Self-Organized Robotic System Design and Autonomous Odor Localization

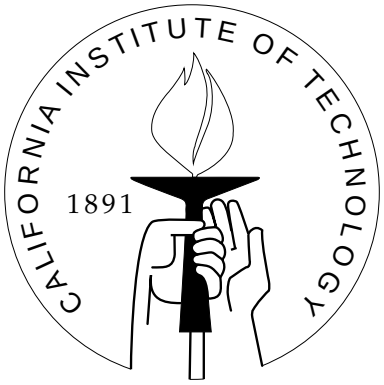
Thesis by

Adam T. Hayes

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2002

(Defended May 24, 2002)

Acknowledgements

I would like to thank Richard Murray for allowing me to complete this work, and Rod Goodman for giving me the opportunity to begin it. Along the way I have benefited from the insight of Shuki Bruck, Pietro Perona, Joel Burdick, and Christof Koch, whose comments have helped mold this work into its final form. I owe much to Owen Holland and Alcherio Martinoli, who introduced me to the study of collective mobile robotics and guided me toward the interesting problems. Many others in the Microsystems Lab and Collective Robotics Group played a role in making everything actually work: Sanza Kazadi, Andrew Lundsten, Jim Pugh, Ladd Van Tol, Robert Enright, Tiago Wright, Mauricio Cordero, and Ian Kelly. I would like to thank Vincent Koosh for showing me the way of the jaded graduate student when I was first starting out, and Kjerstin Easton for giving me the opportunity to continue the tradition. I am forever indebted to my fellow First-Years, without whom I would probably still be working on 185 problem sets. I also especially thank the Guacamoleans, for never being tempted to take my "killing time" literally. I thank my family for their seemingly tireless encouraging support, and for ensuring that I never went hungry. And lastly I thank Sarah, not only for her mathematical expertise and endless appetite for revision, but also for constantly reminding me that the more important things do not run on batteries.

Abstract

This thesis presents a methodology for designing self-organized autonomous robotic systems and demonstrates how this process can be applied to the problem of finding the source of an airborne odor plume. The design methodology is applicable to other task domains and the resulting odor localization system extends the state of the art.

The design procedure centers on the ability to define a specific task performance metric, systematically evaluate performance in a realistic environment, and define abstract relationships between system parameters and system performance. Once such relationships have been experimentally validated in a test environment, they can be used to guide the design of a deployable system. Because this process relies heavily on evaluative feedback, this work emphasizes the development of tools that allow the collection of accurate performance data. It presents a reliable multiple robot test-bed and some task-enabling sensory hardware, as well as validation of the sensory and kinematic models used in simulation. Also, a reinforcement learning methodology is described that provides consistent optimization performance while minimizing the amount of required evaluation.

The design methodology is applied to the task of odor localization. Specifically, this thesis analyzes a basic collective search task and derives the optimal group size and expected performance bounds for random and coordinated search. It also investigates a set of biologically inspired behaviors that permit an agent to traverse an odor plume to its source and describes the common characteristics of successful algorithms. One of these algorithms is implemented on the real test-bed and in simulation to verify that plume traversal is taking place and that the use of multiple collaborating robots can expand the reachable performance space. Collective search and plume

traversal are then combined (along with egocentric source declaration) into the full odor localization task which is optimized in simulation. Then, following the design methodology, a model is presented which can aid in the prediction of performance and choice of algorithm parameters in more complex environments. Finally, a flocking behavior is designed, and the addition of this flocking behavior to the plume tracing algorithm is shown to produce a more capable system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As suggested by the title, this dissertation contains two principal themes. It presents a methodology for designing self-organized robotic systems and demonstrates the application of this procedure on an odor localization task. In the process, it provides a detailed analysis of the odor localization problem, ranging from a treatment of a general search task to a discussion of the common qualities of efficient plume traversal algorithms. Inspiration from biological systems plays a role throughout, reflecting the belief that the imitation of natural systems, when applied in the proper context, can be useful to an engineer.

1.1 Motivation

The creation of autonomous robots, machines that sense and act upon the world to perform useful work without constant human supervision, could free humans from many repetitive or dangerous tasks and increase productivity immensely. However, the traditional Sense-Model-Plan-Act approach to artificial intelligence has proven not to be robust in unknown dynamic environments, and the last fifty years of robotics research has provided little in the way of autonomous systems that can function reliably in the real world. This shortcoming can largely be traced to the extreme difficulty of building and maintaining accurate world models when sensor input is uncertain and the state of the world can change unexpectedly. The problem essentially comes down to one of sensing, that is, the world cannot be sensed accurately enough to be properly

modeled. The prevailing attitude among a large sector of the robotics community is that, in time, sensing (and communication) technology will improve, and eventually the level of reliability currently obtained in highly constrained environments (which is necessary for current control algorithms to function) will be available in the real world.

Newer behavior-based approaches to robotic controller design emphasize a tighter coupling between sensation and action. In these systems the importance of maintaining world models is reduced (because planning is generally absent), but the task of designing a system to achieve a particular goal becomes more difficult because it is not always obvious what global activity will emerge from a particular set of interacting behaviors. Swarm intelligence, a computational and behavioral metaphor that draws inspiration from social insects, combines the behavior-based approach with the redundancy inherent in large numbers of agents. Although systems designed using this concept can be exceptionally robust to agent failure and environmental disturbances (witness the considerable success of the ant, for example), there is even further distance between the local sensing and action programmed into each agent and the overall system objective. Previous attempts at training both single and multiple agent behavior based systems have focused on learning the proper sensory-action mapping to produce the desired behavior. Some success has been achieved in the laboratory domain, but it is unlikely that these techniques will scale to more complicated tasks because the complexity of this learning problem grows exponentially with the number of states in the system. In order to address real applications, different techniques are needed.

1.2 A Design Methodology

This thesis presents a design methodology that relies on a balance of engineering intuition and machine learning to facilitate real system design. The first step in the process requires an engineer to assess the task and develop a set of parameterized behaviors that allows a group of agents to solve it. No magic solution is proposed for

how this is to be done, although familiarity with systems possessing similar function (e.g., as observed in biology) is found to be helpful, and feedback from the subsequent design phases can assist as well. The key point is that this behavioral parameterization drastically reduces the size of the algorithmic search space, which then can be systematically explored both in simulated and real instantiations of the task.

This second phase of design uses a simple reinforcement learning algorithm to optimize system performance while using only a minimal amount of evaluation. Evaluative optimization is required because a priori models of system performance rarely exist. And because performance evaluation in real or realistic environments is typically resource-intensive, it is advantageous to minimize the number of samples required. The results of this optimization procedure not only provide performance levels achievable by the chosen algorithm parameterization, but they also indicate how the behavioral parameters influence task performance.

The third design phase involves the definition of abstract relationships between system parameters and system performance. System parameters encompass both algorithmic parameters as well as any that describe the task environment. Once such relationships have been experimentally validated in a test environment, they can be used to guide the design of a deployable system. If the task resists this sort of characterization, the evaluative optimization can be performed directly on the application environment, but it is typically less expensive to experiment with a scaled-down version of the task.

This design procedure centers on the ability to define a specific task performance metric, which could be considered design phase zero, as it should occur before any attempt is made to design the parameterized behaviors. In other words, before one goes about solving a problem, it is a good idea to know exactly what the problem is. This idea is rather intuitive, although it raises some interesting questions about the current practice of robotics research that are further developed in the next chapter. Also, because this process relies heavily on evaluative feedback, this work emphasizes the development of tools that allow the collection of accurate performance data. It presents a reliable multiple robot test-bed and some task-enabling sensory hardware,

as well as validation of the sensory and kinematic models used in simulation.

1.3 Application: Odor Localization

The design methodology is applied to the task of odor localization–finding the source of an odor plume. The goal is to develop an algorithm that enables a group of relatively simple robots to locate an odor source within an enclosed arena. This task breaks down into three subtasks: plume finding (coming into contact with the odor), plume traversal (following the odor plume to its source), and source declaration (determining from odor acquisition characteristics that the source is in the immediate vicinity). These subtasks, while not wholly independent, can be studied separately for the purposes of building intuition about the operation of successful systems.

Plume finding amounts to a search task, with the added complication, due to the stochastic nature of the plume, that a sequential search is not guaranteed to succeed. This work examines a simplified search task and analytically derives expressions for optimal group size and expected cost for both random (suitable for swarm intelligence implementations) and coordinated search strategies. Coordinated strategies perform better, but the additional localization and communication capabilities required are often expensive to implement. The results are encouraging for the random approach, as they demonstrate that when the probability of target detection is low–which is built in to the plume task, as the location of the plume is time variant–the performance benefit of using coordinated search diminishes.

Plume traversal requires more specialized behavior, both to progress in the direction of the source and to maintain consistent contact with the plume. Single agent plume traversal is dealt with extensively in the biological literature, as there are many species whose livelihood relies on their ability to track an odor plume to its source. Insects are particularly useful in this regard, because they are unlikely to incorporate information from many modalities or utilize complicated cognitive maps of their environment in their search behavior. This work analyzes several simple plume traversal strategies based on moth behavior and concludes that there is a basic pattern of be-

havior that must be followed for an algorithm to be successful. It is worthwhile to note that the strategies being studied at this level already incorporate restrictions of what can be implemented in real hardware. This ensures that the results from these studies will be applicable to the real task.

Nature can only provide a certain degree of guidance, however, because the task being studied need not have a direct natural corollary. In the odor localization task being studied the agents should work collectively to locate the odor source, while in biological systems the individual agents usually compete with each other for use of the resource at the head of the plume. Thus, the collaboration strategies used in this work are guided only by simplicity: attraction, repulsion, and no communication at all. For experimental reasons, attraction is implemented on the real test-bed and in simulation to verify that plume traversal is taking place and that the use of multiple collaborating robots can expand the reachable performance space. The performance impact of the other communication strategies is studied exclusively in simulation.

Source declaration does not necessarily have to be done using odor information, as typically odor sources can be perceived via another type of sensor from short range, though it is possible to do so without any extra sensory apparatus. This phase of the task is not studied extensively because it is likely to be highly dependent on the properties of the particular task and environment, although a functional solution is presented. A source declaration behavior is combined with the collective search and plume traversal algorithms to solve the full plume traversal task, which is optimized in simulation. Then, following the design methodology, a model is presented which can aid in the prediction of performance and choice of algorithm parameters in a larger range of environments.

The design methodology is also applied (in part) to a flocking behavior. A set of basic behaviors that are conducive to implementation on real robots is presented, and it is shown that performance can be tuned to different simulated environments purely evaluatively. No model is presented that describes flocking performance, but it is suggested that if the evaluative process is extended across many different types of environments, a model might be constructed empirically. Furthermore, it is demon-

strated that this flocking capability can be integrated into the plume tracing behavior to produce a better odor localization system.

1.4 Thesis Overview

Chapter 2 provides a review of the robotics literature that is relevant to this work. It briefly surveys the issues surrounding the development of autonomous systems and describes several previous robotic system design approaches. Finally, background on the odor localization problem is provided, including a discussion of both biological and robotic work.

Chapter 3 presents the system design methodology, including the off-line machine learning algorithm. It also provides a detailed description of the odor localization problem and the metrics used in performance evaluation.

Chapter 4 describes the tools used to investigate the plume task. It covers the real-robot platform, the arena and infrastructure developed to carry out systematic experiments, the odor and wind sensors built to enable real-robot plume traversal, the sensor-based simulator used to permit more extensive experimentation, and finally the software architecture that runs the control algorithms.

Chapter 5 presents a quantitative analysis of the tradeoffs between group size and efficiency in collective search tasks that considers both the time-sensitive nature of search completion and the system operating cost. First, the search task is defined and a performance metric is presented that can account for all of the costs associated with the task. Next, for both random and coordinated search strategies, analytical expressions are derived that can be used to predict optimal system performance bounds given a particular task description. Also, the performance benefit of using coordinated search is shown to be dependent on the relative values of the different cost components. Finally, a sensor-based computer simulation is used to support the analytical results, suggesting that the assumptions involved in their derivation are sound.

Chapter 6 presents an investigation into odor source localization algorithms for

turbulent odor plumes. The goal of this chapter is to gain a better understanding of what makes an algorithm successful in order to build more capable and robust chemical plume traversal systems. First, the problem of plume traversal is recast as the task of obtaining the next odor hit, and a set of metrics that provides detailed information about algorithm function is presented. Then, several odor localization algorithms are described, and it is demonstrated that algorithm parameters can be tailored to particular plume characteristics for improved performance. Also, the next-hit analysis is shown to capture the performance of some types of algorithms more accurately than others, and it is concluded that this failure stems from intrinsic shortcomings of some of the algorithms tested. Moreover, based on this analysis, the general properties required of successful turbulent odor plume traversal algorithms are described.

Chapter 7 presents an investigation of plume traversal by groups of autonomous mobile robots and then extends to address the full odor localization task. First, a distributed algorithm is described by which groups of agents can solve the plume traversal task. Next, local position, odor, and flow information tightly coupled with robot behavior is shown to be sufficient to allow a robot to localize the source of an odor plume. Then, the use of multiple agents is demonstrated to increase the size of the solution space that can be reached by a particular system, and the swarm intelligence solution compares well with coordinated search strategies for this task. In addition, the off-line machine learning algorithm is used to optimize algorithm performance on the full odor localization task in several different environments, and it is shown that the optimal system parameters depend on the particular task being studied. Finally, a model is presented that can be used to relate task parameters to system performance.

Chapter 8 presents an investigation of flocking by teams of autonomous mobile robots using principles of swarm intelligence. First, a simple flocking task is presented. Next, a leaderless distributed flocking algorithm that is more conducive to implementation on embodied agents than the established algorithms used in computer animation is described. The design methodology is followed to optimize performance

under different conditions, showing that this method can be used not only to improve performance but also to gain insight into which algorithm components contribute most to system behavior. Then, it is shown that a group of real robots executing the algorithm with emulated sensors can successfully flock (even in the presence of individual agent failure) and that systematic characterization (and therefore optimization) of real-robot flocking performance is achievable. Finally, the integration of a flocking behavior into the odor localization algorithm is demonstrated to produce better odor localization performance.

Chapter 9 concludes this dissertation with a summary of the main results and a discussion of some future directions in autonomous robotics.

1.5 Original Contributions

This thesis contributes to the field of autonomous robotics and advances the study of odor localization. In particular, it presents:

- A self-organized system design methodology that relies on the formulation and evaluation of specific task metrics.
- An improved odor localization system that can derive useful information from the distal part of an odor plume.
- Greater insight into the tradeoffs between sensor reliability, evaluation metrics, and coverage strategy for collective search problems.
- An understanding of the general properties required of successful turbulent odor plume traversal algorithms.
- A flocking algorithm that is well suited to implementation on real hardware.

Chapter 2

Background

This chapter provides a review of the robotics literature that is relevant to this work. It briefly surveys the issues surrounding the development of autonomous systems and describes several previous robotic system design approaches. Finally, background on the odor localization problem is provided, including a discussion of both biological and robotic work.

2.1 Robotics

A robot is defined as "a device that automatically performs complicated and often repetitive tasks" [74]. Its purpose in this sense is to increase human productivity, particularly when the work involved is dull, dirty, or dangerous [110]. Robots have been rather successful in this regard. As of 2001, there were over 750,000 robots in use in industry worldwide, and their share of the workload will likely increase in the future [30]. Robots have made forays into the entertainment industry, from animatronic exhibits at amusement parks to the Sony AIBO and other, simpler toy robots. Service robots abound in professional settings such as medicine and demolition, and they promise to arrive for home use as lawn mowers and vacuum cleaners in the near future [61].

Much progress has been made over the last fifty years since Grey Walter built his robot turtles [100] and George Devol and Joseph Engelberger constructed the first industrial robotic arms [27]. However, many of the dreams of the robotics age, as

exemplified by *The Jetsons*, *2001: A Space Odyssey*, and *Star Wars*, have yet to be realized. This is due in part to the fact that most of the tasks (such as household chores) originally designated for robots can be done quite cheaply by humans [17], so there is little financial incentive to make the necessary investment in development. But there are other reasons behind this shortcoming, as thus far robots have excelled at performing specific tasks in controlled environments, while there are few examples of robots that can function in dynamic sensory-rich settings.

2.1.1 The Difficulty with Sensing

Why have robots been so slow to move out into the real world? When asked what is missing from the field of robotics in a recent interview, Engelberger, who is commonly known as the father of industrial robotics, put it simply:

> The thing that makes the big difference is sensory perception. There isn't any amount of software that can take junk and make it really work in an application. First of all, you need magnificent physical execution. After you have the physical execution, you need great sensory perception. If you have all of that–vision, tactile sense–then you can use software. [17]

This observation argues for continuing to improve sensors and sensory processing systems, and most of this work occurs outside the field of robotics. However, even highly evolved biological sensory systems such as the human visual system cannot be relied upon to accurately report the state of the outside world all of the time [91], so it is reasonable that proper robotics design should be able to account for sensory inconsistencies. According to Sebastian Thrun, a leading robotics researcher, "Robots are inherently uncertain about the state of their environments. Uncertainty arises from sensor limitations, noise, and the fact that the most interesting environments are–to a certain degree–unpredictable" [95]. The question of how to produce reliable performance in the presence of sensory noise is dealt with differently by the different schools of control that have developed within the robotics field.

2.1.2 Classes of Control

Robotics control can be broken into four divisions: deliberative, reactive, hybrid, and behavior-based [71], [2]. The differences between these approaches stem principally from disagreements over the nature of intelligence–whether it stems from extensive cognitive reasoning or from highly tuned and tightly coupled interactions with the environment.

Most of the early work in robotics grew out of the early work in artificial intelligence, which assumed a deliberative view of intelligence. Ronald Arkin, a leading robotics researcher, explains this link well:

> In the original proposal [for what was to become the first conference on artificial intelligence] [73], Marvin Minsky indicates that an intelligent machine "would tend to build up within itself an abstract model of the environment in which it is placed. If it were given a problem it could first explore solutions within the internal abstract model of the environment and then attempt external experiments." This approach dominated robotics research for the next thirty years, during which time AI research developed a strong dependence on the use of representational knowledge and deliberative reasoning methods for robotic planning. [2]

The deliberative approach has proven effective when it is possible to implement, i.e., when an accurate "internal abstract model" can be constructed. This is the case when the sensory burden is low because the environment can be highly controlled (e.g., within a factory) or does not exist physically at all–computers have become quite proficient at the game of chess [90]. However, once the external world departs from the internal construction (in an unknown way), no amount of reasoning is going to reliably produce reasonable actions.

The reactive paradigm takes an entirely different view of the nature of intelligence. It tightly couples the sensory and motor systems, so rather than performing experiments using internal models, robots built in this paradigm function through constant interaction with their environment in a stimulus-response manner. These principles were present in Walter's early work, although they subsequently disappeared for several decades. Valentino Braitenberg revived them in the mid-1980's [14], although Rodney Brooks was responsible for inducing the robotics community to take notice

[16]. Taken to its extreme, robots using reactive control contain no models of the world at all. The problem of maintaining models then disappears, and sensor noise becomes less of an issue because it can be averaged over many actions [35]. However, purely reactive robots also contain no state that might allow them to learn over time, and this inability to adapt has been cited as one of the main shortcomings of this type of control [71].

The purely deliberative and purely reactive strategies represent opposing ends of the control spectrum, and most current work in robotics takes place somewhere in between. One such method, hybrid control, "attempts to combine the real-time response of reactivity with the rationality and optimality of deliberation" [71]. A typical mobile robot under hybrid control might use a reactive controller to attend to immediate problems (such as obstacle avoidance) while relying on a deliberative mechanism to maintain goal-oriented behavior. The key to designing these systems is in the interface module that allows these two systems to communicate effectively [71]. Recently the most successful hybrid systems have assumed a probabilistic approach to world modeling:

> When "guessing" a quantity from sensor data, the probabilistic approach computes a probability distribution over what might be the case in the world, instead of generating a single "best guess" only. As a result, a probabilistic robot can gracefully recover from errors, handle ambiguities, and integrate sensor data in a consistent way. [95]

Probabilistic hybrid robots are nearing the point where they can reliably tackle real-world tasks [18], but there are problems with this approach. The computational cost of dealing with a large number of probability distributions is significant, and even though much current research is devoted to devising computationally efficient methods of dealing with these structures [95], this problem will only intensify as tasks become more complex. Also, there may be a limit to how much sensor noise these systems can handle, as thus far the systems deployed have been dependent on highly accurate (and expensive) sensors to function [96].

Another approach that resides more toward the reactive end of the control spectrum is behavior-based control. It draws much inspiration from biology, dividing up

robot control into sets of interacting behaviors. Behaviors are mappings from sensor input to actuator output, and they may include state (i.e., memory), which enables a wider range of capabilities (including adaptation and planning) than is possible in purely reactive systems [2]. However, typically much less of the environment is represented internally than in a hybrid system, and there is little emphasis on explicit modeling of sensor uncertainty. Instead, proper function is obtained through carefully tuning how each behavior interacts with the rest of the system (both internally and through the environment), and controller design tends to be a difficult and environment-dependent process.

Behavior-based control has been commonly applied to multi-agent systems, perhaps because its limited use of world models scales well to dynamic environments (which are inherent to the multi-agent case) [69]. Another research thread that encompasses a subset of these ideas has grown into the field of Swarm Intelligence, which draws its inspiration from multi-agent biological examples provided by social insects [8]. In most biological cases studied so far, robust and capable biological group behavior has been found to be mediated by nothing more than a small set of simple interactions among individuals and between individuals and the environment [12]. The application of swarm intelligence principles to autonomous collective robotics aims to develop robust task solving by minimizing the complexity of individual units, emphasizing parallelism, and exploiting direct or indirect local interactions. These principles favor the design of behavior-based robotic systems, which emphasize tight coupling between sensation and action, avoidance of representational knowledge, and action decomposition into contextually meaningful units [2]. There are three main advantages of the swarm intelligence approach: first, scalability from a few to thousands of units, second, flexibility, as units can be dynamically added or removed without explicit reorganization, and third, increased system robustness, not only through unit redundancy but also through the design of minimalist units. Several examples of collective robotics tasks solved with swarm intelligence principles can be found in the literature: aggregation [67] and segregation [43], beacon localization [39], stick pulling [45], and collective transportation [56].

To summarize, there are currently two leading approaches to control that can address the problem of imperfect sensing and may yield robotic systems that function well in dynamic environments. One is hybrid control, which attempts to explicitly account for sensory uncertainty and behave optimally with respect to as much information as is computationally feasible. Hybrid systems contain explicit world models, so the design interface is straightforward, although computational issues of scaling up to more complex tasks and more noisy sensing have yet to be fully addressed. On the other side is behavior-based control, particularly of multi-agent systems (in the swarm intelligence sense). This control method can take advantage of the inherent parallelism and robustness of many agents without much additional complexity because precise world models and peer-to-peer communication are not used. The problem with behavior-based systems is that their function is determined implicitly through interactions with the environment, so it can be difficult for a designer to determine the local rules that each agent must follow in order to achieve a particular group goal. The ultimate answer as to which of these control strategies is more appropriate is probably task dependent [71]. It is unlikely that swarms of robots will be cleaning kitchens in the near future, although they could likely be searching for avalanche victims, mining for coal, or even simply mowing (large) lawns. This work focuses on advancing the behavior-based multi-agent/swarm intelligence domain because it may provide a level of robustness for large-scale tasks that is unattainable through other means.

2.1.3 Designing Self-Organized Systems

Before examining the development of the design process, a note on the terminology: 'self-organized' is a term that has grown out of the biological literature [19] that simply represents the decentralized operation that is desired in swarm intelligence based robotic systems. It is often accompanied by the term "emergent", (as in "this function is an emergent property of this self-organized system"), which can be defined as occurring when the "global behavior of a system is not evident from the local

behavior of its elements" [53]. Unfortunately, emergence has become synonymous with the word "magic" in some circles. There is in fact nothing magic about emergent properties or self-organized systems–these words simply mean that the function of a system depends on the interaction of a number of parallel processes and may be difficult to explain in a serial, hierarchical manner.

The initial work into the design of behavior-based systems focused on single agents, which can be considered self-organized if one views the individual behaviors as different interacting processes. The difficulty of designing behavior-based systems was recognized early on, and there was interest in building systems that could essentially design their own solutions to a particular task. As Mahadevan and Connell stated: "If new behaviors could be learned, it would... free the designer from needing a deep understanding of the interactions between a particular robot and its application environment" [64]. In 1991, they studied how to get a behavior-based robot to learn to push a box to the edge of a room. In order to achieve this, they discretized the state and action spaces (18 bits and 5 actions, respectively), and attempted to automatically generate a functional state-action mapping by allowing the robot to interact with its environment. They examined learning performance using two types of reward signals, monolithic and behavior-based. In the former they rewarded the robot only when it was pushing the box, and in the latter they divided the task into three behaviors which were rewarded separately: box finding, box pushing, and unwedging (recovering after bumping into an immovable object). Their methods are notable because they used both a real and a simulated robot to study their algorithms, and they incorporated hand-coded and random controls for performance comparison. However, they did not perform enough trials to generate any statistical analysis of their findings. Overall, they determined that learning was possible and that the behavior-based learning methods were the more effective. These results make intuitive sense because more detailed reward functions provide the learner with more instructive feedback about its progress. However, better performance required more work from the designer, and perhaps even a "detailed understanding" of the task: "it took us several iterations to write reward functions that generated good performance figures" [64].

They foresaw the following challenges:

> Based on our experience with using reinforcement learning on real robots, we think the hard subproblems in reinforcement learning have to do with dealing with large sensory state spaces, long action sequences, and initial task specification [i.e., specification of the reward signal]. [64]

It is difficult to evaluate many states in a reasonable amount of time, and providing detailed reward functions (which are more informative to the learner) requires more knowledge on the part of the designer.

In 1996 Colombetti, Dorigo, and Borghi introduced what they termed "a methodology for behavior engineering" [24], in an effort to bring the tools used in more established disciplines, like software engineering, to bear on the problem of designing robots. Learning played a central role in their system:

> The real world is so complex and unpredictable that directly programming a robot's controller soon becomes an almost impossible job. Recently, machine learning techniques have emerged as an interesting attempt to overcome this difficulty; however, it is not at all clear how machine learning should be integrated with more traditional design methodologies. [24]

They formalized a design process, which began with a mathematical description of the application: "a complete specification of the target behavior should include a formal, quantitative component" [24]. Then the designer would determine the sensors and actuators necessary to complete the task, as well as a training strategy that specified the reward function and how it would be applied (as above, upon task completion or as a progress estimator). The robot would then be trained possibly first in simulation and then on the real environment, learning a state-action mapping that produced a functional system. They demonstrated their methodology on several simple tasks and found it to be successful. They even performed enough real robot trials to generate statistics about their results, although they did not always perform control experiments with hand-coded or random controllers, so they could only draw limited conclusions about the utility of their methodology. It is also unclear whether their methods would extend to harder problems: "A first extension will have to be in the direction of more complex behaviors. This will require a larger amount of

input information to be processed, and therefore will call for more powerful learning mechanisms" [24]. In their implementations they used a combination of classifier systems [13] and genetic algorithms [42], which, due to the implicit manner in which they utilize reward values, require a substantial amount of data to function. These learning techniques are acceptable when continuous feedback about task progress is available, the state spaces being searched are relatively small, and evaluation can be sped up via the use of simulation, but they break down when any of these conditions are not met. Nevertheless, the systematic and quantitative analysis of behavior is an important concept that was advanced by this work.

Around the same time researchers were advancing into techniques for automated controller design in the multi-robot domain. In 1995 Maja Mataric proposed that complex group behavior could be created by appropriate combinations of more simple "basis behaviors" [68] [69]. These simple behaviors, such as avoidance and following, were inspired by biology and could be combined to create more complicated behaviors such as flocking and foraging. These results were demonstrated on groups of real robots, which was an impressive achievement, although quantitative metrics were not assigned for the particular tasks so a statistical characterization of performance could not be performed. In 1997 Mataric applied techniques similar to those of Mahadevan and Connell to learn the task of multi-robot foraging [70]. Mataric split the robot sensory space into a set of four binary conditions and the action space into four low-level behaviors, and aimed to automatically generate a functional mapping between the two. The efficacy of different reward methods was investigated.

> We propose *shaped reinforcement* as a means of taking advantage of as much information as is available to the robot at any point. Shaping is based on principled embedding of domain knowledge in order to convert intermittent feedback into a more continuous error signal. [70]

A more continuous error signal (rather than a single value upon success) should make learning easier because the problem of propagating reward across states temporally is minimized. Results of the learning process were obtained with a group of four real robots, and they revealed that the more detailed reward functions produced a

statistically significant learning increase. However, the learning was measured by similarity of the learned policy to a designated "optimal" hand-coded policy rather than via actual performance of the robots on the foraging task, as again no task metric was ever stated. Also, the process of shaping the reinforcement function itself heavily biases the control policy to develop in a particular way and requires much task-specific knowledge from the designer. Still, it is significant that learning was demonstrated in the group robot domain, as the state space is inherently larger (since multiple agents are learning at the same time), and the dynamic nature of the environment (because the agents interact) renders the reward values more variable.

In 1999 Alcherio Martinoli proposed a twofold approach to distributed controller design that included probabilistic modeling of system performance as well as automatic creation of control algorithms through the use of genetic algorithms [65]. The modeling work incorporated analysis of algorithm flowcharts and simple geometric considerations to generate system performance measures, and this methodology has been used to analyze aggregation and cooperative object transport experiments [67], [11], [45]. These results demonstrate that probabilistic modeling is a useful tool for building intuition about system function because its minimalist essence allows the designer to identify the system characteristics that most influence performance. However, thus far, quantitative agreement with other modes of evaluation (i.e., real-robot results) has required the use of free parameters, so the predictive utility of this approach is unclear. Also, the modeling methodology has been found to be inapplicable when spatial location plays a significant role in task performance [39]. Separate from the modeling, the design aspect of the work investigated different methods of using genetic algorithms to design controllers in simulation. Rather than trying to construct a state-action mapping, the control algorithms were parameterized and good combinations of parameters were searched for via a genetic algorithm. Different reward structures (individual and group) and controller structures (private and public) were studied, and the best combination was determined to be task dependent. It is not clear that this design method will scale to many control parameters, as then the search spaces can become unreasonably large. As described earlier, genetic algorithms

require much performance data to function, and even though other investigators have interleaved simulated and real-world performance evaluation with some success [29], [60], accurate simulation is required to produce working controllers (which becomes more difficult as task complexity increases). Also, control heterogeneity was permitted in all cases, which meant that public policies suffered severely from the state-space size problem, as parameter values for each individual controller had to be determined within the same search space.

A different approach to multi-robot learning was presented by Lynne Parker in 2000 [82]. She introduced a method for allowing a heterogeneous group of robots to adapt their actions to changing environmental conditions over time. The initial algorithm design was left up to a designer, as the motivation behind this approach was to make real-robot teams more robust to dynamic environments. This was done by giving each member of the team a "desire" to accomplish particular parts of a task, which enabled each team member to specialize on a particular subtask during normal system operation. When an unexpected event occurred, such as the failure of a robot or an increase in task difficulty, other agents could compensate by switching from their preferred task to the task that was not being properly accomplished, while a robot that found itself not properly completing its subtask could allow others the opportunity to address it. This dynamic task allocation mechanism was demonstrated with real robots on a box pushing task, but no task metric was defined so the analysis was not quantitative. This work showed that dynamic cooperation between groups of heterogeneous robots can be achieved, although it imposed fairly heavy restrictions. The subtasks being performed must be independent to ensure that robots never work against each other, each of the robots has to be able to accurately determine its progress on its selected subtask, and all of the robots in the system must have access to the state of all the subtasks in the system for proper work distribution to take place. In the experiments presented this global knowledge was achieved through extensive communication among the team members, a solution that renders this control method more suitable for small teams of cooperating robots rather than larger robot systems (where the overhead of constant global communication becomes significant).

The approaches described above are representative of previous research into the design of self-organized systems, but this review is by no means complete. A large amount of work has been done in this area, e.g., [78] [59], [109], [37], [4], [51], [94], [89]. However, the main principles, particularly those that have been applied to real robots, have been addressed. These include the idea of directly learning state-action mappings, using either reinforcement learning or genetic algorithm techniques. The main problem these techniques run up against is that the size of the state space grows exponentially with the number of states an agent can perceive. Structured behaviors have been used to reduce the size of the state and action spaces that must be searched. Likewise, detailed reward functions that incorporate much domain-specific knowledge (describing how a task should be solved) have been used to facilitate learning, although this solution mitigates much of the benefit of using learning controllers because it requires the designer to have an intimate familiarity with the task. Automatic tuning of control parameters has been investigated as an alternative to the state-action approach, as its complexity is determined by the number of plastic parameters rather than the overall complexity of the task. Optimization techniques that have proven effective on more traditional engineering problems (such as Taguchi's robust design method [83, 49]) may enhance the performance of this approach. However, even when the space being searched is small, noisy evaluation due to stochasticity in the environment or unpredictable agent interaction can cause problems for any optimization algorithm.

