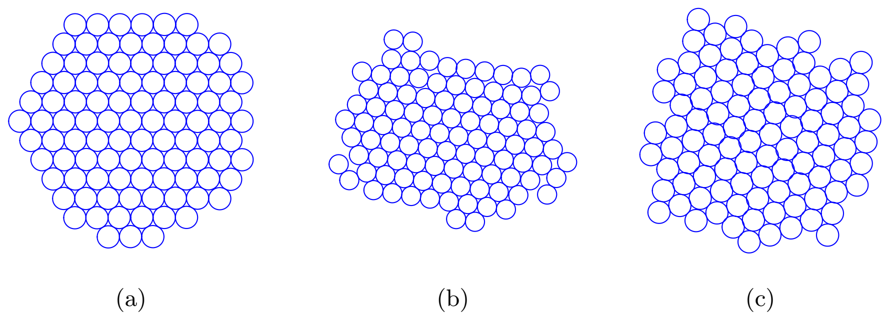


Figure 4.3.: (a): The configuration of 100 robots with the minimum possible dispersion (Equation 3.2). (b): The final configuration of 100 robots after 1800 s with the minimalistic solution. (c): The final configuration of 100 robots after 1800 s with the global information solution. In (b) and (c), the configurations come from the runs that had the median dispersion after 1800 s across 100 simulations with different initial configurations.

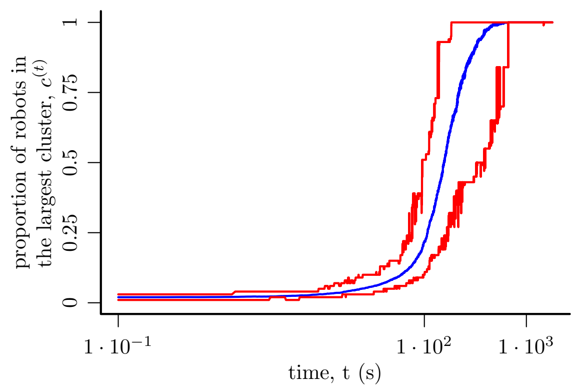
final configuration, as found from the 100 simulations).

Figures 4.4a and 4.4b show how the size of the largest cluster varies with time with the minimalistic and the global information solutions, respectively. With both solutions, all the runs end up in a single cluster. With the minimalistic solution, on average (blue curve), a single cluster is first achieved somewhere between $t = 100$ and 1000 s. On the other hand, with the global information solution, all the runs have converged to a single cluster formation before $t = 100$ s.

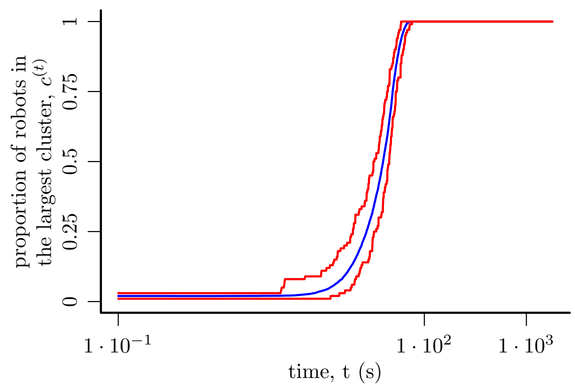
4.1.2. Scalability

We will now investigate the scalability of the two solutions with respect to increasing numbers of robots. For each controller, we performed 100 simulations for each value of $n \in \{100, 200, \ldots, 1000\}$ robots. In each simulation, the maximum cluster size was tracked at every time step, and the simulations were set to stop when the robots first formed a single cluster, recording the time that had elapsed until this point. Every run terminated, meaning that both solutions are capable of achieving consistent, error-free aggregation with at least 1000 robots.

Figure 4.5 shows a box plot of the times taken to achieve a single cluster by the minimalistic and the global information solutions. We performed least squares regression fits



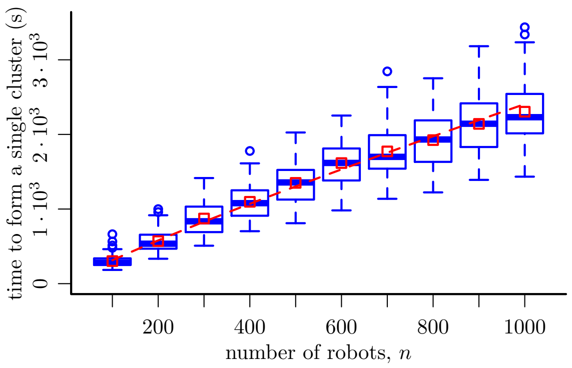
(a) minimalistic solution



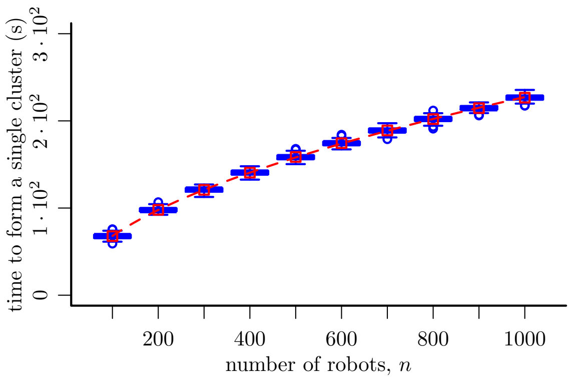
(b) global information solution

Figure 4.4.: These plots show the dispersion dynamics with $n = 100$ robots with (a) the minimalistic and (b) the global information solutions. In both plots, the horizontal axis represents the time, t, and the vertical axis represents the dispersion of the robots (Equation 3.2, lower is better). The blue curves show the mean measure across 100 simulations with different initial configurations, whereas the red curves show the range (minimum, maximum) of the measure across the 100 simulations. In (a), the lower and the upper green lines show, respectively, the minimum possible dispersion for 100 robots, and their expected dispersion at the point of initialization.

of the form An^k (dashed red line in Figure 4.5) to the means of the times (red squares in Figure 4.5) for both controllers. The binary solution yields $A = 5.34$, $k = 0.88$, while the global information solution results in $A = 6.08$, $k = 0.52$. Both solutions therefore scale sub-linearly in the time taken to achieve a single cluster. For $n = 1000$ robots, the global information solution is faster than its counterpart by a factor of around 10.



(a) minimalistic solution



(b) global information solution

Figure 4.5.: These box plots show the time taken by (a) the minimalistic and (b) the global information solutions to aggregate increasing numbers of robots into a single cluster. Each box represents values obtained from 100 simulations with different initial configurations of robots. The red points show the mean values, and the dashed red line shows a least squares regression fit to these points of the form An^k.

4.2. The Effects of Sensor Limitations

In this section, we will investigate the ramifications of real-world sensor limitations with the minimalistic solution. In Section 4.2.1, we will look into the effects on the aggregation performance when the robots' sensory readings are corrupted by noise. In Section 4.2.2, we will analyze the effects of limitations in the robots' sensing ranges.

4.2.1. Sensor Noise

We investigate how the aggregation performance is affected when the readings of the robots' binary line of sight sensors are corrupted by noise. We considered two types of noise on the sensor. False negative noise flips an input of $I = 1$ to $I = 0$ (i.e. a perceived robot is missed) with probability p_n. False positive noise flips an input of $I = 0$ to $I = 1$ (i.e. an imaginary robot is perceived) with probability p_p. We considered a grid of $11 \times 11 = 121$ noise level combinations, with p_n and p_p taking the values $\{0.0, 0.1, \ldots, 1.0\}$. For each combination, we performed 100 simulations, and the mean dispersion of the robots and the proportion of the robots in the largest cluster after 1800 s were recorded.

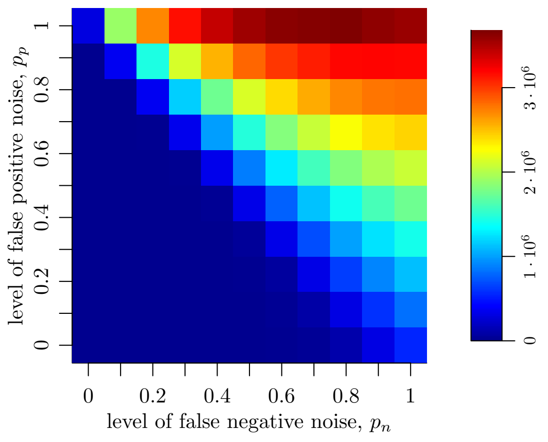
Figure 4.6 shows the noise performance landscapes as color maps. Note that for noise level combinations that lie on the diagonal defined by $p_n + p_p = 1.0$, the reading of the sensor is 0 with probability p_n, and 1 with probability p_p, irrespective of whether or not there is a robot in the direct line of sight. On this diagonal, the sensor provides no meaningful information. In the region $p_n + p_p > 1.0$, the sensor reading could be inverted so as to obtain a performance corresponding to the noise combination that mirrors the actual one in the diagonal $p_n + p_p = 1.0$.

The controller leads to a low dispersion and a high proportion of robots in the largest cluster for a sizeable sub-region of noise level combinations—approximately, for up to $p_n + p_p < 0.6$. The controller is thus highly robust to both types of noise on the sensor.

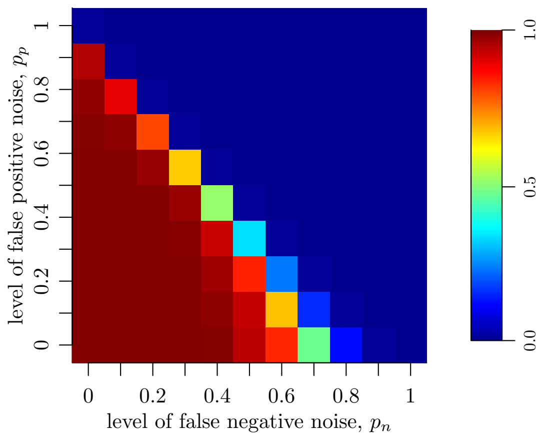
4.2.2. Sensor Range

In this section, we study how the aggregation performance is affected when the sensing range of the robots (assumed to be unlimited in the other sections) is finite. As the robots are initialized within a virtual square of sides 1000 cm, a sensing range of $\delta_\infty = \left(1000^2 + 1000^2\right)^{\frac{1}{2}} = 1414$ cm can be considered as being virtually unlimited. We performed 100 simulations for each $\delta/\delta_\infty = \{0.0, 0.1, \ldots, 1.0\}$. In each simulation, the dispersion of the robots and the proportion of robots in the largest cluster after 1800 s were recorded.

Figure 4.7 shows a box plot with the results. The performance is virtually unaffected as δ/δ_∞ is reduced from 1.0 to 0.3. As δ/δ_∞ is reduced to 0.2, the median performance



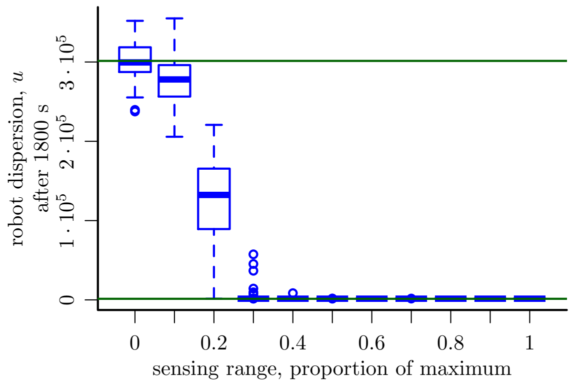
(a)



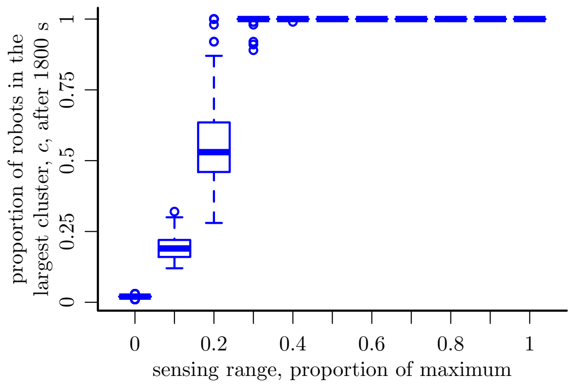
(b)

Figure 4.6.: These plots show the effect of sensory noise on the aggregation performance with $n = 100$ robots. In both plots, the horizontal and the vertical axes show, respectively, the level of false negative and false positive noise (see the text for details). In (a), the colors represent the dispersion of the robots after 1800 s (Equation 3.2, lower is better), and in (b), they represent the proportion of the robots in the largest cluster after 1800 s (Equation 3.3, higher is better). In both plots, the color in each box represents the mean value across 100 simulations with different initial configurations.

drops instantly, almost to the expected value at the point of initialization—equivalent to robots that do not move throughout the simulation (upper green line in Figure 4.7a).



(a)



(b)

Figure 4.7.: These box plots show the effect of the sensing range on the aggregation performance with $n = 100$ robots. In both plots, the horizontal axis represents the sensing range, as a proportion of the maximum possible distance between two robots at the point of initialization. In (a), the vertical axis represents the dispersion of the robots after 1800 s (Equation 3.2, lower is better), and in (b), it represents the proportion of the robots in the largest cluster after 1800 s (Equation 3.3, higher is better). In both plots, each box represents values obtained from 100 simulations with different initial configurations of robots. In (a), the lower and the upper green lines show, respectively, the minimum possible dispersion for 100 robots, and their expected dispersion at the point of initialization.

4.3. A Sensor Variation: Finite Field of View

In Chapter 2, we discussed the “windshield” sensor introduced by Yu et al. [167], and in Chapter 3, we showed how our system implements an even simpler version of the sensor that only has a line of sight, rather than a finite field of view. We demonstrated how this

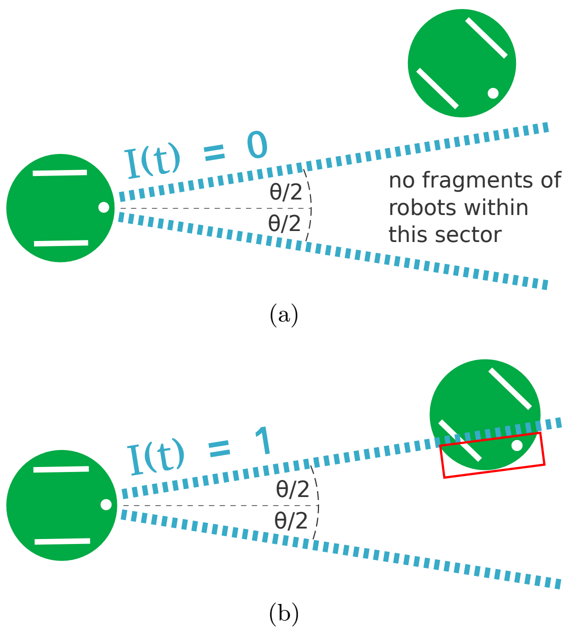


Figure 4.8.: A schematic representation of the binary sensor with a finite field of view (FOV), indicated by the dashed blue lines. The FOV is symmetric about the axis of the robot's orientation (dashed black line), and has an angle of θ. In (a), the sensor returns a reading of $I = 0$, because there are no fragments of other robots within the FOV. In (b), the sensor returns a reading of $I = 1$ because there is at least a fragment of at least one robot (red rectangle) within the FOV.

simplified sensor could be implemented using a single column of pixels from a robot's camera, and in principle, even using one pixel.

In this section, we re-introduce the concept of a finite field of view (FOV) (e.g. implementable by using more columns of pixels from the robot's camera), and systematically study its effect on the system's performance, and its capability to scale with the number of robots. We decide to implement the sensor in a similar way to [167], except that in our case, the robots are embodied. As shown in Figure 4.8, we let the sensor have a FOV of some angle θ, and this FOV is symmetric about the axis of the robot's orientation. The sensor produces a reading of $I = 1$ whenever at least a fragment of at least one other robot intersects with this "triangle" of vision (i.e. the FOV), and a reading of $I = 0$ otherwise. The sensor does *not* provide any information about the location of a perceived robot within the FOV.

The interest of this sensor is that the variation of θ translates into a trade off between the *span* of the environment from which information can be acquired, and the *localization* of the obtained information. For $\theta = 0°$, this sensor reduces to the binary line-of-sight sensor of the "standard" system of Chapter 3. This corresponds to the smallest span, with the highest degree of localization: if a robot j is seen by robot i, then the perpendicular distance between j's center and i's line of sight is guaranteed to be within the robots' radius, r. At the other extreme, $\theta = 360°$ corresponds to the maximal span and the minimal degree localization. The sensor would cease to provide *meaningful* information at this point: in the presence of at least one other robot in the environment, the sensor would constantly produce a reading of $I = 1$.

We would like to answer the following question. How does this trade-off in span versus localization of information translate into an effect on the system's performance as θ is varied? Does this effect carry across unchanged when the controllers are implemented on larger numbers of robots than what they were synthesized for?

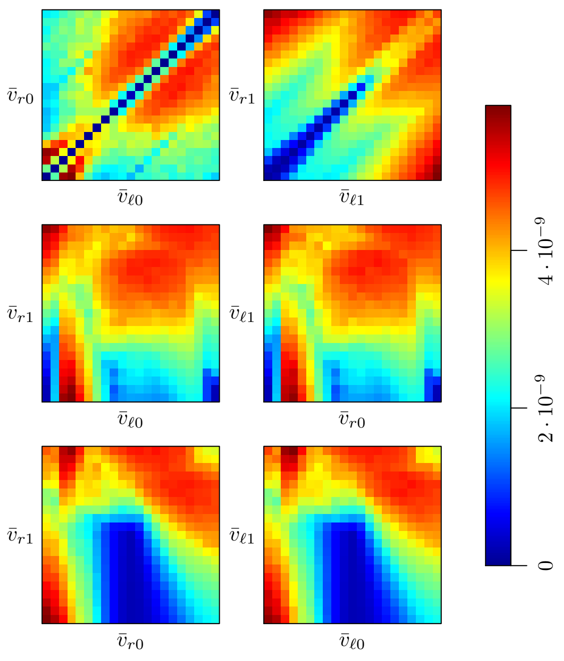
It would not be a fair process to simply evaluate the optimal controller that we obtained in the previous chapter (for $\theta =° 0$) on different sensor angles. Instead, we need to find the optimal controllers for different values of θ and then post-evaluate those on the angle that they are optimal for. We will do this in the following two sections.

4.3.1. Controller Synthesis

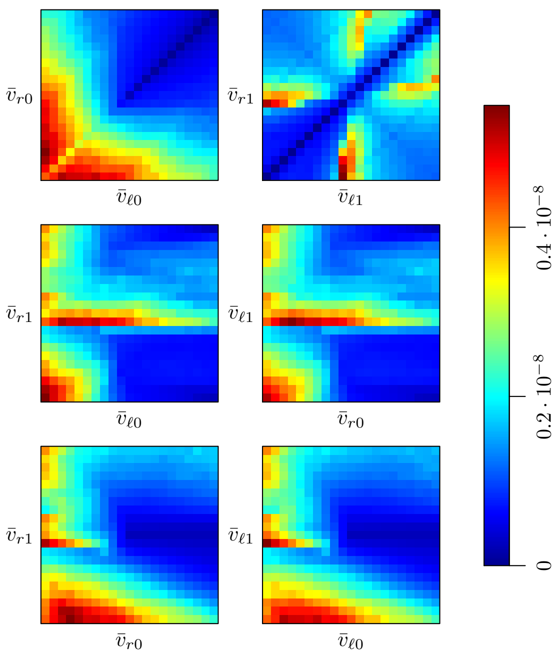
We ran 7 grid searches[1] with $n = 10$ robots for $\theta \in \{0°, 30°, 60°, 90°, 120°, 150°, 180°\}$. Each grid search was run in an identical manner as in the previous chapter (Section 3.4), that is, 100 evaluations per controller using $n = 10$ robots, and using the performance metric of Equation 3.4.

Figure 4.9a shows the performance landscape with $\theta = 0°$ (reproduced from Figure 3.4b for convenience). Figure 4.9b shows the landscape $\theta = 180°$ (we choose the other extreme, from the values that we have evaluated, to analyze the difference between the landscapes). With $\theta = 180°$, the landscape is significantly more contrasting with respect to well-performing and badly-performing regions. Moreover, most of the space is

[1]We also ran an additional 6 grid searches for the remaining angles up to 360° in steps of 30°. It turned out that the performance witch each of these angles was worse than that with 180° for every case we evaluated. In particular, as expected, with 360°, the aggregation performance was never better than in the baseline case of robots that do not move, because at this angle, the sensor does not provide any information about the environment. Therefore, we restrict our presentation here to the range up to 180°, because this is the range where interesting effects were observed.



