Developing Grounded Communication in Multi-Agent Systems

A thesis presented
by

Nathan Wiebe

to
The Department of Computer Science
in partial fulfillment of the requirements
for the degree of
Master of Science
in the subject of

Computer Science

The University of Manitoba
Winnipeg, Manitoba
January 2006

Thesis advisor Author

John Anderson **Nathan Wiebe**

Developing Grounded Communication in Multi-Agent Systems

Abstract

For a mobile robot to be able to communicate usefully with others, the symbols it uses to communicate must be associated with (*grounded* to) physical entities in the environment. While it is common practice to hand-construct such groundings, this does not scale to large problems. In particular, when communicating about useful locations in the environment, there are a large number of potential groundings, even for a relatively simple task such as navigation. The research goal of this thesis was to design, implement, and evaluate an approach that allows a group of robotic agents to develop consistent shared groundings for locations in an environment over time. The approach was implemented in a multi-agent robot simulator and experiments were run in domains of varying size and complexity, and with different robot populations. A number of parameters involved in developing shared groundings were also varied. The results of these experiments illustrate that not only can such shared groundings be developed over time, but that these groundings will improve the effectiveness of communication and ultimately the performance of tasks that require communication.

Contents

List of Figures

Acknowledgments

Thank you to my family, for all your prayers and support.

I would also like to thank Lynx Graphics Ltd., and in particular the President, Dr. Howard Ferch for his generous support while I pursued a Masters Degree. Thank you Richard Bobbie of Lynx Graphics Ltd. for his helpful reviews of early versions of this document.

Thanks also to my advisor, Dr. John Anderson. He has provided me with effective guidance and taken the time to contribute detailed feedback on my writing. He's also not scared to let me know when he knows I can do better.

For Kristin, thank you for your love.

I can't imagine completing this project without your support.

Chapter 1

Introduction

When I first discovered the moon, he said, I gave it a different name.

But everyone kept calling it the moon. The real name never caught on.

– Brian Andreas, *Mostly True*

The benefits of multi-agent systems over single agent systems are well established. Multi-agent systems are often more robust, more reliable, and solve problems more efficiently [Stone and Veloso, 2000; Balch, 1999; Weiss, 1999]. The distributed nature of multi-agent systems allows for multiple perspectives on a problem, which can be combined to form a more accurate, global point of view [Matarić, 1998]. Multi-agent systems can also be heterogeneous, allowing simpler agents with different skills to combine their abilities in situations where no one agent in a population could be expected to possess all necessary skills [Stone and Veloso, 2000].

In order to realize these benefits, agents are often required to communicate to coordinate their actions and share information. While the benefits of communication to everyday human cooperative activities are easily observed, communication has also been empirically shown to be of great benefit when solving problems in multi-agent systems. In multi-agent reinforcement learning, for example, a team receives reinforcement for some event (e.g. scoring a goal in soccer) and individuals are expected to adjust their behaviour as a result. In this situation, there is a problem assigning credit (reinforcement) to individual team members, since not all individuals would have been performing actions that directly led to the reinforcement. Matarić [1998] showed that communication was necessary to deal with such credit assignment problems.

Balch and Arkin [1994] show the advantages of employing communication in foraging tasks with multiple robots. They also illustrate that there is a point at which additional communication does not translate into increased performance, and can ultimately harm performance. That is, the effort devoted to increased communication must be measured against the benefits to performance that the increased communication provides. Beyond explicit communication, research has also been done on the utility of stigmergic communication - communication where information is imparted via indirect changes in the environment (pheromone markers, dropped objects, etc.) as opposed to explicit exchange of symbols. Wurr [2003] illustrates the efficacy of stigmergic communication in robotic navigation, while Sauter et al. [2002] shows its utility in path planning.