2.1.4 The Need for Metrics

In the previous section, it was touched upon that in order to properly use evaluative learning techniques, there needs to be some definitive task and some quantitative way of evaluating performance on that task. This idea is rather intuitive, although not all researchers have defined a concrete task metric when studying learning, e.g., [70], [82]. In fact, throughout the field of robotics, there is a general tendency to shy away from defining specific task performance metrics, particularly as tasks become complex.

Although attempts have been made to introduce uniform quantitative metrics to specific subfields (e.g., [107]), this type of analysis has not taken hold, particularly in the field of autonomous mobile robotics. This observation is evident from the number of papers that use a diagram showing the path that their robot(s) took on a particular trial (i.e., "validation") as proof that their algorithm works, e.g., [5], [22], [69], [23], [96] compared to those that support their claims with statistically valid experimental data, e.g., [55].

There are two underlying reasons that the culture of "proof by demonstration" has enveloped the field. One is technological–it is difficult to build systems that can generate systematic data, although recent developments in wireless networking and computer vision have made this task easier. The other reason is more philosophical–by defining a specific metric and studying only a specific task, the results obtained appear more difficult to generalize. The study of an algorithm that applies whenever a particular set of conditions is met is deemed more worthwhile. The fallacy behind this approach is that design without a specific task in mind often assumes best-case sensory and actuation scenarios. So, rather than developing general solutions, researchers end up with algorithms that function reliably only in the perfect (possibly Gaussian) worlds they construct inside their computers. They end up with solutions that are brittle when exposed to the harsh sensory realities of the real world, and this may be why there are so few fielded autonomous robotic systems.

Of course, purely evaluative design is not a perfect process, as the designation of a metric that captures all important aspects of a task can be difficult, and exhaustive testing of every situation that a system may encounter while in an unconstrained application environment is by definition impossible. There is much to be said for studying algorithms themselves and investigating what sort of systems could be created if certain sensor and actuator characteristics could be guaranteed. An understanding of the fundamental properties of a system and the ways in which they interact can greatly simplify the design process, and at the very least this sort of investigation can be used to inform the sensor community about what capabilities would be most useful. However, robotic algorithm design that relies on realistically unsatisfiable as-

sumptions (perfect localization and communication are common examples) does little to advance the creation of functional robotic systems.

The designation of a specific task performance metric and the systematic, statistical verification that desired performance levels can be achieved is the only way to uncover the precise nature of the unavoidable imperfections of the real world and demonstrate that a robotic system can account for them. Such work can be done with the help of simulation, as long as the sensor and actuator models used are well grounded in reality. The use of real robots to verify that sensors are accurately captured by simulation models has been shown to be useful [44], although eventually actual real-robot experimentation is necessary (again in a systematic, possibly more limited way) to ensure all of the relevant nuances have been captured. "There are many facets of the real world that are very difficult to simulate effectively–for example, how far does a sonar reflect off a metal file cabinet?" [64]. A recognition of the need for experimentation exists in the literature, although it is rarely acted upon.

> Experimental studies might become more rigorous and thorough, e.g., via standard benchmark problems and algorithms. This is challenging in mobile robotics, given the noisy, system-specific nature of the field. Nevertheless, it is necessary for claims about "robustness" and "near-optimality" to be appropriately quantified, and for dependencies on various control parameters to be better understood. [20]

Only after rigorous testing becomes standard in the robotics community, and the difficulties associated with using real sensors and actuators are embraced, will designs emerge that prove to be robust in the real world.

2.2 Odor Localization

Recent advances have been made in understanding biological odor localization and tracking as developed in moths [21], [7] and rats [10] in the air, and lobsters [3] and stomatopods [106] in water. Biology utilizes olfaction for a wide variety of tasks including finding others of the same species, communication, behavior modification, avoiding predators, and searching for food. Animals use a combination of "hardware"

(frequency of receptor adaptation, perhaps), "software" (temporal integration and/or spatial integration), and behavioral search strategies (both intrinsic and landmark-based) to locate odor sources. Odor localization is in essence a behavioral problem that varies from animal to animal. While some animals exploit fluid information at different layers (lobster) or different residues on the ground (ants), others can track odors in the air (moths) or use a combination of information (dogs).

From an engineering standpoint there are advantages to combining odor tracking with mobile robots, such as in the detection of chemical leaks and the chemical mapping of hazardous waste sites [87]. A necessary initial step is to develop robotic systems that use odor tracking algorithms, multiple sensory modalities (e.g., odometry, anemometry, olfaction), and sensory fusion to search out and identify sources of odor. Such systems have been built by various research groups [81], [88], [57], [36], [51], [40], although performance thus far has been limited by the reliability, temporal response characteristics, and sensitivity of available odor transduction mechanisms [80]. The essential problem is that the available odor sensors lack the combination of speed and sensitivity necessary to perceive the complex and dynamic structure of a turbulent odor plume. The approach of moving slowly and continually sampling odor and flow data to reduce environmental noise is used in nature (starfish) and has been applied to robotic systems [81], [51], but environmental and behavioral constraints (e.g., significant plume sparseness or meander, time critical performance) can render these systems ineffective [47]. However, as odor sensory technology improves [47], designers of artificial odor localization systems will be able to focus on algorithm design to achieve better system performance.

Previous robotic odor localization research that derived concentration gradient information from multiple sensors [46], [81] was restricted to operation in the proximal region of the plume (within 2 m of the source) and had to move slowly (.01-.03 m/s) so that concentration gradient information could be extracted with reasonable accuracy. Although these efforts were successful in demonstrating the feasibility of odor localization with mobile robots, it is not clear that any method that involves spatial concentration extraction will extend to more sparse plumes (i.e., longer plume

tracking distances), since as odor information becomes less frequent, concentration integration times will increase, decreasing system performance accordingly [47]. More capable sensors and different plume tracking behaviors may be necessary to efficiently track more complex plumes.

Chapter 3

A Design Methodology and the Odor Localization Task

The reasons why one might want to construct a self-organized robotic system to perform a particular task were laid out in the previous chapter. The difficulties involved in designing this type of system have been described as well. This chapter presents a way to design self-organized systems, including an off-line machine learning algorithm that can be used when evaluation is expensive. It also provides a detailed description of the odor localization task studied in this work and the metrics used in performance evaluation.

3.1 A Design Methodology

3.1.1 Phase Zero: Choose a Task

Before one begins the design process, the task to be accomplished must be specified exactly. This description should be in the form of a quantitative performance metric. A "total system cost" is useful in this regard, because it is often the only way to reduce different task components (e.g., initialization costs, energy used, time-to-completion) to comparable units. Cost metrics are used throughout this work. Some costs may be difficult to specify exactly–e.g., what is the cost of one more minute of exposure to a chemical weapon? Estimates can always be made, however, and their accuracy will play a significant role in determining the ultimate performance level achieved.

3.1.2 Phase One: Parameterize the Control Algorithm

The first step for specifying the control algorithm is to determine what sensors and actuators are necessary (and cost-effective) for task completion. Systems using different sensor and actuator sets may be constructed and compared, but the initial design effort for each set should proceed independently. As others have done previously, this design methodology uses the notion of "behaviors" to partition the sensory and action space of each individual agent. However, rather than choosing a behavior set and using machine learning to derive a sense-action mapping, the designer instead specifies a set of parameterized behaviors that allows a group of agents to solve the given task. The designer determines when each behavior is active, and only a small set of control parameters are left to be searched. This design methodology does not attempt to guide the designer in the process of determining the proper behaviors or specifying which parameters should be free, although familiarity with systems possessing similar function (e.g., as observed in biology) have been found to be helpful, and feedback from the subsequent design phases can assist as well.

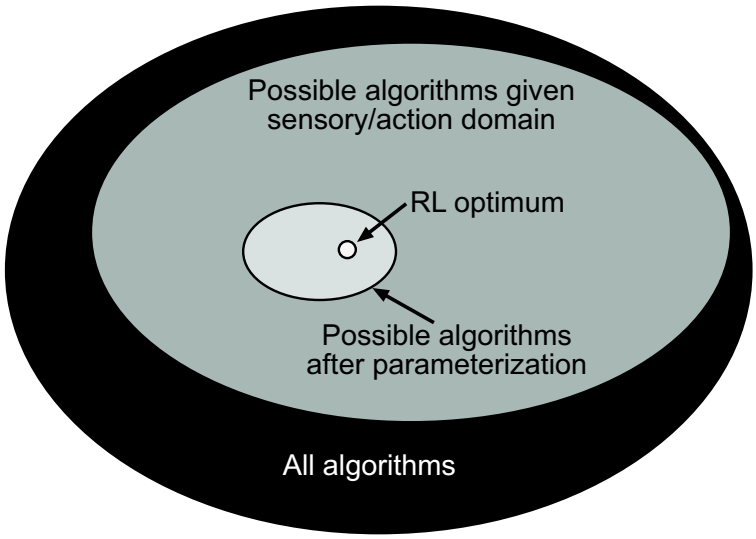


Figure 3.1: Venn diagram depicting the space of algorithms that can solve a particular task.

Venn diagram depicting the space of algorithms that can solve a particular task. The key point of this phase of the design is that the behavioral parameterization

drastically reduces the size of the algorithmic search space, which then can be systematically explored both in simulated and real instantiations of the task. Figure 3.1 provides an abstract representation of the design space. Within the set of all algorithms that can perform a particular task lies a subset restricted by a particular sense-action domain. This subset of algorithms can be large for complicated tasks, so this design methodology calls for the system designer to further restrict the space of possible algorithms by parameterizing the behaviors that perform the sense-action mapping. Within this more limited domain, reinforcement learning techniques can be used to determine the algorithm parameter sets that produce the best performance.

3.1.3 Phase Two: Off-Line Machine Learning Optimization

Once the behaviors and the control parameters have been chosen, the system must be implemented for testing. Simulation is useful in the initial design stages to permit rapid prototyping of behaviors, although testing on real hardware must take place to verify that the simulations accurately model the agent interactions with the environment. If the simulated and real results agree over a test domain, the speed of the simulation can be exploited during the optimization phase. Maximizing system performance involves solving a global optimization problem in the algorithm parameter space. Because self-organized systems depend heavily on sensitive agent-to-agent and agent-to-environment interactions, performance is often stochastic, hence evaluative, rather than gradient based, search methods are appropriate. This type of control optimization has been extensively studied for the case of a single agent [93, 29, 111], as well as for multiple agents [82, 70].

This design methodology assumes that all agents follow the same control policy and that the only metric used for system evaluation is the global task metric. The use of homogeneous controllers with a global reward signal provides a way of addressing the credit assignment problem [76], which represents the difficulty in distributing credit for success among the many decisions that may have played a role in producing it. Credit assignment is a central problem in reinforcement learning [93]. By making

the learning agent operate in the space of algorithm parameters and providing only measures of group performance (rather than feedback from individual actions), there effectively becomes one agent and one action per reward signal, and the credit assignment problem no longer applies [98]. This solution may be an extreme simplification of the reinforcement learning problem, but it does allow performance improvements to be realized on reasonably complex tasks. Note that even though all agents are required to have the same control policy, differentiation is still possible. Each agent could alter control parameters according to experience, or agents could randomly choose one of a fixed number of different policies when the task begins (in the limit of many agents, the number of agents of each type will be stable).

Even if the number of control parameters is small, however, a full dimensional search of the parameter space is not always feasible. In this work, to reduce the size of the search space, sequential 1-D optimizations are performed, with each parameter optimized while the others remain fixed. This restriction may make finding the optimal parameter set difficult in some search domains. However, in the case studies examined in this work, the parameters can be grouped so that each set is effectively independent, and the 1-D search allows performance improvements to be achieved in a reasonable amount of time. In this work the selection of design points (i.e., specific parameter values over which to optimize) is done a priori, although there are techniques for selecting them adaptively [111, 54] which may be utilized in further studies. Each parameter space is bounded and discretized to include a range of important values, as determined by preliminary experiments. At the beginning of each optimization run the variable values are randomly initialized.

The idea behind the optimization procedure itself is rather simple: repeatedly evaluate a set of parameter values until it can be said with some degree of certainty that none of the parameter values performs significantly better than the current estimated "best" value. This optimization algorithm is defined by the initial design choice method and three parameters: η, κ, and ϵ. η defines the margin around the best point in which it is not cost effective to further optimize (e.g., if $\eta = .1$ and all remaining design points are determined to be less than 10% greater than the maxi-

mum, the optimization stops). κ defines the desired level of certainty of achievement of the margin defined by η. ϵ sets the minimum number of trials necessary so that the group comparison procedure is accurate. This work uses a parametric test, Tukey's HSD multiple comparison procedure [84], and while the assumptions underlying its operation (namely normality of data) are not fully met, it performs well enough to demonstrate the utility of this optimization procedure. For real applications, the multiple comparison procedure should be tuned to the underlying distribution of the performance data.

The following describes the details of the algorithm. For each parameter, once the design points χ_i ($i = 1...\psi_\rho$, where ψ_ρ is the total number of points for parameter ρ), are selected, the optimization is performed as follows:

1. Initialize the set of active points B to include all χ_i.

2. At each iteration j, simulate a trial at each χ_i in B, storing the result $\upsilon_i^j = P$ in Υ_i. The system performance P (e.g., as described in equation (3.4)) represents the task metric that is to be maximized and is assumed to be positive.

3. If $j > \epsilon$, first, using Tukey's HSD multiple comparison procedure, determine the critical difference d (to significance κ) which must be equalled or exceeded by the difference of two means in the set for that difference to be declared significant. Next, let

$$E(\Upsilon_{max}) = \max_i E(\Upsilon_i). \tag{3.1}$$

   For each $\chi_i \in B$, if

$$(1 + \eta)E(\Upsilon_{max}) - E(\Upsilon_i) > d \tag{3.2}$$

   and $i \neq max$, remove χ_i from B. $E(x)$ represents the expected value of x.

4. If more than one χ_i remains in B, go to *Step 2*.

At the end of the process, the remaining point χ_{max} in B represents the best guess at the optimum value for the parameter currently being optimized given the other

fixed parameter values. After each cycle through all parameters (in either a fixed or random order), the resulting parameter set is evaluated and then used as the input set for the next cycle. In this work we set the number of cycles per optimization run to 10.

This reinforcement learning algorithm optimizes system performance while using only a minimal amount of evaluation. Because performance evaluation in real or realistic environments is typically resource-intensive, it is advantageous to minimize the number of samples required. The results of this optimization procedure not only reveal the performance levels achievable by the chosen algorithm parameterization, but they also indicate how the behavioral parameters influence task performance. For example, parameters that do not converge to a particular set of values can be fixed to any value and omitted from subsequent optimization runs, as this nonspecificity indicates that the designer chose to parameterize a value that was not critical to task performance. Also, optimizing across a set of environments and analyzing how the optimal parameter values change with environmental characteristics can facilitate the construction of abstract relationships between environmental and algorithmic parameters.

3.1.4 Phase Three: Generate an Abstract Model

The third design phase involves the definition of abstract relationships between system parameters and system performance. System parameters encompass both algorithmic and environmental parameters. Once such relationships have been experimentally validated in a test environment, they can be used to guide the design of a deployable system, as they will allow the designer to predict system performance in a wider range of environments than have been explicitly examined experimentally. Examples of this procedure are given in Chapters 5 and 7. If the task resists abstract characterization, extensive evaluative optimization can be performed directly on the application environment, but it is typically less expensive to experiment with a scaled-down version of the task. Some amount of experimentation will be necessary in the application

environment regardless, to verify that the system performs as expected, but the use of models to guide the choice of control algorithm parameters can eliminate the need for extensive experimental parameter search.

3.1.5 Feedback During the Design Process

The design methodology has been described as a sequential process, although in practice there should be substantial feedback among the different design phases. Behaviors and control parameterizations will be modified after performance has been observed in simulation. Real-world implementation will constrain the interaction models used in simulation. The experimental axis of the design will impact the model generation procedure, and insights from the modeling work may suggest new behaviors. Once a model has been generated that can predict the performance values found by the learning system, real-world problems may be approached with some degree of confidence. A schematic of the design process and the interactions between its components can be seen in Figure 3.2.

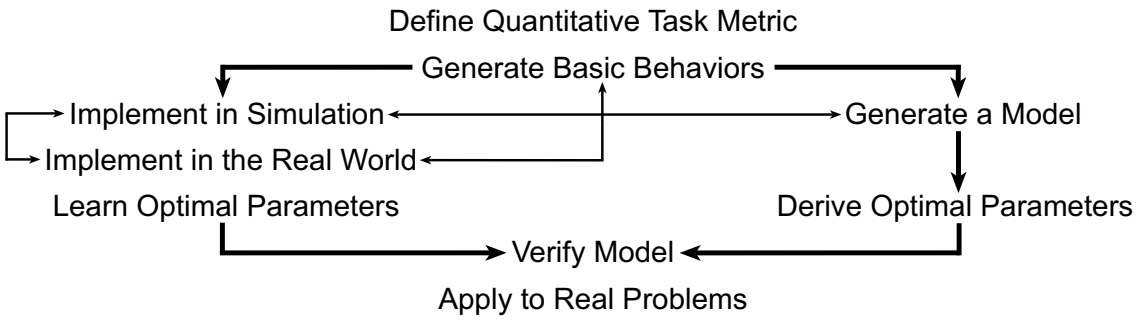


Figure 3.2: A schematic of the design process, beginning with task definition and ending with application to real problems. Arrows indicate significant interplay between design phases.

3.2 The Odor Localization Problem

The following section provides a detailed description of the odor localization task that is the principle application examined in this work.

3.2.1 The Odor Plume

As an odor source dissolves into a fluid medium, an odor plume is formed. The turbulent nature of fluid flow typically breaks the plume into isolated packets, areas of relative high concentration surrounded by fluid that contains no odor [48, 79]. This work focuses on turbulent plumes because the application environments for artificial systems will likely be dominated by turbulent dispersion. Given the packet-like nature of odor plumes and the current (and projected) limitations of odor sensing technology, it is assumed that only binary odor information generated from a single plume sensor is available to odor localizing agents.

3.2.2 Task Definition

The general odor localization task addressed in this paper is as follows: find the source of a single turbulent odor plume in an enclosed 2-D area as efficiently as possible. This can be broken down into three subtasks: plume finding (coming into contact with the odor), plume traversal (following the odor plume to its source), and source declaration (determining from odor acquisition characteristics that the source is in the immediate vicinity). Plume finding amounts to a search task, with the added complication, due to the stochastic nature of the plume, that a simple sequential search is not guaranteed to succeed. Plume traversal requires more specialized behavior, both to progress in the direction of the source and to maintain consistent contact with the plume. Figure 3.3 illustrates this phase of the task. Source declaration does not necessarily have to be done using odor information, as typically odor sources can be perceived via another type of sensor from short range, but it is possible to do so without any extra sensory apparatus [88, 40].

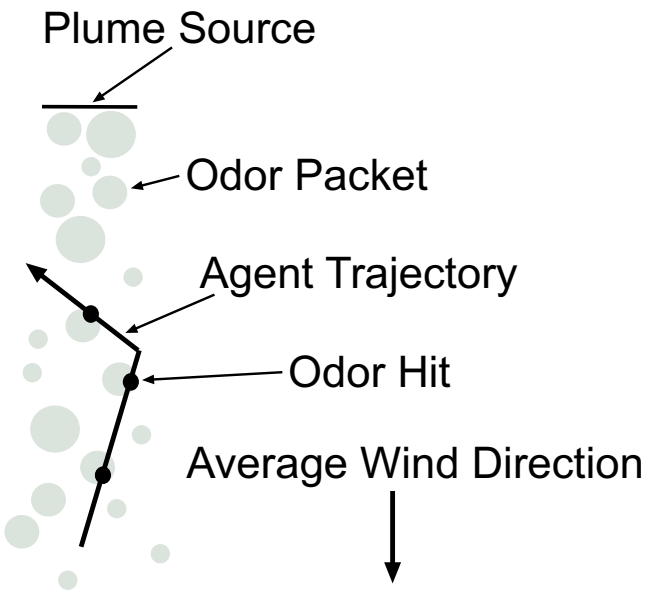


Figure 3.3: The plume traversal task. The issue is how to define the agent trajectory based on odor hit and wind direction information such that the agent approaches the plume source.

3.2.3 Performance

To study odor localization, one or multiple agents are placed inside an enclosed arena containing an odor plume, and over repeated trials the time and distance traveled by the whole group until an individual completes the task are measured. This work assumes that in the case of multiple agents, all are deployed within a minimal distance of a single deployment point. Task completion can be defined in a number of ways: an agent comes within a given radius of the plume source (allocentric determination– useful to emulate a non-odor related target sensor that each robot might carry), an agent declares the plume source found (egocentric determination, no additional sensor necessary), or any combination and extension thereof (i.e., multiple declarations required within a given radius). For the purposes of performance evaluation it is assumed that some measures of time and group energy (which can be considered proportional to the sum of the individual distances traveled) necessary for task completion exist.

Efficiency for the odor localization task cannot be defined in the general case.

Instead, one can combine the time and energy measures of task performance in an application specific manner. Since these measures are physically independent, a composite metric incorporating a particular weighting of these two basic factors can be considered:

$$C = \alpha T_{TC} + \beta D_{TC}. \tag{3.3}$$

T_{TC} is the time needed for task completion, and D_{TC} represents the total distance traveled by all agents during the task. α is taken to be the cost per unit time of not completing the task, and β is the cost per unit distance of running the system. C represents the total cost incurred before the task is completed. To facilitate performance comparison across different environments, it is useful to normalize C by the minimum completion cost of a particular system. This measure is then inverted to generate a more intuitive (and presentable) performance metric:

$$P = \frac{\alpha T_{MIN} + \beta D_{MIN}}{C}. \tag{3.4}$$

T_{MIN} and D_{MIN}, the optimum values for the given task, are determined from the average distance between starting location and target location as well as maximum agent speed. The numerator of P thus represents the minimal completion cost– obtainable only by a system that has prior knowledge of the source location. The form of P ensures that for any α and β greater than 0, the optimal system will achieve a performance of 1, and any system that requires more time or distance (averaged over many initial conditions) will have a performance less than 1. By choosing specific values for α and β, the appropriate relationship between time required and energy used (which typically vary inversely) can be generated for evaluating any particular application.

Chapter 4

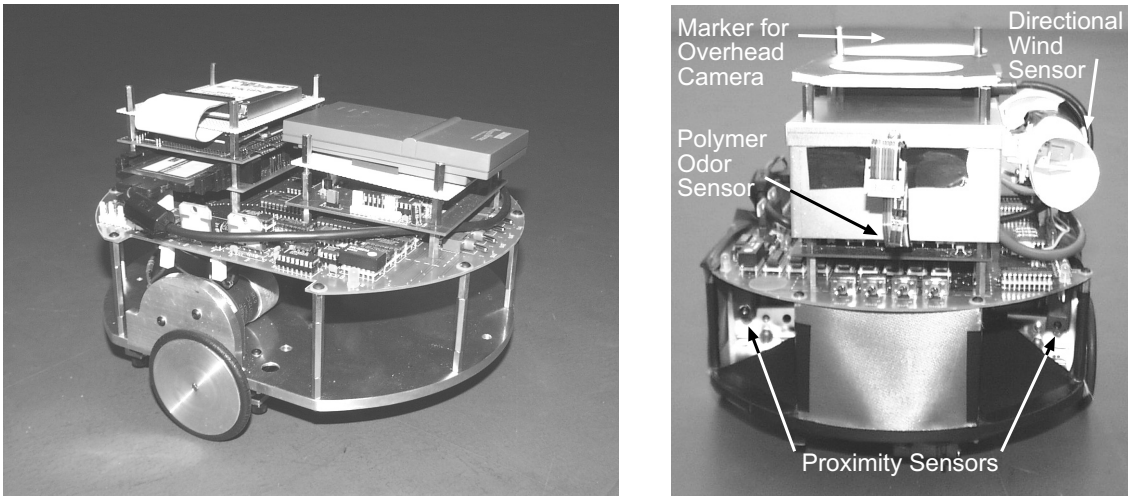
Robots, Sensors, and Simulators

The design methodology described in the previous chapter emphasizes the generation of systematic performance data in real and realistic environments. This chapter describes the tools used to investigate the plume task [41]. It covers the real-robot platform, the arena and infrastructure developed to carry out systematic experiments, the odor and wind sensors built to enable real-robot plume traversal, the sensor-based simulator used to permit more extensive experimentation, and the software architecture that runs the control algorithms.

4.1 Real Robots

This work uses Moorebots, as shown in Figure 4.1a, which were originally designed by Owen Holland at the University of West England, Bristol, U.K. Each 24 cm diameter robot is equipped with two DC motor-driven wheels, a castor wheel, a 2 Mbit wireless LAN transceiver, and 12-bit A/D and D/A converters. See [108] for a more detailed robot description. To perform plume traversal, this basic configuration was supplemented with 4 infrared range sensors for collision avoidance, a single odor sensor, a hot wire anemometer, and a set of markings to assist in overhead tracking. A fully equipped plume-traversing Moorebot is shown in Figure 4.1b, and in this configuration each robot has an energetic autonomy of 1.5 hours. On-board high-level control is provided by a PC104 based Intel 386 processor running Linux. Low level control such as motor speed regulation is executed by dedicated hardware interfaced to the

PC104 bus. Groups of 1 to 6 robots are used to study the plume task, and a group of 10 robots is used to demonstrate a flocking behavior (see Chapter 8).



(**a**) (**b**)

Figure 4.1: (**a**) A basic Moorebot. (**b**) A Moorebot equipped with wind, odor, and proximity sensors, as well as markings for overhead tracking.

4.2 Robot Arena and Infrastructure

Due to physical space constraints, only one of the three odor localization subtasks, plume traversal, is studied on the real robots. A plume of significant length fills a large part of the arena, so the plume search phase is trivial in the real arena, and technical and space constraints make the source declaration phase experimentally difficult to study as well. However, much of the plume-related complexity of the odor localization task is captured by the traversal phase.

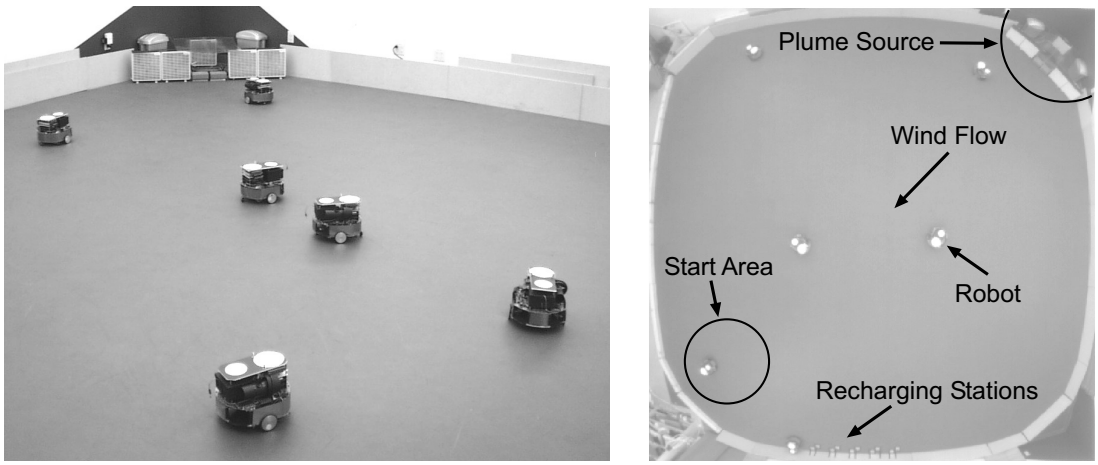
The plume traversal arena is 6.7 by 6.7 m. The odor plume is created by a 23 cm square hot water pan and a bank of 5 fans 30 cm in diameter (see Figure 4.2a), and it extends diagonally from one corner of the arena toward the opposite corner. Flow characteristics based on data taken along the plume axis 15 cm above the floor are summarized in Table 4.1. The coefficient of variation is a measure of the intensity of the flow turbulence. It represents the ratio of the standard deviation of the wind

velocity to the mean wind velocity, and 20% is a value typically measured outdoors [46].

Table 4.1: Wind Field Characterization

Distance from source [m]	1	4	8
Mean wind speed [m/s]	1.13	1.01	.34
Coefficient of variation [%]	15.4	21.2	52.0

The robot start area is located in the corner opposite the plume source. An overhead camera tracking system, combined with a radio LAN among the robots and an external workstation, is used to log position data during the trials, determine trial completion, reposition the robots between trials, and emulate inter-robot communication signals. These signals require each robot to have access to the local range and bearing of the signaling robot, and since hardware to provide this information is not available, virtual sensors are used. The arena layout, as seen from the overhead camera, is shown in Figure 4.2b.



(**a**) (**b**)

Figure 4.2: (**a**) Real-robot arena. Plume source visible in upper left. (**b**) Real-robot arena as seen from overhead camera.

Trials of different group size are interleaved, and inactive robots are automatically positioned at recharging stations. Intermittent charging enables experimental

sessions to run for over 4 hours nonstop. A bank of recharging stations is shown in Figure 4.3. Each robot is equipped with a pair of servos underneath its chassis that make contact with the metal plates on the floor when instructed to do so by the camera system. Automation of the inter-trial repositioning and recharging procedures greatly increases the rate at which experiments can be run and reduces wear on the robots from human handling. These two factors are instrumental in the extraction of systematic data from a multiple robotics test-bed.

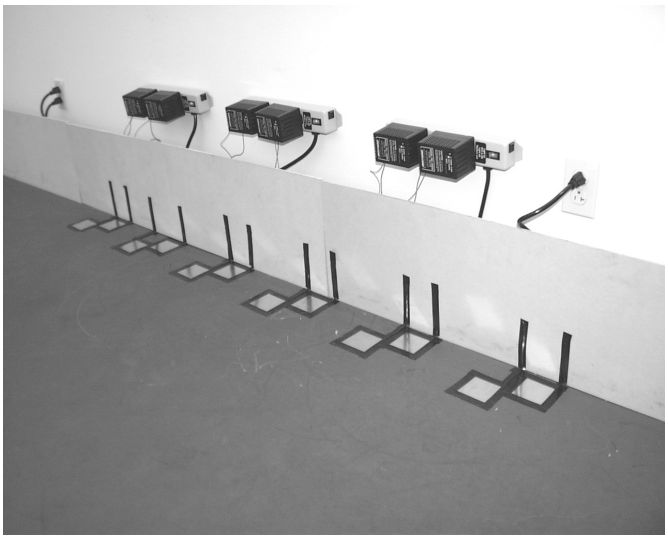


Figure 4.3: One bank of 6 recharging stations. Servos underneath each robot make contact with the metal plates on the ground after the robots are positioned by the overhead camera system.

4.3 Odor Sensor

While many types of odor sensing technology currently exist [80], a good combination of ease of transduction, reversibility, reproducibility, tunability, ease of production, robustness across environments, miniaturization, and speed is offered by carbon-doped polymer sensors [62]. This odor sensor detects the presence of an airborne substance through a change in the electrical resistance of a chemically sensitive carbon-doped polymer film [32]. While this type of sensor can lack baseline stability, it is very fast (response times $<$ .1 s [51]), and signal processing techniques can be used to

counteract its baseline drift. Carbon-doped polymer sensors are used in this work.

Sensors are fabricated from solutions consisting of 20% carbon black and 80% polymer (poly-vinylpyrrolidone) dissolved in dichlormethane, using methods as described in [26]. The conducting polymer solution is spray coated [72] onto a surface mount universal board so that the sensor film closes the circuit between two mounting pads. Polymer solution is applied until sensor resistance nears 100 kOhm, and baseline resistances typically settle to a value between 30 and 300 kOhm after a 24 hour drying period. A sensor closeup can be seen in Figure 4.4.

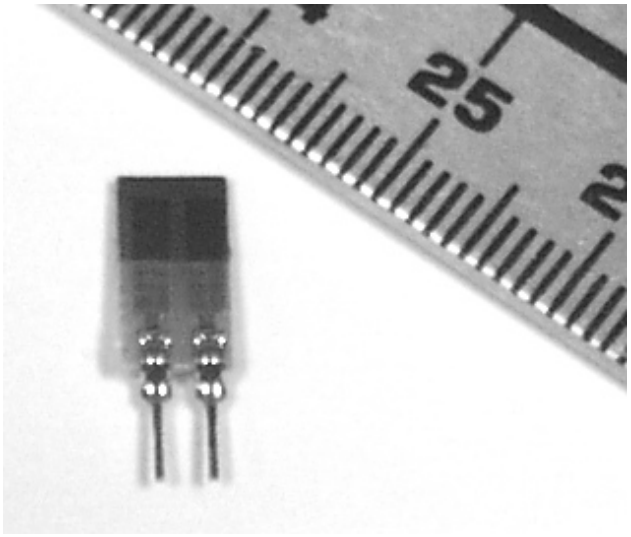
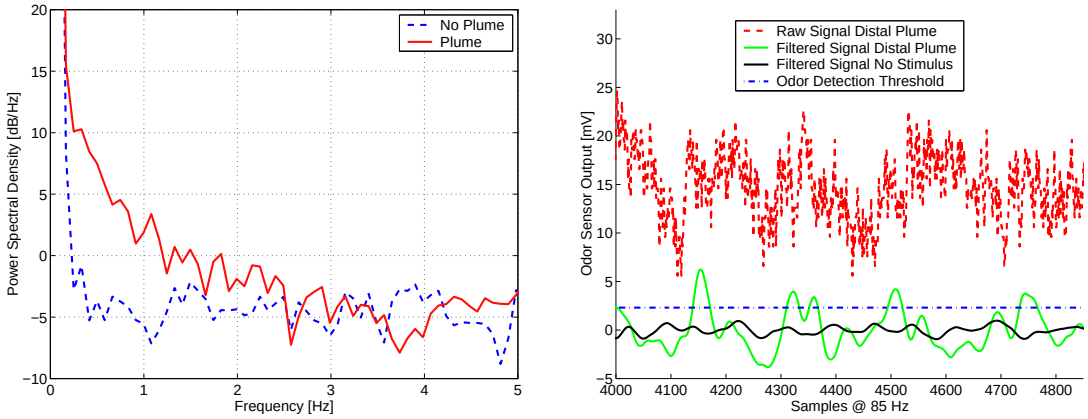


Figure 4.4: Odor sensor closeup.