(a) $\theta = 0$ degrees



(b) $\theta = 180$ degrees

Figure 4.9.: Visualizations of the controller performance landscapes for (a) $\theta = 0°$ and (b) $\theta = 180°$. Each box shows the landscape over two out of the four controller parameters. Each point represents the maximum performance that is obtainable over the space of the two omitted parameters. The performance is shown as the reciprocal of the value of Equation 3.4, meaning that higher (red) is better. See the text of Section 3.4 for more details.

dominated by poorly-performing controllers. A possible rationale for this phenomenon could be the following. With $\theta = 0°$, the information provided to the robots is more localized. Consequently, a robots "experiences" its environment in different ways when it moves with different patterns (i.e. employing different controllers). This gives rise to different regions of performance in the landscape, and also, as we have seen in Chapter 3 (Section 3.5.2), to strikingly different emergent behaviors. In contrast, because the information provided by the sensor with $\theta = 180°$ is much less localized, the patterns observed by the robot are similar for a wide range of motion patterns (i.e. controllers), and this leads to broader sub-spaces of similar performance. There are only small sub-spaces of controllers which lead to information patterns that can be exploited to achieve aggregation.

Table 4.1 shows the parameters $(\bar{v}_{\ell 0}, \bar{v}_{r0}, \bar{v}_{\ell 1}, \bar{v}_{r1})$ of the optimal controllers for $\theta \in$

θ°	$\bar{v}_{\ell 0}$	$\bar{v}_{r0}$	$\bar{v}_{\ell 1}$	$\bar{v}_{r1}$
0	-0.7	-1	1	-1
30	1	-1	1	0.8
60	-0.8	-1	1	-0.9
90	-1	-0.9	-0.8	1
120	-1	-0.9	-0.6	0.8
150	-1	-1	-1	0.5
180	-1	-0.7	-0.1	-1

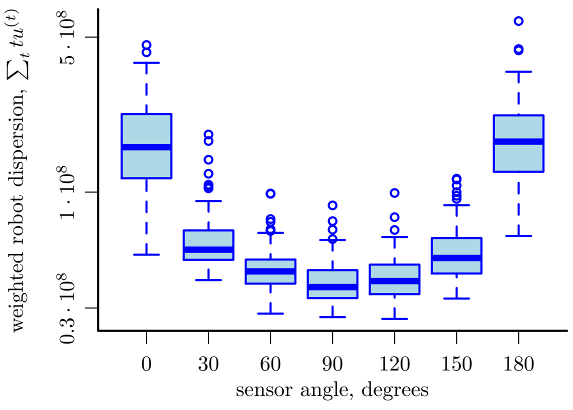
Table 4.1.: The parameters $(\bar{v}_{\ell 0}, \bar{v}_{r0}, \bar{v}_{\ell 1}, \bar{v}_{r1})$ of the optimal controllers for various sensor angles $\theta = 0, 30, 60, \ldots, 180$ degrees, obtained from the grid searches.

$\{0^\circ, 30^\circ, 60^\circ, 90^\circ, 120^\circ, 150^\circ, 180^\circ\}$, obtained from the grid searches. The controller for $\theta = 30^\circ$ is vastly different in structure from the one for $\theta = 0^\circ$. With this controller, a robot rotates on the spot when it does *not* perceive another robot, and when it does, it moves almost directly forward. Interestingly, however, for the larger values of θ, the controller becomes once again structurally similar to the one for $\theta = 0^\circ$, that is a robot moves backwards along a curved trajectory when $I = 0$, and rotates (although not strictly on the spot) when $I = 1$. It is important to note, however, that even though these controllers are *structurally* similar, it does not mean they are necessarily *behaviorally* similar, because the angle of the sensor is different in each case. In fact, as we shall see in the next section, these controllers lead to a variety of performances.

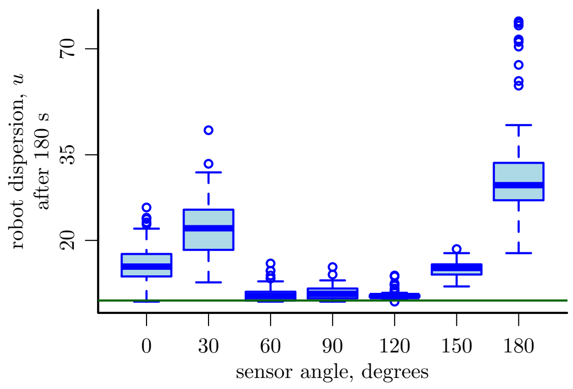
4.3.2. Post-Evaluations

We post-evaluated each of the optimal controllers obtained in the previous section (Table 4.1) for 100 times on $n = 10$ and $n = 100$ robots. In each simulation, we recorded the final dispersion and cluster sizes (Equations 3.2 and 3.3). In addition, we also recorded the *weighted* dispersion over time, which was the metric used in evaluating the grid searches (3.4), in order to be able to analyze the *speed* of aggregation with the different sensor angles.

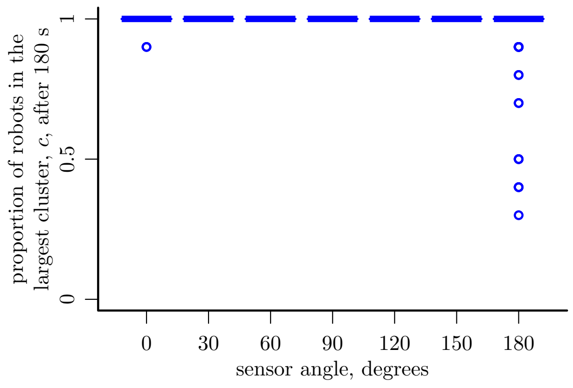
Figure 4.10 shows the performances of the optimal controllers with various sensor angles on $n = 10$ robots. The most interesting metric to observe here is the weighted dispersion



(a) weighted dispersion



(b) final dispersion



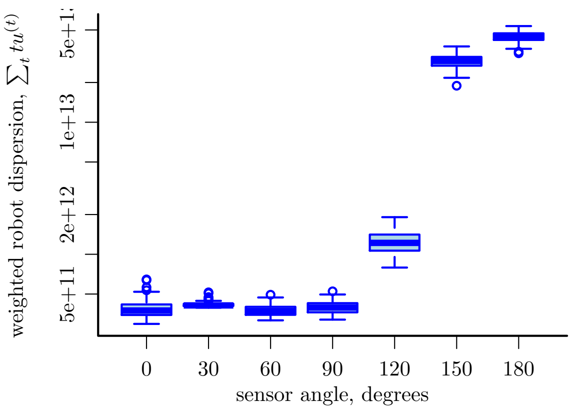
(c) final cluster size

Figure 4.10.: 10 Robots, Varying Sensor Angle

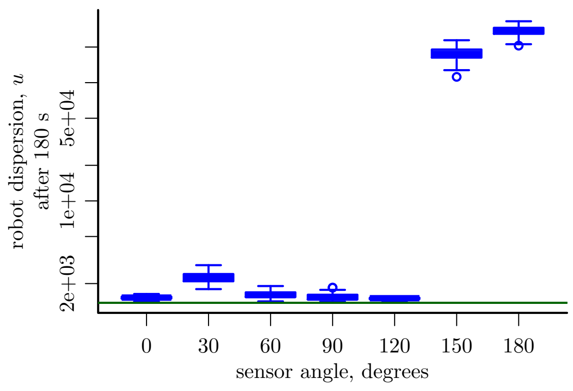
metric (4.10a), which measures the dispersion of the robots over time. The trade-off of the sensor in span versus localization has translated into a bowl-shape for this metric, with the optimum occurring somewhere around $\theta = 90°$. With $\theta = 180°$, the performance of the sensor is around the same as that of the one with a line-of-sight (i.e. $\theta = 0°$) (no statistically significant difference; Mann-Whitney test, 2% significance level). The profile of the performance metric is more or less symmetric around the $\theta = 90°$ point. Looking at the final dispersion of the robots (4.10b), we observe that this metric becomes worse when θ is increased from 0° to 30°, and then better than it was with 0° for $\theta = 60°, 90°, 120°, 150°$ (all statistically significant differences; Mann-Whitney test, 2% significance level). The final dispersion is worst with $\theta = 180°$). In terms of the final cluster sizes (4.10c), all the robots always manage to aggregate for up to $\theta = 150°$, except for one outlier value with $\theta = 0°$. Most of the runs with $\theta = 180°$ also terminate with the robots in a single cluster, but this time there are a number of outliers, some with even less than half of the robots being aggregated into the largest cluster.

We now turn our attention to how the optimal controllers with various sensor angles synthesized with $n = 10$ robots perform when post-evaluated on $n = 100$ robots. Figure 4.11 shows the weighted dispersion, final dispersion, and final cluster size metrics from these post-evaluations. We notice that the profile of the weighted dispersion metric across the angles (Figure 4.11a) does not directly translate from the situation with $n = 10$ robots. The sensors with up to $\theta = 90°$ perform roughly similarly, while the sensor with $\theta = 120°$ performs substantially worse (it performed substantially better than the line-of-sight with $n = 10$ robots). The sensors with $\theta = 150°, 180°$, which were better and no worse than the line-of-sight sensor on $n = 10$ robots, respectively, now seem to exhibit a complete degradation in performance. The profile of the final dispersion metric (Figure 4.11b) is similar to that of the weighted dispersion metric; interestingly, as in the case with $n = 10$ robots, the final dispersion with $\theta = 30°$ is worse than that with $\theta = 0°, 60°, 90°, 120°$ (Mann-Whitney test; 2% significance level). The plot showing the final cluster sizes (Figure 4.11c) reveals that, despite their differences in the other two metrics, the sensors with θ up to 120° all lead to the robots aggregating into a single cluster (excepting some outlier runs with $\theta = 30°$). This plot also confirms the total degredation in performance with $\theta = 150°$ and 180°: less than 20% and 10% of the robots end up in the largest cluster, respectively.

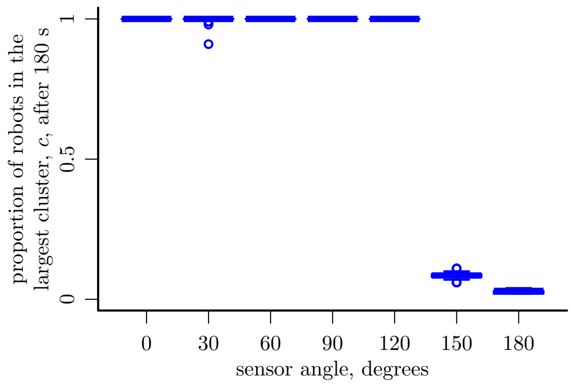
In the above results, there seems to be an anomaly with $\theta = 30°$. Is there something inherent about this specific sensor angle that limits its performance, or is this a consequence of the metric that we have chosen in synthesizing the controller? In order to settle this question, we picked from the grid search with $\theta = 30°$ the controller that led to the lowest *final* dispersion (as opposed to the *weighted*) dispersion, and re-evaluated



(a) weighted dispersion



(b) final dispersion



(c) final cluster size

Figure 4.11.: 100 Robots, Varying Sensor Angle

this controller on $n = 100$ robots. It turns out that this time, the sensor with $\theta = 30°$ achieves a final dispersion that is not significantly worse that than of the line-of-sight sensor (Mann-Whitney test; 2% significance level). We can therefore conclude that this anomaly in the plots at this specific angle is a consequence of the chosen setup (e.g. area of initialization, running time of the simulations). In particular, note that the weighted dispersion metric is sensitive to the running time of the simulations, T. In particular, as this time tends to infinity, ranking controllers by the weighted dispersion metric becomes equivalent to ranking them with the final dispersion metric, because the former places more and more emphasis on later values of $u^{(t)}$, and will therefore converge to a value proportional to the steady-state value of $u^{(t)}$.

4.3.3. Discussion

The most salient observation from the results in this section is that the performance profile across a given parameter of the system—in this case, the sensor angle—does not necessarily carry across from the size of the group on which it was synthesized onto larger swarms. This discrepancy might even be up to the point of a successful solution turning into a complete failure, such as the situation with $\theta = 150°$ above. Interestingly, however, the simplest controller, the one with a line of sight instead of a finite field of view, performed well with both $n = 10$ and $n = 100$ robots. Moreover, while it was outperformed (with respect to a subset of the metrics) by another sensor on the number of robots used in the controller synthesis, this difference became insignificant when scaled up to a larger number robots.

The results suggest that the designer of a swarm-intelligent system must be careful when making inferences from the behavior of the system with some number of agents to a system with more or less agents. Perhaps we may also conclude that a designer should always give serious consideration to the simplest solution (in this case, the line-of-sight sensor), only moving on to more elaborate solutions as necessary and with caution.

4.4. A Controller Variation: Run-Time Memory

In the interest of minimalism, in the original system of Chapter 3, we imposed the constraint that the robots did not have the ability to store any information during run-time (e.g. past sensor inputs). This, combined with the discrete nature of the robots' sensors, resulted in a switching controller that did not rely on performing arithmetic computations (Figure 3.1). Such memory-less controllers are often termed "reactive controllers" [53, 54], because the robot always behaves by reacting to its instantaneous perception of the environment.

We will now consider the introduction of memory into the controller structure, and investigate whether this can result in benefits to the system's performance. We introduce this memory in the form of an artificial neural network (NN) with recurrent connections. More specifically, we choose the Elman network structure[2], which was introduced by Elman [192], and has been used widely in evolutionary robotics experiments (e.g. [127]). Hereafter, we will refer to this controller as the "recurrent NN controller".

In Section 4.4.1, we will see that the recurrent NN controller that we choose is parametrized by 14 unbounded real values. Consequently, it is not computationally feasible to apply the grid search technique used for synthesizing the reactive controllers in the previous chapter (Section 3.4) for synthesizing recurrent NN controllers. We will therefore resort to using a metaheuristic algorithm for synthesizing these controllers. For a fair comparison, we will also re-synthesize reactive controllers using the same algorithm, as the grid search used in Chapter 3 had a relatively coarse resolution of 0.1. As our metaheuristic algorithm, we choose the state-of-the-art Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [193], which we will describe briefly in Section 4.4.2.

4.4.1. New Controller Formulation

Figure 4.12 shows a schematic representation of an Elman network. This network consists of N_i input units (which are not strictly speaking neurons) that are fully connected N_h hidden neurons. In addition, the hidden neurons are also fully connected among themselves. In other words, the output of each hidden neuron at time step $t-1$ is fed back as an input to all the hidden neurons (including itself) at time step t. The hidden neurons are also fully connected to the network's N_o output neurons. Each connection between an input unit and a neuron, or between two neurons, has an associated weight, $w \in \mathbb{R}$; additionally, each neuron (i.e. excluding the input units) has an associated bias term, $\theta \in \mathbb{R}$.

[2]The Elman structure is a special case of the fully-connected structure [191]. The former has been selected for presentation here mainly because it seems to have been more widely adopted in evolutionary robotics experiments than the latter. Nevertheless, all the results presented have also been duplicated with a fully-recurrent network and, in all cases, the performances yielded by the two networks were not substantially different. In particular, both networks were always matched in their performance with respect to the reactive controller (i.e. better, worse or not significantly different).

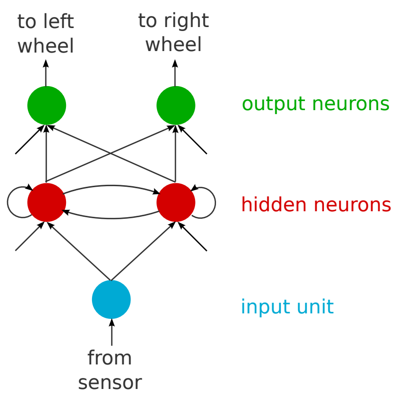


Figure 4.12.: A graphical representation of the Elman neural network controller. We show the case specific to our implementation, that is with $N_i = 1$ input unit being fed from the sensor, $N_h = 2$ hidden neurons, and $N_o = 2$ output neurons whose outputs are used to control the robot's wheel speeds. The free-hanging arrows represent the neurons' bias terms; these may be thought of a connection weights from units that have a constant output of 1.0.

Formally, the inputs to outputs mapping of the network is computed as follows. Let $I_j^{(t)}$, $j \in \{1, 2, \ldots, N_i\}$ represent the value of input unit j at time t. Then, the activations of the hidden neurons $H_k^{(t)}$, $k \in \{1, 2, \ldots, N_h\}$ at time t are given by:

$$H_k^{(t)} = \text{sigmoid}\left[\sum_{j=1}^{N_i} w_{jk}^{IH} I_j^{(t)} + \sum_{j=1}^{N_h} w_{jk}^{HH} H_j^{(t-1)} - \theta_k^H\right], \tag{4.2}$$

where w_{jk}^{IH} is the weight associated with the connection between input unit I_j and neuron H_k, w_{jk}^{HH} is the weight associated with the connection between neurons H_j and H_k, θ_k^H is the bias of neuron H_k, and sigmoid denotes the logistic sigmoid function, given by:

$$\text{sigmoid}\,(x) = \frac{1}{1 + e^{-x}} \quad \forall x \in \mathbb{R}. \tag{4.3}$$

At the first time step ($t = 0$) $H_j^{(t-1)}$ is defined to be sigmoid$(0) = 0.5$, for all $j \in \{1, 2, \ldots, N_h\}$.

Finally, the activations of the output neurons, O_k, $k \in \{1, 2, \ldots, N_o\}$ are computed as:

$$O_k^{(t)} = \text{sigmoid}\left[\sum_{j=1}^{N_h} w_{jk}^{HO} H_j^{(t)} - \theta_k^O\right], \tag{4.4}$$

where w_{jk}^{HO} is the weight associated with the connection between neurons H_j and O_k, and θ_k^O is the bias of neuron O_k.

The nonlinear nature of the logistic sigmoid activation function, combined with the recurrent connections, endow the Elman network with capabilities for rich dynamics. In fact, it can be shown that the Elman network structure can approximate the dynamics of any nonlinear system to an arbitrary accuracy, by choosing an appropriate number of hidden neurons and suitable parameter values (i.e. weights and biases) [194].

Note that the number of parameters in an Elman network is given by $N_i N_h + N_h^2 + N_h + N_h N_o + N_o$. As our robots have one sensor, we have one input unit ($N_i = 1$) which takes its value directly from the sensor's reading (i.e. $I^{(t)}$). Our network has two output neurons ($N_o = 2$), which serve to actuate the robot's two wheels. The activation of each output neuron, O_k, $k \in \{1, 2\}$, which is in the interval $(0, 1)$ (from Equation 4.3), is linearly mapped onto a normalized wheel speed (i.e. in $(-1, 1)$) by computing $2O_k - 1$. We chose to use $N_h = 2$ hidden neurons in our network[3], which makes for a total of 14 parameters (see Figure 4.12).

4.4.2. Covariance Matrix Adaptation Evolution Strategy

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [193] is an algorithm for optimization on real-valued decision vector spaces, $\mathbb{R}^d$. The main feature of CMA-ES is the non-random adaptation of the variance of each decision variable, as well as all the covariances among these variables. CMA-ES is quasi parameter free, because all its internal parameters can be set according to theoretically-justified formulae [193, 195]. The user only needs to specify three exogenous parameters: a starting point for the decision vector, $\mathbf{m}^{(0)}$; the initial step size, $\sigma^{(0)}$; and the population size, λ. CMA-ES is also scale-invariant, because in its selection mechanism, it does not take into account the actual fitnesses of the candidate solutions, but rather selects the μ best solutions from the λ candidate solutions. This is a desirable feature for evolutionary robotics applications, because it ensures that the outcome of the evolution is not overly sensitive to the precise choice of the performance metric. For a detailed explanation of CMA-ES and all its features, we refer the reader to [193, 195]. In the following, we will outline the details that are specific to our implementation.

[3] We have also experimented with other values for N_h, namely 1, and 3–5, and the results were similar in each case. However, note that N_h cannot be chosen to be too large, because as the search space becomes larger (it grows as $\mathcal{O}(N_h^2)$), it becomes more difficult for the evolutionary algorithm to synthesize well-performing controllers.

4.4.2.1. Constraint Handling

In its standard form, CMA-ES operates across the entire real space, $\mathbb{R}^d$. This is well-suited for synthesizing our recurrent NN controllers, whose parameters are also in $\mathbb{R}^d$ ($d = 14$); however, when synthesizing reactive controllers, the parameters are required to be constrained to the interval $[-1, 1]$. In order to achieve this, we opted to let CMA-ES operate in its unconstrained form; however, before evaluating the performance of a candidate solution, we mapped each of its four parameters onto the feasible region by using the logistic sigmoid function of Equation 4.3, as:

$$\bar{v} = 2\,\mathrm{sigmoid}(v) - 1, \tag{4.5}$$

where v is the "raw" value in the algorithm's candidate solution, and $\bar{v}$ is in $(-1, 1)$ and can therefore be used as a normalized wheel speed.

4.4.2.2. Parameter Settings

For synthesizing both types of controller, we set the starting point for the decision vector, $\mathbf{m}^{(0)}$, to the zero vector. For synthesizing recurrent NN controllers, whose parameters are in $\mathbb{R}^d$ ($d = 14$), we chose $\sigma^{(0)} = 1$. For synthesizing reactive controllers ($d = 4$), we set $\sigma^{(0)} = 0.72$: Monte Carlo simulations show that with this setting, the initial population will be approximately uniformly distributed in $[-1, 1]^d$, when it is mapped onto this region by Equation 4.5. For the population size, Hansen [195] suggests a value of around $\lfloor 4 + 3 \ln d \rfloor$, which corresponds to $\lambda = 8$ and $\lambda = 11$ for $d = 4$ and $d = 14$, respectively. We chose to use $\lambda = 10$ in both cases, which we identified to be a reasonable value after a number of preliminary runs.