While inter-agent communication provides many benefits, it also introduces complexities. What form does communication take? How do agents decide what information is relevant to communicate, given the resource costs and returns associated with communication? While these questions are at the forefront of most work in communication in multi-agent systems, there are still important problems at a lower level that remain unsolved. Whenever communication is introduced, for example, we also introduce the requirement that agents share a common *grounding* for the symbols that they will use to communicate. A grounding is a connection between a concept or symbol about which the agent must reason internally, and some entity in the world. Creating and maintaining these groundings is known as the *symbol grounding problem* [Harnad, 1990]. Traditional system designers hand-construct explicit and precise groundings for everything that an agent could possibly reason about ahead of time. This provides a *shared grounding* between all agents in the system. While it is certainly possible to explicitly provide groundings for small environments, it becomes unwieldy very quickly as the complexity of the environment increases, and simply will not scale to environments of any reasonable size [Jung and Zelinsky, 2000].

The problem of scale in providing groundings becomes much more significant when agents are mobile. In mobile robotics, agents must typically communicate spatial information, and thus have potentially a very large number of grounded physical locations that they might wish to communicate to others. This is true even in environments that are sparse in terms of the number of objects. In order to ground these locations and allow agents to share information that includes location references, the system designers must provide a shared coordinate system for all agents - essentially a methodology for providing a shared grounding for every space in the environment at some level of granularity. Even when agents share a coordinate system, there is the additional requirement that each agent be able to find its own location, or *localize* itself within that coordinate system. If an agent receives information about the location of a goal, for example, it must know its current location in order to be able to navigate to the location of the goal.

To see the magnitude of this problem in a real-world environment, consider robotic rescue. This domain involves deploying a team of agents into an unknown and dangerous environment, to explore and map the area and search for humans who may be trapped. In order to explore this environment effectively, some communication needs to be employed. Without it, agents do not have the ability to share partial maps with one another, nor give advice to minimize redundant explorations. To make such communications meaningful, agents must have common points of reference, be they unique physical landmarks (e.g. a numbered door), or agreed-upon locations in some coordinate system.

The Global Positioning System (GPS) may be used to provide a shared coordinate system. Using this system to define locations, agents could easily overlay mapped portions of the environment with each other, and spatial information about locations of interest could be easily exchanged. However, in the robotic rescue domain, interference from fallen debris will likely prevent an agent from ascertaining its location. There are many other places where GPS signals are not available: underground, underwater, distant planets, and inside many buildings. A group of agents that relies solely on GPS, or a similar external coordinate system, will only be able to function within the bounds of that external system. If agents are additionally able to create their own set of shared locations, and rely on these for spatial communication, they are able to function anywhere. This is analogous to the way humans function. When placed in a new environment, a group of people need not find their location in terms of an external coordinate system. They are able to identify landmarks, like *traffic lights* or *the table at the front of the room*. People are then able to define new locations in terms of these landmarks, for example: *one block past the lights* or *the chair beside the table*. System that are able to define their own landmarks are more flexible that those that cannot, although they still may make use of external coordinate system when available.

Beyond the unrealistic assumption of agents implicitly sharing data, there are questions about the basic methodology of developing complex systems through hand-constructed groundings and categorizations made by the human developers [Jung and Zelinsky, 2000]. The problem with hand-constructing groundings is that *anthropocentric* (human centred) categorizations will limit the system in ways that would not occur if those categorizations were made by the system itself [Jung and Zelinsky, 2000]. For example, behaviour-based robots consist of a collection of simple perception-action packages known as behaviours, which operate in parallel and whose output is arbitrated to produce an overall response to the changing world around the robot [Arkin, 1998]. Designers of behaviour-based systems place linguistic labels on the robot's internal behaviours, such as *follow-wall* or *random-wander*. These have meaning to the system designer and affect the way the designer thinks about the system. However, the agent performing the behaviours carries none of the label semantics used by the designer: internally the complex interactions between a collection of such behaviours may be operating in a very different way than might be assumed given their labels. This leads to misunderstandings and errors on the part of developers who use these labels to understand the system. A rift forms between what the designer intended and what the system actually does [Jung and Zelinsky, 2000]. This rift is small in systems with only a few behaviours but as functionality increases and more behaviours are added to the system the rift becomes larger. Eventually these systems will collapse as the rift between how the system is understood to operate, and how the system *actually* operates becomes too large [Jung and Zelinsky, 2000].