The interface circuitry applies an input bias voltage across a multiplexer selectable range resistor to generate a current through the sensor via a Wilson current source. The output voltage across the sensor is then filtered to remove high frequency noise and buffered for reading. The variable bias voltage and selectable range resistor allow a wide range of sensor baseline resistances (10 kOhm to 10 MOhm) and automatic calibration, an important feature because polymer sensors are difficult to fabricate precisely and their baselines drift over time. The calibration procedure consists of switching through all range resistors with the bias voltage centered (and no stimulus present), choosing the resistor that results in an output closest to the desired output,

and then adjusting the bias voltage until the desired baseline output is achieved. The resistor and bias values are then stored for later use. The desired output value is 25% of the ADC's range, as the sensor values are more likely to drift up than down.



(**a**) (**b**)

Figure 4.5: (**a**) Power spectral density of the odor sensor output when no stimulus is present and when the robot is in the distal end of plume. (**b**) Raw distal plume data, filtered distal plume data, and filtered baseline data. The threshold is 4 std above 0.

Previous versions of the interface circuit used a local analog feedback loop to maintain the output voltage at a constant level. However, this low-pass hardware filtering attenuated not only the sensor drift but the signal as well, reducing sensor sensitivity. Sensitivity is crucial for the study of plume tracing, because the agents must be able to sense a meaningful plume structure, not simply respond when very close to the odor source. In our indoor experimental set-up, room ventilation is limited, so enhancement of the plume signal is not an option. Thus, instead of using analog feedback, the output signal is digitally filtered and an odor hit is recorded whenever the filtered signal rises above some threshold. A sixth order Butterworth bandpass filter is used, and the filter parameters are set by comparing the power spectral density given no stimulus with the power spectral density when the robot is stationary in the distal part of the plume. Given that a frequency range that provides the highest possible signal to noise ratio is desired, bandpass cutoff values of .3 to 1.8 Hz were chosen based on the data shown in Figure 4.5a. Although the

sensors can respond at higher frequencies, no information is available above 2 Hz given the transduction circuitry and experimental conditions. The amplitude threshold for odor detection is set at 4 times the baseline standard deviation (recorded from 10,000 samples taken at an average rate of 85 Hz following calibration) to render false positives improbable.

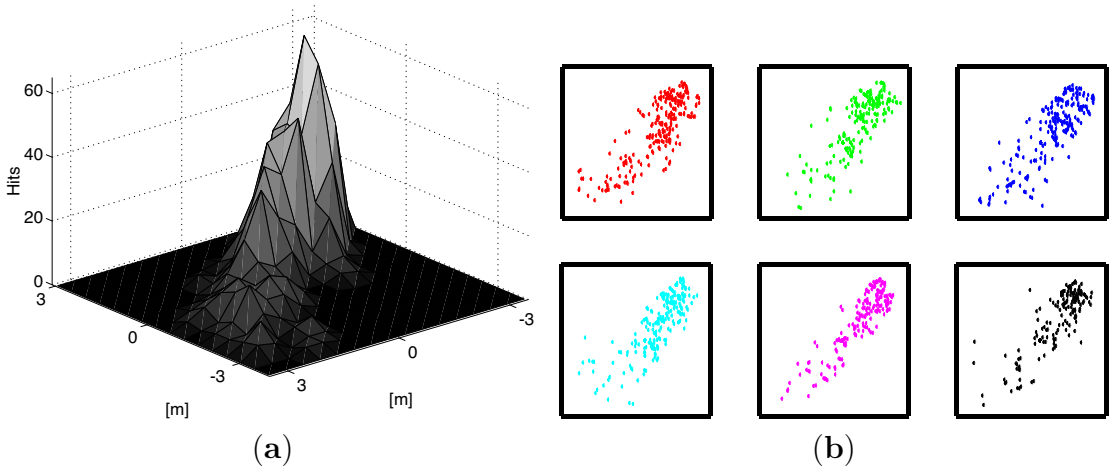
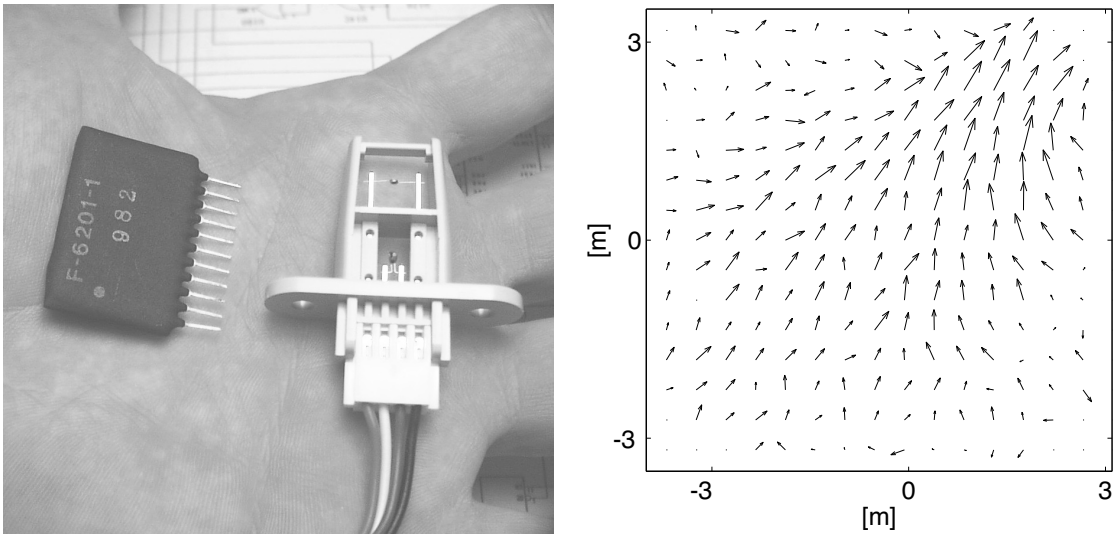


Figure 4.6: (**a**) Total plume hits received by 6 real robots over 1 hour while performing a random walk behavior. The well defined plume boundary indicates the plume envelope is stable over time. (**b**) Plume hits received by 6 individual real robots over 1 hour while performing a random walk behavior. Similarity between maps suggests there are no significant differences between robots.

When executing the odor localization algorithm, the odor sensor polling rate averages 85 Hz. Because the robot CPU is performing the polling, the filtering, and handling all other tasks the robot requires (e.g., communications, high-level motor control, and memory management), the sensor polling rate is not precise, and we do not use a real-time Linux kernel (which could provide reliable, although slower, polling rates) due to the overhead it requires. This timing imprecision is not taken into account by the digital filter, and treatment of the polling jitter, through, for example, the use of a dedicated microcontroller to take sensor readings, could increase sensitivity. However, the combination of the calibration procedure and digital filtering produces a robust binary odor detection sensor. Figure 4.5b compares raw and filtered data from the distal end of the plume against filtered baseline data from the

same sensor. The detection threshold is plotted 4 std above 0, and the raw data has been DC shifted about -3 V for ease of presentation. The presence of odor hits 8 m from the plume source shows that a significant plume stimulus exists to be tracked, even in the distal plume region where odor information is intermittent. Mapping the plume using a random walk behavior indicates that the plume is stable over time and across robots (see Figure 4.6a and Figure 4.6b).

4.4 Wind Sensor



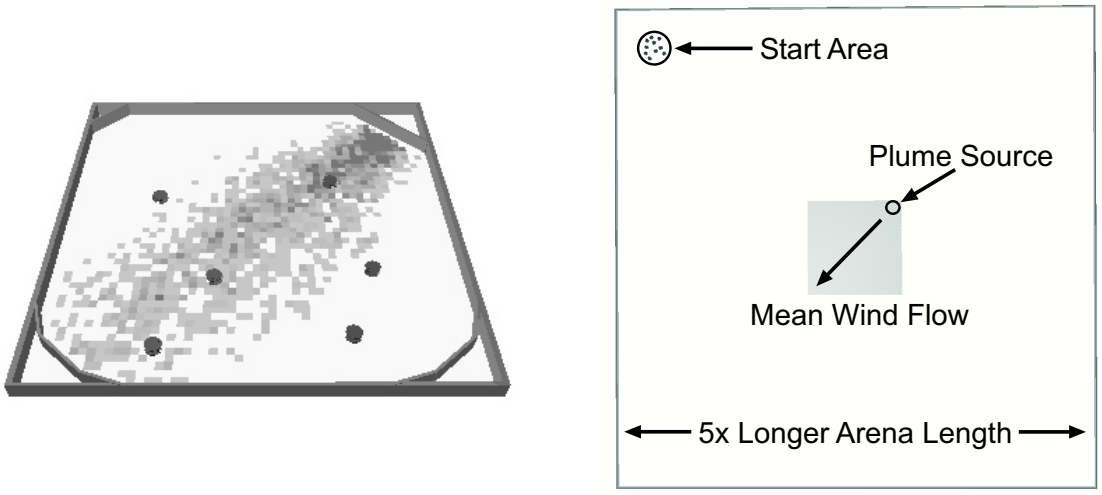
(**a**) (**b**)

Figure 4.7: (**a**) Wind sensor closeup. Sensor circuitry shown on left. (**b**) Average wind direction in plume traversal arena as measured by the real robots (2102 individual samples averaged spatially). Plume source at upper right. Arrow lengths are proportional to the uniformity of flow direction at the tail of each arrow.

The anemometer (shown in Figure 4.7a) is a Shibaura F6201-1 air flow sensor. It has previously been used to study odor localization [46], and it can sense wind flow down to .05 m/s. It is enclosed in a tube to provide unidirectional sensitivity, which, combined with a scanning behavior, allows the robot to measure wind direction. When wind direction information is required, the robot first rotates 90 degrees, then

rotates slowly 360 degrees while reading the wind sensor output, and finally rotates back to the heading corresponding to the highest sensor value. The robot takes the shortest path back to the desired heading, and either over or under rotates to the target to account for the 1 s time delay of the internal anemometer processing circuitry. The initial rotation reduces the probability that the robot begins facing upwind, in which case the discontinuity in the scanning behavior can degrade the resulting wind direction value. Wind sensor performance has not been fully characterized due to the requirements of a suitable testing environment (flow must be laminar), although the data from the odor localization experiments suggests it is sufficient for the given task. A wind map of 2102 individual samples averaged spatially is shown in Figure 4.7b.

4.5 Sensor-Based Simulation



(**a**) (**b**)

Figure 4.8: (**a**) Webots plume traversal arena with average plume intensity map. (**b**) Layout of larger Webot arena.

Simulation models can provide a significantly decreased performance evaluation time, which enables a more complete investigation of the system parameter space. Models also allow treatment of environmental conditions which (due to some technical

limitation) cannot be implemented physically. In this work, the use of a sensor-based simulation, which explicitly embeds models of both the agent's sensors and the sensed environment into the simulation, permits the enlargement of the plume search arena and the examination of agent performance on the full odor localization problem. Because the source declaration phase of the task can lead to elevated agent densities around the source, and thus is very sensitive to inter-agent repulsion parameters, point simulations, which can only approximate such interactions, are not able to provide faithful results. Thus, Webots [75], a 3-D sensor-based, kinematic simulator, originally developed for Khepera robots [77], is employed to systematically investigate odor localization performance in simulation. This kinematic simulator has previously been shown to generate data that closely matches real Khepera [45], [67], [66] and Moorebot [39] experiments, so there is reason to believe that real-robot behavior can be accurately captured.

(**a**) (**b**)

Figure 4.9: (**a**) Georgia Tech plume, taken from a real dye plume in a flume tank. (**b**) Caltech plume, generated by simulating particle transport based on real ocean flow data. In both plumes overall flow moves left to right, although the flow direction is more variable for the Caltech plume.

Initial simulations are performed in an arena modeled after the physical arena, as shown in Figure 4.8a, to verify that the simulator produces accurate results. These results are presented in Chapter 7. In addition, a 25 times larger (area) arena is used to study the full odor localization problem (see Figure 4.8b). The agent behavioral algorithms correspond exactly to those used by the real robots. To properly capture the plume stimulus, a series of 3000 leaky source 2-D Planar Laser-Induced

Fluorescence plume images generated in a water flume by Donald Webster and Philip Roberts at Georgia Tech [105] [103] are incorporated into the simulation. An instantaneous image of this plume is shown in Figure 4.9a. Such "plume movies," even though they do not capture the influence of the agents on plume dynamics, offer a good approximation to the discretized (packet-like) nature of odor stimulus received in real environments.

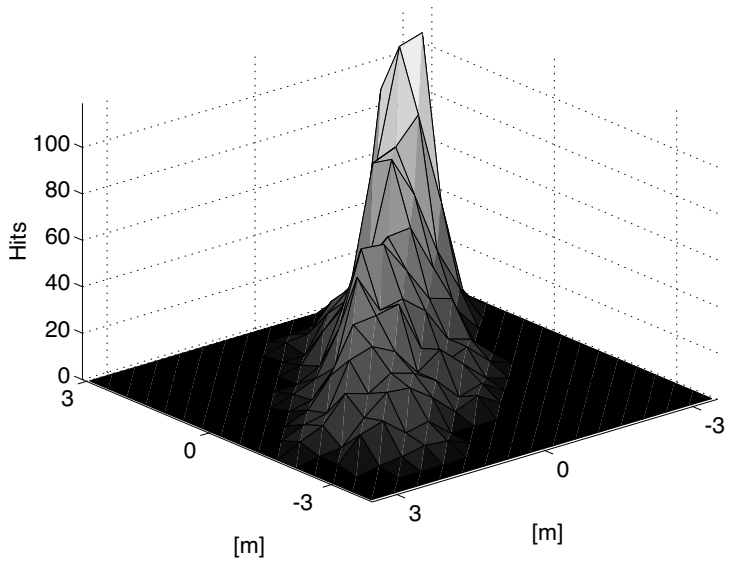


Figure 4.10: Plume hits received by 6 simulated robots over 1 hour.

The Georgia Tech plume data is scaled to imitate the average speed and envelope of the real plume data (compare Figure 4.10 and Figure 4.6a), and the odor sensitivity threshold is tuned (higher threshold leads to less odor information) based on performance observed in our real arena. In the small arena the sensor threshold is .8826 μg/L. The increased odor hit frequency observed in the simulated plume (compared to the real plume) is due to the fact that for efficiency the simulated sensors are bandwidth limited only by the update rate of the plume data (10 Hz) rather than by a bandpass filter like the one used on the real robots (.3-1.8 Hz). For the smaller arena, flow information is taken directly from the real-robot data (as shown in Figure 4.7). In the experiments performed in the larger arena, the Georgia Tech plume data

is increased in size by a factor of 2 (to make the plume traversal phase more relevant to the overall task). To create more sparse plumes, two other sensor thresholds are investigated: 3.678 μg/L (resulting in a plume with $\frac{1}{5}$ the density of the original), and 7.061μg/L (resulting in a plume with $\frac{1}{10}$ the density of the original). In an effort to emulate more open flow patterns, wind information is generated by adding $\pm$10% white noise to a constant direction parallel to the main plume axis.

Also, in the larger arena, a set of experiments incorporates another set of plume data provided by Francois Lekien and Chad Coulliette at Caltech. This plume data (1000 frames) was generated via a detailed simulation model that traced the trajectories of virtual particles as they are dropped into a simulated moving fluid. The flow data is based on readings of ocean currents off of the eastern Florida coast, and it generates a more complex plume than can be observed in the relatively narrow Georgia Tech flume tank. An instantaneous image of the Caltech plume is shown in Figure 4.9b. This plume data is scaled to be the same size as the Georgia Tech plume, and the frame update rate is scaled to render the average velocity of the two plumes equivalent.

4.6 Software

Although robot software architectures are a heavily researched area [2], there are many methods of achieving equivalent function, and particular design choices are typically not important. Therefore, the software developed to control the robots is described only briefly. The control code is written in C. A set of driver functions was developed to enable simplified access to the robot sensor and motor ports. Each controller process is divided into a number of different threads. Sensors that are polled constantly (such as the collision sensors) have a dedicated thread to handle the polling, and the sensor data is stored in memory. Incoming communication from the camera system (via a TCP/IP socket) is also handled by a separate thread. A control thread reads in the communication and sensor data from memory, performs the control logic, and issues commands to the motors. The same control code runs

on the real and simulated robots, as an interface library was developed to translate Moorebot functions into Webots function calls.

The overhead camera software is written in C++ to take advantage of library functions that were provided with the camera hardware. This software tracks the robots in the arena during experiments, repositions the robots between trials, and augments the sensory information available to the robots. Because all of the robots appear the same to the camera system, an initialization procedure is used to differentiate between robots. At the beginning of an experiment, the camera system runs through each robot IP address sequentially and issues a command to move a slight amount. The camera system can then pair each moving robot with an IP address, and it maintains this pairing by tracking the robots throughout the experiment. Tracking failures (in which the camera system confuses the IP addresses of the robots, due to the robots moving too fast or the camera updating too slowly) can be detected during repositioning between trials, and recovery is performed via a re-initialization procedure. A high level of software stability is required to generate systematic data from a multiple robot system.

Chapter 5

Collective Search

Before investigating the full odor localization problem, two of its subtasks will be examined separately, beginning with the search phase. The system design issues are addressed at a high level, and the results described in this chapter will form the basis of the abstract model of odor localization performance presented in Chapter 7. This chapter presents a quantitative analysis of the tradeoffs between group size and efficiency in collective search tasks that considers both the time-sensitive nature of search completion and the system operating cost. First, the search task is defined and a performance metric is presented that can account for all of the costs associated with the task. Next, for both random and coordinated search strategies, analytical expressions are derived that can be used to predict optimal system performance bounds given a particular task description. Also, the performance benefit of using coordinated search is shown to be dependent on the relative values of the different cost components. Finally, a sensor-based computer simulation is used to support the analytical results, suggesting that the assumptions involved in their derivation are sound [38].

5.1 Background

Search tasks, because they submit well to parallelization, are an ideal application for multi-agent systems. Search is a well-studied problem (for a review, see [9]), and there has been a significant amount of investigation into the efficiency tradeoffs

between random and coordinated search strategies [33]. However, how to assess the performance of multi-agent search systems is still an open problem. Some researchers take into account only energy used [33], while others consider only the time required until completion [6] when analyzing the performance of multi-agent systems on similar search tasks. Clearly, the performance metric used must be appropriate for the task being studied, but there is reason to believe that a more complete cost metric might offer further insight into the design tradeoffs present and aid in the comparison of results across research groups.

5.2 Search Task Description

The search task examined in this chapter can be described as follows: a group of N agents each having a sensor radius r must locate a single target contained within an enclosed 2-D arena. For simplicity, consider this arena to be a square of length L, with $L \gg r$ so that the agents are likely to disperse throughout the arena before the target is found. To ensure that the agents do not begin with full coverage of the arena (thus driving the search time to 0), initial agent deployment must be within a single deployment area of radius R. It is assumed that $L \gg R$, although the deployment area may be located anywhere within the arena. Figure 5.1 shows a schematic of an example task layout.

5.2.1 Performance Metric

Performance on this search task can be measured in terms of T_s, the time elapsed before an agent detects the target, and D_s, the sum of the distances traveled by each of the agents. D_s then correlates to the amount of energy needed for system operation. There are also setup costs that need to be considered in a complete system evaluation. Since these measures are physically independent, a composite metric incorporating a task-specific weighting of these basic factors can be considered. For N agents,

$$C = \alpha T_s + \beta D_s + \gamma N. \tag{5.1}$$

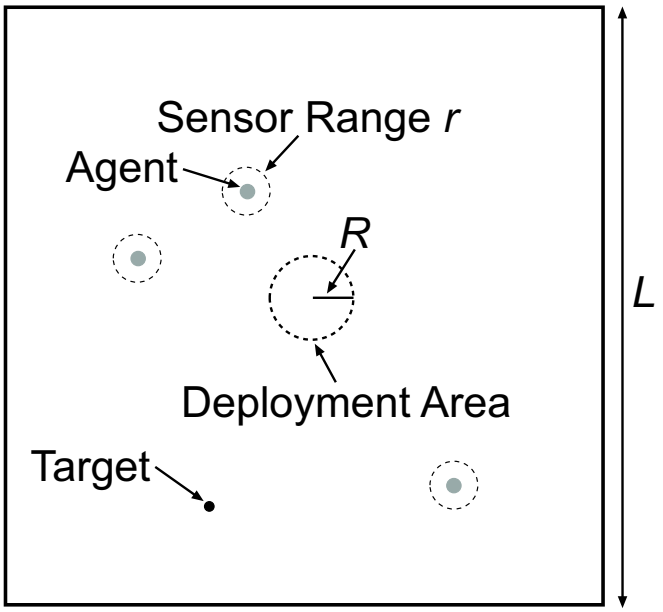


Figure 5.1: Example task layout in which $N = 3$.

There are three basic cost components. α is taken to be the cost per unit time of not completing the task, β is the cost per unit distance of running the system, and γ is the initialization cost per agent. C represents the total cost incurred before the task is completed. By choosing specific values for α, β, and γ the appropriate relationship between time required, energy used, and initial cost can be generated for evaluating any particular application.

To simplify the analysis, if the control algorithm used maintains an average speed v across time, the total distance traveled can be approximated by the time required to complete the task:

$$D_s = T_s N v. \tag{5.2}$$

Substituting into equation (5.1) above,

$$C = \alpha T_s + \beta T_s N v + \gamma N. \tag{5.3}$$

Thus, for any given group size, the system cost can be obtained directly from the

time required. Although C is the metric used in the analysis section of this paper, in order to facilitate comparison across environments, it can be normalized by the minimum completion cost in order to generate a unitless performance metric P. The minimum cost is based on the optimum values for the given task (T_{MIN}, D_{MIN}) for a single agent with prior knowledge of the source location, as determined from the average distance between starting location and target location as well as maximum agent speed:

$$P = \frac{\alpha T_{MIN} + \beta D_{MIN} + \gamma}{C}. \tag{5.4}$$

This form of P ensures that for any cost α, β, or γ greater than 0, the optimal system will achieve a performance of 1, and any that requires more time, distance, or agents will have a performance less than 1.

5.3 Deriving Performance

The stochastic nature of real systems (e.g., from sensor noise, agent movement, or deployment and target location variation) means that for each trial the cost to complete a search task is drawn from some distribution. For some applications the designer is interested in minimizing the average cost of system operation, and for other tasks the value of interest is a composite of the average cost and its variation. This work focuses on bounding the cost of a given percentage of trials, that is, determining the cost C which exceeds the cost of some fraction S of all trials in that particular environment.

Expressions for the optimal cost of random and coordinated search strategies are derived in the following sections. For clarity, a summary of the variables used is provided in Table 5.1.

5.3.1 Random Search

In a system performing random search, the agents move randomly while searching for the target without any explicit attempt to partition the space amongst agents or avoid

Table 5.1: Summary of Parameters and Variables

N	Number of agents	r	Sensor radius
L	Arena length	R	Deployment area radius
T_s	Time to complete task	D_s	Total distance to complete task
α	Cost of not finishing	β	Cost of operation
γ	Initialization cost per agent	C	Total system cost
v	Average agent velocity	P	System performance measure
S	Desired performance bound	g	Probability of system finding target
t	Time interval	k	Minimum dispersion time
η	Sensor detect probability	p	Probability of agent finding target
$x^\star$	Optimal value for variable x	Z	Single agent trial search time

searching the same area multiple times. Given that a system has some probability g of finding the target during a time interval t, the probability of finding the target during a particular interval is simply g multiplied by the probability of not finding the target in all previous intervals. Thus the probability S that the target is found before some time T_s can be expressed as the sum of a geometric series:

$$S = \sum_{t=1}^{T_S} g(1-g)^{t-1}. \tag{5.5}$$

To solve for T_s, the series can be simplified as follows:

$$S - (1-g)S = g - g(1-g)^{T_S} \tag{5.6}$$

$$T_s = \frac{\log(1-S)}{\log(1-g)}. \tag{5.7}$$

The above equation describes the time to complete the task based on search success probability and desired performance bounds. To be more accurate, however, a term needs to be added to account for the fact that the agents cannot begin the task with full coverage of the entire search area (because all agents start within the deployment area):

$$T_s = \frac{\log(1-S)}{\log(1-g)} + k. \tag{5.8}$$

The factor k represents the time required to cover the distance between the deployment area and target, and serves as a lower bound of the time needed to perform the task (i.e., $k = T_{MIN}$).

The probability g can be decomposed in terms of the number of individual agents N performing the task, and the probability p of a single agent scanning the target per time period t. In turn, p can be approximated using the ratio of the area scanned per time t to the total area of the arena L^2. A sensor detection probability η, modeled here as the probability of target detection given that the target enters the sensor range, factors in as well:

$$p = \frac{2rv\eta}{L^2}. \tag{5.9}$$

Assuming that the probability of each agent succeeding is fully independent, given p and a group size of N agents, the probability g of the system locating the target during a time period t can be calculated to be

$$g = 1 - (1 - p)^N. \tag{5.10}$$

Plugging this value into equation (5.8):

$$T_S = \frac{\log(1 - S)}{N \log(1 - p)} + k. \tag{5.11}$$

Now the optimum number of robots, the optimum time, and the optimal cost for a given task can be derived. Let

$$Z = \frac{\log(1 - S)}{\log(1 - p)}. \tag{5.12}$$

Z represents the length of time necessary for S percent of trials using a single agent to locate the target (after the initial dispersion period k). Substituting into equation (5.11):

$$T_s = \frac{Z}{N} + k. \tag{5.13}$$

Substituting this value into equation (5.3), another form of the total system cost is derived:

$$C = \frac{\alpha Z}{N} + \alpha k + \beta v Z + \beta N v k + \gamma N. \tag{5.14}$$

Assuming that all the parameters in the system are fixed except N, determining the critical points leads to an expression for the optimal number of robots $N^\star$. Taking the derivative of C, setting it equal to 0, and then solving for $N^\star$:

$$\frac{\delta C}{\delta N} = -\frac{\alpha Z}{N^2} + \beta v k + \gamma = 0 \tag{5.15}$$

$$N^\star = \sqrt{\frac{\alpha Z}{\beta v k + \gamma}}. \tag{5.16}$$

The positive root is taken because the number of agents must be positive, and the second derivative $\frac{\delta^2 C}{\delta N^2}$ is positive so $N^\star$ occurs at a minimum value of C. Plugging this value into equation (5.13) produces the optimal search time $T_s^\star$:

$$T_s^\star = \sqrt{\frac{Z(\beta v k + \gamma)}{\alpha}} + k. \tag{5.17}$$

Equations (5.14) and (5.16) can be combined to arrive at the optimal cost $C^\star$ for searching a particular environment using random search:

$$C^\star = \alpha k + 2\sqrt{(\beta v k + \gamma)\alpha Z} + \beta v Z. \tag{5.18}$$

$C^\star$ breaks down into essentially three terms. The first, αk, represents the minimum cost of having to disperse throughout the arena before finding the target. Generally, however, because the sensor radius is assumed to be small compared to the arena size, $Z \gg k$ so this term will not have a substantial influence on the overall cost. The second term in equation (5.18) represents the cost of not finishing the task accrued while performing the task (e.g., the damage done by the target before it can

be located and neutralized). This term will dominate when α is the dominant cost component. The coefficients β and γ play a role in this term as well because they influence the optimal number of agents and thus the speed at which the task can be accomplished. The third term represents the cost of searching the required area to complete the task. It will dominate when β is the dominant cost component. It has a relatively simple form because the number of agents in the system does not influence the size of the area that must be searched. Substituting back in for Z, the optimal random search cost can be specified in terms of the component costs and basic task parameters:

$$C^{\star} = \alpha k + 2\sqrt{(\beta v k + \gamma)\alpha \frac{\log(1-S)}{\log(1-\frac{2rv\eta}{L^2})}} + \beta v \frac{\log(1-S)}{\log(1-\frac{2rv\eta}{L^2})}. \tag{5.19}$$

5.3.2 Coordinated Search

The performance of coordinated search algorithms has been well studied [9]. In terms of the variables described in this paper, the results are as follows. Coordinated search for N agents requires breaking the search space into N equal partitions, and assigning a single agent to sequentially search each one. The total amount of time T_{Pass} required for each agent to make a single pass over its entire partition can be stated in terms of the arena size L, agent speed v, and sensor range r:

$$T_{Pass} = \frac{L^2}{2Nrv}. \tag{5.20}$$

Given a sensor detect probability η, the total number of passes M each robot must make can be expressed similarly to equation (5.7) above:

$$M = \frac{\log(1-S)}{\log(1-\eta)}. \tag{5.21}$$

Thus the total time required for the optimal system to search the arena is as follows:

$$T_{_S} = T_{Pass}M + k = \frac{\log(1-S)L^2}{\log(1-\eta)2Nrv} + k. \tag{5.22}$$

Where k represents the time required for the robots to move from the deployment area to their respective partitions. If Z_{cor} is defined as follows:

$$Z_{cor} = \frac{\log(1-S)L^2}{\log(1-\eta)2rv}. \tag{5.23}$$

Equation (5.13) is again reached:

$$T_{_S} = \frac{Z_{cor}}{N} + k. \tag{5.24}$$

All of the optimal value derivations in the previous section now apply.

5.3.3 Performance Comparison

Comparing the optimal costs of different search algorithms can provide insight into the conditions under which each type might be more suitable. This can be done by looking at the ratio of the optimal cost of random search $C^{\star}_{rnd}$ to the optimal cost of coordinated search $C^{\star}_{cor}$. The choice of algorithm influences only the value Z, and Z_{rnd} (equation (5.12)) and Z_{cor} (equation (5.23)) are defined above. As shown in [34], the ratio Z_{rnd} to Z_{cor} simplifies as follows:

$$\frac{Z_{rnd}}{Z_{cor}} = \frac{\frac{\log(1-S)}{\log(1-\frac{2rv\eta}{L^2})}}{\frac{\log(1-S)L^2}{\log(1-\eta)2rv}} \approx \frac{-\log(1-\eta)}{\eta}. \tag{5.25}$$

The approximation holds when $\frac{rv\eta}{L^2}$ is close to 0, as is typical when the search arena is large. This equation indicates that as the sensor reliability decreases, the performance gap between random and optimal search strategies closes. However, the cost components play a role as well.

As stated in Section 5.3.1, when α is the dominant cost component, the second term in the cost function (equation (5.18)) will dominate, so assuming all cost components remain constant across the different algorithms:

$$\frac{C^{\star}_{rnd}}{C^{\star}_{cor}} = \sqrt{-\frac{\log(1-\eta)}{\eta}}. \tag{5.26}$$

Likewise, when β dominates, the third term in the cost function is the most important, thus:

$$\frac{C^{\star}_{rnd}}{C^{\star}_{cor}} = \frac{-\log(1-\eta)}{\eta}. \tag{5.27}$$

Therefore, aside from sensor detect probability, tasks for which there is considerable time pressure will be more suited to random search strategies than tasks that emphasize economy of effort. This is not an unexpected finding, but this analysis formalizes the tradeoffs involved. Because the cost γ of building and maintaining different types of robots suitable for each algorithm is difficult to deal with abstractly, it is not considered here. However, it is worthwhile to note that robots capable of the coordinated action will likely cost more than robots suitable for random search.

5.4 Supporting Simulations

Formulation of the optimal search cost is straightforward, but the analysis of the random search algorithm required assumptions about the independence of the success probability over sequential time periods for a single agent as well as across agents. To verify that these assumptions are valid for this type of task, the search task was implemented in Webots and the time and distance required for groups of various sizes to succeed was recorded. To implement the random search behavior, the agents moved forward at a constant speed, making random turns (between $\frac{\pi}{4}$ and $\frac{3\pi}{4}$ rad) away from obstacles (walls and other agents) when necessary.

5.4.1 Results

The random algorithm was simulated 1000 times for group sizes from 1 to 80 agents, and the time and group distance required to complete the task were measured. The deployment area was always placed in the arena center, and the target was placed randomly throughout the arena for each trial. The dispersal time k was calculated from the arena length and the agent speed. The task and cost parameter values

Table 5.2: Task and Cost Parameter Values

Agent radius		.5 [m]
Sensor radius	r	.5 [m]
Arena length	L	100 [m]
Deployment area radius	R	10 [m]
Average agent velocity	v	2.9 [m/s]
Minimum dispersion time	k	17 [s]
Desired performance bound	S	.95
Sensor detect probability	η	.5
Cost of not finishing	α	10 [$/s]
Cost of operation	β	.0055 [$/m]
Initialization cost	γ	82 [$/agent]

selected are shown in Table 5.2. Note here η is significantly less than one and $\alpha \gg \beta$, so the random algorithm is expected to perform similarly to the coordinated search.