4.4.2.3. Performance Evaluation

Performance evaluations were performed in a similar fashion to that used in running the grid searches to synthesize reactive controllers in Chapter 3. We used $n = 10$ robots, and in order to evaluate each controller (i.e. candidate solution), we performed 10 simulations employing that controller with different initial configurations (the set of initial configurations was identical for all candidate solutions in a given generation). The

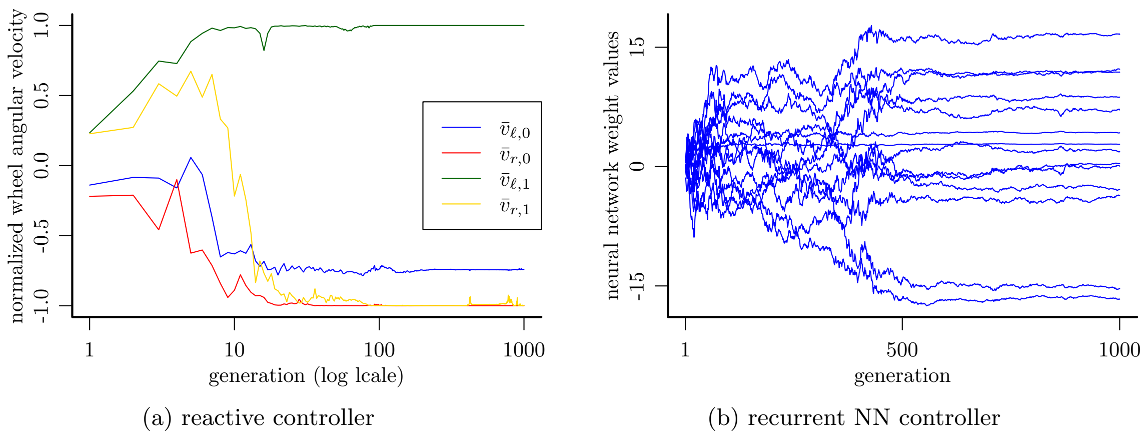


Figure 4.13.: Parameter dynamics in the evolutions with a reactive controller (4.13a) and a recurrent NN controller (4.13a). Note that the scales on the vertical axes are not identical in the two figures.

performance of the controller in each simulation $k \in \{1, 2, \dots, 10\}$, was evaluated as:

$$U(\mathbf{x}) = \sum_{t=0}^{T-1} t\, u^{(t)}. \tag{4.6}$$

The overall performance of the controller was calculated as the average of these values, i.e. $\bar{U}(\mathbf{x}) = \sum_{k=1}^{10} U_k(\mathbf{x})/10$.

4.4.3. Controller Synthesis

We performed 100 runs of the evolution strategy for each type of controller (i.e. reactive and recurrent NN), with each run lasting for $G = 1000$ generations. The mean solution of the final generation (i.e. $\mathbf{m}^{(G)}$) of each run was post-evaluated 100 times using the metric of Equation 4.6. For each type of controller, the run corresponding to the $\mathbf{m}^{(G)}$ with the highest U was chosen as the winning run, and its $\mathbf{m}^{(G)}$ was chosen as the winning controller for that type.

Figure 4.13 shows the evolution of the controller parameters in the two winning runs. In the case of the reactive controller (4.13a), the parameter dynamics stabilize quickly — by around the 30th generation. Beyond the 100th generation, there is virtually no more change in any of the parameters. The parameters of the last generation (i.e. $\mathbf{m}^{(G)}$) are, to within two decimal values: $(\bar{v}_{\ell 0}, \bar{v}_{r0}, \bar{v}_{\ell 1}, \bar{v}_{r1}) = (-0.74, -1.00, 1.00, -1.00)$. As

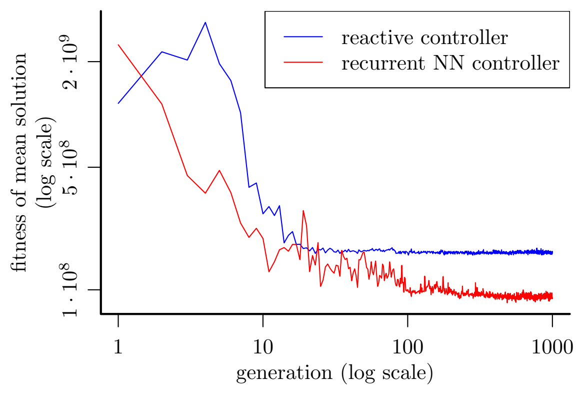


Figure 4.14.: Dynamics of the performance of the mean solution in the two best evolutions with the reactive (blue) and the recurrent NN (red) controllers.

expected, this is structurally identical to the optimal controller obtained from the grid search for $n = 10$ robots; the only difference is in the first parameter which now has a value of -0.74 rather than -0.7, a difference of 5.71%. This is due to the fact that the evolution strategy has a much higher resolution than the grid search (its resolution is limited only be the representation precision on the computer hardware).

The parameter dynamics of the recurrent NN controller (4.13b) are somewhat slower to evolve than their counterparts in the reactive controller. This is to be expected as the search space is significantly larger (14 parameters as opposed to 4). In this case, the dynamics reach steady-state after around the 500th generation. Although the algorithm was not bounded to any interval, all the parameters (in all generations) are contained within $(-20, 20)$, which gives an indication that the algorithm is effective.

4.4.4. Post-Evaluations

We evaluated the mean solution (i.e. $\mathbf{m}$ in the CMAES algorithm) of each generation $g \in \{1, 2, \ldots, 1000\}$ in the two winning evolutionary runs with $n = 10$ robots for 100 times. For each run, we recorded the metric of Equation 4.6. Figure 4.14 shows the dynamics of the mean metrics from these 100 post-evaluations over g for the winning reactive (blue) and recurrent NN (red) evolutionary runs. The fitness of the recurrent NN controller is consistently better than that of the reactive controller across g, except for a short spike in the 15th generation. At steady state, it becomes clear that there is a significant gap between the performances of the two controllers. Figure 4.15 is a box plot showing distributions of the final dispersions of the robots across 100 post-evaluation

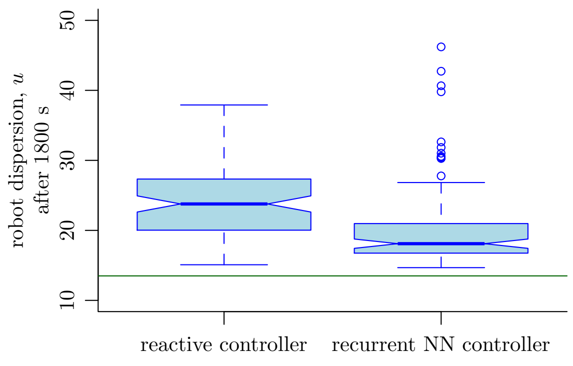


Figure 4.15.: Post-evaluations with the reactive and the recurrent NN controllers with $n = 10$ robots.

simulations with $n = 10$ robots, using the two best controllers. There is a statistically significant difference between the two sets of values (Mann-Whitney test, 2% significance level). The final dispersion with the reactive controller is, on average, 77.90% worse (i.e. larger) than the optimal (i.e. minimum) possible dispersion (green line), while with the recurrent NN controller, this discrepancy is considerably less, at 50.79%. Nevertheless, with the recurrent NN, there are four outlier values that are worse than the worst values with the reactive controller.

That the recurrent controller performs better (on average) as the reactive one is an expected result. This is because the structure of the recurrent controller is a superstructure of that of the reactive controller. In other words, for any given reactive controller, there is at least one recurrent controller that behaves exactly like it. One such equivalent can be found by setting all the recurrent connections to zero and working out the remaining parameters (in general, there will be multiple solutions because of the "multiple representation" nature of neural networks [127]).

We would now like to investigate whether the advantage of the recurrent NN controller over its reactive counterpart carries over when the two are scaled up to larger numbers of robots. Towards this end, we ran 100 post-evaluations per controller using 100 robots and we recorded the dispersion of the robots at the end of each run.

Strikingly, the scenario in Figure 4.16 is now the opposite of that with $n = 10$ robots, as the recurrent NN controller scales much worse than its reactive counterpart onto $n = 100$ robots. The difference is statistically significant (Mann-Whitney test, 2% significance level). The reactive controller now leads to a final dispersion that is only 7.35% worse

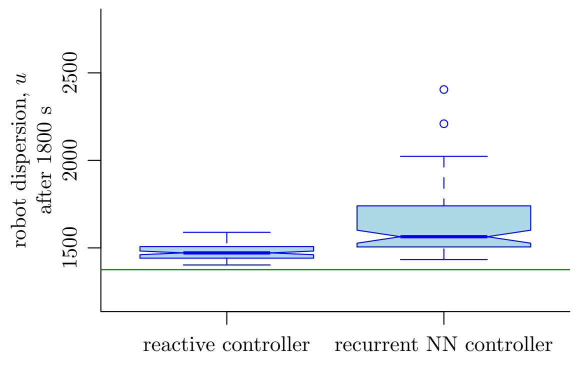


Figure 4.16.: Post-evaluations with the reactive and the recurrent NN controllers with $n = 100$ robots.

(i.e. larger) than the minimum possible dispersion (green line). For the recurrent NN controller, this figure is 19.44%. In the worst cases across the 100 runs (excluding outliers), the reactive and the recurrent NN controllers lead to final dispersions that are 17.87% and 49.86% worse than the optimum, respectively.

It was verified that given enough time, both controllers lead to a single cluster, starting from any configuration, for both $n = 10$ and $n = 100$ robots. This means that the primary differences in between the two controllers are (i) the final dispersion of the robots, as shown above, and (ii) the speed of aggregation.

4.4.5. Discussion

The phenomenon present in the results points to the well-known problem of over-adaptation in evolutionary-driven design in general, and evolutionary robotics in particular (e.g. [196]). This happens because the evolutionary algorithm is "willing" exploit any factor that is available to it in order to improve the objective function. Often, factors that are constant within the performance evaluations tend to be prime candidates in succumbing to such an exploitation.

In our case, one such constant factor happens to be the number of robots, n, used in the simulations for evaluating each candidate solution (i.e. controller). A possible explanation for the difference in performance and scalability of the reactive and the recurrent NN controllers is as follows. The reactive controller, having no run-time memory, has a limited range of motion patterns that it can generate on the robots (we showed in

Chapter 3 (Section 3.3) that only 16 fundamentally distinct motion patterns are possible with a binary sensor). On the other hand, the recurrent NN controller opens up the potential for much richer dynamics, which translate into a wider variety of motion patterns for the robots. Consequently, the evolutionary algorithm is not able to exploit the fixed number of robots with the reactive controller, but in the case of the recurrent NN controller, it is able to select, from the much larger controller space, the one that is the best for $n = 10$ robots specifically.

A common technique for avoiding over-adaptation in evolutionary robotics is to try to avoid, as much as possible, constant factors within the environment. In this case, one could vary the number of robots used in the performance evaluations. For instance, instead of being evaluated 10 times with $n = 10$ robots, each controller could be evaluated 10 times with different numbers of robots. In this case, a simple averaging process would not be ideal to find the "overall" performance of the controller, and a more appropriate method for ranking for the candidate solutions in the algorithm would be required.

However, the above method does not necessarily address the issue of scalability, unless the controllers are also evaluated on large numbers of robots, which can be prohibitively computationally expensive. Even then, one could question whether *those* controllers would scale up well with even *larger* numbers of robots (but in this case, a law of "diminishing differences" would seem to apply[4]).

The results obtained in this section indicate that when scalability is desired in swarm robotic systems, and an evolutionary robotics methodology is used to identify the robots' individual behaviors, the designer should seriously question whether the introduction of any memory is really required in the system, and if so, carefully evaluate how much of it is appropriate.

4.5. Summary

This chapter has furthered the results of the previous one by presenting a number of systematic investigations into the minimal information processing self-organized robot

[4]For example, if the controllers had to be synthesized with $n = 100$ robots, and post-evaluated on $n = 1000$ robots, the degradation in performance of the recurrent NN controller would be expected to less than that which we have observed. This is because the global dynamics of $n = 1000$ (homogeneous) robots "look" less different from those of $n = 100$ robots than the dynamics of $n = 100$ robots "look" from those of $n = 10$ robots. In other, more complex problems, however, this observation may cease to be so clear-cut.

aggregation problem. In Section 4.1, we confirmed our previous hypothesis that the simplicity of our sensor/controller solutions comes at a performance price, especially in terms of speed, as opposed to what could be achieved with more information acquisition on the part of the robots. However, we showed that our minimalistic solution scales well to large numbers of robots: in particular, it manages to aggregate at least $n = 1000$ robots into a single cluster consistently, and the time for doing so scales sub-linearly with the number of robots. In Section 4.2, we demonstrated that the system is robust with respect to noise on the sensor, both in the forms of false negative and false positive noise. We also investigated the practical ramifications of our theoretical result in Chapter 3, that is the need for an unlimited sensing range under our sensor formulation. We showed that, at least for one reasonably practical setup, the range of the sensor does not begin to have an adverse effect until it is reduced to around 20 to 30% of what is effectively unlimited at the point of initialization.

The results of Sections 4.3 and 4.4 turned out to be particularly interesting. They both point towards a “less is more” philosophy in swarm intelligence, at least when an evolutionary methodology is employed for synthesizing the behaviors of the individual agents. Because as human beings we are complex agents, and were not evolved for acting within a “swarm” (in the sense of, for example, ant or honeybee colonies), it may be difficult for us to come to terms with the concept that global complexity of the *system* does not necessarily entail equal local complexity of the *agents*. In other words, the whole can be larger than the sum of the parts. As unremarkable as it may sound, the results of these two sections remind us to adopt a bottom-up methodology in designing swarm-intelligent systems; that is, attempt first the simplest possible solution, and only gradually increase it in complexity until a satisfactory performance is acquired.

5. Object Clustering without Computation

In the previous two chapters, we developed a framework for synthesizing behaviors in swarm robotic systems in which the robots do not possess run-time memory and do not perform any arithmetic computations. This framework was applied to the problem of self-organized robot aggregation. As we have seen, this problem is a fundamental one in swarm robotics, because it can serve as a precondition for other forms of cooperation [126]. At the same time, however, self-organized robot aggregation is arguably on the simpler side of the complexity spectrum of swarm robotics problems.

The relative simplicity of the self-organized robot aggregation problem allowed us to perform an extensive analysis and evaluation of our framework for this specific case study. For instance, in the basic set-up of Chapter 3, the controller consisted of only four bounded real-valued parameters. Thus, we were able to systematically evaluate the performance of the entire controller space , within a finite resolution, through the use of a grid search. This allowed us to locate the optimal controller(s), and also to visualize the whole performance landscape, and analyze how it changed when the controllers were synthesized using different numbers of robots. We were also to analyze the most scalable synthesized controller mathematically, and prove its correctness for some particular cases.

The question remains, however, whether the framework that has been presented can be extended to more complex swarm robotics tasks. Therefore, in this chapter we will consider a progressively more complex task, which is in a sense, a natural extension of the previous one. This time, the robots are expected to aggregate not themselves, but a number of other, passive objects that are initially spread out throughout the environment. We refer to this problem as *self-organized object clustering*.

We have seen in Section 2.5 that this problem has received considerable attention in the literature (although not as widely as self-organized robot aggregation). A number of works have addressed it using robots that can sense the environment using a combination of infra-red proximity sensors, and force sensors [170, 171, 172]. Other works have approached the problem without the use of force sensing [173, 174]. The solution in [173] was partially successful: a good clustering performance was achieved, but the behavior did not work with a single robot, and it degraded in the presence of more than four. The solution in [174] was successfully demonstrated to work, but due to the behavioral rules used, the robots 'clustered' the objects into a linear formation rather than a compact one.

In this chapter, we present our solution to the self-organized object clustering problem by extending the framework developed over the previous two chapters in a straightforward manner. To our knowledge, this work is novel in three main aspects with respect to ones mentioned above:

- Our solution is the simplest solution presented so far in terms of both sensor and controller complexity. The robots in previous works used force sensors and/or infra-red proximity sensors with distance-measuring capabilities. Our robots are only equipped with discrete-valued, line-of-sight sensors that do not provide explicit distance information. This means that the robots have no direct means of knowing when they are in contact with another object or robot. Moreover, all the previous works made use of run-time memory and/or arithmetic computation in their control. In contrast, our controller does not require any run-time memory, and as a result of this combined with the discrete nature of the sensor, the controller does not perform any arithmetic computations.

- We explicitly investigate three different sensing configurations, to investigate how the ability of the robots to recognize each other against the objects being clustered and the background of the environment affects the clustering performance.

- Our solution is the first one to be synthesized using an evolutionary robotics methodology, rather than being hand-designed. The solution is successfully ported from simulation onto a physical system, without any structural modifications.

This chapter is organized as follows. Section 5.1 describes the methods used, including the problem definition (5.1.1), the characteristics of the objects and the robots (5.1.2), the performance metrics (5.1.1.3), and the evolutionary algorithm used to synthesize controllers (5.1.4). Section 5.2 explains how controllers were synthesized using the evolutionary algorithm for three different sensor configurations, and how post-evaluations were used to select the best of these controllers (in each configuration) for further investigation (5.2.2). Section 5.3 presents the results from a number of simulation studies about the emergent behaviors of the synthesized controllers (5.3.1), their scalability with respect to the number of robots in the environment (5.3.2), and their robustness with respect to sensory noise (5.3.3). Section 5.4 starts by describing how one of the sensor/controller configurations was ported onto a physical platform consisting of 5 e-puck robots and 20 objects (5.4.2). It then outlines the experimental setup procedure used for evaluating this implementation (5.4.2), and presents the results obtained (5.4.3). Section 5.5 summarizes the chapter.

5.1. Methods

Note that the methods used in this chapter are all similar to the ones that have been introduced in the previous two chapters, when applying the framework to the self-organized robot aggregation problem. As such, here, we will only briefly review what has already been described before, and will focus on highlighting the important differences and extensions of the methods as they are applied to the new problem of self-organized object clustering.

5.1.1. Problem Definition

As we have seen in Section 2.5, the self-organized object clustering problem has been formulated in different ways throughout the literature. For instance, in some cases the robots had explicit mechanisms for directly manipulating the objects; in others, the robots pushed the objects without directly manipulating them. Some formulations used robots with force sensors; others without. In most cases, the objects being clustered were cylindrical in shape, but cuboid objects were used in others [173, 172].

In our formulation, we consider an environment containing $m \geq 2$ cylindrical objects and $n \geq 1$ differential-wheeled robots that are also cylindrical in shape. The robots do not have any mechanisms for directly manipulating the objects, and can therefore only move the objects by pushing against them. As in all the previous works that we are aware of, the environment does not contain any items that could act as obstacles, other than the robots and the objects being clustered. In our simulations (both for controller synthesis

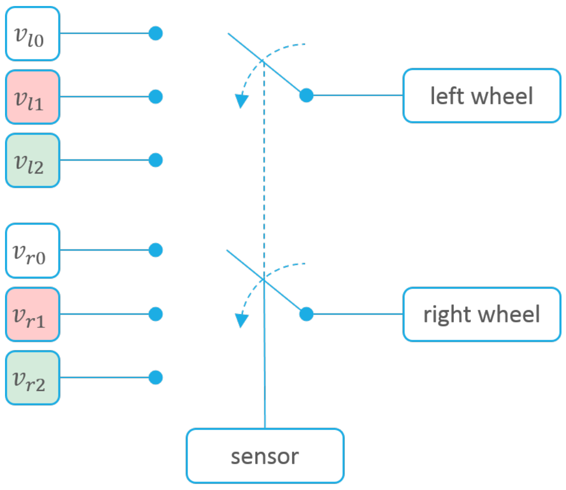


Figure 5.1.: A conceptual schematic of the robot's controller, which is simply a mapping from each possible sensor reading onto a pair of speeds for the robot's wheels. This structure lends itself naturally to a hard-wired implementation.

and post-evaluations), we use an unbounded environment, but we subsequently port a solution onto a physical system where the environment is bounded.

5.1.1.1. Sensor

Each robot is equipped with a line-of-sight sensor I at its front. In its general form, the sensor is ternary. It returns $I = 0$ if it is pointing at the background of the environment (i.e. empty space if the environment is unbounded, or the boundary if the environment is bounded), $I = 1$ if it is pointing at an object, and $I = 2$ if it is pointing at another robot. Note that the sensor does not explicitly provide the distance to a perceived object or robot, nor does it directly provide any information about the numbers of objects or robots.

5.1.1.2. Controller

The robots have no memory, i.e. they are not able to store previous inputs, and therefore, at each time step t, they must decide on their actions based solely on the current sensor reading, $I^{(t)}$. As $I^{(t)}$ is discrete-valued, only one form of controller is possible: a mapping from each of the three possible values onto a pair of angular velocities for the robot's wheels (see Figure 5.1). Note that any other memory-less controller (e.g. a feed-forward neural network) can be reduced to this form.