The same problem with anthropocentric categorizations exists with symbol groundings. Choosing the particular entities worth grounding seems logical and obvious when designing a small system, but as systems become larger and more complex, not only does selecting all groundings become impossible, but the designer-system rift comes into play. The groundings chosen by the designer are based on a human viewpoint of how a complex system should be built (where that viewpoint is based on concrete knowledge of only the isolated elements of that system) and how the agents are *expected* to perform. Were agents to evolve their own set of groundings, those might be very different from the groundings envisioned by the human designers, and would arguably be more effective as well (since unneeded groundings that designers might supply would not develop, and groundings would be tailored to the activity at hand [Jung and Zelinsky, 2000]). Beyond the issue of anthropocentric categorizations, future intelligent systems must be able to acquire groundings on their own: there will simply be too many groundings to be able to hand construct all of them.

This idea has led a number of contemporary researchers to pose solutions to the problem of anthropocentric categorizations by providing alternative means for a system to acquire groundings to the outside world [Billard and Dautenhahn, 1999; Jung and Zelinsky, 2000; Steels, 1997]. In general, these are methods which require agents to learn or discover groundings for the things (concepts and objects) about which they must reason and communicate. The difficulty inherent in requiring multiple agents to learn groundings is that in isolation, different agents will naturally learn different groundings. Obviously this is not suitable for the purposes of communication: if two agents have different groundings for the same symbol, that symbol is not useful. What is needed is a mechanism to develop and reconcile *shared* groundings.

The goal of this thesis is to motivate, design, implement, and evaluate a multi-agent system which learns shared groundings for locations in an environment without sharing a coordinate system. This is done without having any prior knowledge of the environment, or any prior shared points of reference. Previous research into this problem [Jung and Zelinsky, 2000; Billard and Dautenhahn, 1999] contains assumptions which limit the applicability of these systems.

In the remainder of this chapter I will outline the motivation for this thesis, present some definitions, and a general outline of the methods used in the research presented here. Following that I will present the research questions addressed by this thesis, provide a brief summary, and outline the rest of this document.

1.1 Motivation

This thesis is primarily motivated by the desire to remove assumptions present in previous work. As outlined above, it is not feasible for system designers to hand construct groundings for all entities in a large system. Further, system designers will provide anthropocentric groundings, which limit the system in artificial ways [Jung and Zelinsky, 2000; Brooks, 1990]. The groundings provided by a human designer will be based on human senses and experience. The sensory abilities of the robot will necessarily be different from that of the human designer. There is no way for the human designer to provide a perfect description of objects in the environment in the language of robot perceptions. The solution to these problems is to allow agents to learn about their environment as they explore it and ultimately form their own groundings. However, if multiple agents learn independently of one another they will undoubtedly develop different groundings. How then can these agents communicate if their groundings are different? What is needed is a system that allows agents to learn and form groundings independently and still communicate in a meaningful way. There are few examples of systems that try to solve this problem, and those that do contain significant restrictions. These restrictions include having been demonstrated with only two agents, flooding the environment with *far* more points than is necessary for communicating spatial information, and requiring an explicit shared label creation phase [Jung and Zelinsky, 2000] (taking time away from performing work in the domain). Other work in this area has taught agents a predefined language for labelling the environment from either a human or robot teacher, rather than allowing agents to evolve groundings on their own [Billard and Dautenhahn, 1999].