Figure 5.2 shows the results of calculating the costs in this system analytically compared to the costs derived experimentally. There is good quantitative agreement between the analytical and simulated results for the random algorithm, suggesting that for this task the assumptions of independence hold and the analytical results are valid. Also, it is worthwhile to note that the optimal group size for both algorithms is well above 1 (so the interest in multiple agents completing this task is warranted), and the optimal cost of the random algorithm is fairly close to that of the optimal system. This suggests that if the increased cost of adding coordination and fault tolerance into the optimal system is significant, the random system (which has fault tolerance built in because all of the agents perform the same actions) may be the most efficient.

5.5 Conclusion

This chapter presented a quantitative analysis of the tradeoffs between group size and efficiency in collective search tasks that considers both the time-sensitive nature of search completion and the system operating cost. First, the search task was defined and a performance metric was presented that can account for all of the costs associated

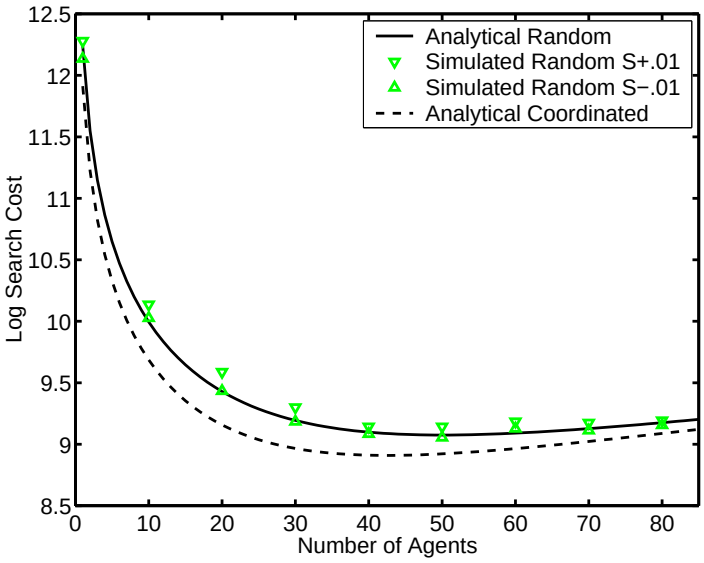


Figure 5.2: Simulated and analytical results for this search task. For the simulated data the lower triangles are above $S - .01$ of the cost data and the upper triangles exceed $S + .01$ of the cost data. Good agreement between the simulated and analytical results indicates the random search model assumptions are sound

with the task. Note that computation of the cost parameters may not be simple, but estimates are feasible. Also, while the costs used in this paper were linear functions of the task metrics, any differentiable function can be used in this framework. Next, for both random and coordinated search strategies, analytical expressions were derived that can be used to predict optimal system performance bounds given a particular task description. This analysis also allowed the prediction of the optimal number of agents required to complete a task most efficiently. In addition, the performance benefit of using coordinated search was shown to be dependent on the relative values of the different cost components, with coordinated search being less favored when the cost of not completing the task significantly outweighs the cost of operating the search system. Finally, a kinematic computer simulation was used to support the analytical results, suggesting that the assumptions involved in their derivation are sound. These assumptions, which include minimal interference between agents and uniform coverage of the given arena, will not hold in all environments, but they will

be approximately correct for many difficult applications where the area to be searched is much larger than the agent extent.

Chapter 6

Single Agent Plume Traversal Algorithms

This chapter presents an investigation into the second phase of the odor localization problem, plume traversal. A better understanding of what makes a plume traversal algorithm successful and how effective algorithms differ across environments can guide the creation and parameterization of plume traversal behaviors. First, the problem of plume traversal is recast as the task of obtaining the next odor hit, and a set of metrics that provides detailed information about algorithm function is presented. Then, several odor localization algorithms are described, and it is demonstrated that algorithm parameters can be tailored to particular plume characteristics for improved performance. Also, the next-hit analysis is shown to capture the performance of some types of algorithms more accurately than others, and it is concluded that this failure stems from intrinsic shortcomings of some of the algorithms tested. Moreover, based on this analysis, general properties required of successful turbulent odor plume traversal algorithms are described.

6.1 Plume Traversal

The initial search procedure is essentially a coverage problem that depends largely on the characteristics of the search area (as examined in the previous chapter), and the source declaration procedure is likely to be highly specialized for each task. Therefore, the aspect of odor localization that depends most heavily on plume characteristics is

that of plume traversal, or following the trail of odor packets upstream to the source. This observation suggests that for the purposes of tailoring system parameters to plume characteristics, it is most useful to focus only on the plume traversal phase. The first step in this process is to define an efficient way in which to quantify algorithm performance so that many plume-algorithm pairs can be analyzed. Eventually, these comparisons should generate a high-level mapping between desirable algorithm properties and the characteristics of the plume stimuli being tracked. Such a mapping will facilitate the matching of system parameters to the application environment (yielding higher efficiency [101]) and will aid in the development of deployable systems. Also, a better grasp of the relationship between plume stimuli and tracking algorithms should enable new avenues of investigation into the function of biological systems.

6.2 The Next-Hit Analysis

In the distal plume region, where plume information is intermittent, sensors must be able to respond quickly to derive information from individual plume packets as they pass (see Figure 3.3 for a diagram of the plume traversal task). As discussed in Chapter 4, the currently available odor sensors that possess the required sub-hertz sample times are noisy, so binary odor information is all they can reliably provide. There may be information encoded within the fine structure in the distal part of the plume [47, 102]. However, due to the highly stochastic nature of turbulent fluid flow and the odor-packet nature of the plume, it is unclear that complex sensing (via graded intensity information or larger fixed sensor arrays) would benefit an odor localizing agent when flow information is available through other means.

Assuming only binary odor information from a single sensor is available, it becomes possible to restate the problem: given that an agent has just received an odor hit, what strategy will maximize the appropriate combination of likelihood, distance, and speed of receiving another odor hit in the upstream direction (i.e., closer to the source)? This formulation necessitates another set of performance metrics: P_H, the probability

of getting another plume hit in the short term (i.e., before it becomes more efficient to revert to the plume search procedure), T the expected time until the next plume hit, and X, Y, respectively the expected next-hit locations in the down-stream and cross-stream directions. Assuming that these metrics approximate the actual range of values encountered along the length of the plume, they can be combined to produce the probability P_{F} that an initial plume hit from the search phase will result in a successful approach of the source:

$$P_{F} = P_{H}^{\frac{L}{X}}. \tag{6.1}$$

L is the expected necessary plume traversal distance. As above, P_{H} is the probability of getting another plume hit in the short term and X is the expected next-hit location in the down-stream direction. Thus, P_{F} is the probability that an agent will receive the required number ($\frac{L}{X}$) of consecutive odor hits to traverse the expected plume length, resulting in a successful approach of the source.

Table 6.1: Summary of Evaluation Metrics and Variables

P_{H}	Next-hit probability during plume traversal
T	Expected time of next odor hit
X,Y	Down and cross stream expected next-hit locations
P_{F}	Probability of approaching plume source
L	Expected plume traversal distance
P_{SR}	Probability of an odor hit during search phase

There are tradeoffs among these metrics: for fastest plume traversal, a low search time T combined with a high inter-hit upstream traversal X is ideal, but because odor information is typically most dense in the direct vicinity of the most recent odor hit, this combination generally calls for a lower probability of approaching the plume source P_{F}. This decrease in P_{F} is due to the fact that when the expected plume traversal distance is much greater than the average inter-hit upstream traversal ($L \gg X$), P_{F} is more sensitive to decreases in the probability of receiving the next hit P_{H} than increases in X. In order to optimize the full odor localization task in a

given environment, the expected cost of losing contact with the plume (i.e., the cost of an additional search phase, which depends upon the probability per unit time of finding the plume in the search phase of the task P_{SR}) must be weighed against the expected plume traversal performance decrease necessitated by increasing P_F. For the purposes of this chapter we will assume that P_{SR} is very low (i.e., there is a high cost of losing the plume) and thus P_F should be maximized for best system performance.

6.3 Plume Traversal Algorithms

6.3.1 Biological Inspiration

It is not obvious how to generate a good plume tracking algorithm. Upon sensing an odor signal, a reasonable policy is to move directly upwind, because a good immediate local indication of source direction under such circumstances is the instantaneous direction of flow [25]. When the odor is no longer present, a good strategy is to perform a local search (known as casting in the biological literature) until an odor packet is reacquired, as the location of the previous packet encounter provides the best immediate estimate of where the next will occur. This type of surge-cast behavior has been observed in both in flying [99, 1] and walking [50] moths and its performance has been studied in simulation [7].

The previous work on the surge-cast category of odor localization algorithms [7] was aimed at studying biology, which limited the sensory and behavioral time scales investigated. When applying these ideas to artificial systems, however, the separation between control algorithm and underlying hardware is much more clear, and it no longer makes sense to constrain behavior strictly by sensory response characteristics. Therefore, key aspects of the search behavior, such as surge duration and casting locality, can be parameterized and subsequently optimized for each plume type.

6.3.2 Algorithm Descriptions

The agent is assumed to have access to binary odor information, flow direction, distance of travel, and time passage. Let A be an algorithm that defines how an agent's location $(x(t), y(t))$ evolves over time t between odor hits. Upon each odor hit, an agent samples the flow direction and proceeds according to A until the next odor hit, at which time the process repeats. Four simple algorithm types are explored: `Straight`, `Step`, `Zig-Zag`, and `Spiral`. `Straight` proceeds upwind at a fixed velocity for a fixed amount of time. `Step` proceeds upwind for a fixed distance at a fixed speed and then waits at that location for a fixed amount of time until declaring the plume lost and reverting to a plume search behavior. `Zig-Zag` performs a counter-turning procedure of a given angle, step length, and speed for a fixed amount of time. `Spiral` begins with a step upwind at a fixed speed and then moves outward in a constant spiral for a fixed amount of time. The traversal speed in the spiral increases toward a maximum as the agent gets farther from the center, as this type of movement can be implemented on two-wheeled vehicles (and has been used in previous work [40]). Diagrams showing these behaviors can be seen in Figure 6.1 and the corresponding parameters can be seen in Table 6.2.

Table 6.2: Algorithm Parameter Definitions

`V`	Agent Velocity
`U`	Time until failure declared
`D`	Length of step/cast
`b`	Heading angle from upwind
`g`	Spiral gap distance

6.4 Algorithm Evaluation

In order to adequately investigate the parameter space and acquire a relationship between algorithm parameters and plume characteristics, methods of evaluating traversal performance metrics must be available. Evaluation could be accomplished by ac-

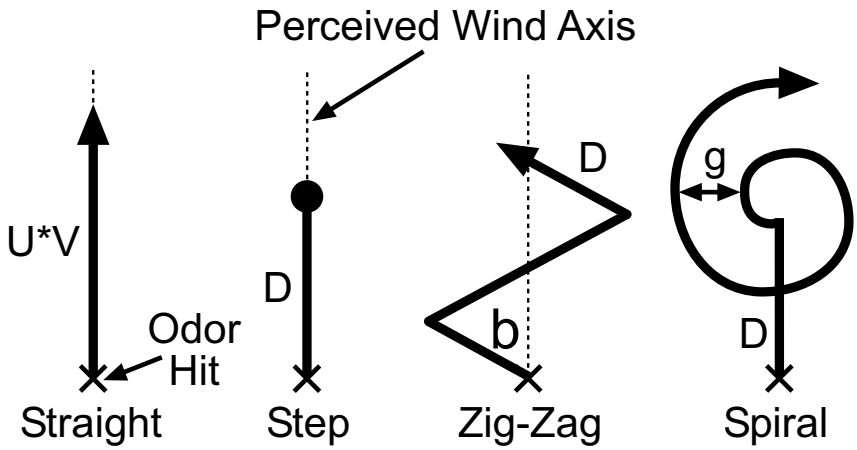


Figure 6.1: Plume traversal algorithms.

tually executing many plume traversal runs in the real world on different plume types using different tracking algorithms, but this method is likely to require too much time to be feasible. In this chapter, two different methods of metric evaluation based on simulated plume input are used. The next-hit analysis allows a detailed view of algorithm function, while direct evaluation provides more accurate performance data.

6.4.1 Next-Hit Metric Generation

Recall that A is an algorithm that defines how an agent's location $(x(t), y(t))$ evolves between odor hits. Let $K(A,t)$ be an instantaneous odor hit probability function which defines the probability of receiving the next odor hit while following a particular algorithm subsequent to the cessation of a previous odor hit (i.e., a "post hit's eye view" of the next-hit landscape averaged over all possible odor hits). For the purposes of this analysis $K(A,t)$ is assumed to be monotonically decreasing at large t, either because the agent has moved away from the plume or vice versa. For a particular environment, given a particular A, $K(A,t)$, and P_{SR}, the plume traversal metrics can be computed as follows:

$$t_S = \min t : K(A,t) < P_{SR}, \qquad t \gg 0 \tag{6.2}$$

$$M(t) = K(A,t)(1 - \sum_{t'=0}^{t-1} M(t')) : M(0) = 0 \tag{6.3}$$

$$P_{_H} = \sum_{t=0}^{t_{_S}} M(t) \tag{6.4}$$

$$T = \sum_{t=0}^{t_{_S}} tM(t) \tag{6.5}$$

$$X = \sum_{t=0}^{t_{_S}} x(t)M(t) \tag{6.6}$$

$$Y = \sum_{t=0}^{t_{_S}} y(t)M(t). \tag{6.7}$$

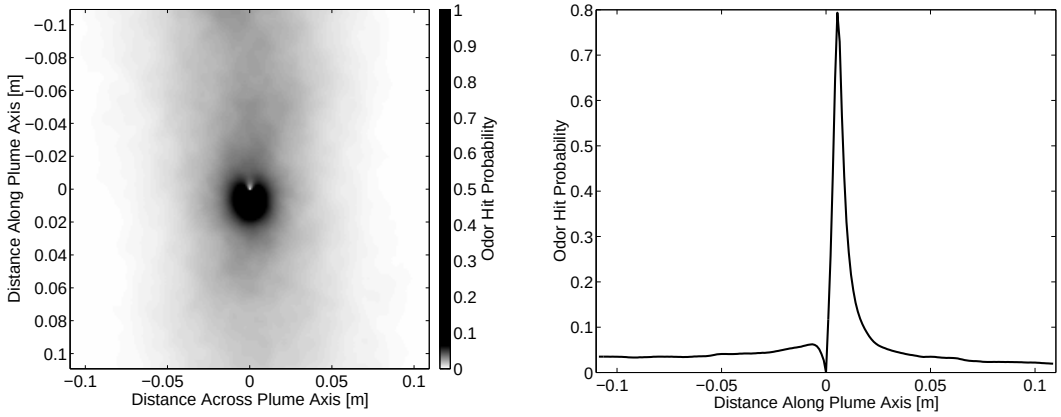
The time $t_{_S}$ represents the point at which continuing to search for the plume results in a lower probability of getting an odor hit than transitioning to the search phase behavior, and $M(t)$ represents the probability density function of getting a hit at each instant of time t. Time is considered to be discrete. Note that because A and therefore $K(A,t)$ are arbitrary, $M(t)$ will be an arbitrary summed series and will not have a closed form. This is acceptable because the total number of calculations needed to generate all the metrics scales linearly with $t_{_S}$ when $K(A,t)$ is known. Using $M(t)$, the calculations of $P_{_H}$, T, X, and Y are straightforward.

The algorithm A is simple to specify, but K turns out to be more elusive. Ideally we could leverage the extensive work done involving plume models, such as the standard Gaussian [79] or one of the more recent dispersion or finite difference models [86, 92]. However, these models were developed to predict time averaged or peak observed concentration profiles, and they do not capture the specific inter-odor packet spatial and temporal relationships that are necessary to generate instantaneous odor hit probabilities. Also, because fine plume structure is not considered, using these models it is not possible to generate accurate conditional hit probabilities necessary to reliably calculate accurate next-hit statistics (representing the concept that odor packets may be clustered within the plume, and not receiving an odor hit for a particular length of time can decrease the probability of getting a future hit).

As one might expect, all of the plume detail needed to generate K is contained in

the instantaneous concentration fields generated by planar laser-induced fluorescence (PLIF) techniques [104]. Using the Georgia Tech plume data as described in Chapter 4, K was calculated for each algorithm directly, by first thresholding the concentration values to generate binary odor hits (at a threshold of .8826 μg/L) and then implementing A starting at each on/off transition that occurred in the distal half of the plume until reaching another odor hit. Only hits in the distal half were used to avoid the need for a plume find area. Time and location of the first hit for each instance were recorded in a set of frames, and the results were summed over all hits to generate the K's used for each experiment. Estimating L to be 0.5 m for the plume being studied, $P_{_F}$ can be calculated (see equation (6.1)) using the X and $P_{_H}$ values generated in equations (6.2)-(6.7). The simulated binary odor sensor does not contain noise, but the wind sensor returns values distributed in a Gaussian manner around the true upwind direction with standard deviation σ. To approximate $P_{_F}(\sigma)$ over a range of σ, we determine $P_{_F}$ at a set of discrete heading offsets in $\pm[1.22, .87, .52, .35, .17, .087, 0]$ [rad] and then combine those values weighted according to the value of σ selected. This weighting is performed by discretizing the selected Gaussian probability density function onto the heading offsets that were explicitly calculated. The heading offset values chosen for calculation were selected to permit an approximate determination of $P_{_F}$ while reducing computation times, although the degree of accuracy achieved has not been explored.

To illustrate the plume dynamics captured in this manner, the instantaneous odor hit probability map immediately post odor hit is shown in Figure 6.2a, along with a streamwise cross-section in Figure 6.2b. The observed concentration peaks would not be seen if a time-averaged map were used to produce these graphs, and these spatio-temporal dynamics could play a role in algorithm performance. The plume is 1 m long and its envelope extends to 0.2 m in the cross-stream direction.



(**a**) (**b**)

Figure 6.2: (**a**) Probability of receiving a new odor hit in the space surrounding the site of a previous odor hit immediately after cessation of that odor hit at (0,0). Negative y values are closer to the plume source. (**b**) The hit probability along the plume axis (i.e., the probability values along x = 0 in part (a)).

6.4.2 Direct Evaluation

To verify that the assumptions involved in the next-hit metric analysis are valid, performance measurements are compared with those from an odor localization test-bed that simulates the full plume traversal behavior. The Georgia Tech plume data is again used. The task layout is shown in Figure 6.3. At the start of each trial, an agent is positioned randomly within the Agent Start Area. The agent remains motionless until it receives an odor hit, and if no odor hit is received within a particular time period (1 s) it is moved to a new random location. This method of commencing each trial is meant to mimic the transition from plume finding to plume traversal that occurs in a full plume tracing task so that the distribution of agent initial locations within the plume is accurate. Upon receiving an initial odor hit, the agent follows a given algorithm until it either enters the Plume Find Area (success) or exceeds its inter-hit time out value (failure). The plume image is updated after every 0.1 s of simulated time, and after all 3000 images have been used the first is used again. Note that the size and layout of the Start and Find areas are arbitrary, and they

have a significant influence on the performance values of each algorithm. Therefore, quantitative agreement between this direct and the next-hit evaluation procedures is not expected. However, as long as relative algorithm performance values are accurate, it is likely that the next-hit analysis is capturing the important aspects of the plume traversal task.

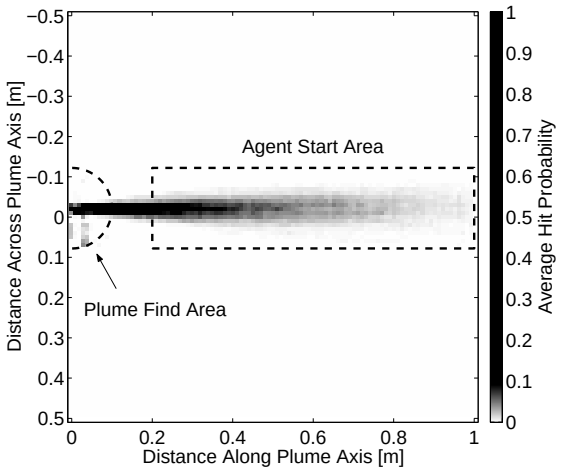


Figure 6.3: Task layout for direct odor localization algorithm evaluation. Agents receive an initial odor hit within the Agent Start Area and attempt to progress into the Plume Find Area. The plume outline shown represents the average odor hit probability generated from 3000 instantaneous plume images. The Plume Find Area has a radius equal to 10% of the plume length, and the Agent Start Area encompasses the distal 80% of the plume. Note the Plume Find Area is sized to eliminate the influence of noisy plume data near the source.

6.5 Results and Discussion

6.5.1 Parameter Search

To maximize P_F on the selected plume data, the next-hit procedure was used to examine a range of parameter values across all algorithm types. In all, 7 `Straight`, 49 `Step`, 94 `Zig-Zag`, and 48 `Spiral` algorithms were evaluated. The parameter ranges searched are shown in Table 6.3. These values were selected to cover a sub-

stantial region of the parameter space but do not represent an exhaustive search. To reduce simulation requirements and simplify evaluation procedures, the maximum agent speed `V` was limited to twice the average flow speed in the plume, and the search time-out value `U` was fixed to 50 seconds for all algorithms.

Table 6.3: Parameter Evaluation Ranges. Parameter definitions can be found in Table 6.2

Behavior	`V` [cm/s]	`U` [s]	`D` [cm]	`b` [rad]	`G` [cm]
`Straight`	0.1-10	50	-	-	-
`Step`	0.1-10	50	1-30	-	-
`Zig-Zag`	1-10	50	2-30	$\pi/2.1$ - $\pi/5$	-
`Spiral`	2-10	50	5-20	-	2-20

To illustrate the evaluative process, data is presented from the best algorithm of each type at $\sigma = 0$. These algorithms are referred to as *Straight*, *Step*, *Zig-Zag*, and *Spiral* respectively, and their parameters are shown in Table 6.4.

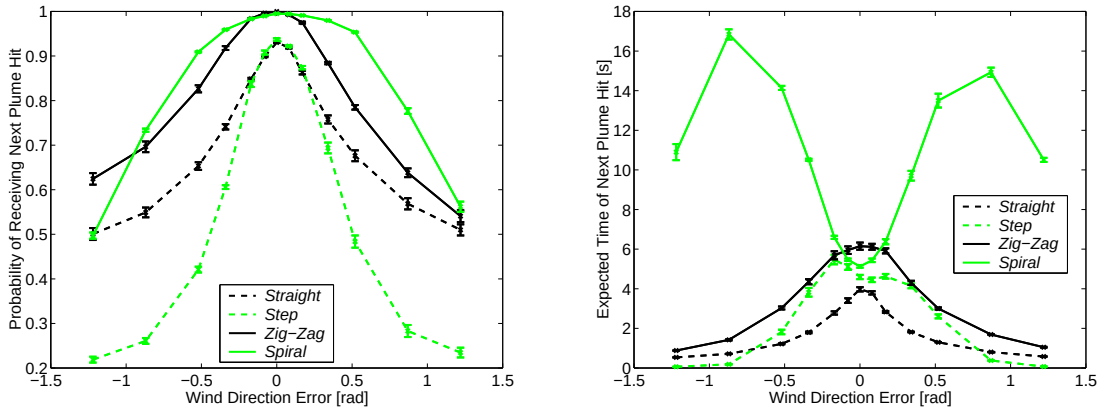
Table 6.4: Optimal Parameter Values at $\sigma = 0$

Algorithm	`V` [cm/s]	`U` [s]	`D` [cm]	`b` [rad]	`G` [cm]
Straight	2	50	-	-	-
Step	10	50	20	-	-
Zig-Zag	5	50	10	$\pi/2.5$	-
Spiral	10	50	20	-	5

For the subsequent graphs, the plume data set was split into 10 sections of 250 frames. The metrics (P_{H}, T, X, Y, P_{F}) were calculated at every 10th frame, pooled within each section, and then compared across sections so standard error information could be generated. In sample tests this procedure was shown to produce results that were not significantly different from data generated from 25 sections of 100 frames each in which all frames were evaluated, and reducing the number of evaluated frames offers a substantial savings in evaluation time. An evaluation of a single algorithm using the reduced data sampling takes 190 minutes on a 1 GHz Pentium III.

Figure 6.4a shows the probability of receiving the next odor hit P_{H} versus wind error. Note that as the wind direction error becomes larger, the probability of receiv-

ing another plume hit decreases across all of the algorithms, although by different amounts. *Step* has the lowest off-axis performance because it moves quickly out of the plume. *Straight* takes the same trajectory but a lower velocity allows more time within the plume envelope to receive an odor hit. *Zig-Zag* and *Spiral* are more effective at maintaining contact with the plume. Performance is not symmetric with respect to the wind error because the algorithms are not symmetric with respect to the plume axis. However, alternating the orientation of the behaviors after each hit will average out the differences, so the absolute value of the wind error is the critical value.



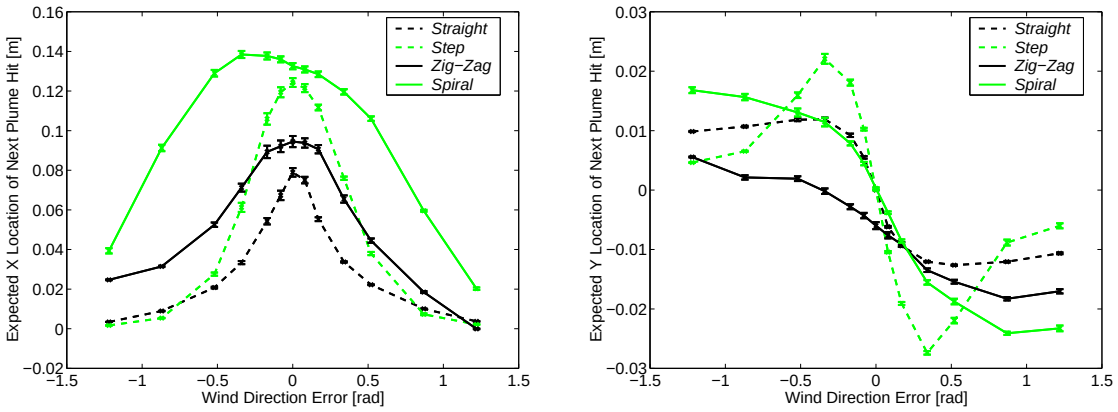
(**a**) (**b**)

Figure 6.4: (**a**) Odor hit probability for each of the different plume tracing algorithms. (**b**) Expected time to next odor hit for each of the different plume tracing algorithms. Note that if the odor hit probability is low then the expected time of the next odor hit is of little importance. All error bars represent standard error of the mean.

Figure 6.4b shows the expected time T of receiving the next odor hit versus wind error. Note that since lower times are better, *Straight* can be said to be performing the best of all the behaviors, although this is largely correlated with its low hit probability and therefore is not a useful feature. The large times shown for *Spiral* suggest that for some wind error values it is able to regain plume contact after initially exiting the plume envelope.

Figure 6.5a shows the expected upstream location X of receiving the next odor

hit for all of the behaviors versus wind error. Larger metric values are better because they indicate that fewer hits are necessary to traverse the plume, so *Spiral* is the performance leader in this category. *Straight* suffers due to its low velocity, because even though it has a higher probability of getting a hit than *Step*, it also requires more consecutive hits to reach the plume source.



(**a**) (**b**)

Figure 6.5: (**a**) Expected downstream traversal before next odor hit for each of the different plume tracing algorithms. Error bars represent standard error of the mean. (**b**) Expected cross-stream traversal before next odor hit for each of the different plume tracing algorithms. Error bars represent standard error of the mean.

Figure 6.5b shows the expected cross-stream location Y of receiving the next odor hit for all of the behaviors versus wind error. The cross-stream movement of *Spiral* and *ZigZag* renders their curves asymmetric, although the behaviors can be mirrored after each hit so the net cross-plume travel tends toward 0.

Figure 6.6a shows that for this plume with 0 wind noise *Zig-Zag* performs the best of this group of algorithms, and its near-perfect performance is not surprising given the rather simple structure of the plume being tracked. *Spiral* does almost as well, but the simpler algorithms do significantly worse. A tougher test of an algorithm's capability comes when wind information is not perfect. The algorithms shown are not the best found for larger wind direction errors, but they demonstrate the relevant trends. As the wind information degrades, performance falls as well, suggesting that investing in the development of a good wind sensor is critical. Also, more complex

plumes in which large-scale meander separates the plume axis from the wind axis may be difficult to track effectively.

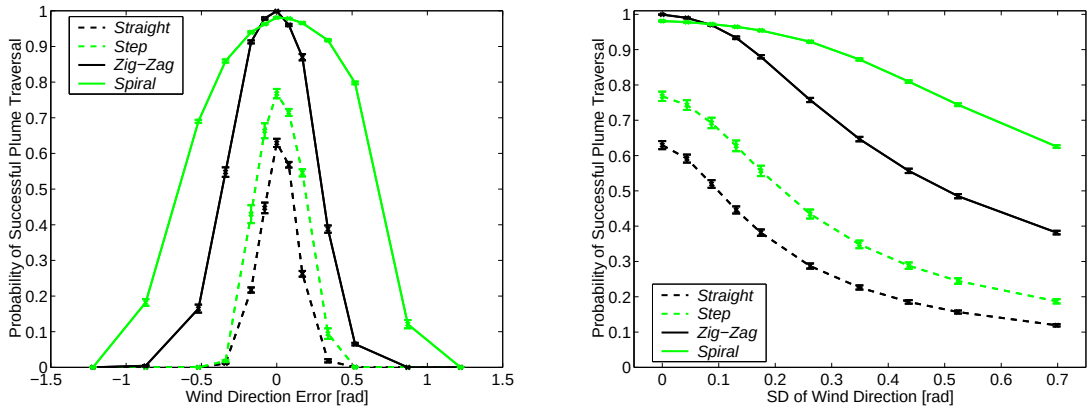


Figure 6.6: (**a**) Expected probability of successful source location $P_{_F}$ for each behavior. (**b**) Expected probability of successful source location $P_{_F}$ for each behavior. Note that the algorithms shown are optimized for $\sigma = 0$, so better performance may be achievable at higher wind variances.

Combining the above data according to the expected heading error frequency for a given wind sensor error standard deviation σ leads to the data seen in Figure 6.6b. This graph directly relates sensor error to algorithm performance, although the relationships between the individual algorithms as the wind error grows are not particularly relevant because these are only the optimal algorithms for $\sigma = 0$. However, as one might expect, the trend of decreasing performance with increasing wind error holds across all algorithms.

6.5.2 Optimized Algorithms

The above data represents the best performing algorithm of each type at $\sigma = 0$. Figure 6.7 shows the performance of the best algorithms of each type for each value of σ plotted. As one can see, `Zig-Zag` performs the best over the range of σ examined, followed closely by `Spiral`, with `Step` and `Straight` performing significantly worse. Although the correlation is approximate, the efficiency of the `Zig-Zag` behavior is not surprising because this type of casting behavior is widely seen in biological systems.

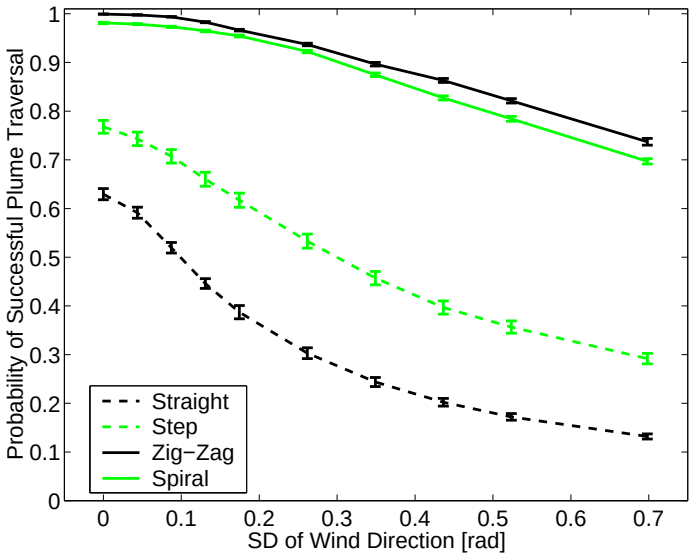


Figure 6.7: Expected probability of successful source location P_F for each behavior. Note that each point represents optimized performance for that particular wind variance.

One of the main goals of this chapter is to illustrate the creation of a mapping between algorithm parameters and plume characteristics, in order to simplify the design of efficient plume tracing algorithms. While the wind sensor noise present in the system is not technically a plume characteristic, it can be treated in a similar way to demonstrate the principles involved. Table 6.5 shows how the parameters of the best performing `Zig-Zag` algorithms evolve as σ increases.