Let $\bar{v}_\ell, \bar{v}_r \in [-1, 1]$ represent the normalized angular velocities of the robot's wheels, where -1 and 1 correspond, respectively, to the wheel rotating backwards and forwards with the maximum possible velocity. The controller can now be represented as a six-valued vector:

$$\bar{\mathbf{v}} = \left(\bar{v}_{\ell,0}, \bar{v}_{r,0}, \bar{v}_{\ell,1}, \bar{v}_{r,1}, \bar{v}_{\ell,2}, \bar{v}_{r,2}\right), \quad \bar{\mathbf{v}} \in [-1, 1]^6 , \tag{5.1}$$

where $\bar{v}_{\ell,0}$ denotes the normalized angular velocity of the left wheel when $I = 0$, etc.

5.1.1.3. Performance Metric

For this task, we use a performance metric that we had already used in the self-organized robot aggregation problem, namely the second moment [182]. This time, however, this metric is applied to the positions of the objects, rather than those of the robots.

Let r_o represent the radius of one object. Let $\mathbf{p}_i^{(t)}$ represent the position of object i at time t, and let $\bar{\mathbf{p}}^{(t)} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{p}_i^{(t)}$ represent the centroid of the positions of the objects. Then, the second moment of the objects is given by:

$$u^{(t)} = \frac{1}{4r_o^2} \sum_{i=1}^{m} ||\mathbf{p}_i^{(t)} - \bar{\mathbf{p}}^{(t)}||^2. \tag{5.2}$$

Recall that the $4r_o^2$ in the denominator serves to normalize $u^{(t)}$ such that it becomes independent of r_o. $u^{(t)}$ does not have an upper bound, because the objects can be arbitrarily dispersed. It has a positive lower bound, because of the physical constraint that the objects cannot overlap with each other, i.e. $||\mathbf{p}_j^{(t)} - \mathbf{p}_i^{(t)}|| \geq 2r_o$, $i \neq j$. Graham and Sloane [182] report lower bounds of $u^{(t)}$ for several values of n up to $n = 499$. Except for $n = 212$, the packings corresponding to the lower bounds of $u^{(t)}$ are optimal among hexagonal packings [183].

Equation 5.2 measures the dispersion of the objects at one point in time. We use the following equation to measure the object clustering performance of a controller $\bar{\mathbf{v}}$ when employed on a number of robots for T time steps, $t \in \{0, 1, \ldots, T-1\}$:

$$U\left(\bar{\mathbf{v}}\right) = \sum_{t=0}^{T-1} t\, u^{(t)}. \tag{5.3}$$

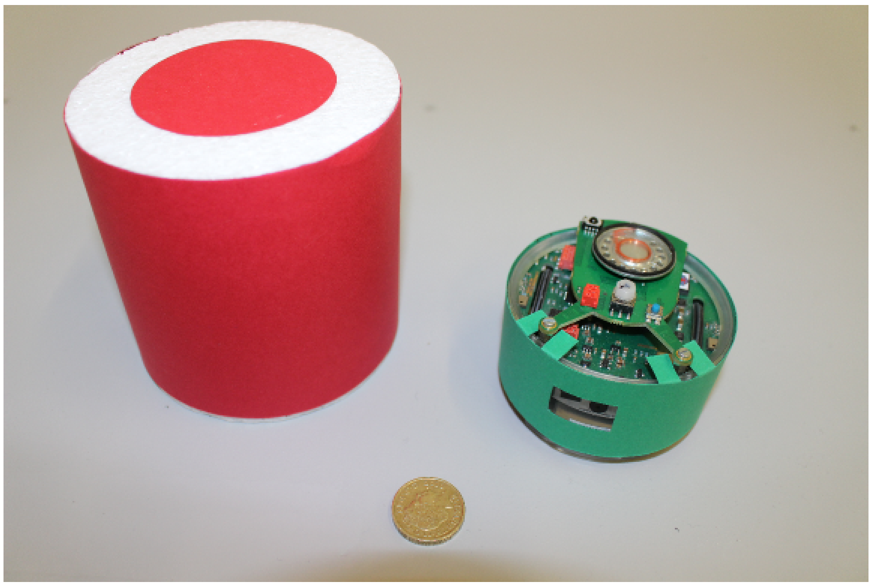


Figure 5.2.: Left: A cylindrical object made of expanded polystyrene (EPS). The object's diameter and height are both 10 cm. The object is wrapped in red paper in order to make it visually distinguishable to the robots. A red marker is also attached to its top, in order to facilitate the tracking of its position by an overhead camera. Right: An e-puck robot. The e-puck's diameter and height are approximately 7.4 cm and 5.5 cm, respectively. The e-puck is fitted with a green 'skirt' that makes it visually distinguishable to the other robots.

Equation 5.3 is designed to reward both a low dispersion at the end of the time interval, as well as the speed with which the robots cluster the objects. It penalizes large values of $u^{(t)}$ at every time instant (except for the very first one), but gives increasing importance to small values of $u^{(t)}$ for increasing values of t.

5.1.2. Objects and Robots

For the objects, we used cylinders made of expanded polystyrene (EPS), shown in Figure 5.2. These cylinders have a diameter and a height of 10 cm. We covered these objects with red paper, to allow the robots to visually distinguish them in the physical implementation (see Section 5.4.1). Moreover, we attached a red paper circle to the top of the objects, to allow for their tracking by an overhead camera during physical experiments. The mass of the objects, including the paper attachments, was measured to be approximately 35 g.

We determined the coefficient of static friction between the objects and the floor of our experimental arena by performing a ramp experiment, as follows. A sample of the arena floor (i.e. a slab made of the same type of wood, coated with the same type of paint) was aligned horizontally, and an object was placed on one end of it. The slab was then gradually lifted from the end at which the object was sitting, hinging it on the opposite end. The angle at which the object began to slide down the ramp was measured, and from this, the coefficient of static friction between the objects and floor of the experimental arena was determined to be 0.58.

The robotic platform used is the same as in our previous case study, namely, the e-puck. Recall that the e-puck is a miniature, differential wheeled mobile robot with an inter-wheel distance of around 5.2 cm. Its diameter and height are approximately 7.4 cm and 5.5 cm, respectively, and its weight is approximately 150 g.

The e-puck is equipped with a directional camera located at its front, which has been used in this study to realize the line-of-sight sensor in the physical implementation (see Section 5.4.1). The e-puck's processor is a Microchip dsPIC micro-controller with 8 KB of RAM and 144 KB of flash memory.

5.1.3. Simulation Platform

As in the case of self-organized robot aggregation, the simulations presented here were performed using the open-source Enki library [186]. Recall that in Enki, the body of an e-puck is modeled as a disk of diameter 7.4 cm and mass 152 g. The inter-wheel distance is 5.1 cm. The velocities of the left and right wheels along the ground can be set independently in $[-12.8, 12.8]$ cm/s. The objects were modeled as disks of diameter 10 cm, mass 35 g, and a coefficient of friction with the ground of 0.58. The line-of-sight sensor of the robots was realized by projecting a line from the robot's front and checking whether it intersects with the body of an object or another robot (and if it intersects with both, checking with which it intersects first). The length of the control cycle was set to 0.1 s, and the physics were updated at a rate of 10 times per control cycle (i.e. 100 times per second).

5.1.4. Evolutionary Algorithm

Although the controller is similar in structure to the one we had for the previous case study, it now has 6 parameters instead of 4. If we were to use a grid search with the same resolution as before (i.e. 21 settings per parameter) for locating the optimal controller, this would require $21^2 = 441$ times as many performance evaluations. This option was not viable given our computational resources.

We therefore resorted to using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [193] in order to optimize the performance measure defined by Equation 5.3. Note that this is the same algorithm that we used when synthesizing controllers with (and, for the sake of fairness, without) memory when we considered a controller variation in Section 4.4. Recall that in CMA-ES, the user only needs to specify three exogenous parameters: a starting point for the decision vector, $\mathbf{m}^{(0)}$; the initial step size, $\sigma^{(0)}$; and the population size, λ. Below, we will outline the details of the algorithm that are specific to our implementation.

5.1.4.1. Constraint Handling

In its standard form, CMA-ES operates across the entire real space, $\mathbb{R}^d$; however, in out case, the decision variables need to be constrained within the interval $[-1, 1]$, as explained in Section 5.1.1.2. In order to achieve this, we opted to let CMA-ES operate in its unconstrained form; however, before evaluating the performance of a candidate solution, we mapped each of its four parameters onto the feasible region by using the logistic sigmoid function (see Equation 4.3), as:

$$\bar{v} = 2\,\mathrm{sigmoid}(v) - 1, \tag{5.4}$$

where v is the "raw" value in the algorithm's candidate solution, and $\bar{v}$ is in $(-1, 1)$ and can therefore be used as a normalized wheel speed.

5.1.4.2. Parameter Settings

We set the starting point for the decision vector, $\mathbf{m}^{(0)}$, to the zero vector, and the initial step size, $\sigma^{(0)}$, to 0.72. Monte Carlo simulations show that with this setting, the initial population will be approximately uniformly distributed in $[-1, 1]^d$, when it is mapped onto this region by the logistic sigmoid function (Equation 4.3).

For the population size, we use $\lambda = 10$, based on the 'default' setting suggested in [195, 193] of $\lambda = \lfloor 4 + 3 \ln d \rfloor$ (here, $d = 6$).

5.2. Controller Synthesis

We wish to study the importance of the robots being able to detect the presence of other robots, and distinguish them from the objects in the environment. For this reason, we performed 3 sets of evolutions, with 25 evolutions per set, using the following settings:

1. In the first set of evolutions, the robots' sensors work as described in Section 5.1.1.1, i.e. they can distinguish between pointing at the background of the environment, an object, or another robot. In this case, the controller has six parameters, as given in Equation 5.1. We term this setting *Robots as Robots*.

2. In the second set of evolutions, the robots are unable to detect the presence of other robots. Therefore, when their line-of-sight sensor is pointing at another robot, it gives a reading of $I = 0$ rather than $I = 2$. Consequently, only the first four parameters of the controller of Equation 5.1 are utilized. We term this setting *Robots as Background*.

3. In the third set of evolutions, the robots are able to detect the presence of other robots; however, they are not able to distinguish them from the objects. In other words, their line-of-sight sensor returns $I = 0$ if it is pointing at the background of the environment, and $I = 1$ if it is pointing at an object or another robot. As in the previous case, only the first four parameters of the controller of Equation 5.1 are utilized. We term this setting *Robots as Objects*.

Each evolution was run for $G = 1000$ generations. Note that for the sake of fairness, all the evolutions operated in a 6-dimensional space (i.e. $d = 6$). In the evolutions for the cases *Robots as Background* and *Robots as Objects*, the last two parameters of the controller of Equation 5.1 were simply not used by the robots, and therefore had no impact on the selection of the candidate solutions.

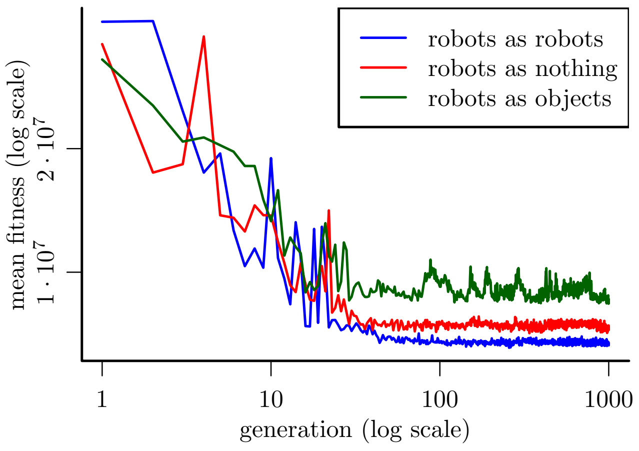


Figure 5.3.: Evolution Fitness Dynamics.

5.2.1. Evaluation of Candidate Solutions

In each generation, each of the $\lambda = 10$ candidate solutions (i.e. controllers) was evaluated by running it for 100 s on $n = 2$ robots in an environment containing $m = 5$ objects. The objects and the robots were initialized with a uniform distribution in a virtual square of sides 111.80 cm, such that on average, the area per object was 2500 cm^2. Additionally, the initial orientation of each robot was chosen randomly in $[-\pi, \pi]$. Each candidate solution was evaluated 10 times with different initial configurations of the objects and the robots, and the mean value of U (see Equation 5.3) across these 10 runs was assigned as its fitness measure. Note that the set of initial configurations was identical for each candidate solution within each generation, but a new set of was chosen in every new generation.

5.2.2. Post-Evaluations

The selection of the controller for the three cases *Robots as Robots*, *Robots as Background* and *Robots as Objects*, was performed as follows. Each of the controllers in the last generation of the respective 25 evolutions was post-evaluated using the same mechanism used within the evolutions, with the difference that each controller was evaluated 100 times (rather than 10 times) with different initial object/robot configurations. The controller with the highest mean fitness (see Equation 5.3) across the 100 runs was selected to be used in the experiments presented in the rest of this chapter.

Figure 5.3 shows the fitness dynamics of the three 'best' evolutions, i.e. the ones that led to the best controllers. For this plot, the controllers generated by these three evolutions in each generation were post-evaluated in the same way as the controllers of the last generation. Figure 5.4 shows the dynamics of the parameters in these evolutions (the parameters $\bar{v}_{\ell,2}$ and $\bar{v}_{r,2}$ are not shown in Figs. 5.4b and 5.4c, as they are irrelevant). Note that in each of the three cases, the dynamics of the evolutions have reached steady state well before the last generation.

The evolution with *Robots as Robots* led to the best fitness at the last generation (see Figure 5.3), followed by *Robots as Background* and *Robots as Objects*. From Figure 5.4, we see that the first four parameters of the controller (see Equation 5.1) in the last generations are very similar across the three evolutions.

5.3. Simulation Experiments

5.3.1. Emergent Behaviors

It turns out that each of the three controllers leads, in a qualitative sense, to the same emergent behavior. Figure 5.5 shows snapshots of 50 robots and 20 objects over 1000 seconds where the robots employ the *Robots as Robots* controller (these settings are scaled up by a factor of 10 from the ones used within the evolutions). The robots first find their way to the periphery of (most of) the objects. Following this, they move in a circular formation around the objects, with each robot pushing each object slightly inwards on each contact. Sometimes, some objects are initially not included within the robots' circle, but at some point, the robots branch out from their circular formation to encircle these stray objects as well.

This emergent behavior is robust, as we have observed with many different parameter settings (e.g. number of objects/robots, initial dispersion, and initial configuration). The robots always manage to find their way to the periphery of most of the objects, and after some time, most of the objects end up in one cluster (although sometimes, some objects that are initially very far from the rest are missed, potentially due to the discrete nature of the sensor/controller cycle). Interestingly, the behaviour also works if there is only one robot in the environment.

5.3.2. Scalability Study

In this section, we study the effect of the number of robots relative to the number of objects on the clustering performance, for each of the three controllers.

We ran simulations using $m = 50$ objects and lasting for 1000 seconds. With each controller, we ran 100 simulation with each number of robots in the set $n = \{10, 20, \ldots, 50\}$, and for each simulation, we recorded the fitness measure given by Equation 5.3. Figure 5.6 shows a box plot[1] of the results, where the blue, red, and green boxes correspond to the *Robots as Robots*, *Robots as Background* and *Robots as Objects* controllers, respectively. The dotted line indicates the expected value of the *baseline fitness measure*, i.e. the fitness measure if the robots do not move throughout the run (found by performing a Monte Carlo simulation with 100 evaluations; note that this does not depend on the number of robots).

With the *Robots as Objects* controller, the performance degrades as the number of robots is increased, until with 40 robots it is not significantly different from the baseline (paired Wilcoxon signed-ranked test, $p < 0.02$). With the *Robots as Robots* and *Robots as Background* controllers, the performance has a bowl-shaped profile with respect to the number of robots, with the optimum occurring at $n = 30$ (with the resolution used). However, for each value of n, the performance of the *Robots as Robots* controller is significantly better than that of the *Robots as Background* controller.

5.3.3. Noise Study

In this section, we investigate the effect of sensory noise using the *Robots as Robots* controller. We chose this controller as it is the best-performing one, as shown in the previous section, and as we also use it in our physical experiments (see Section 5.4).

We use the following sensory noise model, which we have found to be realistic from experiments with our physical setup. The sensor always gives the correct reading of $I = 0$ when it is pointing towards the background (in our physical setup, the boundaries

[1]The box plots presented here are all as follows. The line inside the box represents the median of the data. The edges of the box represent the lower and the upper quartiles (25-th and 75-th percentiles) of the data, and the whiskers represent the lowest and the highest data points that are within 1.5 times the inter-quartile range from the lower and the upper quartiles, respectively. Circles represent outliers.

of the arena). When the sensor is pointing at either an object or another robot, its reading is corrupted with some probability p. In this case, with equal probability, the sensor either misses the object or robot, giving a reading of $I = 0$, or registers the wrong type of item (i.e. a robot if there is actually an object, and vice-versa).

We performed simulations with $p \in \{0, 0.1, \ldots, 1\}$. For each value of p, we performed 100 runs with 50 objects and 20 robots, with each run lasting for 1000 seconds. In each run, we recorded the fitness measure given by Equation 5.3. Figure 5.7 shows a box plot of the results, where the dotted line indicates the expected value of the *baseline fitness measure*, i.e. the fitness measure if the robots do not move throughout the run (found by performing a Monte Carlo simulation with 100 evaluations). With $p = 0.1$, all of the 100 runs achieve a fitness that is better than the baseline. With up to $p = 0.4$, the 75-th percentile of the fitness is still better than the baseline. With $p = 0.5$, the performance is still significantly better than the baseline (paired Wilcoxon signed-rank test, $p < 0.02$). Interestingly, with $p = 0.6$ and 0.7, the fitness is significantly worse than the baseline, meaning that the robots *disperse* the objects more than they were at the start. With $p = 0.8$ and $p = 0.9$, the fitness is still statistically significantly different from the baseline, but the difference is minimal. With $p = 1$, the fitness is not significantly different from the baseline.

5.4. Physical Experiments

5.4.1. Sensor and Controller Implementation

The sensor was implemented using the e-puck's directional camera. For this reason, the objects were wrapped with red paper, and the robots were fitted with green 'skirts', in order to make them distinguishable from each other and from the white walls of the arena (see Figure 5.2). Recall that the e-puck's camera is a CMOS RGB color camera with a resolution of 640 (horizontal) by 480 (vertical) pixels, with corresponding viewing angles of 56° and 42°, respectively.

In principle, using one pixel of the camera is sufficient for implementing the line-of-sight sensor. However, in order to account for misalignments among the robots' cameras, as well as the presence of noise in a real-world environment, we used the following method in order to achieve a more robust implementation of the sensor. Firstly, the image from the camera is sub-sampled to obtain a 40×15 pixel image, spanning the whole of the original image. Then, a 10×10 pixel window from center of this sub-sampled image is used to implement the sensor, as follows. Each pixel is compared against threshold to decide whether it is white, red or green. If there are less than 2 green pixels, and less than 2 red pixels, then the sensor gives a reading of $I = 0$, indicating that it is pointing at a wall. Otherwise, the majority of the colored pixels decides whether the robot gives a reading of $I = 1$ or $I = 2$, indicating that it is pointing at a red object or a green robot, respectively (if the number of green and red pixels is equal, green is given precedence). The implemented sensor has been found to provide reliable readings for up to a range of around 150 cm (with some misperceptions).

5.4.2. Experimental Setup and Procedure

The arena used for the experiments is a rectangle of size 400 cm × 225 cm. It has a light gray floor, and is surrounded by white walls that are 50 cm in height. Its floor is marked with a grid of $15 \times 8 = 120$ points, spaced 25 cm from each other and from the walls. For each trial, 25 of these points were chosen randomly to serve as the initial positions of the objects and the robots. With the objects and the robots positioned on these points, an infrared signal was issued to instruct each robot to turn on the spot through a randomly generated portion of a revolution, such that robots now faced in random directions. Another infrared signal was then issued to instruct the robots to start executing the controller. The robots were programmed to stop automatically after 600 s. We performed 10 trials with $m = 20$ objects and $n = 5$ robots. Each trial was recorded by an overhead camera. All the 10 videos are available in the online supplementary material [188].

5.4.3. Results

Figure 5.10 shows a sequence of snapshots from one of the trials. We observe the same emergent behavior as that in simulation (see Section 5.3.1), and in this particular trial, the robots have gathered the objects into one cluster after around 300 s. Figure 5.9 shows the final configurations (i.e. after 600 s) of the the objects and the robots in the 10 trials. On average, at the end of the trials, the largest cluster of objects[2] contains

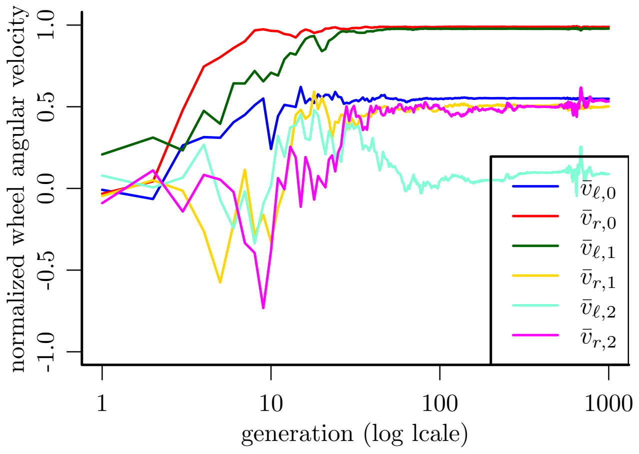
[2]A cluster is defined as a maximal connected subgraph of the graph defined by the objects' positions, where two objects are considered to be adjacent if another object cannot fit in between them (in other words, objects i and j are adjacent if $||\mathbf{p_j}^{(t)} - \mathbf{p_i}^{(t)}|| < 4r_o$).