This thesis presents a system that requires no explicit location labelling phase, does not flood the environment with groundings, is demonstrated to be functional with up to 16 agents, and has an initially undefined set of grounded locations. This system is evaluated in a domain where communication about locations in the environment can improve performance, and is shown to provide significant benefits, indicating that agents using this approach can reap the benefits of grounded communication without predefined groundings.

While providing fewer groundings directly as part of the system limits anthropocentric categorizations, it can also increase the burden on the system itself. Human experts can provide valuable insight into problems based on their experience, that could be used as a basis for the agents learning efforts. Without the use of human provided groundings, agents are forced to acquire knowledge that could otherwise be innate. It takes time to acquire this knowledge: time that could otherwise be spent on performing the task for which the agent was designed. It may take a long time for an agent to discover the solution to a particular problem, when knowledge of that solution could have been provided directly to the agent. However, agents that are allowed to learn their own groundings may find more effective solutions than that provided by the system designer. There are also problems that current agents simply cannot solve, as there is no known method for artificial agents to acquire this knowledge in a symbolic way. In these cases, knowledge about solutions to the problem must be provided to the agent.

Agents that are expected to learn are also likely to be more complicated than agents that are built with specific domain knowledge for a specific task, at least in small domains. In large domains, it may be simpler to provide the agent with a learning mechanism, than to define all elements of the domain. There is also a trade off between the amount of learning that an agent is capable of, and that agent's generality. The more specific the circumstances under which the agent is expected the function, the less learning is required. However, if an agent is unable to learn, it will not be able to adapt to its changing circumstances.

1.2 Terminology

Having outlined my motivations above, and before describing the system in more detail, I must first ensure a common understanding by defining some terms used throughout the remainder of this thesis.

An *environment* is a contained universe in which agents reside. Environments also contain goals and obstacles and allow agents to communicate with each other. Environments may exist in the real world, or be simulated in software.

Though many different authors have defined the term *agent* [Maes, 1995; Hayes-Roth, 1995], the most common definition is that given by Russell: “An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.” [Russell and Norvig, 1995]. Beyond Russell’s definition, for the purposes of this thesis, agents are able to move around in the environment, and reason about their own location in their own coordinate system. An agent can detect goals, obstacles, and other agents relative to itself through its sensors. *Agents* can also communicate with other agents by broadcasting explicit messages through the environment. Agents may be simulated in software, or implemented on physical robots.

A *multi-agent system* is an environment in which multiple agents exist. Agents may cooperate with each other to achieve their goal, compete against each other for limited resources, or may not even be aware of the existence of other agents. The agents in the research presented here are cooperative and actively work together.

A *symbol* is an agent's internal representation of some entity or concept in the environment. Many symbols can be combined to form and reason about complex world models, as described by the *physical system symbols hypothesis* [Newell and Simon, 1976]. The symbols used in my research are used only to designate locations in the environment. My research employs two different kinds of symbols. The first is used in a simple association between an integer and a coordinate in the environment. The second form is a much more complicated *symbolic reference* (see below) that designates an arbitrary location in the environment based on two other known locations.

A *grounding* is a connection between a symbol within an agent and some entity within the real world. For the purposes of this thesis, a grounding is a connection between a location in the environment, and an integer label for that location.

A *shared grounding*, also called a *common grounding*, exists between two agents when the same symbol is connected to the same real world entity. For the purposes of the experiments presented in this thesis, groundings are shared between two (or more) agents when they use the same symbol to designate the same location in the environment.

Grounded Communication is communication between agents using symbols where the agents share a common grounding for those symbols. Grounded communication may be extremely simple: it does not imply a complex grammar or syntax. Anytime a symbol is exchanged between agents, where that symbol is grounded to the same concept in each agent, the communication is grounded. For example, if one agent sends the message “foo” to another, and the agents agree on the meaning of “foo”, the communication is grounded. The symbols exchanged in my research are integers which designate locations in the environment. If the same integer, say 42, refers to the same location in the environment in two different agents, then communication between those agents using the symbol 42 is grounded. Note that each agent is free to use any mechanism they wish to ground the symbol 42. Grounded communication does *not* imply the grounding within each agent is represented in the same way.