Table 6.5: Optimal `Zig-Zag` Parameter Values versus σ. Parameter definitions can be found in Table 6.2

σ [rad]	0	.085	.175	.262	.349	.524	.698
`V` [cm/s]	5	10	10	10	10	10	10
`D` [cm]	10	15	20	15	20	20	30
`b` [rad]	1.26	1.37	1.37	1.50	1.50	1.50	1.50

As the wind error increases, the casting angles `b` become more shallow and the agent speed `V` increases, with casting distances `D` increasing within each casting angle. These findings suggest that increasing the agent speed might increase performance

at higher levels of wind error. Also, it might be possible to create a mapping from wind error and plume envelope to optimal casting angle and distance, at least when the plume envelope is well defined during the search period.

6.5.3 Evaluation Comparison

The next-hit analysis relies on two assumptions: first, the success of a plume traversal algorithm is dependent on its ability to acquire odor hits, and second, the use of a single probability P_H for receiving future plume hits is a reasonable approximation to the actual plume traversal case. To investigate these assumptions, the next-hit optimized algorithms of each type were examined on the direct evaluation test-bed described in Section 6.4.2. Ten sets of 1000 individual trials of each algorithm were run, and the means of each set were averaged to produce the performance values. Each algorithm evaluation took 245 minutes on a 1 GHz Pentium III. All error bars represent standard error.

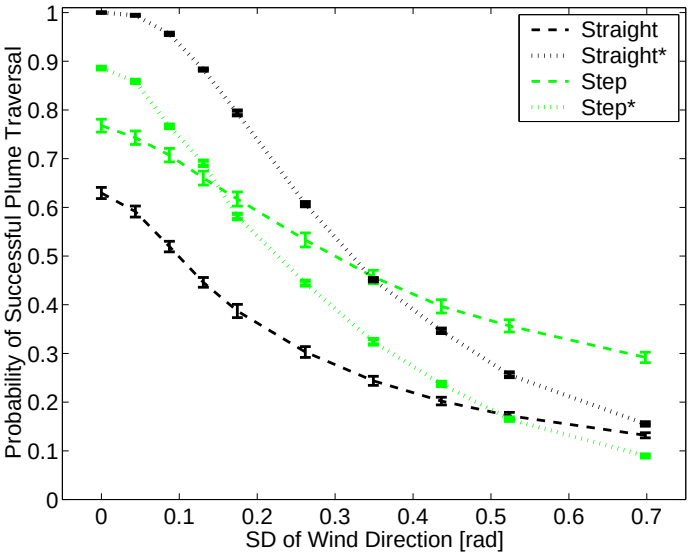


Figure 6.8: Comparison of the optimized `Step` and `Straight` algorithms under next-hit (`Step`, `Straight`) and direct (`Step`*, `Straight`*) evaluation.

Figure 6.8 shows a comparison of the algorithm performances of the `Step` and

`Straight` algorithm types as evaluated by the next-hit analysis (`Step`, `Straight`) and by the direct test-bed (`Step`*, `Straight`*). Quantitative correspondence is not expected, because the next-hit analysis does not factor in some task parameters like the size of the Source Find Area, but the relative performance of the two algorithms does not agree either. This suggests that the assumptions required for the next-hit analysis are not accurate for these particular algorithms.

`Straight`* significantly outperforms `Step`* because in the direct test-bed the Plume Find Area spans the entire plume width (see Figure 6.3). Because the best `Straight` algorithm at 0 wind noise moves straight ahead at 0.02 m/s for 50 s (i.e., 1 m), this algorithm will always enter the source found area, even though it may not always receive a second odor hit. `Step` does not receive a performance enhancement because it must receive multiple plume hits before reaching the source found area (i.e., getting one initial hit on the plume periphery will likely lead to a failure, because the plume becomes more narrow as it approaches the source). This performance advantage (which is heavily dependent upon the particular geometry chosen for the direct test-bed) extends across all wind direction variances and can explain why `Straight`* consistently performs better than `Step`*. Thus the assumption that acquiring plume hits is related to performance is violated, and one cannot expect the next-hit analysis to properly evaluate `Straight`.

The fact that the performances on the direct test-bed decrease much more quickly at higher wind direction variances than those predicted by the next-hit analysis suggests that the single-probability assumption of the next-hit analysis does not hold either. That is, these two algorithm types do not compose well, because their next-hit probabilities change significantly with each new odor hit. It is unclear how to model this effect so that the next-hit analysis can accurately capture the performance of these algorithms, and it is also largely unnecessary. Algorithms can be evaluated via the direct method at little extra computational cost, and the primary contribution of the next-hit analysis is that its limitations uncover an inherent weakness in these algorithms.

If the initial plume search process covers the area containing the plume uniformly,

the location of the initial odor hit is dependent on average plume concentration, which can be modeled by a Gaussian type plume equation. Because there is no mechanism for moving laterally within the plume in these algorithms, the agent movement is entirely determined by a Gaussian wind reading. Therefore the second-hit sampling distribution is generated by a convolution of the initial Gaussian agent distribution with a Gaussian distributed movement, which will result in a broader agent distribution than the initial. The location of the second odor hit can at best mimic the sampling distribution, which can be achieved if the plume is perfectly Gaussian and the agent is allowed to sample indefinitely. However, in real situations the plume simply does not extend beyond some distance from its axis. Some portion of the agents will spread outside the plume boundary and will not be able to reacquire the plume. Moreover, because the spatial distribution of agents spreads with each iteration, the percentage of active agents near the plume boundary will grow with each odor hit. Thus a greater percentage of agents will fail to receive each subsequent odor hit–P_H will decrease.

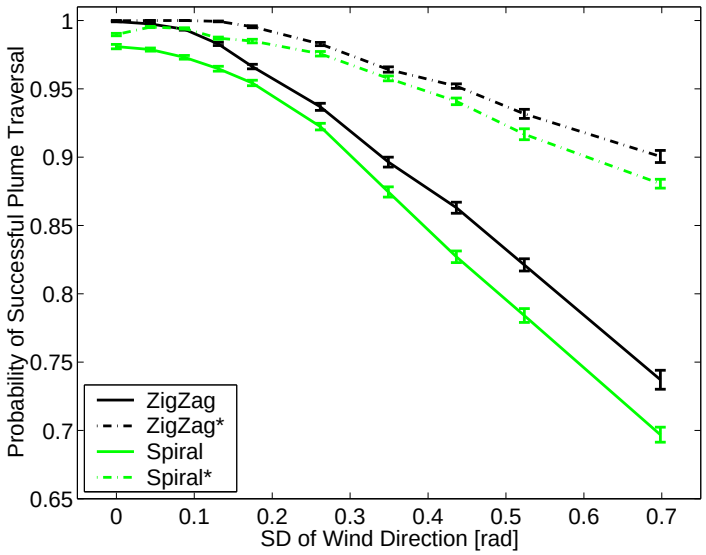


Figure 6.9: Comparison of the optimized `ZigZag` and `Spiral` algorithms under next-hit (`ZigZag`, `Spiral`) and direct (`ZigZag` *, `Spiral`*) evaluation.

Not all algorithms suffer from this limitation. Figure 6.9 shows a comparison of the algorithm performances of the `ZigZag` and `Spiral` algorithm types as evaluated by the next-hit analysis (`ZigZag`, `Spiral`) and by the direct test-bed (`ZigZag`*, `Spiral`*). Qualitative comparison holds across evaluation types, suggesting that these algorithms are better modeled by the next-hit analysis. Neither algorithm has non-local aspects that allow it to take advantage of the particular task description chosen for the direct test-bed, so no performance anomalies are observed. Likewise, both algorithms are able to move across the plume axis as part of their search, so the above analysis of the next-hit location as the composition of two Gaussians does not hold. Instead, the broad sampling of each algorithm allows the next-hit location to follow the plume concentration map itself, so there is no systematic spread of the next-hit location distribution and decline in P_H. Instead, the hit probabilities will likely increase as the agent traverses up the plume (because the plume becomes more dense), and this nonstationarity (which is not accounted for in the next-hit analysis, where the agent is essentially solving the original "difficult" initial problem repeatedly) could explain the fact that the direct performance levels consistently exceed the next-hit levels.

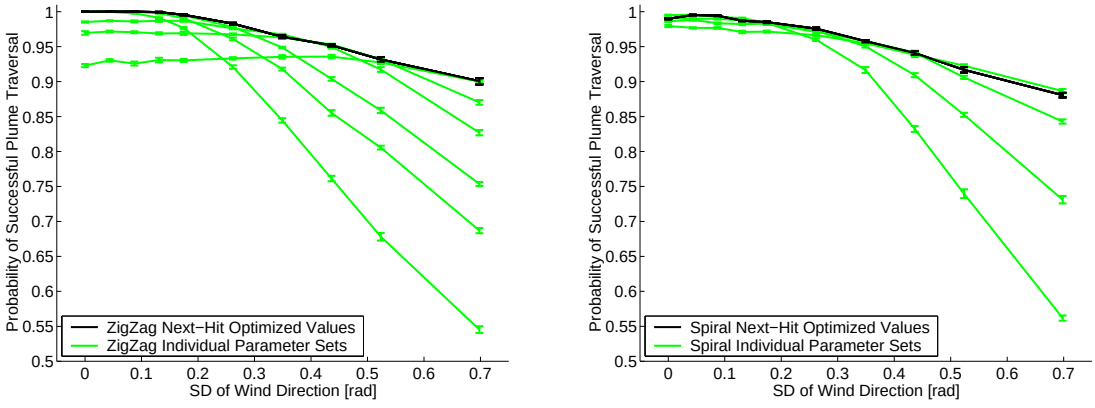


Figure 6.10: Direct evaluation values of all optimal parameter sets across all wind values compared to the direct performances of the next-hit optimized parameter sets, for the (**a**) `ZigZag` and (**b**) `Spiral` algorithms. In general the next-hit parameter sets perform well, indicating that the next-hit optimization can transfer to the real plume plume traversal task for these algorithms.

It is worthwhile to note that even though the next-hit evaluation procedure is not quantitatively accurate, it can still be useful as an optimization procedure when its assumptions are met. Recall that several different parameter sets (6 `ZigZag`, 4 `Spiral`) are employed across wind variances. All of these algorithms were evaluated across all wind variances using the direct procedure and compared with the direct performances of the next-hit optimized values. Figure 6.10 shows that the next-hit optimization generally found the best parameter values as evaluated by the direct evaluation procedure (at least in this parameter subset). In the two cases that the direct procedure evaluated a different parameter set as performing significantly better than the next-hit optimized set (ANOVA, $p = .05$), the absolute performance difference was less than 0.6 %. Note that these differences are not visible at the scale of the plots.

6.6 Conclusion

This chapter presented an investigation into plume traversal algorithms for turbulent odor plumes. A better understanding of what characteristics make an algorithm successful will enable the construction of more capable and robust chemical plume tracing systems. Restricting the sensory capabilities of the robots allows a reduction of the problem to the task of acquiring the next odor hit, which provides a detailed way to evaluate algorithm performance. Several odor localization algorithms were described in detail, and it was shown that performance characterization is possible and that algorithm parameters can be tailored to particular plume characteristics.

The next-hit analysis captures the performance of some types of algorithms more accurately than others, and this stems from intrinsic shortcomings of some of the algorithms tested. The next-hit analysis requires that the probability of getting the next odor hit remains approximately the same as an agent traverses the plume. This condition cannot hold for agents that do not actively cast across the perceived wind axis, because as the agent distribution spreads, a larger percentage of agents will fail to get each subsequent plume hit even when the plume is stable and the inter-hit time

is not limited. Based on this analysis, it can be concluded that strategies that actively cast across the plume will tend to be better than those that do not, and the problem of optimizing the tradeoff between upwind travel distance and probability of receiving the next hit can be addressed as a search over particular algorithm parameters.

However, evaluating plume traversal algorithms based on experimental data, while accurate, has two significant drawbacks. First, because the evaluation values are valid only for the type of plume originally filmed, in order to be able to test a broad cross section of plume types, many plume experiments will need to be run. An instantaneous plume computer model (in which meander parameters could be adjusted freely) could potentially be used [28], although a substantial amount of validation (i.e., comparison to experimental data) would need to be performed to ensure that the pertinent plume structure has been properly captured. The second drawback to these methods of evaluating traversal performance is that each plume-algorithm pair requires a substantial amount of data analysis and a correspondingly significant amount of time, limiting the extent to which the algorithm parameter space can be explored. However, continuing advances in computer technology should alleviate this problem in the near future.

The results shown in this chapter should extend to larger time and length scales due to self-similarity in turbulent plume structure. But because plume tracing really becomes difficult only as odor packets become more sparse (due to source intermittency and diffusion below detectable levels) and more dispersed spatially (due to flow meander), the conditions most likely to be faced when fielding real odor localization systems have yet to be investigated. Plume data sets incorporating large scale meander (i.e., 3-10 times instantaneous plume width) would be very useful for analyzing whether it is possible to track such plumes. The Caltech plume, as described in Chapter 4, while not representing actual plume data, may capture a greater degree of flow complexity, and it is investigated in the context of the full odor localization task in the next chapter.

Chapter 7

Designing an Odor Localization System

The previous two chapters examined individual phases of the odor localization problem in detail, generating a deeper understanding of the design issues involved. Now, this knowledge guides the application of the design methodology described in Chapter 3 to the full multi-agent odor localization problem. A quantitative performance metric for this task has already been defined in Chapter 3. To begin this chapter, a distributed algorithm–a set of parameterized behaviors– is described by which a group of agents can solve the full odor localization task. Next, because experimental constraints allow only the plume traversal phase to be investigated on real robots (as discussed in Chapter 4), it is shown that local position, odor, and flow information tightly coupled with robot behavior enable a robot to traverse a real odor plume. The use of multiple agents is demonstrated to increase the size of the solution space that can be reached by a particular system, and the swarm intelligence solution compares well with a sequential search strategy for this task. Also, the kinematic simulator described in Chapter 4 is validated against the real-robot data. In addition, the off-line machine learning algorithm is used to optimize system performance on the full odor localization task in several different simulated environments, and it is shown both that performance can be enhanced and that the optimal system parameters depend on the particular task being studied. Finally, a model is presented that can be used to relate task parameters to system performance.

7.1 The Spiral Surge Algorithm

The basic odor localization algorithm used in this study, Spiral Surge (SS), is shown in Figure 7.1. It consists of different behaviors related to the three different subtasks.

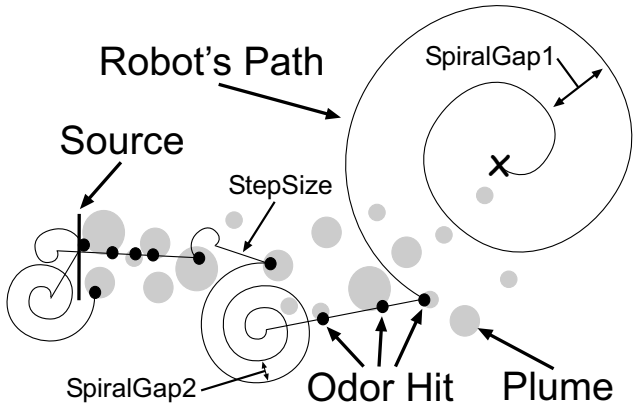


Figure 7.1: Spiral surge odor localization behavior.

Table 7.1: Spiral Surge Algorithm Parameters

SpiralGap1	Initial spiral gap width
SpiralGap2	Plume reacquisition spiral gap width
StepSize	Surge distance post odor hit
CastTime	Length of time before reverting from reacquisition to initial search spiral
SrcDecThresh	Significance threshold between consecutive separate odor hits
SrcDecCount	Number of significant differences before source declaration

Plume finding is performed by an initial outward spiral search pattern with a constant inter-cycle distance (SpiralGap1). This allows for thorough coverage of the local space if the total search area is large and initial information can be provided by the deployment point (an external "best guess" as to source location). Alternatively, if no a priori knowledge is available, a spiral with a gap much greater than the arena

size (producing essentially straight line search paths) provides an effective search procedure, as shown in Chapter 5.

Plume traversal is performed using a type of upwind surge algorithm. When an odor packet is encountered during spiraling, the robot samples the wind direction and moves upwind for a set distance (StepSize). If during the surge another odor packet is encountered, the robot resets the surge distance but does not resample the wind direction. After the surge distance has been reached, the robot begins a spiral casting behavior, looking for another plume hit. Even though zig-zag casting performed marginally better in the single-agent plume traversal analysis presented in Chapter 6, a spiral cast procedure is used because it allows a simple integration of a source declaration behavior. The casting spiral (SpiralGap2) can be tighter than the plume finding spiral, as post-surge the robot has information about local packet density and a thorough local search is a good strategy. If the robot subsequently re-encounters the plume, it will repeat the surging behavior, but if there is no additional plume information for a set amount of time (CastTime), the robot will declare the plume lost and return to the plume finding behavior (with a wider, less local, spiral gap parameter).

Source declaration can be accomplished using the fact that a robot performing the plume traversal behavior at the head of a plume will tend to surge into an area where there is no plume information, and then spiral back to the origin of the surge before receiving another odor hit. If the robot keeps track internally of the post spiral inter-hit distances (using odometry, for example, which is sufficient because information must be accurate only locally), a series of small differences can indicate that the robot has ceased progress up the plume, and must therefore be at the source. However, because small inter-hit distances can occur in all parts of the plume, this method is not foolproof, and tuning of the difference threshold (SrcDecThresh), as well as the number of observed occurrences before source declaration (SrcDecCount), is required to obtain a particular performance within a given plume. See Table 7.1 for a summary of individual SS parameters.

SS uses only binary odor information generated from a single plume sensor because

this is the most simple and reliable type of information that can be obtained from real hardware in the temporal operating regime of interest. There may be information encoded in distal fine plume structure [102], however, due to the highly stochastic nature of turbulent fluid flow and the odor-packet nature of the plume, it is unclear that more complex sensing (via graded intensity information or larger fixed sensor arrays) would benefit an odor localizing agent when flow information is available through other means.

7.1.1 Collaborative Spiral Surge

While more complex odor sensing may be beneficial to the odor localization task, another possible route to greater efficiency is physical distribution of the odor sensing elements, which in principle could improve system speed and robustness via parallelization of the search procedure. This performance benefit can be achieved by constructing an arbitrarily large and complex single robot or, perhaps more conveniently, distributing a number of sensors throughout a group of smaller, more simple communicating robots. With a suitable command and control interface, this collective can be viewed as an 'odor localization sensor' in much the same way a single larger robot, or more generally device, could. One way to increase the performance of such a robot swarm is collaboration between individual nodes. In particular, if collaboration is obtained with simple explicit communication schemes such as binary signaling, the team performance can be enhanced without losing autonomy or significantly increasing complexity at the individual level.

Several simple types of communication can be integrated into basic SS. This chapter examines the performance impact of three types of communication: no communication (NONE), a "come here" signal emitted by upwind surging robots that causes all robots downwind or with no plume information to surge in the direction of the calling robot (ATTRACT), and a "stop" signal emitted by the first robot to receive odor information that causes all other robots to surge away from the signaling robot and then enter a power save mode from which they cannot be awakened (KILL).

To enhance the performance of ATTRACT, an extension of this communication type, ATTRACT3, is studied which includes a "stop" signal sent to all additional agents after three have entered the plume. The influence of these types of communication is analyzed across group size to determine their impact on system efficiency.

7.2 Plume Traversal Results

7.2.1 Real Robots

The real-robot experiments focus on the plume traversal subtask because it contains most of the plume related complexity present in the full odor localization task, and due to experimental limitations, it is currently not feasible to study all phases with real robots. To justify the high density of agents in the plume (which would be unlikely given that in the general problem the plume area is a small percentage of the total search area), ATTRACT communication is employed between the agents to hold the group together as it traverses the plume. Since source declaration is not being studied, a trial is defined to be complete when a robot reaches a given distance, the Source Fnd Radius, from the plume source. Task performance is described by equations (3.3) and (3.4), combined here for convenience:

$$P = \frac{\alpha T_{MIN} + \beta D_{MIN}}{\alpha T_{TC} + \beta D_{TC}}. \tag{7.1}$$

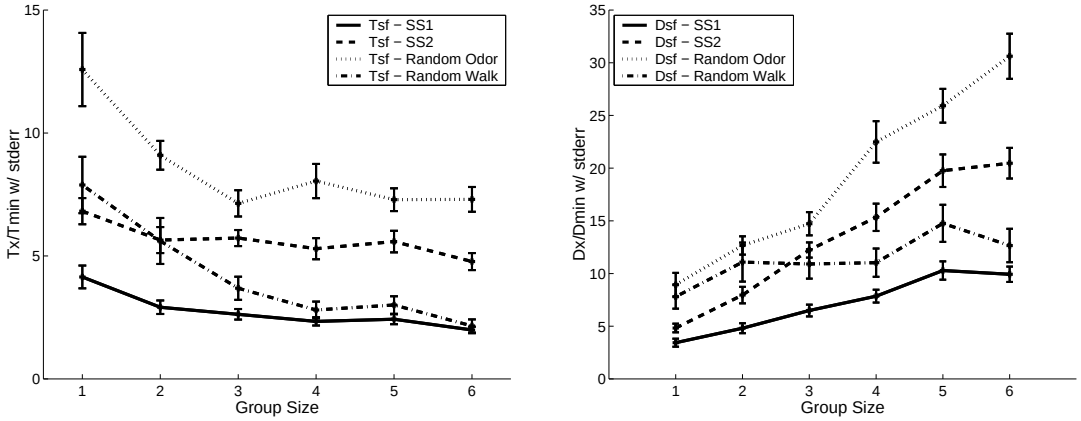
Again, T_{TC} is the time needed for task completion, and D_{TC} represents the total distance traveled by all agents during the task. α is taken to be the cost per unit time of not completing the task, and β is the cost per unit distance of running the system. The optimum values for the task (T_{MIN}, D_{MIN}) are calculated from an agent executing the optimal behavior (a straight line path from start to goal areas at maximum speed). Maximum speed, which determines the relationship between the time and distance values, is the maximum safe operating speed of the agent in the given environment. In this example α and β are set so that the time and energy components of the task factor equally into the minimum cost, so $\frac{\alpha}{\beta} = \frac{D_{MIN}}{T_{MIN}}$.

Table 7.2: Plume Traversal Parameter Values

Agent speed	.325 m/s
Arena length	6.7 m
Plume length	8 m
Plume speed	∼1 m/s
Source find radius	.88 m
Plume:Arena area	1:2.3
Goal:Search perimeter	1:18.0
T_{MIN}	19.0 s
D_{MIN}	6.2 m
$\frac{\alpha}{\beta}$	.326 [m/s]
`SS1`: SpiralGap2	1785 km
`SS1`: StepSize	9.1 m
`SS2`: SpiralGap2	.357 m
`SS2`: StepSize	.91 m

Real-robot plume traversal performance was tested using two sets of SS parameters and two control experiments. Only SpiralGap2 and StepSize are considered because only the plume traversal phase of the task is being studied. The parameter set `SS1` represents a nonlocal search in that its search paths are straight and its surges extend to the boundaries of the arena. `SS2` uses a smaller spiral gap and surge length to perform a more local exploration of the arena. One control, `Random Odor`, uses `SS2` parameters, and receives odor hits that are generated from the time sequence of `SS2` odor hits but are not correlated with robot position in the arena. This experiment investigates whether an algorithm incorporating precise odor packet location information is more efficient than a blind upwind surging behavior. An alternative experiment could be to decouple the wind source from the odor source by creating a wind field with an array of fans, but due to practical limitations in our experimental set-up, the `Random Odor` case was easier to implement and provided equivalent information from a proof of concept point of view. The second control, `Random Walk`, takes straight line paths and random avoidance turns at boundaries (using no odor or flow information) to provide a traversal performance baseline. Specific parameters relating to the real-robot tests are listed in Table 7.2. 15 trials of each group size were

run for `SS1`, `SS2` and `Random Odor`, and 30 trials were run for `Random Walk` due to the high variance of performance values. All error bars in the plots represent standard error unless otherwise specified.



(**a**) (**b**)

Figure 7.2: (**a**) Normalized time to finish task across group size for real-robot trials. Lower values are better. (**b**) Normalized distance across group size for real-robot trials. Lower values are better.

Figures 7.2a and 7.2b show that for all conditions studied, traversal time decreases with group size while group distance traveled increases. This indicates, as expected for a search task, that as time becomes more important to performance than energy usage, larger group sizes will be preferred.

Figure 7.3 shows that while single robots are generally most efficient in this arena (given this particular choice of α and β), `SS1` generates the best results for each group size (significant via Kolmogorov-Smirnov test to $p < .01$ for group size $\in \{1, 2, 3\}$), demonstrating successful real-robot plume traversal. `Random Odor` performs worse than `SS2` for all group sizes (significant as above for group size $\in \{1, 2, 4, 6\}$), indicating that location of odor information is an important aspect of the search algorithm. This means that SS is actually plume tracing rather than simply localizing the source of the wind. If only wind localizing were taking place, one would expect `Random Odor` to perform exactly the same as `SS2`. Also, `SS2` performs worse than `SS1` (significant as above for all group sizes), suggesting that local search is not a good strategy in

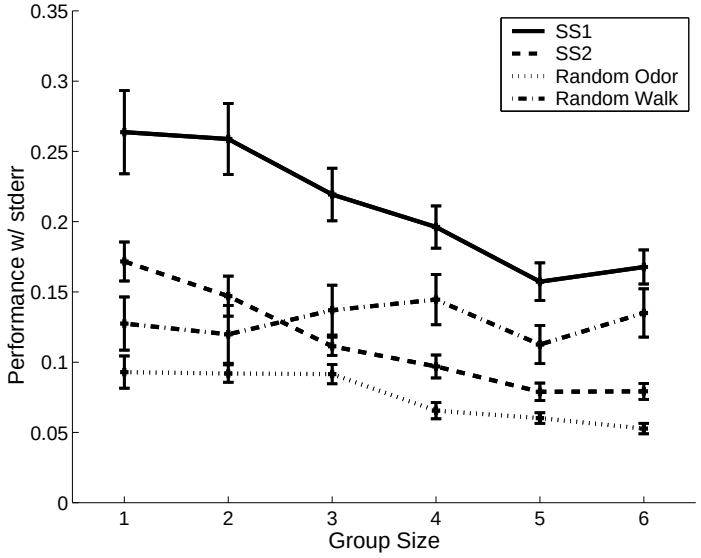


Figure 7.3: Performance P across group size for real-robot trials. Higher values indicate better performance.

this small arena where the goal-to-search perimeter ratio is high (i.e., it is likely to find the goal by chance). The `Random Walk` behavior retains relatively constant performance across group size, and at the larger group sizes its performance tends to approach the optimal observed performance. This suggests that as a search arena becomes overcrowded, random movement becomes the best strategy.

7.2.2 Sequential Search Comparison

SS can perform better than a random search strategy, but another way of gauging SS performance is to compare it to a basic sequential search. Since this task is complete when the agent comes within some distance of the source, the odor plume aspect can be ignored, reducing the task to a search problem (as examined in Chapter 5). Note here the source sensor is assumed to be perfect. As shown in equation (5.20), the total amount of time T_s required for a single agent to make a single pass over the entire arena can be estimated in terms of the arena length L, agent speed v, and source sensor range r:

$$T_S = \frac{L^2}{2rv}. \tag{7.2}$$

Assuming a uniform target distribution throughout the arena and perfect collaboration among team mates, the expected time to find the source for a group of N agents can be approximated as $\frac{T_S}{2N}$. Note this formulation does not account for all of the implementation details and may overestimate performance (e.g., sometimes search overlap is necessary, and groups of agents must spread out from the deployment area to their designated areas), but it is accurate enough for the purposes of this comparison.

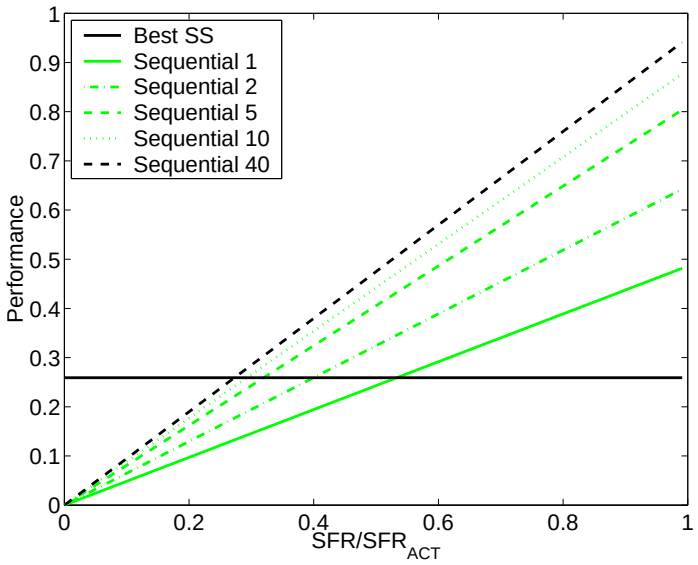


Figure 7.4: Performance of the best SS algorithm and a basic sequential search for different group sizes, as the source find radius given to the sequential algorithm approaches the actual source find radius. Higher values indicate better performance. SS does not explicitly use the source find radius, so performance does not vary.

The arena length and agent speed are known quantities, but the sensor range, here the source find radius, is not necessarily known a priori to the system designer. Sensor ranges can depend on unknown variables in the environment (such as the depth of buried mines in a minefield [33]). If the specified range is greater than the actual range, incomplete coverage will result, so conservative estimates are typically

necessary. The above equation for the expected search time can be used to calculate the expected distance traveled by the sequential search system (see equation (5.2)), and these values can be used to calculate performance, as in equation (7.1).

In Figure 7.4, system performance P is plotted against the ratio of the programmed source find radius (SFR) to the actual source find radius (SFR_{ACT}). The best average performance observed by an SS algorithm (which does not depend on a predicted value of the source find radius) is included for comparison. The sequential search for a single agent exceeds the SS algorithm only when the programmed source find radius is within a factor of 2 of the actual value. It may be difficult to reach this level of accuracy, particularly when the cost of failing to fully cover the search area is high. For large team sizes, the programmed source find radius must be greater than 25% of its actual value to exceed the SS performance. Even when the proper radius is known, the single agent sequential search performance is within a factor of 2 of the SS performance, and the large team performance is within a factor of 4 (and is overestimated here, as discussed above). Also, the added cost of the localization mechanism required to perform a sequential search and the communication network needed for group coordination is not considered.

7.2.3 Kinematic Simulations

The real-robot performance data was successfully reproduced in Webots, as shown in Figure 7.5. Simulated plume parameters (envelope size and sensor threshold–see Chapter 4 for details) were tuned to the `SS2` data and then fixed for the other conditions. Data represents 1000 trials per group size. All parameters in Table 7.2 apply to the Webots data as well. Only `SS1` for group size of one robot produces significantly different results (as determined by a two-tailed Kolmogorov-Smirnov test with $p < .01$) between Webots and the real robots, and even in this case the error bars overlap. Because our Webots data closely matches our available real-robot data, it is reasonable that further simulated experiments will accurately reflect real-world behavior.

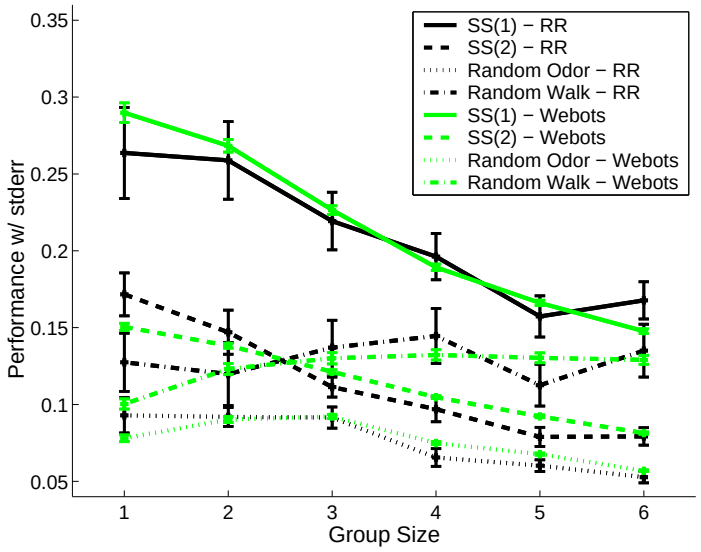


Figure 7.5: Performance of real-robot (RR) and Webots trials across group size. Higher values indicate better performance.

7.3 The Full Odor Localization Task

The principal limitation of the experiments described thus far is the relatively small arena available for the real robots. In simulation we can expand the arena size and move the start area outside the plume extent. This enables the study of all phases of the odor localization task and calls for a change to the task stopping condition. In the following experiments the task is declared complete when greater than half of the source declaration points (from all agents) are within the source find radius of the plume source. This definition was chosen to avoid any notion of task failure–a bad source declaration can be overcome–while not explicitly factoring the distance between the declare point and the source location into the performance measure (which would require another arbitrary cost value, and would introduce greater variation into the performance measure).