86.5% of the objects.

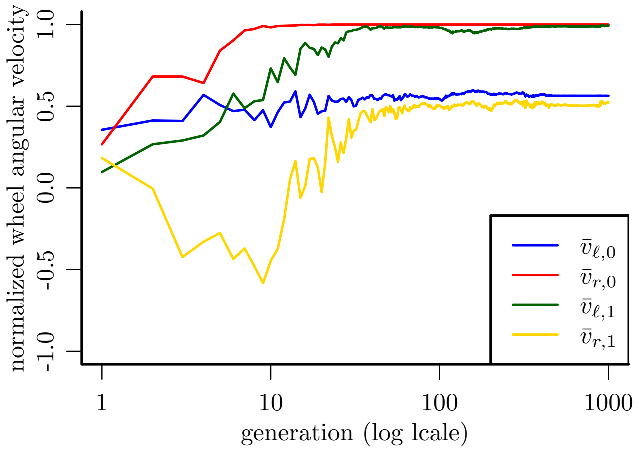
Figure 5.8 shows plots of the clustering dynamics. Figure 5.8a shows dynamics of the second moment of the objects (see Equation 5.2), while Figure 5.8b shows the dynamics of the proportion of objects in the largest cluster. In both plots, the different colored curves correspond to the 10 individual trials, while the dashed black curve represents the mean measure across the 10 trials. In Figure 5.8a, the horizontal dotted black line shows the theoretical lower bound of the second moment for 20 objects, as given in [182].

5.5. Summary

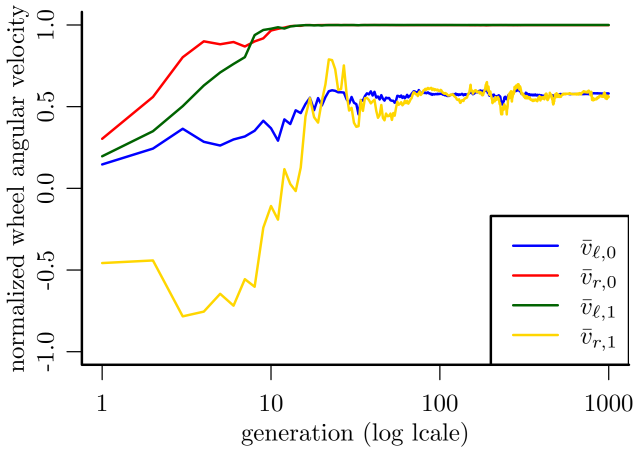
This chapter has presented the simplest solution so far to the problem of clustering objects with a swarm of robots. The robots are memory-less and are unable to perform arithmetic computations. They are only able to detect the presence of an object or other robot is in their line of sight. As they do not possess any distance information about a perceived object (nor any force sensors), they are unable to tell whether they are manipulating it. Despite these limitations, we identified a controller that is capable of consistently gathering the objects into a single cluster. Simulation results have shown that the ability of the robots to distinguish between objects and other robots is beneficial to the task; indeed, if the robots can only perceive other robots as objects, the behavior does not scale well with increasing numbers of robots. Simulations have also shown that the controller is fairly robust with respect to sensory noise. The sensor/controller solution was implemented on a physical system of 5 e-puck robots, and favorable results have been obtained in systematic experiments with 20 objects, with 86.5% of the objects being gathered into one cluster after 10 minutes (average across 10 trials). In the future, we intend to implement a solution of the object clustering task with robots at the sub-millimeter scale.



(a) Robots as Robots



(b) Robots as Background



(c) Robots as Objects

Figure 5.4.: Evolution Parameter Dynamics.

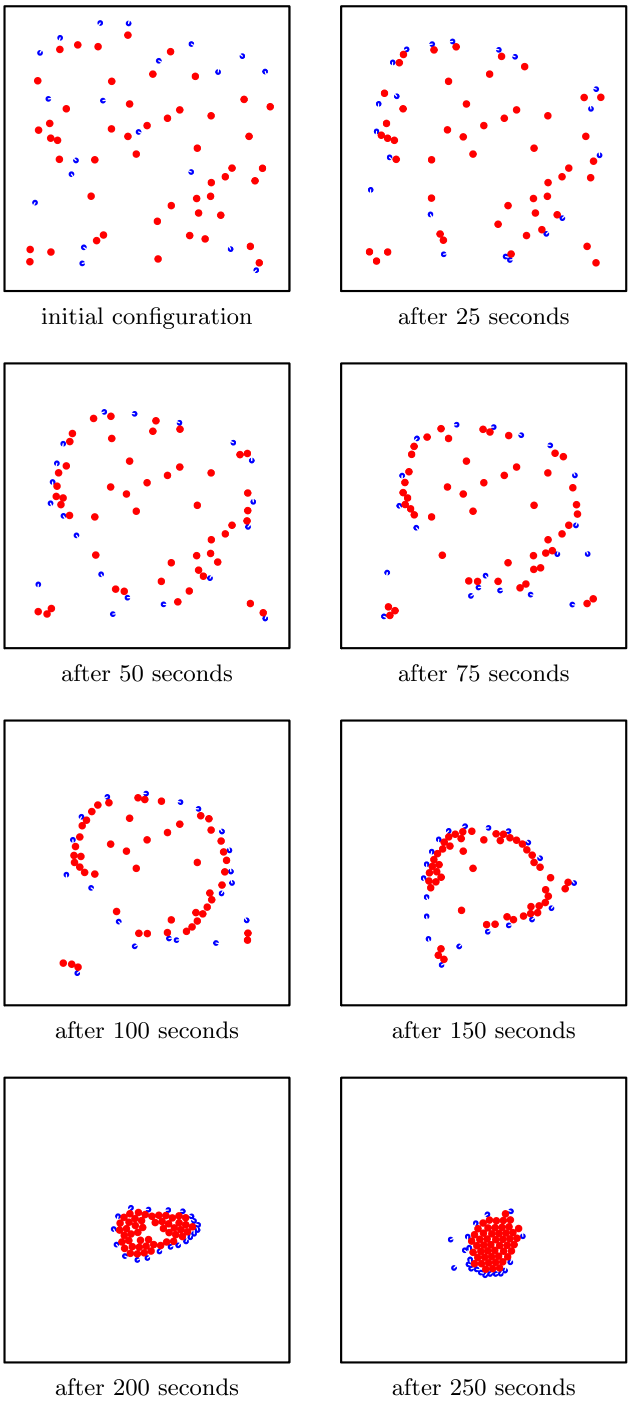


Figure 5.5.: Emergent Behavior. The robots first find their way to the periphery of (most of) the objects. Following this, they move in a circular formation around the objects, with each robot pushing each object slightly inwards on each contact.

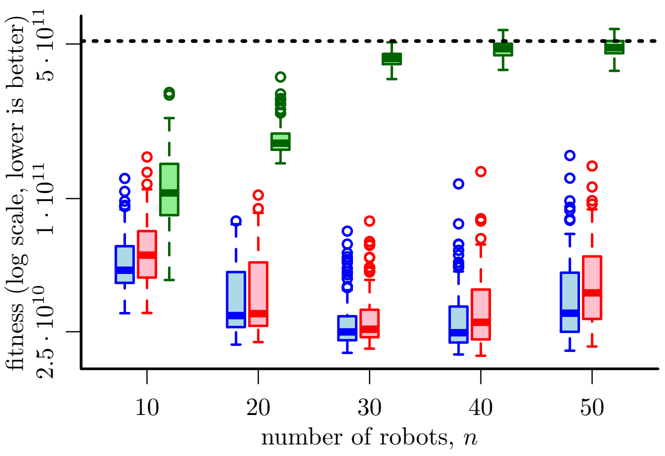


Figure 5.6.: Scalability Study. The blue, red and green boxes correspond to the *Robots As Robots*, *Robots As Background*, and *Robots As Objects* controllers, respectively. The dotted black line shows the mean of the baseline fitness measure (i.e. the fitness if robots do not move throughout the run).

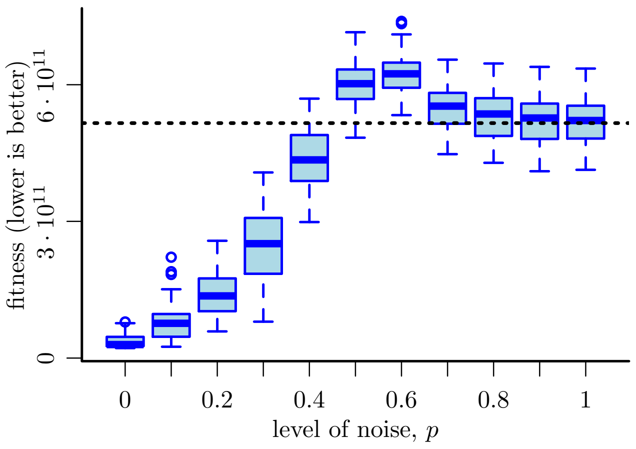
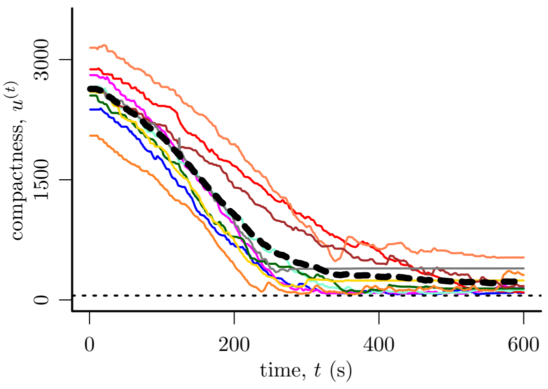
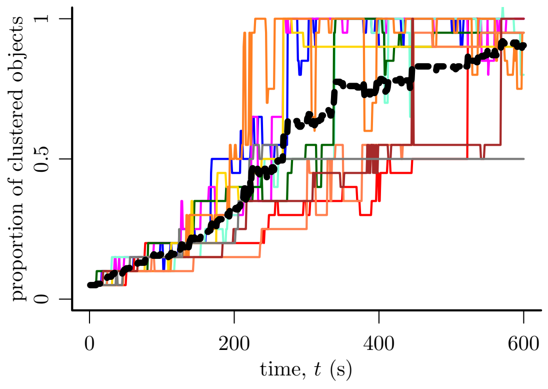


Figure 5.7.: Noise Study. The dotted black line shows the mean of the baseline fitness measure (i.e. the fitness if robots do not move throughout the run).



(a)



(b)

Figure 5.8.: Physical Dynamics.

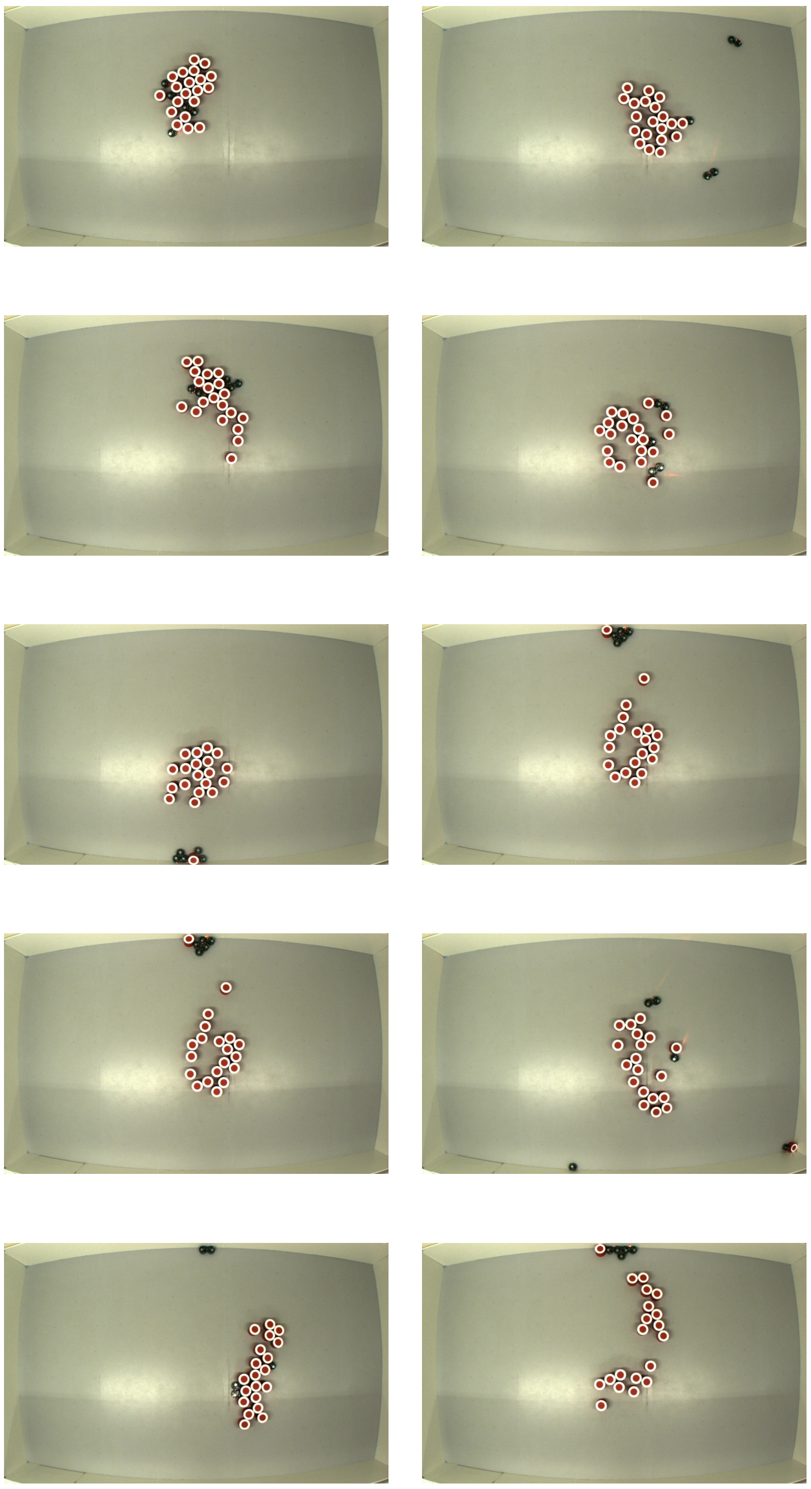


Figure 5.9.: The Final Configurations of the Objects and the Robots in the **10** Physical Trials.

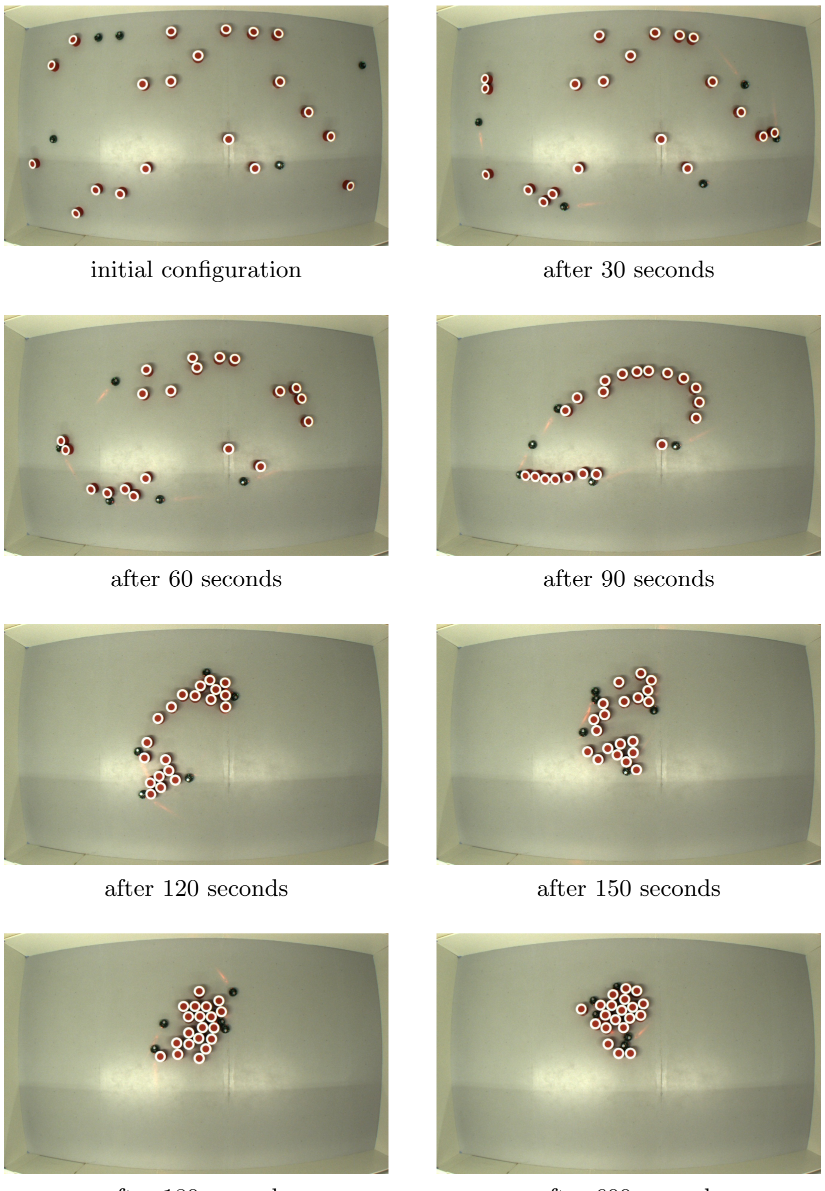


Figure 5.10.: Snapshots From a Physical Trial at Several Time Instances.

6. Conclusion

This thesis has presented a framework for synthesizing meaningful collective behaviors in swarm robotic systems with minimal information acquisition and processing. This framework was successfully applied to two swarm robotic tasks: self-organized aggregation of robots (Chapters 3 and 4), and self-organized clustering of objects by robots (Chapter 5). It was shown that both tasks can be successfully achieved even if the robots do not possess run-time memory and do not perform any arithmetic computations. For both tasks, the developed sensor/controller solutions were seen through from synthesis in simulation, to mathematical and/or computational analysis, and implementation and systematic experimentation on a swarm of physical e-puck robots.

In the case of aggregation, we investigated how the number of robots used in synthesizing the controller influences the scalability of the resulting, collective behaviors. We also compared and contrasted the performance of our minimalistic sensor/controller solution with one that makes use of global knowledge of the environment. It turned out that there is a significant performance trade-off when using the minimalistic solution; however, this may be acceptable — or even necessary — at small scales where more elaborate solutions are harder to implement. We explored the possibility of varying the sensor configuration from a line-of-sight sensor to one with a finite field of view. This analysis concluded that, overall, the resulting gain in performance is relatively small, especially if during the solution synthesis stage, one does not know the exact conditions in which the final system will operate, and cannot therefore optimize the sensor's field of view. Moreover, we explored the introduction of a small amount of memory and computation into the controller, in the form of a neural network with recurrent connections. The result was that, while this provides a marginal improvement in performance when evaluated on the same number of robots that was used in the controllers' synthesis, the more elaborate controller performs worse on larger numbers of robots than the minimalistic one. This is because the more elaborate controller has more potential to specialize to a given number of robots. Together, the results from the sensor and controller variations lend evidence to a *less is more* design philosophy for swarm robotic systems. After all, a major motivation for these systems is that they shall be flexible enough to operate in environmental conditions that are not well-defined *a priori*.

In the case of object clustering, we investigated how the performance of the system benefits from an additional state of the line-of-sight sensor. We compared the performance of three different sensor configurations: in the first, *Robots as Robots*, a ternary sensor could distinguish between robots, objects and the background; in the second, *Robots as Objects*, a binary sensor could identify the presence of robots *or* objects against the background, but not distinguish between robots and objects; in the third, *Robots as Background*, a binary sensor could *only* identify the presence of objects against the background, but not the presence of robots. It turned out that all of these configurations deliver a satisfactory performance when the robots-to-objects ratio is sufficiently small. When this ratio increases, the *robots as objects* sensor configuration breaks down because the robots attempt to cluster themselves as well as the objects. On the other hand, both the *Robots as Robots* and *Robots as Background* configurations continue to perform successfully, implying that the third sensor state is not strictly required. At the same time, however, it was statistically shown this third state does lead to a significant improvement in performance, in terms of the speed of clustering. This suggests that even at the minimal extreme of information acquisition and processing, the robots in a swarm can successfully co-operate to better effect than if they had to act individually. In other words, this means that at physically small scales, it should still be possible to exploit one of the main powers of swarm systems: that the whole can be more than the sum of the parts.