An *iconic reference* [Deacon, 1997] is a association based on features an agent is capable of observing. An iconic reference is grounded to the raw sensor values that allow the agent to identify that symbol. For example, the grounding for a location mentioned above is iconic in each agent because it is represented using only the coordinate values reported by the robot’s motion hardware.

A correlation or association between icons is an *indexical reference*. For example, since smoke occurs with fire, smoke is an index for fire. Sirens are also an index for danger, since the two often occur together. The word ‘seat’ is an index for the physical objects that people sit on. When an agent creates a grounding for a location, it associates a number with the coordinates of that location. This number is an index for the coordinates.

The most powerful type of reference is a *symbolic reference* [Deacon, 1997]. Symbolic references represent arbitrary relationships between icons, indices, and other symbols. Learning symbolic references is difficult: the learner must be shown a series of relationships and grasp a high level pattern from those relationships [Jung and Zelinsky, 2000]. For example, the concept of *higher* may be demonstrated with differences of elevation, social status, and performance metrics. There are no known techniques for artificial systems to learn symbolic references [Jung and Zelinsky, 2000].

Symbolic communication has been defined multiple times by different researches. Jung & Zelinsky [Jung and Zelinsky, 2000], whose work is most closely related to that of this thesis, define symbolic communication as communication using a symbolic reference. Others, such as Stone & Veloso [Stone and Veloso, 2000] do not provide a precise definition, but use the term to refer to any communication which involves symbols (not just symbolic references). This thesis will adopt the usage of Stone & Veloso as it does not introduce confusion.

With several important terms related to this thesis now defined, I will now provide an outline of the methods I used to accomplish my goals.

1.3 Method

The methodology used in my research to develop grounded communication is described in detail in Chapters 3 and 4. Here I provide a brief outline in order to provide a context for the rest of this chapter and the literature review that follows in Chapter 2.

An approach to developing grounded communication must provide three important things: conditions under which groundings are to be made; methodologies for sharing and reconciling groundings; and some representation for the groundings themselves.

In my approach, all groundings will take the form of a random integer used as a label. A robotic agent will associate a particular location in the environment (within its *own* coordinate scheme, which may be very different from other agents in the system) with one of these labels. Associated with each label will be a reference count which will serve as a heuristic indicator of how much this label has been spread through the population. Labels and reference counts can be communicated to other individuals in the environment, while coordinates for labels will not be communicated (and since coordinate schemes are not identical between agents, communicating coordinates would be meaningless).

Having given the agents the ability to create labels, a methodology must be employed for creating individually grounded locations. I will be comparing three different approaches for allowing an agent to decide when a location is worth grounding. The intent of all of these is that agents construct a grounding for their location at *useful* times, rather than at contrived times as in previous approaches such as that of Jung and Zelinsky [Jung and Zelinsky, 2000].

The simplest of the three alternatives will be to provide the ability, upon two agents meeting in the environment, to have both the agents involved make a shared location grounding for that point. The rationale behind this alternative is that agents will encounter one another more often where they spend the most time, which will presumably be in areas of the environment where they are doing the most useful work or are having the most trouble navigating. Both of these situations are likely sources of useful location labels. A second approach will be to attempt to identify specific areas of the environment where changes in terrain are occurring (e.g. open areas to obstacles, doorways to walls, etc.) in a domain-independent fashion, and ground these locations individually. This approach will use an information-theoretic measure, based on the concept of spatial entropy defined in [Baltes and Anderson, 2003], in order to do this.

Both of these are domain-independent schemes for grounding locations in the world without resorting to human-centred categorizations. I will compare these alternatives to a third domain-dependent grounding methodology that provides the agents with the opportunity to create groundings only at specifically marked places in the environment, such as at corners and along sections of open wall. As in the previous approach, agents will share a nearby grounding when they happen to meet.