7.3.1 Algorithm Optimization

The performance impact of the three types of communication described earlier, NONE, ATTRACT3, and KILL, was investigated across four different plumes. ATTRACT3 was studied rather than ATTRACT because preliminary tests indicated that the addition of an energy saving component to the communication algorithm reduced inter-agent interference and increased system performance. Three of the plumes are variants of the Georgia Tech plume data: GT0, GT1, and GT2. GT1 and GT2 are $\frac{1}{5}$ and $\frac{1}{10}$, respectively, of the density of GT0, which is a 2 times larger (in length) version of the plume used in the small arena. The fourth plume, CT0, is based on the Caltech plume data set. It has roughly the same size and density as GT0 but a much more complicated flow pattern. See Chapter 4 for details on the plume stimuli. There were a total of 12 plume-communication pairs studied. These are referred to as "conditions." Each condition was evaluated across a range of 5 group sizes: 1, 3, 5, 10 and 15 agents. For each condition and group size combination, a single optimization run, as described in Chapter 3, was executed over the 6 SS parameters (listed in Table 7.1). Two of the parameters, SPIRALGAP2 and STEPSIZE, were found to be inter-dependent, so they are combined into a single list of parameter pairs to be searched. The parameter values were chosen to cover a functionally significant partition of the parameter space, and they are listed in Table 7.3. Optimization parameter values (as defined in Section 3.1.3) were as follows: $\eta = .1$, $\kappa = .05$, and $\epsilon = 10$. Environmental and algorithmic parameter values that differ from the real-robot experiments are shown in Table 7.4.

Table 7.3: Searched Parameter Values. Parameter definitions can be found in Table 7.1

SPIRALGAP1	{2.83, 35700} m
SPIRALGAP2, STEPSIZE	{(.283, .650), (.357, .910), (.425, .542), (.567, .650), (.567, .2168), (.708, .902), (.850, .975), (1.42, 2.52)} m
CASTTIME	{48, 96, 192} s
SRCDECTHRESH	{.27, .55, 1.09} m
SRCDECCOUNT	{1, 2, 3, 5}

Table 7.4: Full Task Parameter Values (Simulation)

Arena length	33.5 m
GT0 Length	16 m
GT0:Arena area	1:14.5
Wind noise	$\pm$10%
Agent speed	.325 m/s
Plume speed	$\sim$2 m/s
Source find radius	1.6 m
T_{MIN}	72.5 s
D_{MIN}	22.0m
$\frac{\alpha}{\beta}$	1.48 [m/s]

Although group size and communication type are technically algorithm parameters, they are evaluated separately, rather than within the optimization procedure, so that trends across these variables can be analyzed. A total of 60 optimization runs (4 plumes by 3 communication types by 5 group sizes) were performed. Each was executed on a 1.5 GHz Athlon XP, and run times ranged from 1 day to 3 weeks. As detailed in Chapter 3, each optimization run consists of 10 cycles in which each of the parameters is optimized while the others remain fixed. The initial parameter selections are random, and the parameters chosen in each cycle serve as the input set for the subsequent cycle. At the start of the run and after each cycle, the performance of the current parameter set is measured by executing the task 50 times and averaging the observed performance values. By the end of the run there is a set of 11 performance measurements.

To demonstrate that the optimization process succeeds in improving performance, the performance values of all 60 runs were first normalized by the maximum value observed during the run and then averaged across all runs. Figure 7.6 shows that performance increases over the initial cycles, and then plateaus over the remainder of the run. Note that because the performance evaluation has a stochastic component, and the optimization process does not consider values that are within 10% of each other, the mean performance, as analyzed in this manner, is not expected to converge to 1. Also, from this data, it is not possible to determine that the optimization

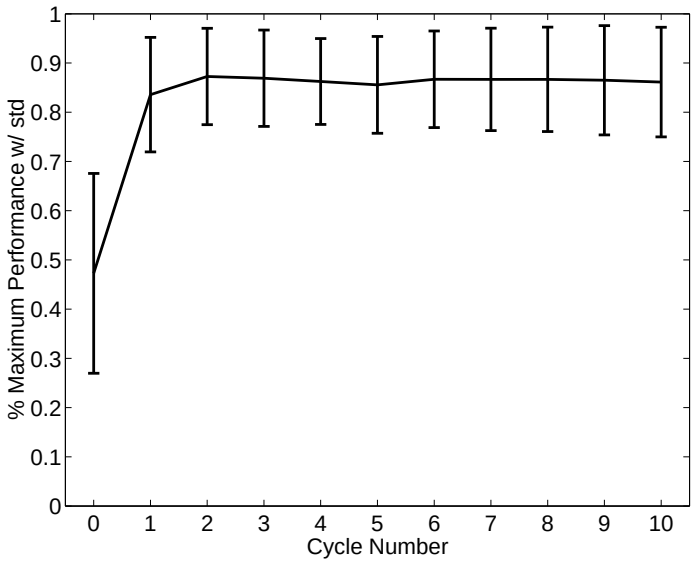


Figure 7.6: Performance during each optimization run, first normalized by the maximum value of each run and then averaged across all 60 runs. Error bars indicate standard deviation.

process does not get trapped in local minima. Nevertheless, for the purposes of further analysis, all runs are assumed to have converged to global optima, and the output of cycles 3 through 10 are considered optimal performance values.

The time and distance taken for the 8 optimized cycles of each run (cycles 3-10) are averaged to produce overall values for each run. Figure 7.7 shows the time and distance necessary for each group of robots to complete the task on each plume. Note that qualitatively the curves resemble those of Figures 7.2a and 7.2b, with distance increasing roughly linearly with group size, and time decreasing with group size. NONE consistently uses the most energy across all plume types, followed by ATTRACT3, and then KILL. This indicates that the energy-saving measures built into ATTRACT3 and KILL are successful. Also, in general, the more sparse plumes have a greater influence on the distance traveled by the smaller group sizes. The distance traveled by a single agent increases by almost a factor of three from GT0 to GT2, while the distance of the groups of 15 agents increases by at most 25%. KILL requires the largest amount of time across all plume types, indicating that its energy

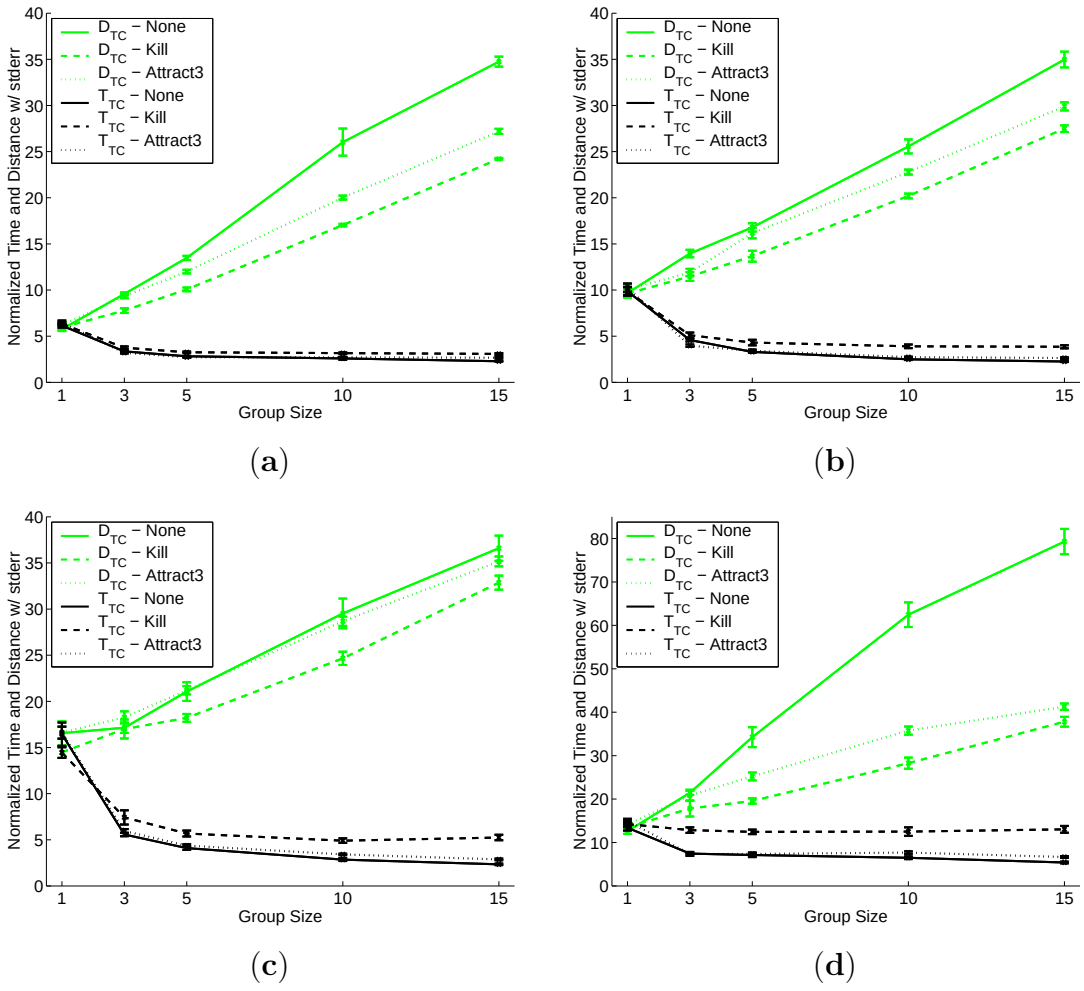


Figure 7.7: Normalized time (T_{TC}) and distance (D_{TC}) across group size for (**a**) GT0, (**b**) GT1, (**c**) GT2, and (**d**) CT0. Lower values are better. Error bars represent standard error.

savings comes at a cost. However, NONE and ATTRACT3 do not differ significantly in time requirements, so ATTRACT3, though its energy savings are not as significant, does not seem to suffer a time penalty. In addition, the temporal speedup with group size increases for the more sparse plumes.

Performance for each run is shown in Figure 7.8. On average, a group size of 3 agents performs best on GT0, GT1, and CT0, while a group of 5 agents performs best on GT2. ATTRACT3 performs at or near the best across all plumes, so communication is beneficial for this task. NONE performs the worst on GT0, all communication types perform similarly on GT1, and KILL performs the worst on GT2.

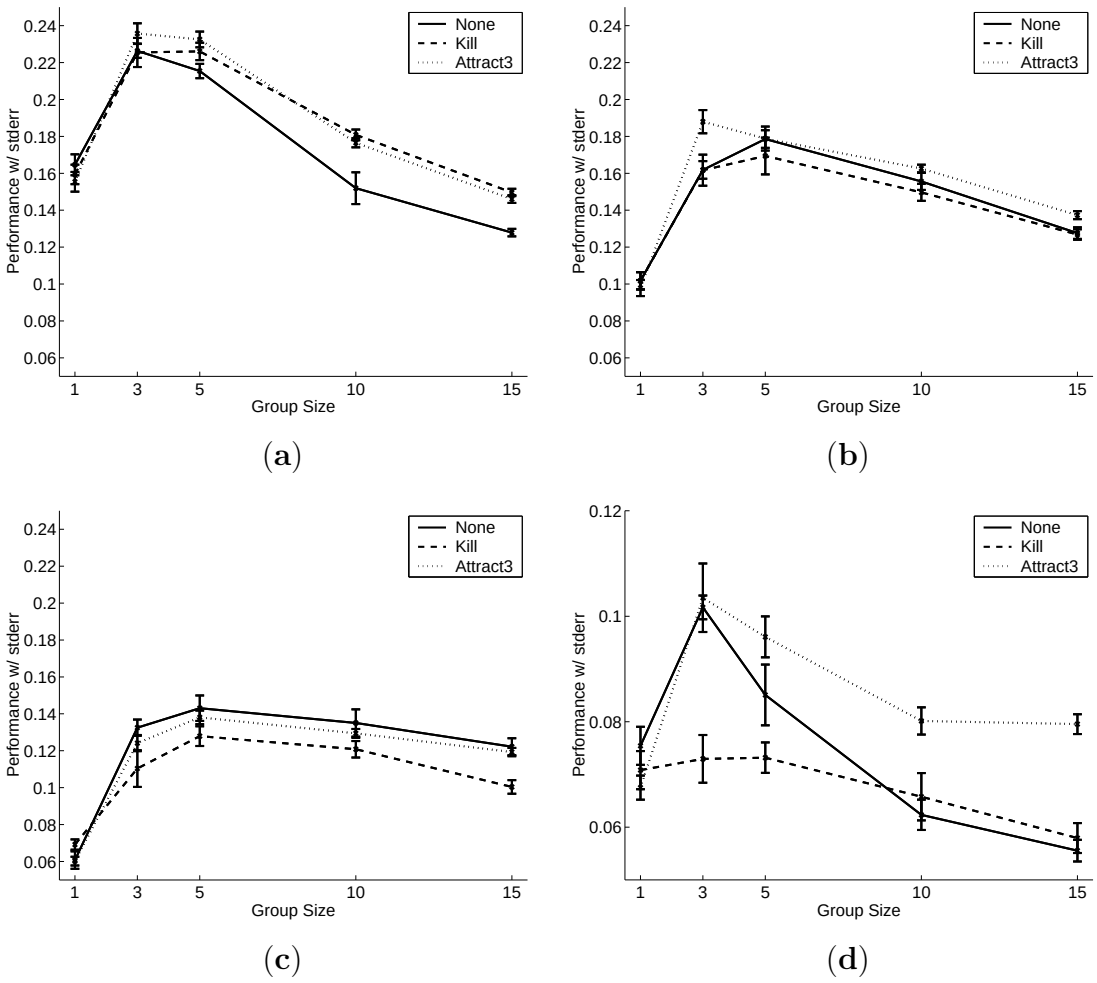


Figure 7.8: Performance across group size for (**a**) GT0, (**b**) GT1, (**c**) GT2, and (**d**) CT0. Higher values are better. Error bars represent standard error.

None and Attract3 share the best performance on CT0, although Attract3 performs better for the larger group sizes.

The above results can be explained in terms of the phases of the odor localization task that are emphasized by each plume. As the plume becomes more sparse, the search phase becomes more prominent. It was shown in Chapter 5 that harder search problems favor larger group sizes, because this phase of the task can be parallelized quite easily. Therefore, for more sparse plumes, task completion times drop more drastically for larger group sizes, and the optimal group size increases. Likewise, the distance required by large group sizes will not increase dramatically when the

plume becomes sparse, as a thorough search process is automatic, while the distance required by small groups does increase significantly because the only way to achieve better coverage is by increasing the distance traveled by each individual unit.

When the traverse or declare phases are emphasized, however, smaller group sizes benefit. These tasks can typically be effectively carried out by a small number of robots, and large groups at best burn extra energy, and at worst cause destructive interference. Interference can explain the poor performance of `None` on GT0 and CT0 at large group sizes. Both of these plumes are dense enough to hold many agents, and the presence of other agents can disturb the casting spirals that are necessary for plume traversal and (particularly) source declaration. GT1 and GT2 appear to be sparse enough that too few robots are drawn into the plume before task completion to cause any problems. In fact, the traverse and declare phases are possibly done best serially, at least when they are relatively easy. KILL, which uses only one agent for these subtasks, requires shorter group distances than the other communication types. However, note that KILL does not fare well when the cost of losing the plume is high (as for GT2) or the probability of losing the plume is high (as for CT0–note there is no temporal speedup with group size). These failures reflect the high temporal cost of plume loss when the parallel plume-reacquisition search capability is lost.

7.3.2 Trends in Optimization

The optimization procedure can do more than improve system performance, because by looking at the optimized parameter values one can gain insight into the operation of the algorithm itself. The optimized parameters are analyzed by combining cycle results 3-10 from each run, and then examining how often each parameter value is selected (this results in an optimization result frequency curve). A simple example of this type of data is shown in Figure 7.9a, which shows the optimization result frequency curve for SPIRALGAP1 averaged over all runs. Two SPIRALGAP1 values are possible (see Table 7.3), and this data indicates that nearly 100% of the optimized parameter sets contain the larger value. For this task the agent start area is not near

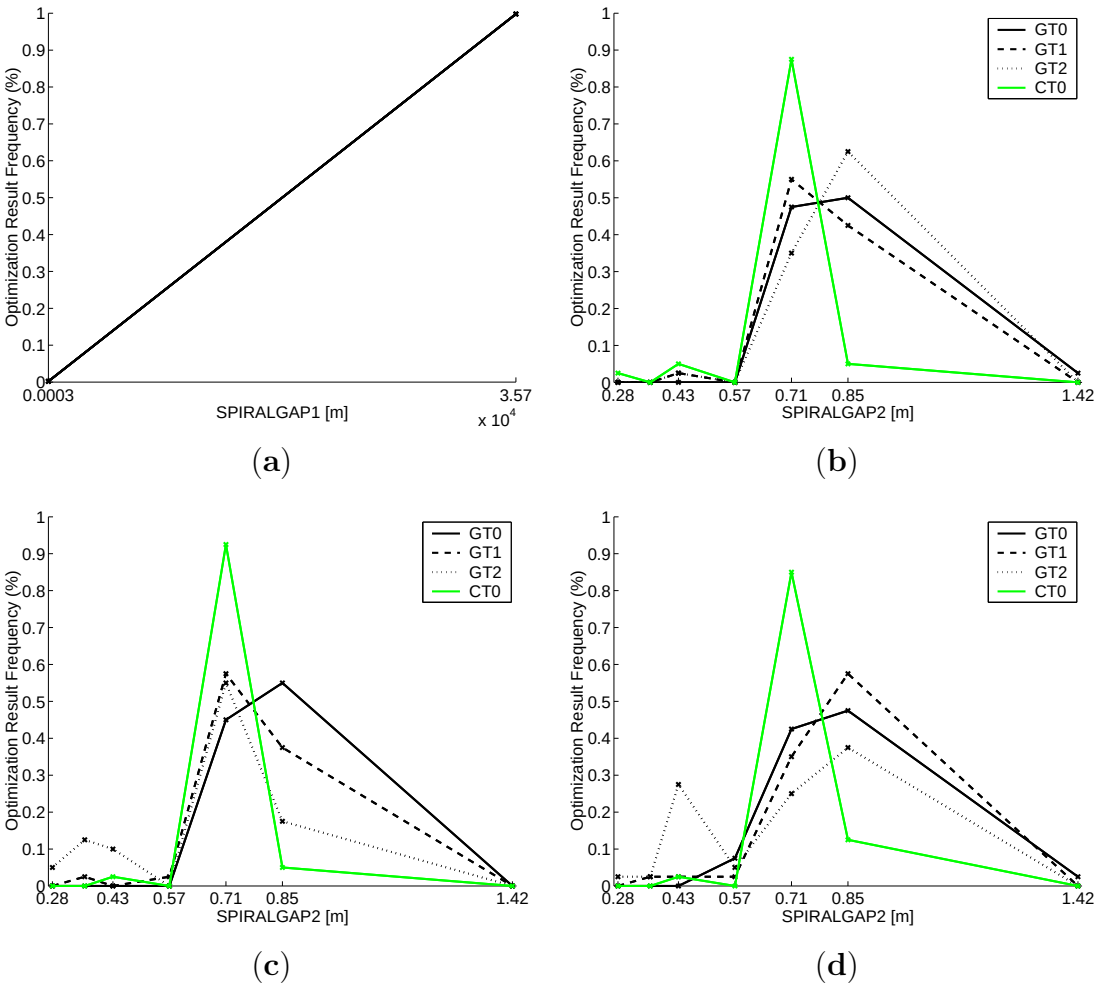


Figure 7.9: (**a**) The optimization result frequency curve for SpiralGap1, as averaged over all runs, (**b**) The SpiralGap2 optimization result frequency curves for each plume type, averaged over group size for None, (**c**) Kill, and (**d**) Attract3.

the plume source, and an initial local search wastes time and energy. It is therefore not surprising that the large SpiralGap1 value is almost always chosen, and the reliability of its selection suggests that the optimization procedure is functioning properly.

Other parameters exhibit a greater degree of complexity. Figures 7.9b-d show the optimization result frequency curves for SpiralGap2. The data is averaged across group size and plotted for each plume type and communication method. Recall that SpiralGap2 is actually searched as a pair with StepSize, but only one Spiral-

GAP2 value appears in more than one pair (.567 appears twice), and since this value is rarely selected, analysis of only the SPIRALGAP2 data captures the relevant features. The most obvious result from this data is that CT0 heavily favors a particular SPIRALGAP2, regardless of communication type. This supports the idea that source declaration is the most difficult phase of the CT0 task, and thus the SPIRALGAP2 - STEPSIZE pair that is best at that procedure is critical for performance. Also, while GT0 and GT1 appear to have similar performance landscapes across communication types, for GT2, which renders traversal most difficult, smaller SPIRALGAP2 values become more favored under KILL, and ATTRACT3– when few agents are performing the traversal. This shift suggests that for this particular plume, smaller SPIRALGAP2 values reduce the probability of losing the plume once it is acquired.

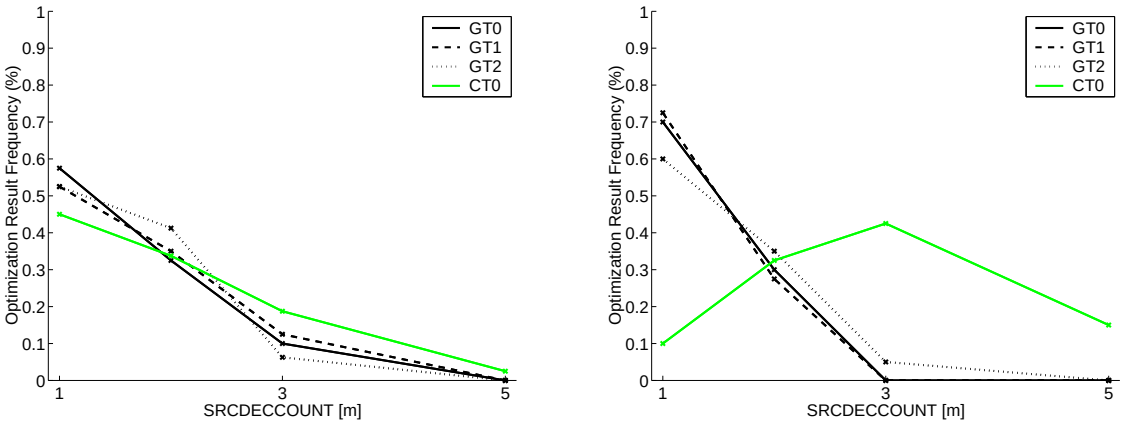


Figure 7.10: (**a**) The SRCDECCOUNT optimization result frequency curves for each plume type, averaged over group size and across the KILL and ATTRACT3 communication types. (**b**) The SRCDECCOUNT optimization result frequency curves for each plume type, averaged over group size for NONE.

Similar themes can be observed in other parameters. Figure 7.10a shows the selected SRCDECCOUNT values across plumes for the KILL and ATTRACT3 communication types. Smaller values are favored, and there is no difference across plumes. However, Figure 7.10b shows the same data for NONE communication, and here the performance landscape for CT0 is markedly different. For this plume and communi-

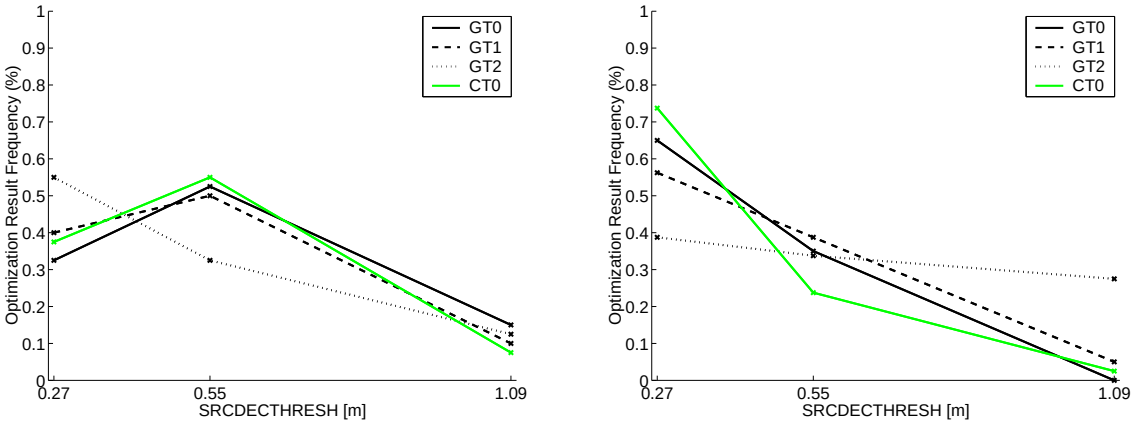


Figure 7.11: (**a**) The SrcDecThresh optimization result frequency curves for each plume type, averaged over group size and across for Kill. (**b**) The SrcDecThresh optimization result frequency curves for each plume type, averaged over group size and across the None and Attract3 communication types.

cation type, larger SrcDecCount values become favored, which indicates that when there are large numbers of agents traversing the plume (which can only occur under None), they can interfere with each other and produce spurious source declarations that inhibit performance. Note this effect is not observed for GT0, which is just as dense (and can therefore contain just as many agents), but is easier to declare because its flow patterns are much less complex. Also, it appears that GT2 is difficult to declare as well, but for a different reason. Figure 7.11a shows the SrcDecThresh optimization result frequency curves for Kill across plume type. There are no major differences across plumes, although perhaps GT2 favors slightly smaller values, which could be explained by the smaller size of the GT2 plume head. However, Figure 7.11b shows the same data for the other two communication types, and it suggests that when there are multiple agents in the plume, GT2 favors larger source declaration distances. This could be explained by the fact that multiple agents operating concurrently in the same small region of space will interfere with each other, and the precise declare procedure will not be possible, so looser requirements are necessary. Likewise, when multiple agents are tracking the plume CT0 favors smaller

SrcDecThresh values, which is consistent with the need to limit false declarations due to the presence of many agents.

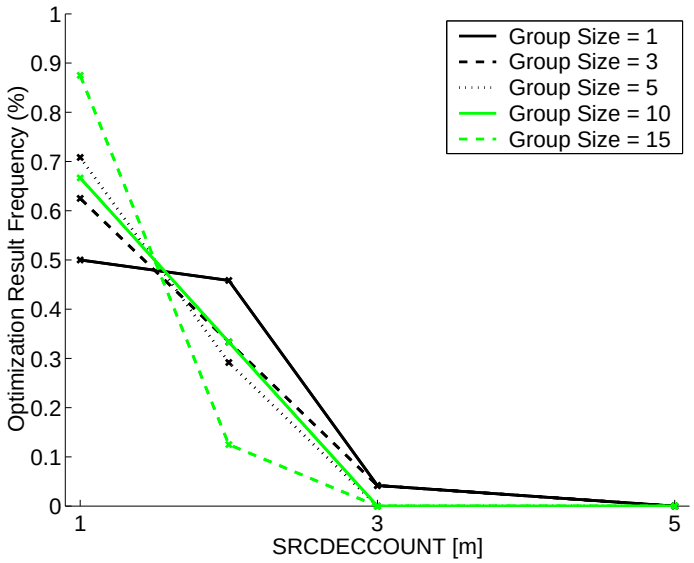


Figure 7.12: The SrcDecCount optimization result frequency curves for each group size, averaged over GT0, GT1, and GT2 for None communication.

Finally, observations can be made across group size as well. Figure 7.12 shows the SrcDecCount optimization result frequency curves for each group size, averaged over GT0, GT1, and GT2 for None communication. The smallest group sizes favor larger SrcDecCount values, presumably to avoid declaring the plume found while still traversing the plume, while the largest group size favors small SrcDecCount values, perhaps to facilitate declaration when interference from other agents renders repeated declaration cycles difficult. This analysis is not intended to be conclusive, but rather representative of the type of information that can be gleaned from examining the optimization results.

7.3.3 A Model of Performance

The specification and optimization phases of the design process have been demonstrated in the previous sections. The third design phase, as discussed in Chapter

3, involves the definition of abstract relationships between system parameters and system performance. Once such relationships have been experimentally validated in a test environment, they can be used to guide the design of a deployable system, as they will allow the designer to predict system performance in a wider range of environments than have been explicitly examined experimentally.

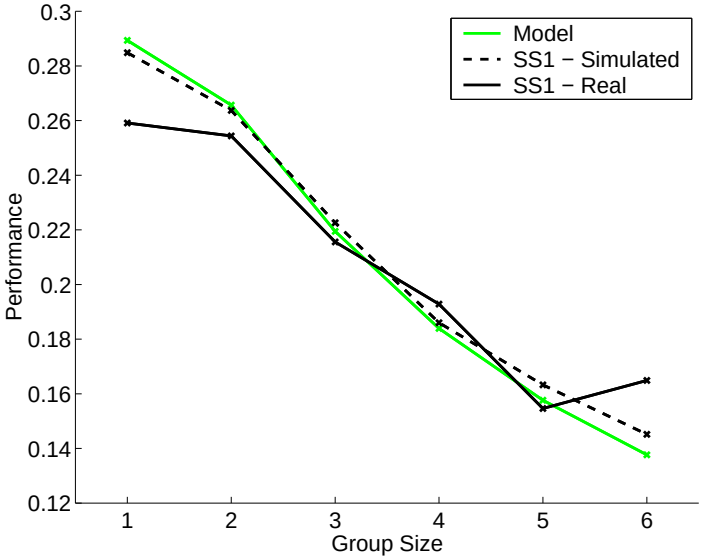


Figure 7.13: Performance versus group size for the odor localization task in the small arena as generated by the model, the kinematic simulator, and the real robots.

The model for the odor localization task is drawn directly from the analysis of the coordinated search problem described in Chapter 5. Specifically, if the task is complete when an agent comes within a given range of the source (as in the experiments in the real arena), the time to find the source can be described as follows:

$$T_{TC} = \frac{1}{(1 - P_{LC}^{N})NP_{SR}} + T_{TR} + k. \tag{7.3}$$

P_{LC} is the probability of an agent losing contact with the plume once it has been acquired, N is the number of agents, and P_{SR} is the probability of a single agent getting an odor hit during the search phase. The first term of the sum represents the expected amount of time needed to search for the plume, and it assumes that

ATTRACT communication is used. T_{TR} is the expected time to traverse the plume, and k is the minimum required dispersion time throughout the arena. T_{TC} represents the expected time to locate the source. The required distance is then simple to specify:

$$D_{TC} = NvT_{TC}, \tag{7.4}$$

where v is the average robot speed. For the real arena, $P_{SR} = .0518$ is calculated from the mapping experiments (see Chapter 4), $v = .27$ m/s is measured directly from the real robot experiments (collision avoidance and wind scanning reduce actual speed from maximum speed), and $k = 13$ is determined from simple geometry. Choosing reasonable values $T_{TR} = 30$ and $P_{LC} = .33$ results in good agreement between the real, simulated, and modeled results on this task, as shown in Figure 7.13.

Additions to the model are necessary so that it can capture the source declaration phase of the task as well as different communication strategies. Note that this model is meant only to capture general performance trends, so not all of the specific interactions present in the system are modeled. Four additional parameters are used: T_{DE}, the time required to perform the source declaration; D_{SU}, the distance travelled by each inactivated robot to move away from the plume; λ, used to factor in interference due to inactive robots; and ω, which represents the speedup in declaration possible when multiple agents are in the plume. Another parameter, ϕ varies with the communication type, and it corresponds to the number of agents that are not inactivated. The time to complete the task is defined in three steps:

$$T_1 = \frac{1}{NP_{SR}} + k. \tag{7.5}$$

T_1 is the expected time before an agent makes contact with the plume. The search time necessary after the first contact (and subsequent plume loss) can be expressed as follows:

$$T_2 = \frac{\frac{1}{1-P_{LC}^{\min(N,\phi)}} - 1}{\min(N,\phi)P_{SR}}. \tag{7.6}$$

T_2 is the expected time that a system will spend reacquiring a plume. The numerator is the expected number of times the plume will be lost, and the denominator accounts for the reacquisition time. For the NONE and KILL communication types, T_3, the time to traverse the plume and declare the source is as follows:

$$T_3 = T_{TR} + (N - \min(N, \phi))\lambda + \frac{1}{\frac{1}{T_{DE}} \min(N, \phi)^{\omega}}. \tag{7.7}$$

The first term in the sum is the traversal time. The second term represents the interference of inactive agents during source declaration, and the third term represents the source declaration time (which can decrease with the number of agents, as it can be done in parallel spatially). For ATTRACT3, T_3, has a greater speedup in traversal time with group size, reflected in the first term:

$$T_3 = \frac{T_{TR}}{\frac{\min(N,\phi)+1}{2}} + (N - \min(N, \phi))\lambda + \frac{1}{\frac{1}{T_{DE}} \min(N, \phi)^{\omega}}. \tag{7.8}$$

The expected time for task completion is:

$$T_{TC} = T_1 + T_2 + T_3. \tag{7.9}$$

Similarly to equation (7.4), the distance traversed can be expressed in terms of the traversal time, although here it must reflect the fact that robots can be placed in a low-power mode:

$$D_{TC} = vNT_1 + v(T_2 + T_3)(\min(N, \phi) + \zeta(N - \min(N, \phi))) + \max(0(N - \phi))D_{SU}. \tag{7.10}$$

ζ is the relative power savings of an inactive robot compared to an active one. The first term in the sum represents the distance required for the first plume hit. The second term represents the distance traveled after the extra group members have been inactivated. The third term accounts for the distance traveled by the extra group members after being inactivated but before entering the low-power mode.