We believe that these results are meaningful within a broader context, and will contribute in paving the way to the implementation of massively-distributed robotic systems at small scales, which in the future may potentially be composed of millions of robots [116, 117]. In the case of aggregation, for instance, the dense packings that have been achieved are desirable as a pre-condition if the robots are required to form and operate as a physically-connected ensemble [189, 15]. This can also help in blending the boundary between swarm robotic systems and smart materials [197]. On the other hand, the results in object clustering are reminiscent of the future potential for small-scale swarm robotic systems to act within the human body, for example to assist the immune systems in fighting pathogens.

6.1. Future Work

In spite of the encouraging results obtained in this work, we do not claim that the framework presented in this thesis, at least in its current form, is generally applicable to the design of swarm robotic systems at physically small scales. For instance, we certainly do not expect that tasks of arbitrary complexity can be achieved strictly without run-time memory and arithmetic computation on the part of the robots. Rather, we see the work in this thesis as one of the first steps towards easing the information acquisition and processing requirements for swarm robotic systems.

There are still several important questions to be answered: Will discrete-state sensors that do not provide explicit distance information still be effective in more complex environments, such as ones with obstacles, or in three-dimensions? How would the collective behaviors demonstrated in this thesis be affected if they were to be synthesized and/or implemented on swarms of robots with different physical shapes [190]? How far can we increase the task complexity before run-time memory and arithmetic computation on the part of the robots become an unavoidable necessity? In complex tasks that cannot be solved without run-time memory and computation, how do we go about systematically and incrementally adding complexity into the robots? Can we still ensure that a given task is achieved with the minimum possible amount of information acquisition and processing? How can different behaviors be integrated into the same swarm robotic system in order to tackle different tasks? How can the robots know when a task has been achieved? In more complex environments, will the reality gap [198, 3] become an issue in successfully porting sensor/controller solutions that have been synthesized in computer simulation onto physical swarm robotic systems, and still achieving the desired collective behaviors? How could this problem be addressed if it arises?

The practical implementation of robotic systems at small scales is also in itself is a subject that will require much research [199]. How can these robots sense their environment, and act and move within it in an 'intelligent' manner? This is in many ways a problem of reverse-engineering. Much like the way in which Brooks [46] used the artificial flight parable to illustrate that artificial intelligence must be possible, a similar argument can be made for autonomous systems at small scales. We may observe many examples of such systems in nature. For instance, at the centimeter down to the millimeter scales, we find social insect colonies. Further down, at the micro- down to the nano-scales, we may look at the immune systems that operate within the bodies of complex organisms. For this reason, it is expected that the field of biomimetics will have much to offer in terms of inspiration for the implementation of artificial, small-scale autonomous systems [200].

The questions above are all high-level questions that may be approached in different ways. In the following sections, we will briefly discuss a selection of concrete ideas for future research projects based on the work presented in this thesis. We believe that each of these projects may help to shed light on one or more of the general research questions presented above, and perhaps others that we have failed to explicitly identify.

6.1.1. Quantifying Performance Trade-Offs

In Section 4.3, we compared our minimalistic sensor/controller solution for the problem of self-organized robot aggregation, with a solution that assumes global knowledge of the environment. This analysis, however, only looked at the extreme cases, and it would be useful to quantitatively investigate what happens at intermediate levels of information acquisition and processing.

As we have seen in Section 2.4, a wide range of solutions has been presented in the literature for the self-organized robot aggregation problem, with various degrees of complexity. Small scales may preclude some of these solutions because of their nature (e.g. sensors that are not possible to implement or integrate), but in general, a subset of solutions may be possible depending on the scale. The main criteria for choosing one of these solutions above the others may then become their level of performance (e.g. speed of task execution) and their impact on the robot's energy consumption.

One possible way to investigate this would be to consider a 2-dimensional evaluation space, where one axis represents the expected performance, such as the time to complete a task, and the other axis represents the computational power required to complete this task, such as the number of floating point operations (FLOPS). Each of the solutions in the subset would be evaluated, and the result would represent a point in the 2-dimensional evaluation space. In this case, note that the minimalistic solution presented in this thesis would, in principle, have a measure of zero on the computational power axis; however, in practice, some computation may be required for instance to deduce the binary reading of the sensor from the raw image returned by a camera, as we described in Section 3.7.1.

Another, more direct method for comparing different candidate solutions in the case where the robots have a finite energy storage would be as follows. Each solution is implemented on a swarm of robots, all of which start with a full energy reserve. Repeated trials are then performed, with each trial being stopped at the point where the task is considered to have been successfully completed, according to some pre-defined criterion. This process continues until the swarm of robots is not able to successfully complete a trial due to energy depletion. Statistical analysis could then be used to analyze the levels of performance of the different solutions as the robots' energy reserves deplete.

6.1.2. More Complex Environments

The two swarm robotic tasks that have been addressed in this thesis have only been investigated in very simple environments. Two aspects that would make an environment more complex readily suggest themselves: the presence of obstacles, and three-dimensional space.

Obstacles could be introduced into the environment in the form of fixed or dynamic objects. It could then be investigated whether successful solutions can still be synthesized in the presence of these obstacles. In particular, it would be interesting to analyze how the performance is affected as the density of obstacles is increased. It would also be informative to ask whether the addition of an extra sensor state to recognize obstacles would improve the performance.

Three-dimensional space would certainly be challenging for the line-of-sight sensors used in this thesis. This is because, whereas in 2 dimensions, a line-of-sight sensor can provide coverage of the whole environment as a robot rotates on the spot, 3-dimensional space would introduce the problem of 3 rotational axes for the robots. Would this make the information provided by a line-of-sight sensor too sparse for achieving meaningful collective behaviors? Could this issue be addressed by using a wider field-of-view for the sensors (e.g. a 3-dimensional cone)?

6.1.3. More Complex Tasks

As we have mentioned before, one of the main questions that arise from the work in this thesis is whether, and how the developed framework could be extended to more complex tasks. Here, we have started with self-organized robot aggregation, and then proceeded to investigate the relatively more complex task of self-organized object clustering.

Both these problems could be made more challenging by converting them into segregation problems. In the first case, the swarm would consist of k groups of robots, with each group consisting of a potentially different number of robots, N_k. These robots would start out distributed randomly across the environment, and the problem would be for them to separate themselves into the different groups. The task completion criterion could be, for example, when each pair of groups of robots are arranged in a linearly separable configuration (i.e. a straight line can be drawn that divides the space into two regions, each of which only contains one type of robots). This completion criterion immediately suggests a performance metric for driving the controller synthesis, based on some measure of the discrepancy between the actual configuration of the robots and the ideal one. This could be, for instance, the *minimum* number of misclassifications that can be made by separating each pair of groups of robots with a straight line. A time-weighted version of this measure could then be used as the overall performance evaluation, analogously to what we did, for example, in Section 3.4. For this problem, the robots could have a ternary sensor, which is able to distinguish between the background, robots from the same group, and robots from other groups.

In the case of object segregation, the problem would be for a swarm of robots to sort out a number of objects into different groups. The same completion criterion and performance metrics could be used as for the case of robot segregation, only this time, applied to the the objects, rather then robots. However, an interesting distinction arises in this case in terms of the choice of the robots' sensors and controllers. The swarm could be either homogeneous or heterogeneous:

- In the homogeneous case, the robots would have identical sensors and controllers. The sensor could provide $k + 2$ values: one for each of the k groups of objects, one for other robots, and one for the background. As a result, the controller would need to be synthesized with the number of groups of objects (i.e. k) in mind.
- In the heterogeneous case, the robots would themselves be divided into k groups. Each group of robots would be matched with one of the groups of objects. The robots' sensor could provide 4 values: one for objects from its corresponding group, one for objects from other groups, one for other robots, and one for the background. This this time, the structure of the controller is independent of k. Note that all the robots employ the same controller; the only difference is that the sensors of robots in different groups are matched to different groups of objects.

Each of these two setups comes with its associated potential advantages and disadvantages, which would open up interesting research questions. In the homogeneous case, the system might be more robust, because each robot that fails does not affect one specific group of objects. On the other hand, it is possible that the homogeneous configuration would break down with increasing k, as the 'space' of different behavioral patterns in the robots to react to different types of objects becomes too 'cluttered'. The heterogeneous configuration has the potential advantage that the same controller could scale up to larger numbers of groups of objects, without the need for re-synthesis. On the other hand, it would introduce the question of how precisely the numbers in each group of robots must be matched to the numbers of objects in the corresponding group.

Appendices

A. Differential Wheeled Robot Kinematics

This appendix provides a derivation of the forward kinematics of a differential wheeled robots, which were stated in Chapter 3, Equation 3.5. For a comprehensive treatment of differential wheeled robot locomotion, see [1, Section 3.1.5].

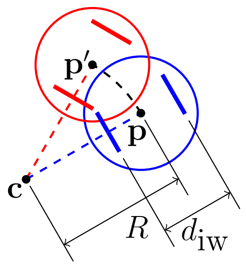


Figure A.1.: Differential wheeled robot kinematics. Over a short time interval δ_t, the robot moves along a circular trajectory from position $\mathbf{p}$ (blue) to position $\mathbf{p}'$ (red), sweeping an angle δ_ω. R and $\mathbf{c}$ denote the radius and the center of curvature, respectively, and d_{iw} denotes the robot's inter-wheel distance.

In general, the speeds of the left and right wheel are functions of time: $v_\ell^{(t)}$ and $v_r^{(t)}$. Let these be approximated by constant values, v_ℓ and v_r, over a short time interval δ_t. During δ_t, the robot therefore moves along a circular trajectory of some constant radius R, with a fixed angular velocity ω.

Consider the path traced by the point of contact of the left wheel with the ground during this time interval. This is approximated by (a) a straight line of length $v_\ell \delta_t$, and (b) an arc of radius $R - \frac{d_{\mathrm{iw}}}{2}$ and angle $\delta_\omega = \omega \delta_t$, whose length is $\left(R - \frac{d_{\mathrm{iw}}}{2}\right) \omega \delta_t$. By letting $\delta_t \to 0$, we can therefore write:

$$v_\ell^{(t)} = \left(R^{(t)} - \frac{d_{\mathrm{iw}}}{2}\right) \omega^{(t)}, \tag{A.1}$$

where we have denoted the time-dependent quantities explicitly. A similar argument for the right wheel shows that:

$$v_r^{(t)} = \left(R^{(t)} + \frac{d_{\text{iw}}}{2}\right)\omega^{(t)}. \tag{A.2}$$

By dividing Equation A.1 by Equation A.2, we get:

$$R^{(t)} = \frac{d_{\text{iw}}}{2}\left(\frac{v_r^{(t)} + v_\ell^{(t)}}{v_r^{(t)} - v_\ell^{(t)}}\right), \tag{A.3}$$

and by subtracting Equation A.1 from Equation A.2, we get:

$$\omega^{(t)} = \frac{1}{d_{\text{iw}}}\left(v_r^{(t)} - v_\ell^{(t)}\right). \tag{A.4}$$

B. Aggregation Performance Landscapes

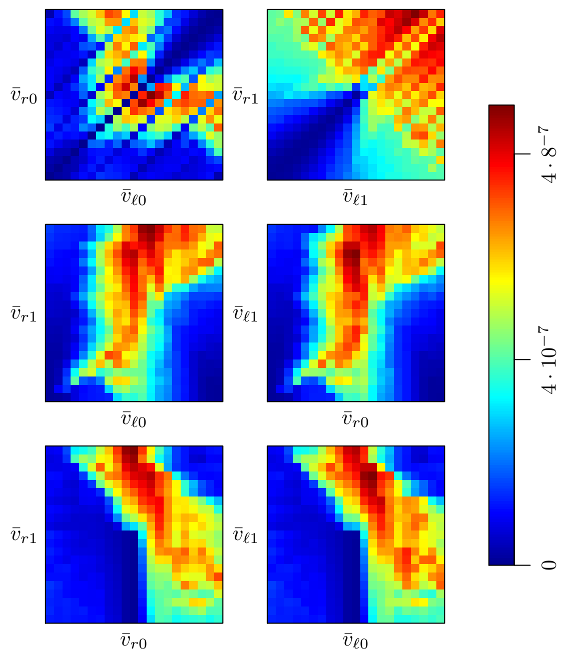


Figure B.1.: Performance landscape with $n = 2$ robots.

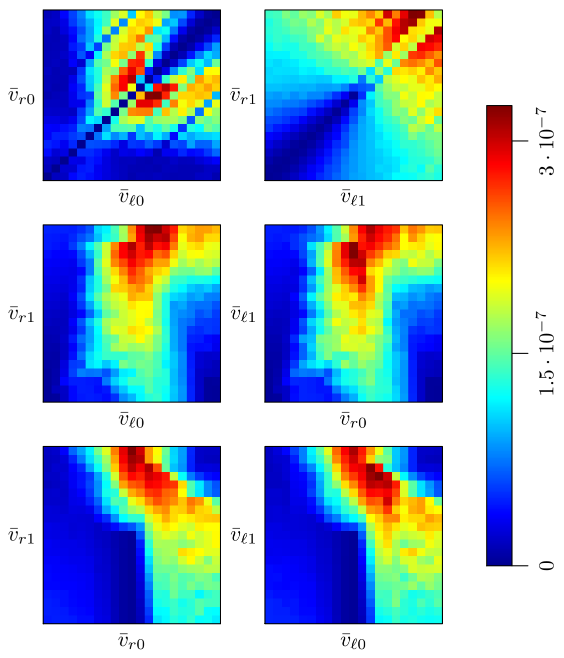


Figure B.2.: Performance landscape with $n = 3$ robots.

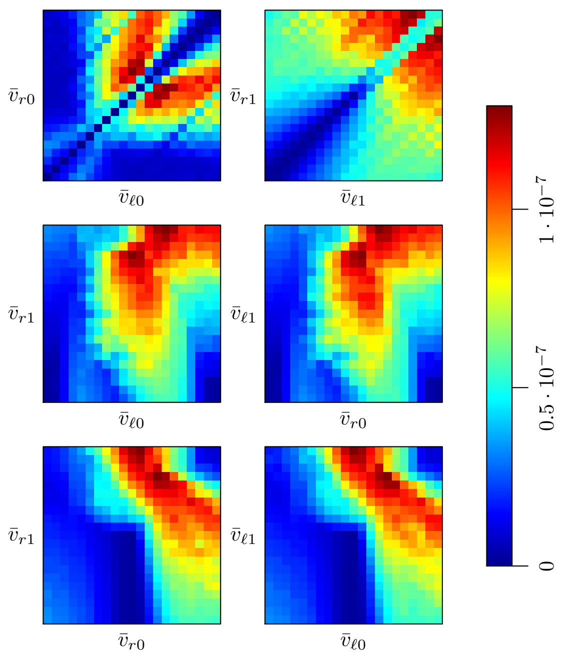


Figure B.3.: Performance landscape with $n = 4$ robots.

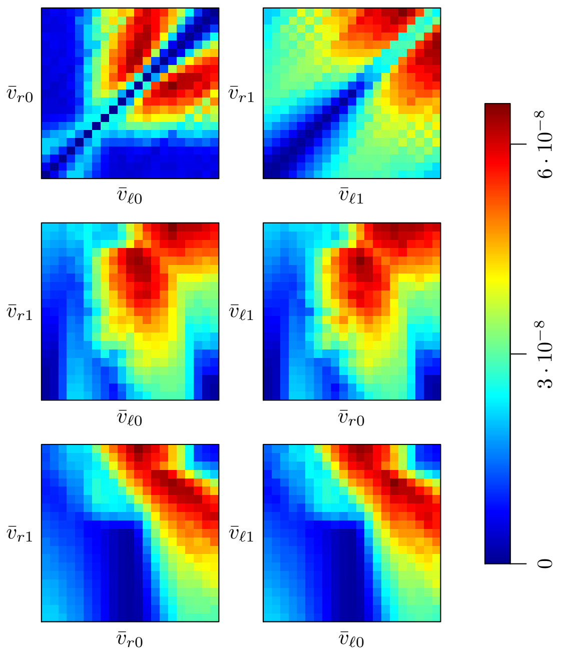


Figure B.4.: Performance landscape with $n = 5$ robots.

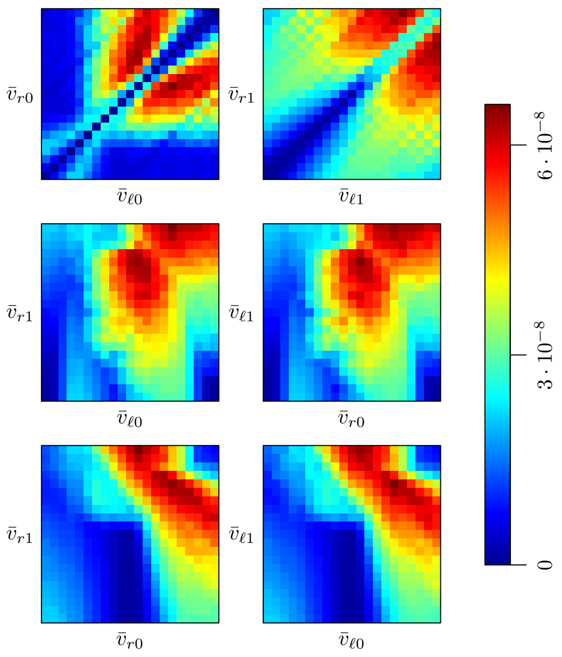


Figure B.5.: Performance landscape with $n = 6$ robots.

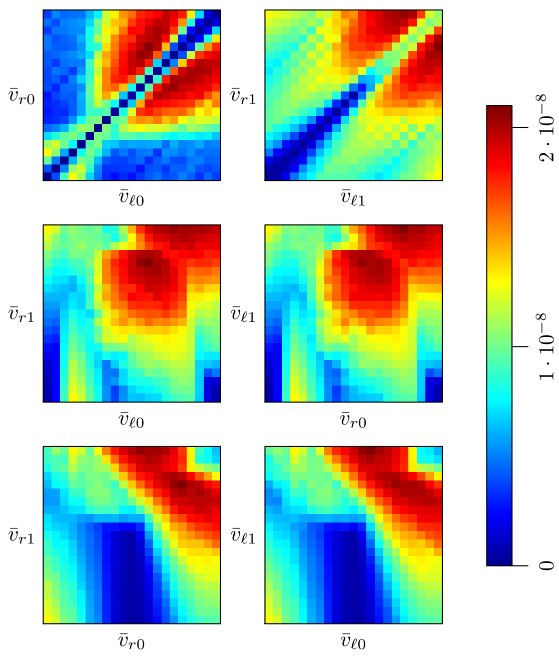


Figure B.6.: Performance landscape with $n = 7$ robots.

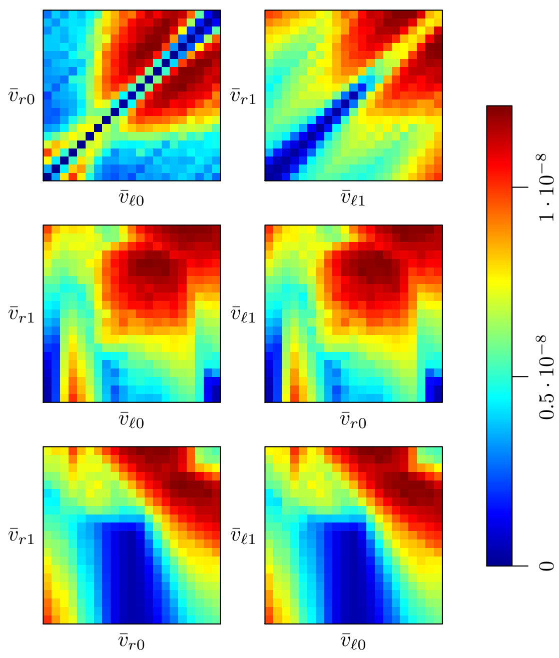


Figure B.7.: Performance landscape with $n = 8$ robots.

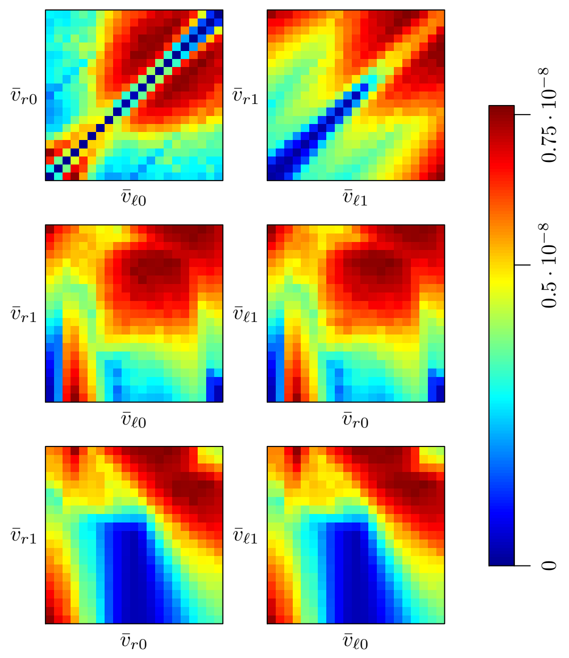


Figure B.8.: Performance landscape with $n = 9$ robots.