An approach to grounded communication must also have conditions under which groundings should be shared between agents and a method for accomplishing this. My approach will have no explicit training phase - shared groundings are expected to develop as a consequence of performing domain-specific activities. In the simplest grounding approach above, some sharing is already built into the initial grounding (since it is created by two agents at the same time). I will be using the the opportunity provided by agents encountering one another to allow for demonstrations of existing nearby grounded locations (since such encounters provide a shared spatial context). In the case of conflicts (e.g. the demonstrated location is already grounded to another name by the agent receiving the demonstration), methods are provided to ensure the propagation of the most globally consistent knowledge.

The distance at which one agent can demonstrate a grounding to another, and the distance at which an existing grounding can be considered the same as a new one are important parameters to these methodologies.

To implement and evaluate this approach, I employ USC's Player/Stage [Gerkey et al., 2001] simulation software. Player/Stage is generally accepted across the research community and has been verified as accurately simulating the physical behaviour of Pioneer robots. While the main reason for using this simulation package is control over experimentation, I also do not have access to enough Pioneer robots to adequately perform this work on physical robots. The performance of the system is evaluated as the improvement in the average time it takes the agents to navigate a physical environment and arrive at a randomly placed goal location using grounded communication developed over time, as opposed to using no communication. When an agent discovers the goal, it broadcasts the grounded location it knows of nearest the goal. Others sharing this grounding can then navigate to the goal more quickly (possibly to the point of being able to plan a path directly to it). As agents develop more consistent groundings, they should be more and more successful operating in the same environment. The evaluation described in Chapter 5 examines my approach to grounded communication under a number of different variations: differences in grounding strategy, different numbers of robots and varying domain sizes, as well as variations in a number of parameters that are important to my approach.

1.4 Research Questions

I will use the methodology outlined above to answer the following research questions:

1. Can agents develop consistent shared groundings for locations in an environment, despite not sharing a coordinate system, in order that agent performance

can be improved in a domain that benefits from communication about locations?

2. Does the manner in which agents create individual groundings impact the performance of the approach for sharing groundings?

While the two research questions above are central to this thesis, the approach I have developed to answer these questions allows a number of factors to be changed, leading to the following secondary research questions:

1. Does the distance at which an agent can begin a conversation with another agent affect the performance of the approach?
2. Does the maximum distance two locations can be apart, and still be considered equal (ϵ), affect the performance of the approach?
3. How does using a symbolic reference, instead of an indexical reference, to designate the goal's location affect performance?

1.5 Summary

In this section I have introduced the concepts of groundings, shared groundings, and symbolic communication in multi-agent systems and shown the need for agents in multi-agent systems to be able to learn groundings on their own, and later reconcile them with their peers. I provided a high level description of the methodology employed in my experiments to evaluate my approach to learning shared groundings for locations in an environment. I have also posed research questions which this thesis will answer. The remainder of this section will provide a brief outline off the subsequent sections of this thesis.

1.6 Thesis organization

The remainder of this document is organized as follows:

Chapter 2: Related Literature

Chapter 2 reviews the academic literature related to the work in this thesis.

Chapter 3: Developing Grounded Communication

Chapter 3 discusses my approach to developing grounded communication.

Chapter 4: Implementation

Chapter 4 provides implementation details for my approach presented in Chapter 3.

Chapter 5: Evaluation

Chapter 5 evaluates the results of the experiments, along with analysis.

Chapter 6: Conclusion and Future Work

Chapter 6 provides answers to the research questions, and additional discussion.

Chapter 2

Related Literature

This chapter will equip the reader with the background necessary to understand the academic context of this thesis. It will first outline the history of the use of symbols within agents. Following that, a brief overview of the progression of agent control architectures is presented, as well as some basic background information on mapping and path planning. Having equipped the reader with the necessary background, the remainder of the chapter then is devoted to a review of selected specific works that are most closely related to the research presented in this thesis.