Recall that $\phi = N$ is the number of agents that are not inactivated during a trial.

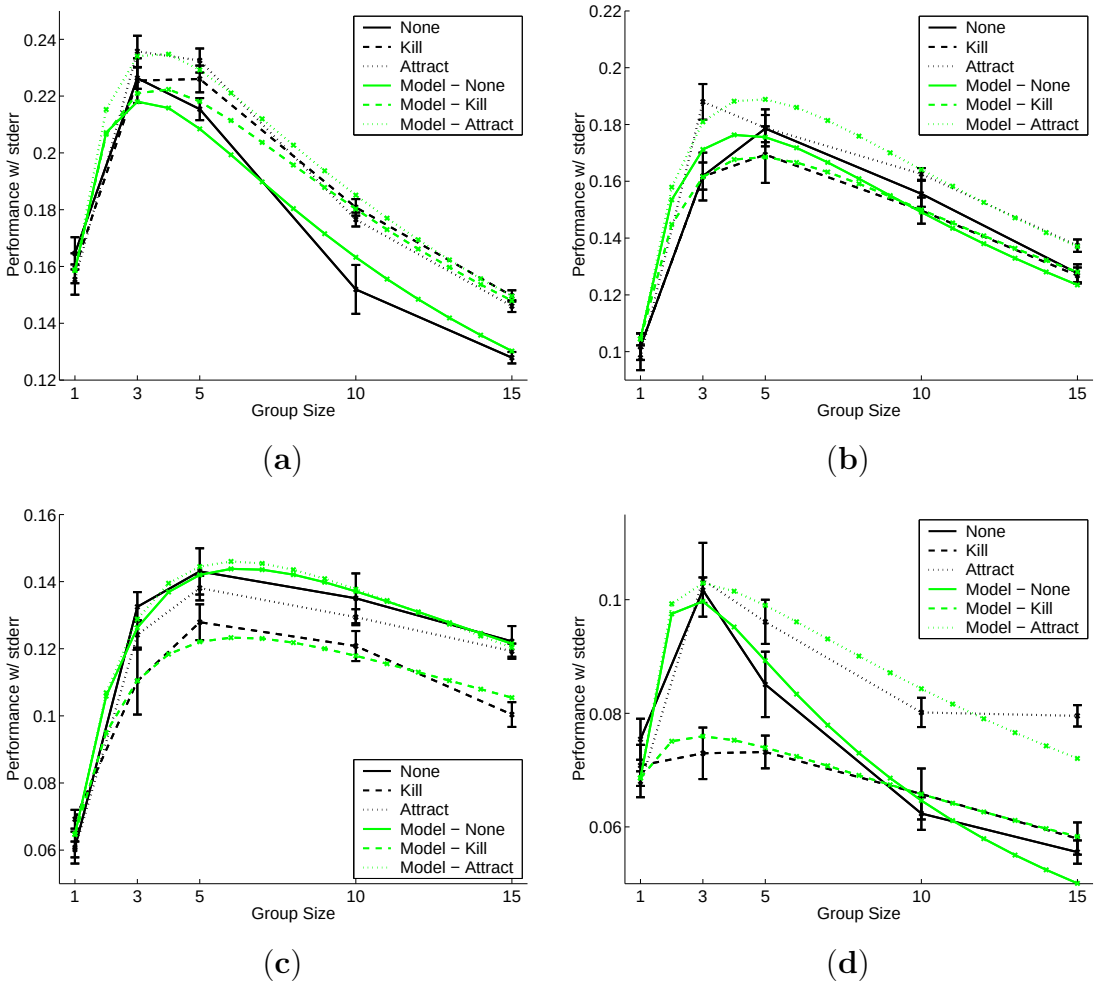


Figure 7.14: Performance across group size for (**a**) GT0, (**b**) GT1, (**c**) GT2, and (**d**) CT0 as generated by the model and the kinematic simulator. Higher values are better. Error bars represent standard error. Note there is good agreement between the simulator and the model across plumes and communication types.

Table 7.5: Model Parameter Values

	GT0	GT1	GT2	CT0	
P_{SR} [1/s]	.00332	.00196	.00110	.00348	Fit
T_{TR} [s]	34	34	17	34	Fit
P_{LC}	.02	.06	.08	.53	Fit
T_{DE} [s]	55	55	55	350	Fit
λ [s]	4	4	4	10	Fit
ω	.04	.08	.14	.04	Fit
D_{SU} [m]	6.5	6.5	6.5	6.5	Fixed
ζ	.04	.04	.04	.04	Fixed
k [s]	61.3	61.3	61.3	61.3	Fixed
v [m/s]	.31	.31	.31	.31	Fixed

For NONE, $\phi = N$. For ATTRACT3 and KILL, $\phi = 3$ and $\phi = 1$, respectively. The other parameters chosen for the model are shown in Table 7.5. Some are fixed by the algorithm. Some can be calculated directly from the environment. Others have been tuned to fit the data, although they are based on observable data and an effort has been made to respect the similarities and differences across the different plumes. For example, P_{SR} for GT0 can be estimated from the small arena, although the resulting value must be further reduced because the location of odor hits is correlated in space (so the density does not scale linearly). Likewise, P_{SR} decreases with plume sparseness, and since CT0 has roughly the same plume density as GT0, its P_{SR} value is similar. T_{TR} stays the same for the three more dense plumes, because they have roughly the same length. GT2 does not extend as far from the source, so its T_{TR} is smaller. All of these values are small compared to the small arena because the expected point of first contact with the plume is dependent only on the distribution of odor (which is more dense near the plume source), while in the small arena the starting location skews initial contact toward the distal end of the plume. P_{LC} increases wth plume sparseness and is much greater for the more turbulent CT0, and the values are lower than in the small arena because in the small arena the particular algorithm used and the presence of arena walls increased the incidence of plume loss. T_{DE} remains the same across the GT plumes, and increases drastically for CT0, as the

changing wind direction makes declaration much more difficult. The parameter λ represents that the source declaration process requires a larger area for CT0 (as the wind direction is more variable), so there is more opportunity for inactive agents to interfere. Finally, ω reflects the fact that the less dense plumes can benefit more from parallel declaration because they are less likely to be overcrowded with agents.

Overall, the model is able to fit the data well, as shown in Figure 7.14. Not every point matches perfectly (e.g. 3 agents for ATTRACT3 in GT1), but this is not unexpected, as the optimized data averages only 8 values per data point, and may itself be skewed. Further experimentation could improve these values, although the current set of data took over 60 processor weeks to obtain, and at this stage more simulation is unwarranted because the purpose of the model is not to reproduce data previously generated via other means. The real value of the model–its predictive power–has yet to be tested because there is no complex real-world plume task being addressed. However, the above formulation is representative of the type of abstract model that is intended to be produced by, and later guide, this design process.

7.4 Conclusion

This chapter presented the design of an algorithm for odor localization by groups of autonomous mobile robots. First, a distributed algorithm was described by which groups of agents can solve the odor localization task. Because this algorithm is based upon both odor and flow information, it is not designed to function in environments in which flow is too weak to detect reliably (typically < 0.05 m/s [47]). Still, there is a broad range of military and industrial situations that involve stronger flows (in particular any outdoor environment) for which it does apply.

Next it was demonstrated that simple sensory information tightly coupled with robot behavior is sufficient to allow a robot to find the source of an odor plume. This shows the power of integrating actuation into sensory systems, and suggests that complicated sensory transduction may not be necessary when a behaving sensory mechanism is well tuned to its designated task [101]. In addition, it was shown

that integrating the information collected by a group of agents in an elementary manner can increase the efficiency of the odor localization system performance, an avenue that has not been previously explored using real robots. If the entire system is viewed as an odor localization sensor, the distributed approach opens up a new axis of optimization (inter-agent communication) not available when only a single unit is considered, and the organizational principles of swarm intelligence allow such distributed systems to remain scalable and require minimal additional complexity. The particular communication types explored in this paper represent the most basic interactions available, and as the complexity of the task description increases (more complicated plume types, higher frequencies of false-positive odor hits), correspondingly more complicated interaction schemes (greater number of signals, variable signaling range) will likely be necessary to yield a performance benefit. However, as long as these more complex interactions can be reduced to a small set of parameters, they can be integrated into the algorithm using the design methods presented here.

This chapter demonstrated that key aspects of the design methodology are feasible. The kinematic simulator can reproduce real data, even when sensory stimuli are complex. The optimization process can improve system performance and provide insight into algorithm function. And it is possible to create abstract relationships to guide further development.

Finally, it may seem contradictory that while the swarm intelligence approach stressed in this work emphasizes minimalism, the actual robots used in this study feature general purpose microprocessors and high bandwidth communication. However, because care was taken to keep system requirements low, the algorithms used in this study can be ported directly to much less expensive or smaller platforms. Only when robot swarms can be implemented on a large scale will the robust nature of these systems be fully exploited. As more advanced sensors become available which can combine sensitivity, discrimination, and mobility, truly useful real-world odor localization systems will become feasible.

Chapter 8

Flocking as Improved Collaboration

The previous chapter examined the performance impact of three simple types of collaboration on the full odor localization task. This chapter describes the design of a more complex mode of interaction–flocking–and then investigates its influence on odor localization performance. First, a simple flocking task is presented. Next, a leaderless distributed flocking algorithm is described that is more conducive to implementation on embodied agents than the established algorithms used in computer animation. The design methodology is followed to optimize flocking performance under different conditions, showing (as in the previous chapter) that this process can be used not only to improve performance but also to gain insight into which algorithm components contribute most to system behavior. Then, it is shown that a group of real robots executing the algorithm with emulated sensors can successfully flock (even in the presence of individual agent failure) and that systematic characterization (and therefore optimization) of real-robot flocking performance is achievable. Finally, the integration of a flocking behavior into the odor localization algorithm is demonstrated to shorten task completion times for large group sizes, although corresponding gains in distance traveled offset performance gains for the chosen cost metrics.

8.1 Background

Flocking, the formation and maintenance of coherent group movement, has long been studied in natural systems, and more recently efforts have been made to reproduce this type of behavior in artificial systems. The first such work appeared in the context of computer animation [85]. Since then this behavior has been extensively studied in simulation (e.g., [15]), and less so on real robots [52, 63]. Theoretical treatments of the stability of flocking behavior have also been presented [97, 112, 58], although these studies tend to capture only limited aspects of the flocking problem or rely on unrealistic agent capabilities (such as perfect global communication). The study of flocking is distinct from that of formation control (e.g., [6, 31]), because the goal of flocking is simply to achieve and maintain coherent group movement rather than to govern specific inter-agent position relationships. Flocking is better suited for implementation on large groups of agents (hundreds to thousands) where the overhead of extensive inter-agent communication and unique agent identification renders formation control inefficient. Also, like formation control, flocking is not an end in itself, but rather can be used as a component of a larger multi-agent system, perhaps simplifying the transport of large numbers of agents or organizing the nodes of a distributed sensing system. The majority of this chapter will focus on the design of a scalable flocking algorithm, and the last section will explore the performance impact when a flocking behavior is integrated into the odor localization system studied in the previous chapter.

8.2 The Flocking Task

8.2.1 Task Definition

The flocking task examined in this paper is similar in form to the cooperative movement task studied in [5]. The agents begin each trial at random positions and orientations within an area A located in the corner of a square arena. The agents move diagonally across the arena through an obstacle field toward an area B in the opposite

corner (see Figure 8.3). The trial is declared finished when half of the agents have entered area B. During traversal, there is a uniform probability θ per time step that an agent will "fail," meaning that it stops moving but other agents can still recognize it as a teammate. Note that some trials will not be able to finish (as failed agents can obstruct the movement of operational agents), and these trials are declared failed after some period of time τ. To reduce the number of trials that can never complete the task, the number of robot failures is capped at half of the total number of agents.

For the purposes of this work system performance is defined to be a combination of the time required to complete the task $T_{_F}$, the sum of the distances traveled by each of the successful agents $D_{_F}$, and the average inter-agent distance between operational agents $I_{_F}$. These factors can be combined to form a cost metric C:

$$C = \alpha T_{_F} + \beta D_{_F} + \varphi I_{_F}. \tag{8.1}$$

α is taken to be the cost per unit time of not completing the task, β is the cost per unit distance of running each agent, and φ is the cost incurred per unit distance of inter-agent separation (e.g., if the agents provide mutual protection when grouped together, looser groups would be associated with less protection and higher costs due to agent loss). C represents the total cost incurred before the task is completed. By choosing specific values for α, β, and φ, the proper relationship between time required, energy used, and inter-agent spacing can be generated for evaluating any application. Failed runs are assigned a cost lower than any successful run could receive.

8.2.2 The Leaderless Distributed Flocking Algorithm

Craig Reynolds [85] identifies three behavior types that lead to simulated flocking: separation, alignment, and cohesion. However, much of the robotic work on leaderless flocking ([52, 63]) relies solely on balanced combinations of separation and cohesion (i.e., flock centering) to produce flocking behavior. It is likely that the inclusion of an alignment term into robotic flocking algorithms will improve performance, but there is a cost to making heading information explicitly available within a system. The lead-

erless distributed flocking algorithm (LD) described here is essentially an extension of the flock centering algorithm presented in [15], incorporating an explicit collision avoidance mechanism (for separation, as they suggest) as well as an implicit velocity matching behavior (i.e., an alignment term) via the comparison of sequential flock centering data. Thus, LD should exhibit better flocking performance than previous robotic algorithms (though comparative data is unavailable) while not significantly complicating implementation on real robots. Because LD does not explicitly use the alignment of other group members, individual agents need not be able to sense their neighbors' orientation, so range and bearing data suffice.

Specifically, LD is defined as follows. There are two basic behaviors, collision avoidance and velocity matching flock centering. Collision avoidance is activated whenever an agent's collision sensors detect the presence of an obstacle (which may be either an environmental obstacle or another team member), and it mediates a turn away from the obstacle. Flock centering is active whenever collision avoidance is not, and it involves the generation of a target vector and a target difference vector as well as a mapping from those vectors to wheel speed commands. The details of this behavior are explained below.

After every sensory input cycle, each agent can utilize information from up to Q closest neighbors residing in a region surrounding the agent defined by a maximum range M, as shown in Figure 8.1. Range ($\|n_i\|$) and bearing ($-\pi \leq \hat{n}_i \leq \pi$) information from this set of m neighbors ($i = 1...m, 0 \leq m \leq Q$), along with the desired cushion distance H between each agent and its neighbors, can be used to generate an instantaneous center of mass vector CoM for each agent:

$$CoM = \sum_{i=1}^{m} \left(\frac{\|n_i\| - H}{Q}, \hat{n}_i \right) + (J, \hat{\jmath}). \tag{8.2}$$

CoM is normalized by the maximum number of neighbors to reduce the vector sizes seen at large values of Q. J is a tunable system parameter that represents the strength of the attraction to the goal area, and $\hat{\jmath}$ is the agent centered heading of the goal area (e.g., supplied by a GPS signal). Because the flocking task being studied not only

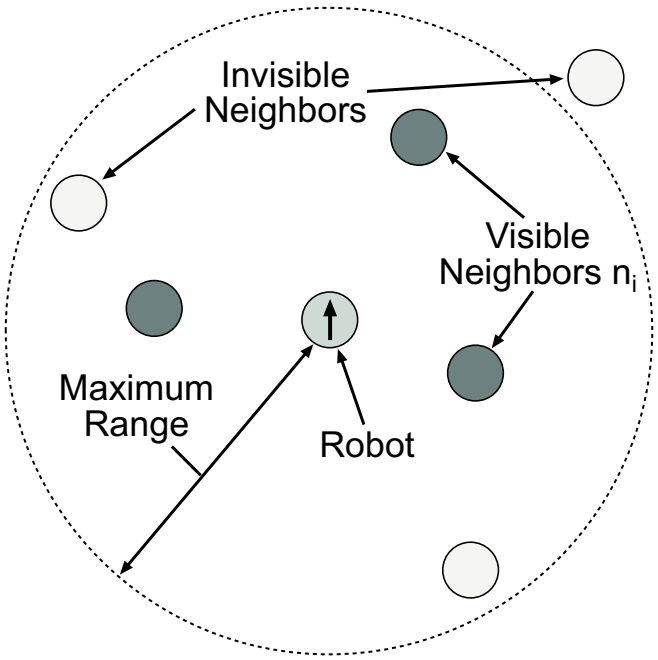


Figure 8.1: Each robot in the flock can sense the range and bearing of up to Q neighbors within a sensory area defined by a maximum range M. In this example $Q = 3$.

favors coherent movement with flock neighbors but also directed movement toward the goal, $(J, \hat{\jmath})$ is added to CoM to induce movement in the proper direction.

CoM is all that is needed to implement flock cohesion, but alignment requires information about how that vector is changing over time. This information is represented by ΔCoM. To generate ΔCoM, the value of CoM generated in the previous sensory cycle (CoM_{prev}) is transformed into the current agent coordinates and combined with the current CoM. Δh is the agent's change in heading between sensory cycles, and e is the agent's change in position:

$$\Delta CoM = CoM - \left[\left(\left(CoM_{prev} - \frac{\Delta h}{2}\right) - e\right) - \frac{\Delta h}{2}\right]. \tag{8.3}$$

The relationships between the algorithm components are summarized in Figure 8.2.

The agent has access to its desired position with respect to its neighbors (CoM) as well as how that location is moving with respect to the agent (ΔCoM). These values are used to generate the motor commands. The gain factor Φ allows the agents to speed up or slow down to approach CoM using a gain parameter K_2 and

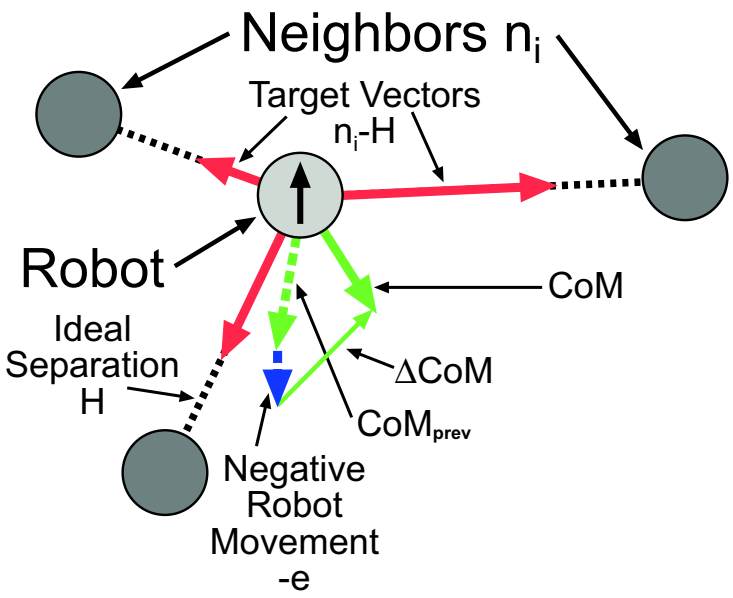


Figure 8.2: A summary of the generation of CoM and ΔCoM.

the maximum sensor range M:

$$\Phi = \frac{M + K_2 \left\| CoM \right\| \cos(\widehat{CoM})}{M}. \tag{8.4}$$

As above, $\|CoM\|$ denotes the magnitude and $\hat{CoM}$ the direction of the vector CoM. Φ is factored into the motor commands as follows:

$$LSpeed = (V - K_0(\Delta\widehat{CoM} + K_1\widehat{CoM}))\Phi \tag{8.5}$$

$$RSpeed = (V + K_0(\Delta\widehat{CoM} + K_1\widehat{CoM}))\Phi. \tag{8.6}$$

The motor speeds are biased at a desired travel speed V. They are changed differentially to rotate toward the heading specified by a weighted sum of the direction of the desired location and the direction of movement of the desired location. K_0 is a weighting parameter that determines how fast an agent can approach this target heading. K_1 weights the influence of the desired location direction versus the desired location movement direction. A small K_1 ($\ll 1$) will induce agents to align with their neighbors (thus minimizing $\Delta\widehat{CoM}$) rather than to move toward their desired

locations, although once alignment is achieved the agents will gradually steer toward CoM (provided $K_1 > 0$).

Note that it is not necessary to calculate the optimal movement necessary to reach the goal position in order to have a functional system. As long as the commanded wheel speeds bring each agent closer to its desired position during each sensory cycle (and CoM moves slower than the agent itself), in steady state all agents will approach their goal positions. Formal stability conditions and proofs are not examined in this chapter, although stable flocking systems were observed over a broad range of the 8 tunable algorithm parameters. A summary of these parameters is shown in Table 8.1.

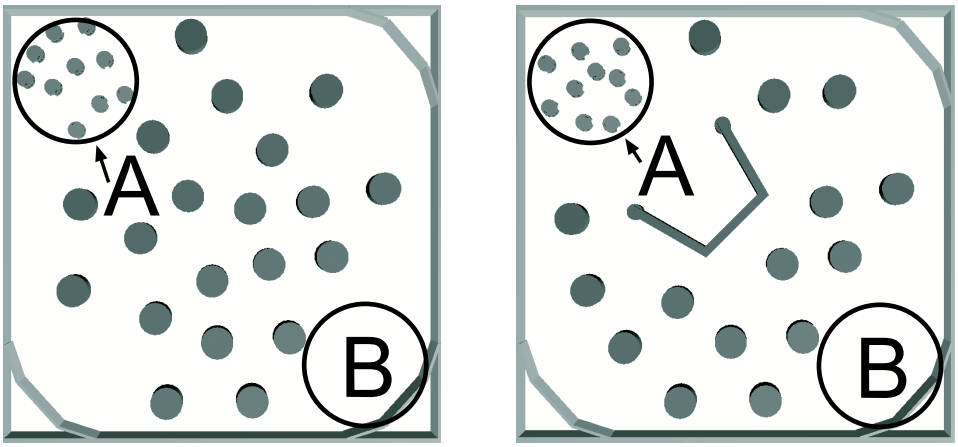
Table 8.1: Leaderless Distributed Flocking Algorithm Parameters

V	Desired forward speed
Q	Maximum number of neighbors
H	Desired distance between agents
M	Maximum sensor range
K_{0-2}	Motor speed gain parameters
J	Target attraction

8.3 Test Environments

8.3.1 Kinematic Simulation

The physical arena was reproduced in Webots to allow comparison with data generated by the real robots, and two different obstacle fields were studied. The simpler of the two (`Obs1`) contained only cylindrical obstacles that were twice as large as each agent, while the more complex (`Obs2`) also contained a three-sided barrier that obstructed the direct path between the start (A) and goal (B) areas. These environments are shown in Figure 8.3. The timeout value τ was 400 s.



(**a**) (**b**)

Figure 8.3: (**a**) `Obs1` and (**b**) `Obs2`, seen from above. The start (A) and goal (B) areas are indicated. The large disks are the obstacles, and the smaller disks (shown here within the start area) are the agents.

8.3.2 Real Robots

A group of 10 Moorebots was used to demonstrate flocking in real robots, as shown in Figure 8.4. The flocking arena is 6.7 by 6.7 m. The layout of the arena is the same as shown in Figure 8.3a, except in this case a single obstacle was placed in the center of the arena. In addition to the standard configuration, as described in [108], each robot is equipped with four Sharp GP2-D02 infrared range sensors for collision avoidance. The overhead camera tracking system, combined with a radio LAN among the robots and an external workstation, is used to log position data during the trials, reposition the robots between trials, and emulate the range and bearing sensor signals.

8.4 Results and Discussion

8.4.1 Optimization with the Kinematic Simulator

The optimization procedure for this flocking task involves the off-line tuning of 8 parameters. Since a full 8-D optimization is not computationally feasible, instead 8 sequential 1-D optimizations are performed, with each parameter optimized while the

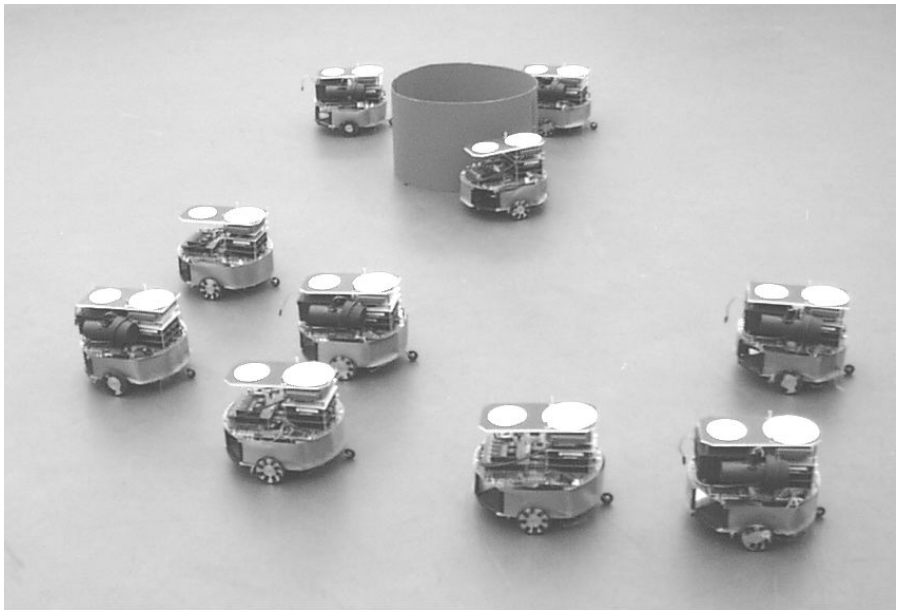


Figure 8.4: 10 Moorebots flocking.

others remain fixed. While this restriction may make finding the optimal parameter set difficult in some search domains, it does not do so in the particular case being studied, and it allows performance improvements to be achieved in a reasonable amount of time. In this study the selection of design points (i.e., specific parameter values over which to optimize) is done a priori, although there are techniques for selecting them adaptively [111] which may be utilized in further studies. Each parameter space is bounded and linearly discretized to include a range of important values, as determined by preliminary experiments. At the beginning of each optimization run the variable values are randomly initialized. The optimization process is described in detail in Chapter 3.

Four different conditions were optimized, consisting of 10 runs each: `Obs1` and `Obs2`, each with (`F`) and without failures (`NF`). The `F` conditions set θ so that there were $\sim$2 failures per trial given near-optimal algorithm parameters for the `Obs1` environment. The cost values were the same for all trials, and they were fixed to balance the cost components for near-optimal algorithm parameters under the `Obs1NF` condition: $\alpha = .0033$ [\$/s], $\beta = .0066$ [\$/m], and $\gamma = .22$ [\$/m]. Optimization parameter

values were as follows: $\eta = .05$, $\kappa = .05$, and $\epsilon = 10$.

Figure 8.5a shows the flocking performance with standard deviation at every cycle for each of the four conditions. For ease of presentation, $\frac{C_{min}}{C}$ is plotted, where C_{min} represents the lowest cost observed over all trials. Cycle 0 data represents the performance of the initial parameter sets. `Obs1NF` converges to the highest performance value, with `Obs1F` slightly worse, followed by `Obs2NF` and then `Obs2F`. This shows that `Obs2` is a more difficult environment than `Obs1`, and the presence of agent failures can hurt performance. Under all four conditions the means and standard deviations stabilize after 4 optimization cycles, showing that optimization does improve performance and is complete after a small number of cycles. The fact that all conditions have a small standard deviation across runs once optimized (after cycle 4) suggests that even though the optimization algorithm searches only one dimension at a time, it is performing an effective search of the fitness landscape and is not susceptible to being trapped in local minima. The standard deviations for the `F` and `Obs2` conditions are larger because in these environments occasional runs fail to complete within the timeout period, and thus the performance metrics (for individual parameter sets) have higher variances.

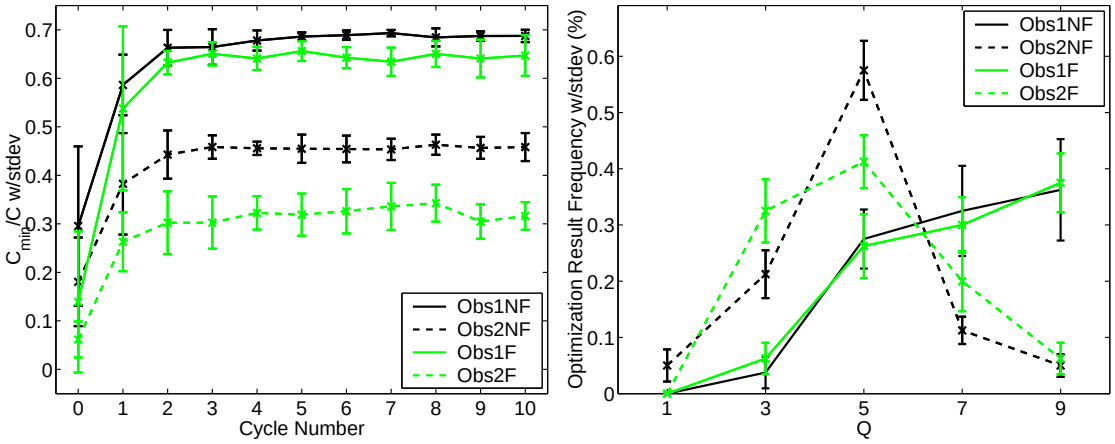


Figure 8.5: (**a**) Per-cycle flocking performance for each experimental condition. Higher values are better. (**b**) The optimal result frequency curves for Q, the maximum number of neighbors observed while flocking.

The optimization procedure can do more than improve system performance, be-

cause by looking at the optimized parameter values one can gain insight into the operation of the algorithm itself. The optimized parameters were analyzed first by combining cycle results 4-10 from each run, and then averaging the chosen parameter distributions across the 10 runs for each condition (this results in an optimization result frequency curve). Figure 8.5b shows that there are optimal values of Q for each environment and that they are different, with `Obs2` preferring smaller neighborhoods. This is an intuitive result because when an agent is listening to fewer neighbors, it is less likely to be impeded by a neighbor that is caught behind a barrier (which is more common in `Obs2`), while in more open environments larger neighborhoods allow for tighter flocks and thus a higher performance level. Using pointwise one-way ANOVA comparisons ($p < .01$) and a threshold of > 1 significant difference, it was determined that the Q optimization result frequency curves for the `Obs1` conditions differ from those of `Obs2`, while they remain the same within each simulated environment across failure rates. In fact, for the `Obs1` conditions the result frequency curves did not differ for any parameter, indicating that the best solution in that environment remained the same even in the presence of agent failure. This makes sense because a failed agent simply becomes another circular obstacle. It might have been expected that the presence of failed agents would favor a reduced agent neighborhood (so failed agents do not impede the progress of those still active), but Figure 8.5b clearly shows that there is no preference for smaller Q in `Obs1F` as compared to `Obs1NF`.

In the `Obs2` environment, the optimal parameter values are influenced by the presence of agent failure, as shown in Figure 8.6a. In the absence of agent failure, a low value of J is optimal, so that the attractive force of the goal does not break groups apart as they move around the barrier. When agents can fail, however, the task becomes so difficult (because failed agents can trap others within the barrier) that the best solution is to move as individuals toward the goal whenever the opportunity presents itself (so a high J is best). Note that for the `Obs1` conditions, there is a broad region of the parameter space over which the performance landscape is effectively flat. This type of finding suggests that in some cases the size of the parameter space being searched may be reduced, resulting in faster optimization runs without a loss of

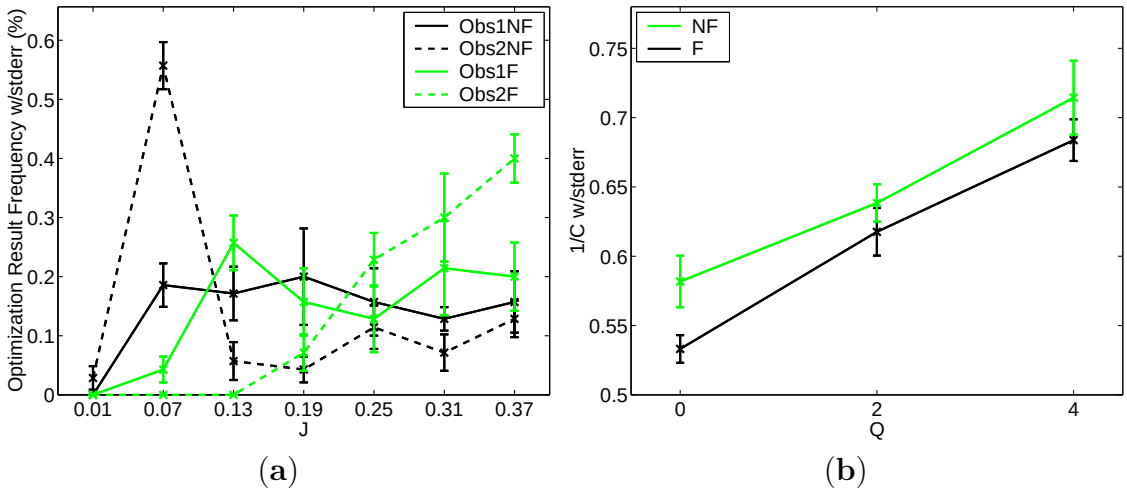


Figure 8.6: (**a**) The optimal result frequency curves for J, the attractive power of the goal area. (**b**) Flocking performance of a group of 10 real robots versus Q, the maximum number of visible neighbors. Higher values are better.

performance.