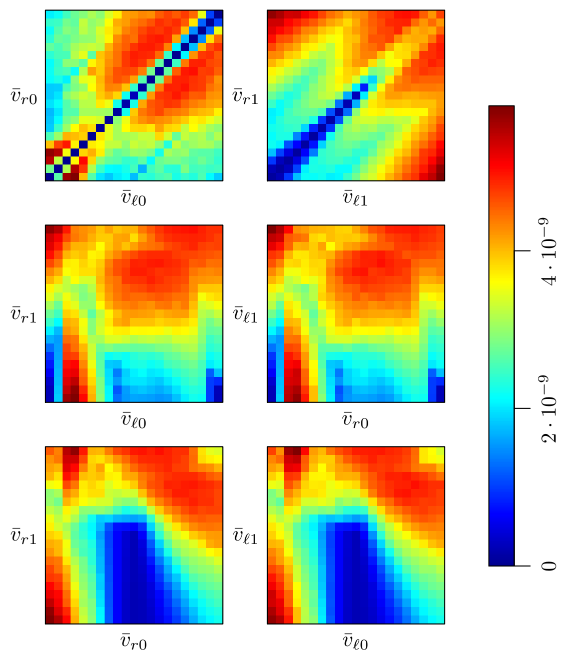


Figure B.9.: Performance landscape with $n = 10$ robots.

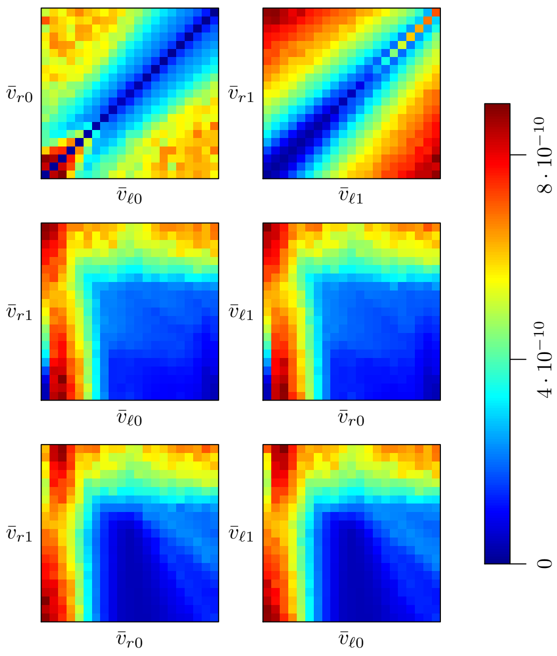


Figure B.10.: Performance landscape with $n = 25$ robots.

C. Aggregation Source Code Listing

This appendix provides the complete source code listing used for implementating the aggregation sensor and controller on the physical e-puck robotic platform (see Section 3.7). Functions who name starts with "e_" are ones provided by the e-puck software library [201]. The datasheet for the e-puck's camera is available at [202].

```
// e-puck Library Headers
#include <motor_led/e_epuck_ports.h>
#include <motor_led/e_init_port.h>
#include <motor_led/e_led.h>
#include <motor_led/e_motors.h>
#include <motor_led/advance_one_timer/e_agenda.h>
#include <motor_led/advance_one_timer/e_remote_control.h>
#include <camera/fast_2_timer/e_poxxxx.h>
#include <camera/fast_2_timer/e_po6030k.h>

const int max_speed = 1000;

// Controller Parameters
const double left_speed_0  = -0.7;
const double right_speed_0 = -1.0;
const double left_speed_1  =  1.0;
const double right_speed_1 = -1.0;

void control_step();
void increment_counter();

int Main()
{
    e_init_port();
    e_init_motors();
    e_init_remote_control();

    e_poxxxx_init_cam();
    e_poxxxx_config_cam(0, 0, 640, 480, 16, 16, GREY_SCALE_MODE);

//  Set Auto White Balance (AWB) and Auto Exposure (AE) off
//  (see p. 44 and p. 73 of the datasheet).
    e_po6030k_write_register(BANK_C, 0x04, 0b10011110);
    e_po6030k_write_register(BANK_C, 0x55, 0x00);
    e_po6030k_write_register(BANK_C, 0x56, 0x00);

//  Configure manual exposure settings
//  (see p. 73 of the datasheet):
//  External Integration Time
//  is set to default (see p. 18 of the datasheet), i.e.
//  C-24 ExtIntTime(H) = 0b00000000
//  C-25 ExtIntTime(M) = 0b10000000
//  C-26 ExtIntTime(L) = 0b00000000
//  External Global Gain is set to maximum i.e.
//  C-26 = 0x10, C-29 = 0x00
    e_po6030k_write_register(BANK_C, 0x28, 0x10);
    e_po6030k_write_register(BANK_C, 0x29, 0x00);

//  Set the counter to be incremented every 1 s.
    e_activate_agenda(increment_counter, 10000);

//  Set the control step to be executed every 0.1 s.
    e_activate_agenda(control_step, 1000);

//  Wait for remote control signal.
    while(e_get_check());

    e_start_agendas_processing();
    while(1) {}

    return 0;
}

void increment_counter()
{
    static int counter = 0;
    if (counter++ > 600)
    {
        RESET();
    }
}

void control_step()
{
    static char camera_buffer[40*30] = {0};
    bool sensor_reading;
    int left_speed;
    int right_speed;
    int i;

    e_poxxxx_launch_capture(camera_buffer);

    sensor_reading = 0;
    for(i = 0; i <= 14; i++){
        if(camera_buffer[20 + 40*i] < 170U){
            sensor_reading = 1;
            break;
        }
    }

    e_set_led(0, (int)sensor_reading);

    if (sensor_reading == 0)
    {
        left_speed  = (int)(max_speed*left_speed_0);
        right_speed = (int)(max_speed*right_speed_0);
    }
    else
    {
        left_speed  = (int)(max_speed*left_speed_1);
        right_speed = (int)(max_speed*right_speed_1);
    }

    e_set_speed_left(left_speed);
    e_set_speed_right(right_speed);
}
```

Bibliography

[1] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics*, 2nd ed. New York, NY, USA: Cambridge University Press, 2010.

[2] A. Clark and R. Grush, "Towards a cognitive robotics," *Adaptive Behavior*, vol. 7, no. 1, pp. 5–16, 1999.

[3] D. Floreano and C. Mattiussi, *Bio-inspired artificial intelligence: theories, methods, and technologies.* Cambridge, MA, USA: MIT Press, 2008.

[4] R. Pfeifer, M. Lungarella, and F. Iida, "Self-organization, embodiment, and biologically inspired robotics," *Science*, vol. 318, no. 5853, pp. 1088–1093, 2007.

[5] R. Jones, "Soft machines: Nanotechnology and life," *Oxford University Press*, 2004.

[6] R. Pfeifer, M. Lungarella, and F. Iida, "The challenges ahead for bio-inspired 'soft' robotics," *Communications of the ACM*, vol. 55, no. 11, pp. 76–87, 2012.

[7] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida, "Cognitive developmental robotics: a survey," *IEEE Tran. Autonomous Mental Development*, vol. 1, no. 1, pp. 12–34, 2009.

[8] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 3, pp. 99–117, 2008.

[9] M. Raibert, K. Blankespoor, G. Nelson, R. Playter, *et al.*, "Bigdog, the rough-terrain quadruped robot," in *Proc. 17th World Congress*, 2008, pp. 10 823–10 825.

[10] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems.* Oxford, UK: Oxford University Press, 1999.

[11] G. Beni, "From swarm intelligence to swarm robotics," in *Swarm Robotics*, ser. Lecture Notes in Computer Science, E. Şahin and W. M. Spears, Eds. Berlin/Heidelberg, Germany: Springer, 2005, vol. 3342, pp. 1–9.

[12] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *Swarm Robotics*, ser. Lecture Notes in Computer Science, E. Şahin and W. M. Spears, Eds. Berlin/Heidelberg, Germany: Springer, 2005, vol. 3342, pp. 10–20.

[13] E. Şahin and A. Winfield, "Special issue on swarm robotics," *Swarm Intelligence*, vol. 2, no. 2, pp. 69–72, 2008.

[14] 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.

[15] R. Groß and M. Dorigo, "Self-assembly at the macroscopic scale," *Proc. IEEE*, vol. 96, no. 9, pp. 1490–1508, 2008.

[16] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[17] J. Purcell, A. Avril, G. Jaffuel, S. Bates, and M. Chapuisat, "Ant brood function as life preservers during floods," *PloS one*, vol. 9, no. 2, 2014.

[18] R. Bongiovanni and J. Lowenberg-DeBoer, "Precision agriculture and sustainability," *Precision Agriculture*, vol. 5, no. 4, pp. 359–387, 2004.

[19] A. McBratney, B. Whelan, T. Ancev, and J. Bouma, "Future directions of precision agriculture," *Precision Agriculture*, vol. 6, no. 1, pp. 7–23, 2005.

[20] D. Floreano, J.-C. Zufferey, M. V. Srinivasan, and C. Ellington, *Flying insects and robots.* Berlin/Heidelberg, Germany: Springer, 2009.

[21] K. Nonami, F. Kendoul, S. Suzuki, W. Wang, and D. Nakazawa, *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles.* Berlin/Heidelberg, Germany: Springer, 2010.

[22] R. A. Freitas, "Current status of nanomedicine and medical nanorobotics," *J. Computational and Theoretical Nanoscience*, vol. 2, no. 1, pp. 1–25, 2005.

[23] P. Gomes, "Surgical robotics: Reviewing the past, analysing the present, imagining the future," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 2, pp. 261 – 266, 2011.

[24] G. Iddan, G. Meron, A. Glukhovsky, and P. Swain, "Wireless capsule endoscopy," *Nature*, vol. 405, pp. 417–418, 2000.

[25] J. Timmis, A. Tyrrell, M. Mokhtar, A. Ismail, N. Owens, and R. Bi, *An Artificial Immune System for Robot Organisms.* Berlin/Heidelberg, Germany: Springer, 2010, pp. 268–288.

[26] D. S. Blank, L. A. Meeden, and J. B. Marshall, "Exploring the symbolic/subsymbolic continuum: A case study of RAAM," in *The Symbolic and Connectionist Paradigms: Closing the Gap*, ser. Cognitive Science Series : Technical Monographs and Edited Collection, J. Dinsmore, Ed. London, UK: Psychology Press, 1992, p. 113.

[27] J. Haugeland, *Artificial Intelligence: The Very Idea.* Cambridge, MA, USA: MIT Press, 1989.

[28] A. K. Mackworth, "On seeing robots," in *Computer Vision: Systems, Theory, and Applications.* Singapore: World Scientific Press, 1992, pp. 1–13.

[29] D. Vernon, *Artificial Cognitive Systems: A primer.* Cambridge, MA, USA: MIT Press, 2014.

[30] R. Pfeifer and J. Bongard, *How the body shapes the way we think: a new view of intelligence.* Cambridge, MA, USA: MIT Press, 2006.

[31] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proc. London Mathematical Society*, vol. 2, no. 42, pp. 230–265, 1936.

[32] B. J. Copeland, "The modern history of computing," in *The Stanford Encyclopedia of Philosophy*, Fall 2008 ed., E. N. Zalta, Ed., 2008. [Online]. Available: http://plato.stanford.edu/archives/fall2008/entries/computing-history/

[33] N. J. Nilsson, *The Quest for Artificial Intelligence: A History of Ideas and Achievements.* Cambridge, UK: Cambridge University Press, 2009.

[34] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, pp. 433–460, 1950.

[35] B. Whitby, *Reflections on Artificial Intelligence The Legal, Moral and Ethical Dimensions.* Bristol, UK: Intellect Books, 1996.

[36] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, "A proposal for the dartmouth summer research project on artificial intelligence," 1955. [Online]. Available: http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html

[37] A. Newell and H. A. Simon, "Computer science as empirical inquiry: Symbols and search," *Communications of the ACM*, vol. 19, no. 3, pp. 113–126, 1976.

[38] R. R. Murphy, *Introduction to AI Robotics*, ser. Intelligent Robots and Autonomous Agents, R. C. Arkin, Ed. Cambridge, MA, USA: MIT Press, 2000.

[39] N. J. Nilsson, Ed., "Shakey the robot," SRI International Technical Note 323, 1984.

[40] M. Newborn, *Kasparov versus Deep Blue: computer chess comes of age.* New York, NY, USA: Springer-Verlag, 1997.

[41] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: The next frontier for innovation, competition, and productivity," McKinsey Global Institute, May 2011. [Online]. Available: http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovation

[42] J. Searle, "Minds, brains and programs," *Behavioral and Brain Sciences*, vol. 3, no. 3, pp. 417–424, 1980.

[43] P. McCorduck, *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence.* Natick, MA, USA: AK Peters Ltd, 2004.

[44] H. Dreyfus, "Alchemy and artificial intelligence," RAND Corporation, 1965.

[45] ——, *What Computers Can't Do: A critique of artificial reason.* New York, NY, USA: MIT Press, 1972.

[46] R. Brooks, "Intelligence without representation," *Artificial Intelligence*, vol. 47, pp. 139–159, 1991.

[47] R. A. Brooks, "Intelligence without reason," in *Proc. 12th International Joint Conference on Artificial Intelligence*, vol. 1. San Francisco, CA, USA: Morgan Kaufmann, 1991, pp. 569–595.

[48] "Elephants don't play chess," *Robotics and Autonomous Systems*, vol. 6, pp. 3–15, 1990.

[49] R. A. Brooks, "New approaches to robotics," *Science*, vol. 253, no. 5025, pp. 1227–1232, 1991.

[50] H. Robinson, "Dualism," in *The Stanford Encyclopedia of Philosophy*, Winter 2012 ed., E. N. Zalta, Ed., 2012. [Online]. Available: http://plato.stanford.edu/archives/win2012/entries/dualism/

[51] M. Boden, *Artificial Intelligence and Natural Man*, 2nd ed. Cambridge, MA, USA: MIT Press, 1977.

[52] R. A. Brooks and M. J. Mataric, *Real Robots, Real Learning Problems*, ser. The Springer International Series in Engineering and Computer Science. Berling/Heidelberg, Germany: Springer, 1993, vol. 233, pp. 193–213.

[53] R. C. Arkin, *Behavior-Based Robotics.* Cambridge, MA, USA: MIT Press, 1998.

[54] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology.* Cambridge, MA, USA: MIT Press, 1986.

[55] R. A. Brooks, "A robust layered control system for a mobile robot," Cambridge, MA, USA, Tech. Rep., 1985.

[56] O. H. Bruinsma, "An analysis of building behaviour of the termite *macrotermes subhyalinus*," Ph.D. Thesis, Landbouwhogeschool te Wageningen, Wageningen, The Netherlands, 1979.

[57] P.-P. Grasse, "Termitologia, tome ii," *Fondation des Sociétés. Construction*, 1984.

[58] J. E. Lloyd, "Bioluminescent communication in insects," *Annual Review of Entomology*, vol. 16, no. 1, pp. 97–122, 1971.

[59] ——, "Sexual selection in luminescent beetles," *Sexual selection and reproductive competition in insects*, pp. 293–342, 1979.

[60] P. Laurent, "The supposed synchronal flashing of fireflies," *Science*, vol. 45, no. 1150, 1917.

[61] E. Buck and J. Buck, "Synchronous fireflies," *Scientific American*, vol. 234, 1976.

[62] A. Pikovsky, M. Rosenblum, and J. Kurths, *Synchonization*, ser. Cambridge Nonlinear Science Series. Cambridge, England, UK: Cambridge University Press, 2001, vol. 12.

[63] R. Dawkins, *The extended phenotype: The long reach of the gene.* Oxford, England, UK: Oxford University Press, 1999.

[64] T. G. Whitham, W. P. Young, G. D. Martinsen, C. A. Gehring, J. A. Schweitzer, S. M. Shuster, G. M. Wimp, D. G. Fischer, J. K. Bailey, and R. L. Lindroth, "Community and ecosystem genetics: a consequence of the extended phenotype," *Ecology*, vol. 84, no. 3, pp. 559–573, 2003.

[65] R. Dawkins, *The selfish gene*, 3rd ed. Oxford, UK: Oxford University Press, 2006.

[66] D. J. Futuyma, *Evolution*, 3rd ed. Sunderland, MA, USA: Sinauer Associates, Inc., 2013.

[67] J. Buck, "Synchronous rhythmic flashing of fireflies," *Quarterly Review of Biology*, pp. 265–289, 1988.

[68] D. Otte and J. Smiley, "Synchrony in texas fireflies with a consideration of male interaction models," *Biology of Behaviour*, 1977.

[69] K. J. Aström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers.* Princeton, NJ, USA: Princeton University Press, 2010.

[70] P. Bak, C. Tang, and K. Wiesenfeld, "Self-organized criticality: an explanation of $1/f$ noise," *Physical Review Letters*, vol. 59, pp. 381–384, 1987.

[71] H. M. Jaeger and S. R. Nagel, "Physics of the granular state," *Science*, vol. 255, no. 5051, pp. 1523–1531, 1992.

[72] D. Angeli, J. E. Ferrell, and E. D. Sontag, "Detection of multistability, bifurcations, and hysteresis in a large class of biological positive-feedback systems," *Proc. National Academy of Sciences of the United States of America*, vol. 101, no. 7, pp. 1822–1827, 2004.

[73] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeu, *Self-organization in biological systems*, ser. Princeton Studies in Complexity, Princeton, NJ, USA, 2001.

[74] A. Becskei, B. Séraphin, and L. Serrano, "Positive feedback in eukaryotic gene networks: cell differentiation by graded to binary response conversion," *The EMBO Journal*, vol. 20, no. 10, pp. 2528–2535, 2001.

[75] S. L. Harris and A. J. Levine, "The p53 pathway: positive and negative feedback loops," *Oncogene*, vol. 24, no. 17, pp. 2899–2908, 2005.

[76] K. M. Ansel, V. N. Ngo, P. L. Hyman, S. A. Luther, R. Förster, J. D. Sedgwick, J. L. Browning, M. Lipp, and J. G. Cyster, "A chemokine-driven positive feedback loop organizes lymphoid follicles," *Nature*, vol. 406, pp. 309–314, 2000.

[77] L. Niswander, S. Jeffrey, G. R. Martin, and C. Tickle, "A positive feedback loop coordinates growth and patterning in the vertebrate limb," *Nature*, vol. 371, 1994.

[78] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proc. 14th Annual Conference on Computer Graphics and Interactive Techniques.* New York, NY, USA: ACM Publishers, 1987, pp. 25–34.

[79] "Evolution of supercolonies: the argentine ants of southern Europe," *Proc. National Academy of Science*, vol. 99, no. 9, 2002.

[80] "Communal hunting and pack size in african wild dogs, *lycaon pictus*, author=Creel, Scott and Creel, Nancy Marusha, journal=Animal Behaviour, volume=50, number=5, pages=1325–1339, year=1995."

[81] M. Jacobson, "Natural insect attractants and repellents, new tools in pest control," ser. Advances in Chemistry, D. G. Crosby, Ed., vol. 53. Washington, DC, USA: ACS Publications, 1966, pp. 17–26.

[82] M. S. Mayer, J. R. McLaughlin, *et al.*, *Handbook of insect pheromones and sex attractants.* Boca Raton, FL, USA: CRC Press Inc., 1990.

[83] J. A. Tillman, S. J. Seybold, R. A. Jurenka, and G. J. Blomquist, "Insect pheromonesan overview of biosynthesis and endocrine regulation," *Insect Biochemistry and Molecular Biology*, vol. 29, no. 6, pp. 481–514, 1999.

[84] B. Hölldobler, *The ants.* Cambridge, MA, USA: Harvard University Press, 1990.

[85] D. J. Sumpter and M. Beekman, "From nonlinearity to optimality: pheromone trail foraging by ants," *Animal Behaviour*, vol. 66, no. 2, pp. 273–280, 2003.

[86] J. F. A. Traniello and S. K. Robson, "Trail and territorial communication in social insects," in *Chemical Ecology of Insects 2.* Berlin/Heidelberg, Germany: Springer, 1995, pp. 241–286.

[87] R. Beckers, J.-L. Deneubourg, and S. Goss, "Trail laying behaviour during food recruitment in the ant *Lasius niger*," *Insectes Sociaux*, vol. 39, no. 1, pp. 59–72, 1992.

[88] ——, "Trails and u-turns in the selection of a path by the ant *Lasius niger*," *J. Theoretical Biology*, vol. 159, no. 4, pp. 397–415, 1992.

[89] F. Galton, "Vox populi," *Nature*, vol. 75, no. 1949, pp. 450–451.

[90] E. O. e. a. Wilson, "The insect societies." *The Insect Societies*, 1971.

[91] T. D. Seeley, *Honeybee democracy.* Princeton, NJ, USA: Princeton University Press, 2010.

[92] T. D. Seeley, S. Camazine, and J. Sneyd, "Collective decision-making in honey bees: how colonies choose among nectar sources," *Behavioral Ecology and Sociobiology*, vol. 28, no. 4, pp. 277–290, 1991.

[93] K. A. Pastor and T. D. Seeley, "The brief piping signal of the honey bee: Begging call or stop signal?" *Ethology*, vol. 111, no. 8, pp. 775–784, 2005.

[94] J. C. Nieh, "A negative feedback signal that is triggered by peril curbs honey bee recruitment," *Current Biology*, vol. 20, no. 4, pp. 310–315, 2010.

[95] ——, "The stop signal of honey bees: reconsidering its message," *Behavioral Ecology and Sociobiology*, vol. 33, no. 1, pp. 51–56, 1993.

[96] T. D. Seeley, P. K. Visscher, T. Schlegel, P. M. Hogan, N. R. Franks, and J. a. R. Marshall, "Stop signals provide cross inhibition in collective decision-making by honeybee swarms," *Science*, vol. 335, no. 6064, pp. 108–11, 2012.

[97] M. Dorigo, E. Tuci, R. Gro, V. Trianni, T. Labella, S. Nouyan, C. Ampatzis, J.-L. Deneubourg, G. Baldassarre, S. Nolfi, F. Mondada, D. Floreano, and L. Gambardella, "The swarm-bots project," in *Swarm Robotics*, ser. Lecture Notes in Computer Science, E. ahin and W. Spears, Eds. Berlin/Heidelberg, Germany: Springer, 2005, vol. 3342, pp. 31–44.

[98] B. Liu, L. Wang, and Y.-H. Jin, "An effective pso-based memetic algorithm for flow shop scheduling," *IEEE Tran. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 18–27, 2007.

[99] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European J. Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008.

[100] A. P. Engelbrecht, *Fundamentals of computational swarm intelligence.* Hoboken, NJ, USA: John Wiley and Sons, 2006.

[101] M. Dorigo, V. Maniezzo, A. Colorni, and V. Maniezzo, "Positive feedback as a search strategy," 1991.

[102] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Tran. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[103] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Tran. Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[104] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical Computer Science*, vol. 344, no. 23, pp. 243 – 278, 2005.

[105] M. Dorigo and M. Stützle, *Ant Colony Optimization.* Cambridge, MA, USA: MIT Press, 2004.

[106] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. 1995 IEEE Int. Conf. Neural Networks*, vol. 4. IEEE Press, 1995, pp. 1942–1948.

[107] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. part i: Background and development," *Natural Computing*, vol. 6, no. 4, pp. 467–484, 2007.

[108] ——, "A review of particle swarm optimization. part ii: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications," *Natural Computing*, vol. 7, no. 1, pp. 109–124, 2008.

[109] L. E. Parker, "Multiple mobile robot systems," *Springer Handbook of Robotics*, pp. 921–941, 2008.

[110] L. E. Parker and F. Tang, "Building multirobot coalitions through automated task solution synthesis," *Proc. IEEE*, vol. 94, no. 7, pp. 1289–1305, 2006.

[111] L. E. Parker, "Reliability and fault tolerance in collective robot systems," in *Handbook on Collective Robotics: Fundamentals and Challenges.* Pan Stanford Publishing, to appear.

[112] J. D. Bjerknes and A. F. Winfield, "On fault tolerance and scalability of swarm robotic systems," in *Distributed Autonomous Robotic Systems.* Berlin/Heidelberg, Germany: Springer, 2013, pp. 431–444.

[113] A. F. Winfield and J. Nembrini, "Safety in numbers: fault-tolerance in robot swarms," *Int. J. Modelling, Identification and Control*, vol. 1, no. 1, pp. 30–37, 2006.

[114] A. L. Christensen, R. O'Grady, and M. Dorigo, "From fireflies to fault-tolerant swarms of robots," *IEEE Tran. Evolutionary Computation*, vol. 13, no. 4, pp. 754–766, 2009.

[115] J. Chen and R. Groß, "Cooperative multi-robot box pushing inspired by human behaviour," in *Proc. 12th Annual Conference on Towards Autonomous Robotic Systems.* Berlin/Heidelberg, Germany: Springer, 2011, pp. 380–381.

[116] R. Fitch and Z. Butler, "Million module march: Scalable locomotion for large self-reconfiguring robots," *Int. J. Robotics Research*, vol. 27, no. 3-4, pp. 331–343, 2008.

[117] M. P. Ashley-Rollman, P. Pillai, and M. L. Goodstein, "Simulating multi-million-robot ensembles," in *2011 IEEE Int. Conf. Robotics and Automation.* IEEE Press, 2011, pp. 1006–1013.

[118] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective memory and spatial sorting in animal groups," *J. Theoretical Biology*, vol. 218, no. 1, pp. 1–11, 2002.

[119] D. S. Brown and M. A. Goodrich, "Limited bandwidth recognition of collective behaviors in bio-inspired swarms," in *Proc. 2014 International Conference on Autonomous Agents and Multi-agent Systems.* Richland, SC: IFAAMAS, 2014, pp. 405–412.

[120] S. Kerman, D. Brown, and M. Goodrich, "Supporting human interaction with robust robot swarms," in *Proc. 5th Int. Symp. Resilient Control Systems.* IEEE Press, 2012, pp. 197–202.

[121] M. Dorigo and E. ahin, "Guest editorial," *Autonomous Robots*, vol. 17, no. 2-3, pp. 111–113, 2004.

[122] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Mag.*, vol. 40, no. 8, pp. 102–114, 2002.

[123] R. Groß, S. Magnenat, and F. Mondada, "Segregation in swarms of mobile robots based on the Brazil nut effect," in *Proc. of the 2009 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems.* IEEE Computer Society Press, Los Alamitos, CA, 2009, pp. 4349–4356.

[124] J. Chen, M. Gauci, M. J. Price, and R. Groß, "Segregation in swarms of e-puck robots based on the brazil nut effect," in *Proc. 11th Int. Conf. on Autonomous Agents and Multiagent Systems.* Richland, SC, USA: IFAAMAS, 2012, pp. 163–170.

[125] M. Dorigo, V. Trianni, E. Şahin, R. Groß, T. H. Labella, G. Baldassarre, S. Nolfi, J.-L. Deneubourg, F. Mondada, D. Floreano, and L. Gambardella, "Evolving self-organizing behaviors for a swarm-bot," *Autonomous Robots*, vol. 17, no. 2-3, pp. 223–245, 2004.

[126] V. Trianni, *Evolutionary Swarm Robotics*, ser. Studies in Comput. Intell. Berlin & Heidelberg, Germany: Springer-Verlag, 2008, vol. 108.

[127] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines.* Cambridge, MA, USA: MIT Press, 2000.

[128] R. Bellman, *Dynamic Programming and Lagrange Multipliers.* Princeton, NJ, USA: Princeton University Press, 1957.

[129] Planck Collaboration, "Planck 2013 results. I. Overview of products and scientific results," *arXiv*, 2013, preprint.

[130] K. Deb, *Multi-objective optimization using evolutionary algorithms.* Hoboken, NJ, USA: John Wiley & Sons, 2001.

[131] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing.* springer, 2003.

[132] V. Braitenberg, *Vehicles: Experiments in Synthetic Psychology*, ser. Bradford Books. The MIT Press, 1986.

[133] A. L. Nelson, G. J. Barlow, and L. Doitsidis, "Fitness functions in evolutionary robotics: A survey and analysis," *Robotics and Autonomous Systems*, vol. 57, pp. 345–370, 2009.

[134] M. Mataric and D. Cliff, "Challenges in evolving controllers for physical robots," *Robotics and Autonomous Systems*, vol. 19, no. 1, pp. 67–83, 1996.

[135] G. Baldassare and S. Nolfi, "Strengths and synergies of evolved and designed controllers: A study within collective robotics," *Artificial Intelligence*, vol. 173, pp. 857–875, 2009.

[136] K.-L. Fok, T.-T. Wong, and M.-L. Wong, "Evolutionary computing on consumer graphics hardware," *IEEE Intelligent Systems Magazine*, vol. 22, no. 2, pp. 69–78, 2007.

[137] D. Floreano, P. Husbands, and S. Nolfi, "Evolutionary robotics," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer-Verlag, 2008, pp. 1423–1451.

[138] N. Jakobi, "Half-baked, ad-hoc and noisy: minimal simulations for evolutionary robotics," in *Proceedings of the 4th European Conference on Aftificial Life*, P. Husbands and I. Harvey, Eds., 1997, pp. 348–357.

[139] S. Nolfi and D. Floreano, *Evolutionry Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*, ser. Intelligent Robotics and Autonomous Agents, R. C. Arkin, Ed. The MIT Press, 2000.

[140] R. Bianco and S. Nolfi, “Toward open-ended evolutionary robotics: evolving elementary robotic units able to self-assemble and self-reproduce,” *Connection Science*, vol. 4, pp. 227–248, 2004.

[141] J. Koza and R. Poli, “Genetic programming,” in *Search Methodologies*, E. K. Burke and G. Kendall, Eds. Springer, 2005, pp. 127–164.

[142] C. Lazarus and H. Hu, “Using genetic programming to evolve robot behaviours,” in *Proceedings of the 3rd British Conference on Autonomous Mobile Robotics Autonomous Systems*, 2001.

[143] H.-G. Beyer and H.-P. Schwefel, “Evolution strategies: A comprehensive introduction,” *Natural Computing*, vol. 1, pp. 3–52, 2002.

[144] F. Hoffmann, “The role of fuzzy logic control in evolutionary robotics,” in *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, ser. Studies in Fuzziness and Soft Computing, D. Driankov and A. Saffiotti, Eds., 2001, vol. 61.

[145] N. Sharkey and T. Ziemke, “Life, mind and robots: The ins and outs of embodied cognition,” in *Hybrid Neural Systems*, S. Wermter and R. Sun, Eds. Springer Verlag, 2000.

[146] J. K. Parrish and L. Edelstein-Keshet, “Complexity, pattern, and evolutionary trade-offs in animal aggregation,” *Science*, vol. 284, pp. 99–101, April 1999.

[147] J. Halloy, G. Sempo, G. Caprari, C. Rivault, M. Asadpour, F. Tâche, I. Saïd, V. Durier, S. Canonge, J. M. Amé, C. Detrain, N. Correll, A. Martinoli, F. Mondada, R. Siegwart, and J.-L. Deneubourg, “Social integration of robots into groups of cockroaches to control self-organized choices,” *Science*, vol. 318, pp. 1155–1158, November 2007.

[148] S. Kernbach, R. Thenius, O. Kernbach, and T. Schmickl, “Re-embodiment of honeybee aggregation behavior in an artificial micro-robotic system,” *Adaptive Behavior*, vol. 17, pp. 237–259, June 2009.

[149] N. Fatès, “Solving the decentralised gathering problem with a reaction-diffusion-chemotaxis scheme - social amoebae as a source of inspiration,” *Swarm Intelligence*, vol. 4, no. 2, pp. 91–115, 2010.

[150] J.-L. Deneubourg, J.-C. Gregoire, and E. L. Fort, “Kinetics of larval gregarious behavior in the bark beetle *Dendroctonus micans* (Coleoptera: Scolytidae),” *J. Insect Behavior*, vol. 3, pp. 169–182, March 1990.

[151] R. Jeanson, C. Rivault, J.-L. Deneubourg, S. Blanco, R. Fournier, C. Jost, and G. Theraulaz, “Self-organized aggregation in cockroaches,” *Animal Behavior*, vol. 69, pp. 169–180, January 2005.

[152] S. Garnier, C. Jost, J. Gautrais, M. Asadpour, G. Caprari, R. Jeanson, A. Grimal, and G. Theraulaz, “The embodiment of cockroach aggregation behavior in a group of micro-robots,” *Artificial Life*, vol. 14, no. 4, pp. 387–408, 2008.

[153] L. Bayindir and E. Şahin, “Modeling self-organized aggregation in swarm robotic systems,” in *Proc. 2009 IEEE Swarm Intelligence Symp.* IEEE Press, 2009, pp. 88–95.

[154] O. Soysal and E. Şahin, “Probabilistic aggregation strategies in swarm robotic systems,” in *Proc. 2005 IEEE Swarm Intelligence Symp.*, 2005, pp. 325–332.

[155] N. Correll and A. Martinoli, “Modeling and designing self-organized aggregation in a swarm of miniature robots,” *Int. J. Robotics Research*, vol. 30, pp. 615–626, April 2011.

[156] S. Kernbach, D. Häbe, O. Kernbach, R. Thenius, G. Radspieler, T. Kimura, and T. Schmickl, “Adaptive collective decision-making in limited robot swarms without communication,” *Int. J. Robotics Research*, vol. 32, pp. 35–55, January 2013.

[157] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, “Distributed memoryless point convergence algorithm for mobile robots with limited visibility,” *IEEE Trans. Robot. Autom.*, vol. 15, no. 5, pp. 818–828, 1999.

[158] J. Cortés, S. Martinez, and F. Bullo, “Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions,” *IEEE Tran. Autom. Control*, vol. 51, no. 8, pp. 1289–1298, 2006.

[159] N. Gordon, I. A. Wagner, and A. M. Bruckstein, “Gathering multiple robotic a(ge)nts with limited sensing capabilities,” in *Proc. 4th Int. Conf. Ant Colony Optimization and Swarm Intelligence*, ser. Lecture Notes in Computer Science, vol. 3172. Berlin/Heidelberg, Germany: Springer, 2004, pp. 142–153.

[160] N. Gordon, Y. Elor, and A. M. Bruckstein, "Gathering multiple robotic agents with crude distance sensing capabilities," in *Proc. 6th Int. Conf. Ant Colony Optimization and Swarm Intelligence*, ser. Lecture Notes in Computer Science, vol. 5217. Berlin/Heidelberg, Germany: Springer, 2008, pp. 72–83.

[161] M. D. Gennaro and A. Jadbabie, "Decentralized control of connectivity for multi-agent systems," in *Proc. 45th IEEE Conf. Decision and Control*, 2006, pp. 3628–3633.

[162] A. Gasparri, A. Priolo, and G. Ulivi, "A swarm aggregation algorithm for multi-robot systems based on local interaction," in *Proc. 2012 IEEE Int. Conf. Control Applications*, 2012, pp. 1497–1502.

[163] V. Gazy and K. M. Passino, "Stability analysis of swarms," *IEEE Trans. Autom. Control*, vol. 48, pp. 692–697, April 2003.

[164] W. Li, "Stability analysis of swarms with general topology," *IEEE Trans. Sys., Man, Cybern., Syst.*, vol. 38, pp. 1084–1097, August 2008.

[165] A. Gasparri, G. Oriolo, A. Priolo, and G. Ulivi, "A swarm aggregation algorithm based on local interaction for multi-robot systems with actuator saturations," in *Proc. 2012 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2012, pp. 539–544.

[166] A. Leccese, A. Gasparri, A. Priolo, G. Priolo, and G. Ulivi, "A swarm aggregation algorithm based on local interaction with actuator saturations and integrated obstacle avoidance," in *Proc. 2013 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*. IEEE Press, 2012, pp. 1857–1862.

[167] J. Yu, S. LaValle, and D. Liberzon, "Rendezvous without coordinates," *IEEE Tran. Automatic Control*, vol. 57, no. 2, pp. 421–434, 2012.

[168] E. Bahceci and E. Şahin, "Evolving aggregation behaviors for swarm robotic systems: a systematic case study," in *Proc. 2005 IEEE Swarm Intell. Symp.*, 2005, pp. 333–340.

[169] J.-L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien, "The dynamics of collective sorting robot-like ants and ant-like robots," in *Proc. 1st Int.Conf. Simulation of Adaptive Behavior, From Animals to Animats*, 1991, pp. 356–363.

[170] R. Beckers, O. E. Holland, and J. L. Deneubourg, "From local actions to global tasks: Stigmetry and collective robotics," *Artificial Life*, vol. 4, pp. 181–189, 1994.

[171] O. Holland and C. Melhuish, "Stigmergy, self-organization, and sorting in collective robotics," *Artificial life*, vol. 5, no. 2, pp. 173–202, 1999.

[172] Y. Song, J.-H. Kim, and D. A. Shell, "Self-organized clustering of square objects by multiple robots," in *Proc. 8th Int. Conf. Swarm Intelligence.* Berlin/Heidelberg, Germany: Springer, 2012, pp. 308–315.

[173] M. Maris and R. te Boekhorst, "Exploting physical constraints: Heap formation through behavioural error in a group of robots," in *Proc. 2012 IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 1996, pp. 1655–1660.

[174] A. Martinoli, A. J. Ijspeert, and F. Mondada, "Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots," *Robotics and Autonomous Systems*, vol. 29, no. 1, pp. 51–63, 1999.

[175] S. Kazadi, A. Abdul-Khaliq, and R. Goodman, "On the convergence of puck clustering systems," *Robotics and Autonomous Systems*, vol. 38, no. 2, pp. 93–117, 2002.

[176] N. Fatès and N. Vlassopoulos, "A robust scheme for aggregating quasi-blind robots in an active environment," *Int. J. Swarm Intelligence Research*, vol. 3, no. 3, pp. 66–80, 2012.

[177] M. Bodi, R. Thenius, T. Schmickl, and K. Crailsheim, "How two cooperating robot swarms are affected by two conflictive aggregation spots," in *Advances in Artificial Life. Darwin Meets von Neumann*, ser. Lecture Notes in Computer Science, G. Kampis, I. Karsai, and E. Szathmáry, Eds. Berlin/Heidelberg, Germany: Springer, 2011, vol. 5778, pp. 367–374.

[178] F. Arvin, A. E. Turgut, F. Bazyari, K. B. Arikan, N. Bellotto, and S. Yue, "Cue-based aggregation with a mobile robot swarm: a novel fuzzy-based method," *Adaptive Behavior*, 2014, in press.

[179] V. Gazi and K. M. Passino, "A class of attractions/repulsion functions for stable swarm aggregations," *Int. J. Control*, vol. 77, no. 18, pp. 1567–1579, 2004.

[180] J. Clement, X. Dfago, M. G. Potop-Butucaru, T. Izumi, and S. Messika, "The cost of probabilistic agreement in oblivious robot networks," *Information Processing Letters*, vol. 110, no. 11, pp. 431 – 438, 2010.

[181] G. F. Tóth, P. Gritzmann, and J. M. Wills, "Finite sphere packing and sphere covering," *Discrete Computational Geometry*, vol. 4, no. 1, pp. 19–40, 1989.

[182] R. L. Graham and N. J. A. Sloane, "Penny-packing and two-dimensional codes," *Dicrete Computational Geometry*, vol. 5, no. 1, pp. 1–11, 1990.

[183] T. Y. Chow, "Penny-packings with minimal second moments," *Combinatorica*, vol. 15, no. 2, pp. 151–158, 1995.

[184] D. E. Knuth, *The Art of Computer Programming. Volume 1: Fundamental Algorithms*, 3rd ed. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1997.

[185] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Canci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proc. 9th Conf. Autonomous Robot Systems and Competitions*, vol. 1, no. 1, 2009, pp. 59–65.

[186] S. Magnenat, M. Waibel, and A. Beyeler, "Enki: The fast 2D robot simulator," 2011. [Online]. Available: http://home.gna.org/enki/

[187] O. Michel, "Webots: Professional mobile robot simulation," *Int. J. Advanced Robotic Systems*, vol. 1, no. 1, pp. 39–42, 2008.

[188] M. Gauci, "Online supplementary material." [Online]. Available: http://naturalrobotics.group.shef.ac.uk/supp/mgauci_thesis

[189] S. C. Goldstein, J. D. Campbell, and T. C. Mowry, "Programmable matter," *Computer*, vol. 38, no. 6, pp. 99–101, 2005.

[190] P. F. Damasceno, M. Engel, and S. C. Glotzer, "Predictive self-assembly of polyhedra into complex structures," *Science*, vol. 337, pp. 453–457, 2012.

[191] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computing*, vol. 1, no. 2, pp. 270–280, 1989.

[192] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[193] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[194] P. Rodriguez, J. Wiles, and J. L. Elman, "A recurrent neural network that learns to count," *Connection Science*, vol. 11, no. 1, pp. 5–40, 1999.

[195] N. Hansen, "The CMA-ES evolution strategy: A tutorial," https://www.lri.fr/~hansen/cmatutorial.pdf, 2011, accessed 10 October 2013.

[196] F. Hoffmann, "Evolutionary algorithms for fuzzy control system design," *Proc. IEEE*, vol. 89, no. 9, 2001.

[197] N. Correll and R. Groß, "From swarm robotics to smart materials. Guest editorial. Special issue on Swarm robotics," *Neural Computing & Applications*, vol. 19, no. 6, pp. 785–786, 2010.

[198] N. Jakobi, "Half-baked, ad-hoc and noisy: Minimal simulations for evolutionary robotics," in *Proc. 4th European Conf. Artificial Life*. Cambridge, MA, USA: The MIT Press, 1997, pp. 348–357.

[199] M. Sitti, "Micro-and nano-scale robotics," in *Proc. 2004 American Control Conf.*, vol. 1. IEEE Press, 2004, pp. 1–8.

[200] M. Sarikaya, C. Tamerler, A. K.-Y. Jen, K. Schulten, and F. Baneyx, "Molecular biomimetics: nanotechnology through biology," *Nature Materials*, vol. 2, no. 9, pp. 577–585, 2003.

[201] L. Meier, F. Mondada, M. Bonani, and J. Besuchet, "e-puck software library," 2010. [Online]. Available: http://www.e-puck.org/

[202] Pixelplus Co. Ltd., "PO3030K camera data sheet," 2006. [Online]. Available: http://projects.gctronic.com/E-Puck/docs/Camera/PO3030K.pdf