8.4.2 Real Robots

Because local range and bearing hardware has not been completed, the Moorebots must rely on emulated sensory information from the overhead camera system to perform LD. The processing burden thus placed on the camera system limits the maximum speed of the robots, as the camera system must be able to track the robots from frame to frame by position only. This restriction, along with the fact that control is not truly distributed, renders extensive experimental effort unwarranted. However, to demonstrate that the capability to quantitatively characterize real-robot flocking performance (and thus in principle can reproduce the simulated optimization experiments presented above in the real world), a set of reasonable flocking parameter values was chosen and the influence of varying Q on the performance of a group of 10 real robots was examined.

For each value of Q, 10 trials were run under both the `F` and `NF` conditions, and the resulting performance values are shown in Figure 8.6b. The average number of failures over all `F` trials was 2.46. LD at $Q = 0$ represents a baseline traversal behavior (because there is no cooperation among agents). The data demonstrates

that LD does enable this group of robots to flock, as flocking performance is greater at $Q = 4$ than $Q = 0$ (significant via ANOVA to $p < .01$), while agent failure does not significantly influence performance (via ANOVA to $p < .01$– although it is likely that larger sample sizes would uncover a significant difference). Because the specifics of these results are likely to be highly dependent on the particular parameter values chosen (most of which are arbitrary rather than optimized), detailed comparison with the simulation results is not meaningful.

8.5 Flocking as Collaboration

Flocking is integrated into the odor localization as a more complex ATTRACT behavior. Rather than simply surging toward the location of the most recent odor hit by any agent, agents performing the search behavior fix the most recent hit location as the target destination (i.e., the center of area B) and move toward it while flocking with other agents. Flocking parameters are based on an optimized set from the `Obs1NF` condition, and the desired inter-agent spacing is increased to minimize interference within the plume. So as not to interrupt the source declare behavior, agents not in the search phase do not flock (although they are visible to flocking agents). Also, to make the influence of flocking clear, once three agents have received plume hits, the rest of the agents are inactivated, as in ATTRACT3. The odor localization with flocking system is termed FLOCK3.

The same optimization process performed on the other collaboration methods was run in simulation on FLOCK3 (see Section 7.3.1 for a full description), and the performance results are shown in Figure 8.7. Overall, there is little difference between the performance of ATTRACT3 and FLOCK3. This could indicate that flocking simply does not influence odor localization, or it may be due to the fact that the flocking phase lasts only briefly, because as soon as three agents receive odor hits the two communication types become identical.

The plume that is most likely to show a difference between these two communication types is GT2, as it is least dense and the period of time between the first

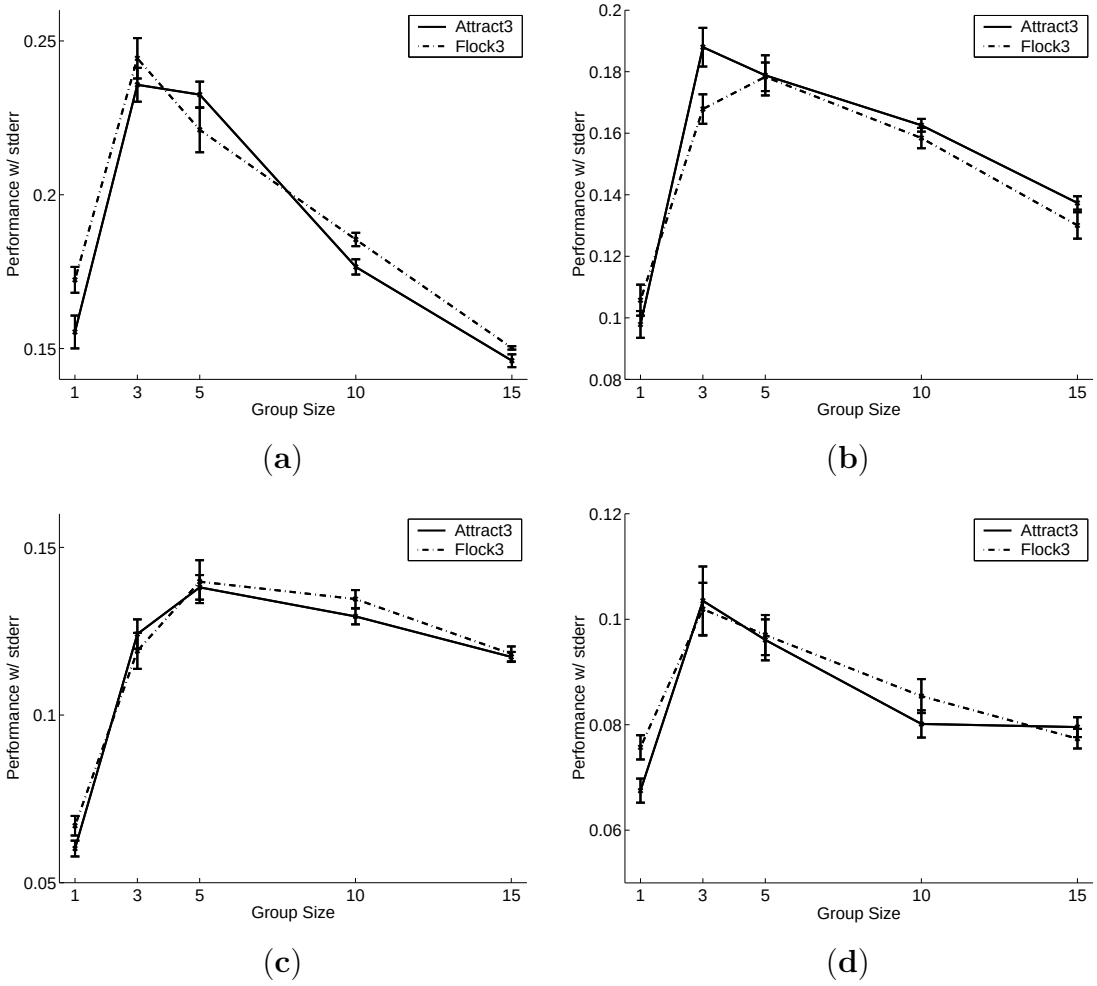
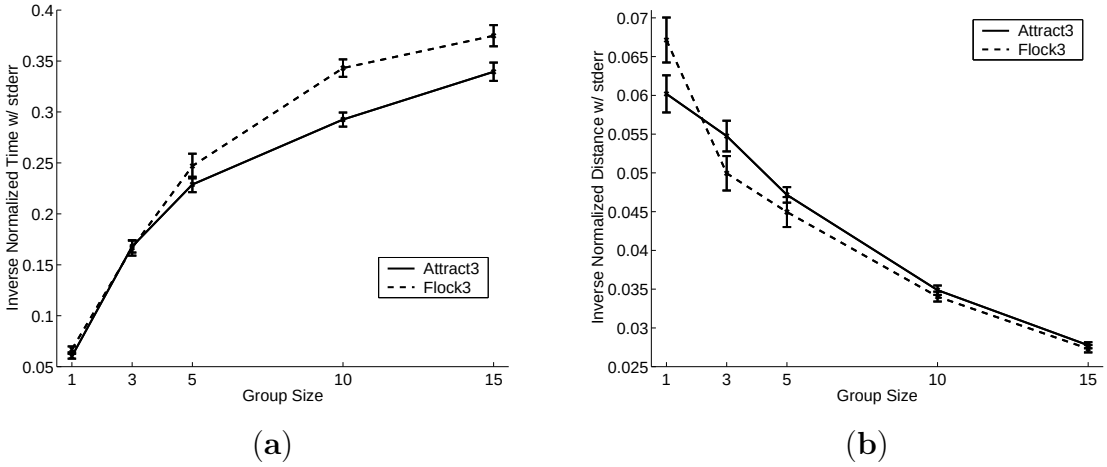


Figure 8.7: Performance across group size for (**a**) GT0, (**b**) GT1, (**c**) GT2, and (**d**) CT0. Higher values are better. Error bars represent standard error.

and third agents receiving odor hits is the greatest. The time and distance required for task completion (first normalized by the minimum values and then inverted for ease of presentation) are shown in Figure 8.8. It can be seen from this data that at larger group sizes, FLOCK3 requires less time than ATTRACT3 (significant for group sizes 10 and 15 via ANOVA to $p < .05$–2 runs were performed for the case of 15 agents, so each average consists of 16 data points rather than 8). However, this speed benefit is offset by the larger distance traveled by the ATTRACT3 system, presumably because the flocking agents are spending less time interfering with each other (thus moving at a higher average speed). Therefore, the incorporation of flocking into the



(**a**) (**b**)

Figure 8.8: (**a**) Inverse of normalized time required for GT2. (**b**) Inverse of normalized distance required for GT2. Higher values are better. Error bars represent standard error.

odor localization behavior can extend the performance space of the system, and if time were a much larger factor in performance than energy used (a tradeoff that can only be determined for a specific application), it could increase system performance. Also, integrating the flocking behavior with the source declaration procedure, allowing agents to share information (locally) during the declare process, could lead to further performance benefits.

8.6 Conclusion

This chapter described the design of a flocking behavior and investigated its influence on odor localization performance. A simple flocking task was presented and a leaderless distributed flocking algorithm was described that is more conducive to implementation on robots than the established algorithms used in computer animation. The key point of this algorithm is that it uses the time derivative of the perceived center of the flock to align the robots without explicit knowledge of robot heading. The design methodology was followed to optimize performance under different conditions, showing that this method can be used not only to improve performance but also to gain insight into which algorithm components contribute most to system behavior.

An issue for further study is the automation of the selection, or perhaps improvement of, the parameter ranges and discretization levels that are searched. In addition, using optimization data it may eventually be possible to construct models that directly relate environmental characteristics to parameter values. It was also demonstrated that a group of real robots executing LD with emulated sensors can successfully flock and that systematic characterization of real-robot flocking parameters is achievable. Finally, the integration of a flocking behavior into the odor localization algorithm was demonstrated to speed task completion for large group sizes, although corresponding gains in distance traveled offset performance gains for the chosen cost metrics.

Chapter 9

Conclusion

The creation of autonomous robots, machines that sense and act upon the world to perform useful work without constant human supervision, could free humans from many repetitive or dangerous tasks and increase productivity immensely. However, currently there are very few examples of robotic systems that can operate with any degree of autonomy in unstructured environments, and there are none that approach the level of robustness observed in relatively simple biological organisms, such as ants and termites. Part of the biological advantage is that natural systems possess a distributed control structure consisting of many parallel local processes (i.e., they are self-organized) which are not susceptible to single failures and do not rely on perfect sensing or communication. This thesis presented a methodology for designing self-organized autonomous robotic systems, and demonstrated how this process can be applied to the problem of finding the source of an airborne odor plume. The design methodology is applicable to other task domains and the resulting odor localization system extends the state of the art.

Specifically, the contributions of this thesis are as follows:

- A self-organized system design methodology that relies on the formulation and evaluation of specific task metrics.

The design procedure centers on the ability to define a specific task performance metric, systematically evaluate performance in a realistic environment, and define an abstract model that relates system parameters and system performance. Once such

relationships have been validated in a test environment by comparing experimentally generated performance distributions with those derived from the model, they can be used to guide the design of a deployable system. Because this design process relies heavily on evaluative feedback, this work emphasizes the development of tools that allow the collection of accurate performance data. Also, a reinforcement learning methodology is described that provides consistent optimization performance while minimizing the amount of required evaluation.

- An improved odor localization system that can derive useful information from the distal part of an odor plume.

The design methodology is applied to the task of odor localization. A plume traversal algorithm is implemented both on the real test-bed and in simulation to verify that plume traversal is taking place and that the use of multiple collaborating robots can expand the reachable performance space. Also, parameter optimization performed in simulation is shown to produce better performance on the real robot platform. Systematic experiments on the real odor plume required development of the robot platform, the arena infrastructure, and the odor and wind sensors. Collective search and plume traversal are combined (along with ego-centric source declaration) into the full odor localization task which is optimized in simulation. The odor localization algorithm is shown to be functional on both sparse and meandering plumes. Then, following the design methodology, a model is presented which captures system performance across algorithms and environments. The real value of the model–its predictive power–has yet to be tested because there is no complex real-world plume task being addressed. However, it is representative of the type of abstract model that is intended to be produced by, and later guide, this design process.

- Greater insight into the tradeoffs between sensor reliability, evaluation metrics, and coverage strategy for collective search problems.

A quantitative analysis of the tradeoffs between group size and efficiency in collective search tasks is presented that considers both the time-sensitive nature of search

completion and the system operating cost. For both random and coordinated search strategies, analytical expressions are derived that can be used to predict optimal system performance bounds given a particular task description. Also, the performance benefit of using coordinated search is shown to be dependent on the relative values of the different cost components. Finally, a sensor-based computer simulation is used to support the analytical results, suggesting that the assumptions involved in their derivation are sound.

- An understanding of the the general properties required of successful turbulent odor plume traversal algorithms.

The problem of plume traversal is recast as the task of obtaining the next odor hit, and a set of metrics that provides detailed information about algorithm function is presented. Several odor localization algorithms are described, and it is shown that algorithm parameters can be tailored to particular plume characteristics for improved performance. Also, the next-hit analysis is demonstrated to capture the performance of some types of algorithms more accurately than others, and it is concluded that this failure stems from intrinsic shortcomings of some of the algorithms tested.

- A flocking algorithm that is well suited to implementation on real hardware.

A leaderless distributed flocking algorithm is described that is more conducive to implementation on embodied agents than the established algorithms used in computer animation. The design methodology is followed to optimize flocking performance under different conditions, showing that this process can be used not only to improve performance but also to gain insight into which algorithm components contribute most to system behavior. It is shown that a group of real robots executing the algorithm with emulated sensors can successfully flock (even in the presence of individual agent failure) and that systematic characterization (and therefore optimization) of real-robot flocking performance is achievable.

This thesis has focused on the design of self-organized robotic systems. There exists skepticism in the robotics community that such systems will ever prove useful,

particularly because it is deemed difficult to integrate them into traditional (military) command structures, and military funding drives most robotic research. There is understandable concern about deploying robotic systems and relinquishing the ability to control (or even communicate with) them at all times, but this is necessary when dealing with large numbers of agents. In order to prove that self-organized systems are feasible, a successful large scale system must be demonstrated. However, before this can be done, a suitable task must be found that both benefits from the extreme parallelism of many agents and warrants the significant investment required to develop a complete system. Odor localization, because it is essentially a search problem, fulfills the former requirement, and future circumstances may see it satisfy the latter as well.

Bibliography

[1] E. A. Arbas, M. A. Willis, and R. Kanzaki. Organization of goal-oriented locomotion: Pheromone-modulated flight behavior of moths. In R. D. Beer, R. E. Ritzmann, and T. McKenna, editors, *Biological Neural Networks in Invertebrate Neuroethology and Robotics*. Academic Press, New York, 1993.

[2] R. C. Arkin. *Behavior Based Robotics*. MIT Press, Cambridge, MA, 1998.

[3] J. Atema. Eddy chemotaxis and odor landscapes: Exploration of nature with animal sensors. *Biological Bull.*, 191:129–138, 1996.

[4] T. Balch. Integrating rl and behavior-based control for soccer. In *Proc. of the IJCAI Workshop on RoboCup*, Nagoya, Japan, 1997.

[5] T. Balch. *Behavioral Diversity in Learning Robot Teams*. Ph.D Thesis, College of Computing, Georgia Institute of Technology, December 1998.

[6] T. Balch and R. C. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Robotics and Automation*, 14(6):926–939, December 1998.

[7] J. H. Belanger and M. A. Willis. Adaptive control of odor guided locomotion: Behavioral flexibility as an antidote to environmental unpredictability. *Adaptive Behavior*, 4:217–253, 1996.

[8] G. Beni and J. Wang. Swarm intelligence in cellular robotic systems. In *Proceedings of the NATO Advanced Workshop on Robots and Biological Systems*, Il Ciocco, Tuscany, Italy, 1989.

[9] S. Benkowki, M. Monticino, and J. Wiesinger. A survey of the search theory literature. *Naval Research Logisitics*, 38(4):469–494, 1991.

[10] U. Bhalla and J. M. Bower. Multi-day recording from olfactory bulb neurons in awake freely moving rats: Spatial and temporally organized variability in odorant response properties. *J. of Computational Neuroscience*, 4:221–256, 1997.

[11] A. Billard, A. J. Ijspeert, and A. Martinoli. A multi-robot system for adaptive exploration of a fast changing environment: Probabilistic modelling and experimental study. *Connection Science*, 11:359–379, 1999.

[12] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, US, 1999.

[13] L. Booker, D. E. Goldberg, and J. H. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40(1-3):235–282, 1989.

[14] V. Braitenberg. *Vehicles. Experiments in Synthetic Psychology*. MIT Press, Cambridge, MA, 1984.

[15] D. C. Brogan and J. K. Hodgins. Group behaviors for systems with significant dynamics. *Autonomous Robots*, 4:137–153, 1997.

[16] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Robotics and Automation*, RA-2:14–23, March 1986.

[17] L. Bruno. Mister roboto: Pioneer Joseph Engelberger shares his passion for robotics. *Red Herring*, August 2000.

[18] W. Burgard, A. B. Cremers, D. Fox, D. Hanel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55, 1999.

[19] S. Camazine. *Self-organization in biological systems*. Princeton University Press, Princeton, N.J., 2001.

[20] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:1–23, 1997.

[21] R. T. Carde and A. Mafra-Neto. Effect of pheromone plume structure on moth orientation to pheromone. In R. T. Carde and A. K. Minks, editors, *Perspectives on Insect Pheromones. New Frontiers*, pages 275–290. Chapman and Hall, N.Y., 1996.

[22] H. Choset. Topological simultaneous localization and mapping (slam): Toward exact localization without explicit mapping. *IEEE Transactions on Robotics and Automation*, 17(2):125–137, April 2001.

[23] H. Choset, J. Burdick, S. Walker, and K. Diamsa-Ard. Sensor based exploration: Incremental construction of the hierarchical generalized Voronoi graph. *International Journal of Robotics Research*, 19(2):126–148, February 2000.

[24] M. Colombetti, M. Dorigo, and G. Borghi. Behavior analysis and training: A methodology for behavior engineering. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(3):365–380, 1996.

[25] C. T. David, J. S. Kennedy, J. S. Ludlow, and J. N. Perry. A re-appraisal of insect flight towards a point source of wind-borne odor. *Journal of Chemical Ecology*, 8:1207–1215, 1982.

[26] B. J. Doleman, M. C. Lonergan, E. J. Severin, Vaid T. P., and Lewis N. S. Quantitative study of the resolving power of arrays of carbon black-polymer composites in various vapor-sensing tasks. *Anal. Chem.*, 70:4177–4190, 1998.

[27] J. F. Engelberger. *Robotics in practice : management and applications of industrial robots*. AMACOM, New York, 1980.

[28] J. Farrell. Plume tracing simulator. `http://www.ee.ucr.edu/farrell/`.

[29] D. Floreano and Mondada F. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man and Cybernetics*, 26:396–407, June 1996.

[30] United Nations Economic Commission for Europe and The International Federation of Robotics. *World Robotics 2001 - Statistics, Market Analysis, Forecasts, Case Studies and Profitability of Robot Investment*. Geneva, 2001.

[31] J. Fredslund and M. J. Mataric. Robot formations using only local sensing and control. In *Proceedings, International Symposium on Computational Intelligence in Robotics and Automation (IEEE CIRA 2001)*, Banff, Alberta, Canada, July 2001.

[32] M. S. Freund and N. S. Lewis. A chemically diverse conducting polymer-based electronic nose. *Proceedings of the National Academy of Sciences USA*, 92:2652, 1995.

[33] D. W. Gage. Randomized search strategies with imperfect sensors. In *Proceedings of SPIE Mobile Robots VIII*, volume 2058, pages 270–279, Boston, September 1993.

[34] D. W. Gage. Many-robots MCM search systems. In A. Bottoms, J. Eagle, and H. Bayless, editors, *Proceedings of Autonomous Vehicles in Mine Countermeasures Symposium*, pages 9.55–9.63, Monterey, April 1995.

[35] D. Goldberg and M. J. Mataric. Design and evaluation of robust behavior-based controllers. In T. Balch and L. E. Parker, editors, *Robot Teams: From Diversity to Polymorphism*. AK Peters, Ltd., 2002.

[36] F. W. Grasso, T. R. Consi, D. C. Mountain, and J. Atema. Biomimetic robot lobster performs chemo-orientation in turbulence using a pair of spatially separated sensors. *Robotics and Autonomous Systems*, 30:115–131, 2000.

[37] A. R. Graves and C. A. Czarnecki. Design patterns for behaviour-based robotics. *IEEE Transactions on Systems, Man and Cybernetics - Part A:Systems and Humans*, 30(1):36–41, January 2000.

[38] A. T. Hayes. How many robots? Group size and efficiency in collective search tasks. In *Proc. of the sixth Int. Symp. on Distributed Autonomous Robotic Systems DARS-2002*, Fukuoka, Japan, June 2002. Springer Verlag. To appear.

[39] A. T. Hayes, A. Martinoli, and R. M. Goodman. Comparing distributed exploration strategies with simulated and real autonomous robots. In L. E. Parker, G. Bekey, and J. Barhen, editors, *Proc. of the fifth Int. Symp. on Distributed Autonomous Robotic Systems DARS-2000*, pages 261–270, Knoxville, Tennessee, October 2000. Springer Verlag.

[40] A. T. Hayes, A. Martinoli, and R. M. Goodman. Swarm robotic odor localization. In *Proc. of the IEEE Conf. on Intelligent Robots and Systems*, pages 1073–1078, Wailea, HI, October 2001. IEEE Press.

[41] A. T. Hayes, A. Martinoli, and R. M. Goodman. Distributed odor source localization. *IEEE Sensors*, 2002. To appear.

[42] J. H. Holland. *Adaptation in natural and artificial systems.* The University of Michigan Press, Ann Arbor, 1975.

[43] O. E. Holland and C. Melhuish. Stigmergy, self-organization, and sorting in collective robotics. *Artificial Life*, 5:173–202, 1999.

[44] D. F. Hougen, P. E. Rybski, and M. Gini. Repeatability of real world training experiments: A case study. *Autonomous Robots*, 6(3):281–292, 1999.

[45] A. J. Ijspeert, A. Martinoli, A. Billard, and L. M. Gambardella. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, 11(2):149–171, 2001.

[46] H. Ishida, Y. Kagawa, T. Nakamoto, and T. Moriizumi. Odor-source localization in the clean room by an autonomous mobile sensing system. *Sensors and Actuators B*, 33:115–121, 1996.

[47] H. Ishida, T. Nakamoto, T. Moriizumi, T. Kikas, and J. Janata. Plume-tracking robots: A new application of chemical sensors. *Biological Bulletin*, 200:222–226, April 2001.

[48] C. D. Jones. On the structure of instantaneous plumes in the atmosphere. *Journal of Hazardous Materials*, 7:87–112, 1983.

[49] R. N. Kachar. Off-line quality control, parameter design, and the Taguchi method. *Journal of Quality Technology*, 17:176–209, 1985.

[50] R. Kanzaki, N. Sugi, and T. Shibuya. Self-generated zigzag turning of Bombyx mori males during pheromone-mediated upwind walking. *Zoological Science*, 9:515–527, 1992.

[51] S. Kazadi, R. Goodman, D. Tsikata, and H. Lin. An autonomous water vapor plume tracking robot using passive resistive polymer sensors. *Autonomous Robots*, 9(2):175–188, 2000.

[52] I. D. Kelly and D. A. Keating. Flocking by the fusion of sonar and active infrared sensors on physical autonomous mobile robots. In *Proc. of the The Third Int. Conf. on Mechatronics and Machine Vision in Practice*, pages 1/1–1/4, Guimaraes, Portugal, 1996.

[53] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Academic Press, San Diego, CA, 2001.

[54] J. R. Koehler, A. A. Puhalskii, and B. Simon. Estimating functions evaluated by simulation: A bayesian/analytic approach. *The Annals of Applied Probability*, 8(4):1184–1215, 1998.

[55] M. J. B. Krieger, J. Billeter, and L. Keller. Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406:992–995, August 2000.

[56] C. R. Kube and E. Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30:85–101, 2000.

[57] Y. Kuwana, I. Shimoyama, Y. Sayama, and H. Miura. Synthesis of pheromone-oriented emergent behavior of a silkworm moth. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1722–1729, 1996.

[58] N. E Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. In *Proceedings of the 40th IEEE Conference on Decision and Control*, pages 2968–2973, 2001.

[59] K. Lerman. Learning real team solutions. In *Lecture Notes in Artificial Intelligence (LNAI) 1871*. Springer Verlag, Berlin, 2001.

[60] H. Lipson and J. B. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978, 2000.

[61] C. Loizos. Special feature: Service-sector robotics: Robots are breaking out of the factory. *Red Herring*, September 1998.

[62] M. C. Lonergan, E. J. Severin, B. J. Doleman, S. A. Beaber, R. H. Grubbs, and N. S. Lewis. Array-based vapor sensing using chemically sensitive, carbon black-polymer resistors. *Chem. Mater.*, 8:2298–2312, 1996.

[63] Mataric M. *Interaction and Intelligent Behavior*. Ph.D Thesis, MIT, Boston, May 1994.

[64] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55(2-3):311–365, June 1992.

[65] A. Martinoli. *Swarm Intelligence in Autonomous Collective Robotics: From Tools to the Analysis and Synthesis of Distributed Control Strategies*. Ph.D Thesis Nr. 2069, EPFL, Lausanne, Switzerland, October 1999.

[66] A. Martinoli, A. J. Ijspeert, and L. G. Gambardella. A probabilistic model for understanding and comparing collective aggregation mechanisms. In D. Floreano, F. Mondada, and J.-D. Nicoud, editors, *Proc. of the Fifth Int. European Conf. on Artificial Life ECAL-99, Lecture Notes in Computer Science*, pages 575–584. Springer Verlag, Lausanne, Switzerland, September 1999.

[67] A. Martinoli, A. J. Ijspeert, and F. Mondada. Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots. *Robotic and Autonomous Systems*, 29:51–63, 1999.

[68] M. J. Mataric. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4(1):51–80, December 1995.

[69] M. J. Mataric. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16(2-4):321–331, December 1995.

[70] M. J. Mataric. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, March 1997.

[71] M. J. Mataric. Learning in behavior-based multi-robot systems: Policies, models, and other agents. *Cognitive Systems Research, special issue on Multidisciplinary studies of multi-agent learning*, 2(1):81–93, April 2001.

[72] A. J. Matzger, C. E. Lawrence, R. H. Grubbs, and N. S. Lewis. Combinatorial approaches to the synthesis of vapor detector arrays for use in an electronic nose. *J. Comb. Chem.*, 2:301–304, 2000.

[73] J. McCarthy, M. Minsky, N. Rochester, and C. Shannon. *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*. August 1955.

[74] Merriam-Webster, editor. *Merriam-Webster's Collegiate Dictionary*. International Thomson Publishing, London, 1998.

[75] O. Michel. Webots: Symbiosis between virtual and real mobile robots. In *Proceedings of the First International Conference on Virtual Worlds, VW'98*, pages 254–263, Paris, France, July 1998. Springer Verlag.

[76] M. L. Minsky. Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49:8–30, 1961.

[77] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturization: A tool for investigation in control algorithms. In T. Yoshikawa and F. Miyazaki, editors, *Proc. of the Third International Symposium on Experimental Robotics ISER-93*, pages 501–513, Kyoto, Japan, 1993. Springer Verlag.

[78] A. Murciano, J. R. Millan, and Z. Zamora. Specialization in multi-agent systems through learning. *Biological Cybernetics*, 76:375–382, 1997.

[79] J. Murlis, J. S. Elkington, and R. T. Carde. Odor plumes and how insects use them. *Annu. Rev. Entomol.*, 37:505–532, 1992.

[80] H. T. Nagle, R. Guitierrez-Osuna, and S. S. Schiffman. The how and why of electronic noses. *IEEE Spectrum*, 35(9):22–31, September 1998.

[81] T. Nakamoto, H. Ishida, and T. Moriizumi. A sensing system for odor plumes. *Analytical Chemistry*, 71(15):531A–537A, August 1999.

[82] L. E. Parker. Lifelong adaptation in heterogeneous multi-robot teams: Response to continual variation in individual robot performance. *Autonomous Robots*, 8(3):239–267, 2000.

[83] M. S. Phadke. *Quality Engineering Using Robust Design*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[84] P.H. Ramsey. Multiple comparisions of independent means. In L. K. Edwards, editor, *Applied analysis of variance in behavioral science*, volume XI, pages 25–62. Marcel Dekker, New York, 1993.

[85] C. W. Reynolds. Flocks, herds, and schools: A distibuted behavioral model. *Computer Graphics*, 21(1):79–98, 1987.

[86] P. J. W. Roberts. Modeling Mamala Bay outfall plumes. II: Far field. *J. of Hydraulic Engineering*, 125(N6):574–583, 1999.

[87] R. A. Russell. *Odor detection by mobile robots*. World Scientific, Singapore, 1999.

[88] R. A. Russell, D. Thiel, R. Deveza, and A. Mackay-Sim. A robotic system to locate hazardous chemical leaks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 556–561, Nagoya, 1995.

[89] J. C. Santamaria and A. Ram. Learning of parameter-adaptive reactive controllers for robotic navigation. In *Proceedings of the World Multiconference on Systemics, Cybernetics, and Informatics*, Caracas, Venezuela, July 1997.

[90] H. A. Simon. Kasparov vs. Deep Blue: The aftermath - AI lessons. *Communications of the ACM*, 40(8):23–25, August 1997.

[91] D. J. Simons. Current approaches to change blindness. *Visual Cognition: Special Issue on Change Detection and Visual Memory*, 7:1–16, 2000.

[92] M. T. Stacey, E. A. Cowen, T. M. Powell, E. Dobbins, S. G. Monismith, and J. R. Koseff. Plume dispersion in a stratified, near coastal flow: measurements and modeling. *Continental Shelf Research*, 20:637–663, 2000.

[93] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Cambridge, MA, 1998.

[94] Y. Takahashi and M. Asada. Behavior acquisition by multi-layered reinforcement learning. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 716–721, 1999.

[95] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.

[96] S. Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5):335–363, 2001.

[97] J. Toner and Y. Tu. Flocks, herds, and schools: a quantitative theory of flocking. *Physical Review E*, 58(4):4828–4858, 1998.

[98] C. Versino and L. M. Gambardella. Learning real team solutions. In G. Weiss, editor, *DAI Meets Machine Learning, Lectures in Artificial Intelligence*, pages 298–311. Springer Verlag, Berlin, 1997.

[99] N. J. Vickers and T. C. Baker. Reiterative responses to single strands of odor promote sustained upwind flight and odor source location by moths. *Proceedings of the National Academy of Sciences USA*, 91:5756–5760, 1994.

[100] G. Walter. *The Living Brain*. Norton, New York, 1953.

[101] B. Webb. View from the boundary. *Biological Bulletin*, 200:184–189, April 2001.

[102] D. R. Webster, S. Rahman, and L. P. Dasi. On the usefulness of bilateral comparison to tracking turbulent chemical odor plumes. *Limnology and Oceanography*, 46(5):1048–1053, 2001.

[103] D. R. Webster, S. Rahman, and L.P. Dasi. Laser-induced fluorescence measurements of a turbulent plume. *ASCE Journal of Engineering Mechanics*. Submitted.

[104] D. R. Webster, P. J. W. Roberts, and L. Ra'ad. Simultaneous dptv/plif measurements of a turbulent jet. *Experiments in Fluids*, 30:65–72, 2001.

[105] D. R. Webster and M. J. Weissburg. Chemosensory guidance cues in a turbulent chemical odor plume. *Limnology and Oceanography*, 46(5):1034–1047, 2001.

[106] M. J. Weissburg. From odor trails to vortex streets: Chemo and mechanosensory orientation in turbulent and laminar flows. In M. Lehrer, editor, *Orientation and Communication in Arthropods*, pages 215–246. Birkhauser, Basel, 1997.

[107] L. L. Whitcomb, A. A. Rizzi, and D. E. Koditschek. Comparative experiments with a new adaptive controller for robot arms. *IEEE Transactions on Robotics and Automation*, 9(1):59–70, February 1993.

[108] A.F.T. Winfield and O.E. Holland. The application of wireless local area network technology to the control of mobile robots. *Microprocessors and Microsystems*, 23:597–607, 2000.

[109] D. Wolpert, K. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. *Europhysics Letters*, 49(6), March 2000.

[110] D. Wright. Bold new vision for robotics: Improved robotics vision systems bring us one step closer to next generation manufacturing. *Advanced Manufacturing*, March 2001.

[111] S. Yakowitz, P. L'Ecuyer, and F. Vazquez-Abad. Global stochastic optimization with low-dispersion point sets. *Operations Research*, 48(6):939–950, November-December 2000.

[112] H. Yamaguchi and G. Beni. Distributed autonomous formation control of mobile robot groups by swarm-based pattern generation. In *Proc. of the Second Int. Symp. on Distributed Autonomous Robotic Systems DARS-96*, pages 141–155. Springer Verlag, 1996.