2.1 Agents and Symbols

Symbols have been central to cognitive science since the 1950's, when they emerged from information processing psychology [Sun, 2000]. The modern basis for symbol systems is the *physical symbol system hypothesis*, proposed by Newell and Simon in 1976 [Newell and Simon, 1976]. This hypothesis states that symbol manipulation is both necessary and sufficient for generally intelligent behaviour. While this hypothesis has been extremely pervasive in artificial intelligence, there are those that argue philosophically that the hypothesis is too strong (most notably Searle [1980]), as well as those that have actually built systems that demonstrate intelligent behaviour (albeit not generally intelligent) without using symbols. Examples of such systems include neural networks, as well as non-neural approaches such as those of Brooks [1991b, 1990]. Despite these exceptions, symbol manipulation is still an extremely important part of many areas of artificial intelligence - logical reasoning revolves around associations between symbols, as does language processing and communication. To reconcile these differences, a temporal boundary is generally accepted between unconscious/subsymbolic reasoning and conscious/symbolic reasoning; an activity falls into the former category or the latter depending on whether a response is generated from stimuli before or after the 100ms level [Wright, 1990].

Much practical reasoning (any application situated in a physical environment, for example, such as a mobile robot) employs symbols that are representative of entities and concepts in the domain in which the system is situated. When a mobile robot reasons about avoiding obstacles, for example, it can create symbols to designate the obstacles it knows about, and then reason about them. These symbols are placeholders for physical objects, as opposed to purely hypothetical structures. Such groundings for perceptually-identifiable physical objects are also known as *anchors* [Saffiotti, 1994]. The problem of developing and maintaining anchors for mobile agents through perception (the *anchoring problem*, a subset of the broader symbol grounding problem) was formally defined in 1994 by Saffiotti. Coradeschi and Saffiotti presented a preliminary formal logic solution to the anchoring problem in 2000 [Coradeschi and Saffiotti, 2000], defining the major components in predicate logic, as well as initial methodologies for using perception to create new anchors and track them as the orientation of the robot changed. This work was later extended specially to address using symbols in actions and plans [Coradeschi and Saffiotti, 2001, 2003].

While this work provides a formal framework for maintaining anchors to symbols via perception in single agent systems, it does not yet deal with the multitude of practical issues that arrive when the shared groundings necessary for communication in multi-agent systems are considered. This problem is an order of magnitude more difficult than the single-agent symbol-grounding problem, as it involves reconciling inconsistencies between the groundings of various agents. Since an agent can arbitrarily decide to ground whatever symbols it believes might be useful, the differences in perceived utility between many agents greatly increase the potential number of symbols that could exist across the system. Moreover, the bounded resources at each agent's disposal preclude simply sharing each and every grounding between all agents - communication to agree on symbol grounding in any reasonably complex domain would be combinatorially more significant than the amount of communication necessary for problem-solving itself. While this is true in any rich domain, it is especially true in the context of the research presented in this thesis, where the groundings are useful physical locations in an environment. From a multi-agent systems standpoint, the goal in developing shared groundings is to ground what is *most useful* to the group given the domain, and to share groundings with a *minimum of communication*. This is an attempt to parallel human use and sharing of symbols. Humans need not ground everything they come into contact with, only that which is useful. Further, when humans wish to impart information to each other they do not (hopefully) tell each other about all symbols which they know. They selectively share only symbols which are useful given the current situation.

Comparatively little work has been done in grounding in multi-agent systems compared to single-agent environments. Steels has done work in multi-agent systems examining how individual agents may be able to generate discrimination trees to distinguish one object from other in the environment, without the aid of a teacher [Steels, 1997]. Using these trees, agents engage each other in a series of adaptive language games. Each game consists of a speaker and a hearer. At the beginning of the game, both the speaker and hearer rotate 360 degrees, which provides a shared context. The speaker then identifies a topic by moving toward it and pointing to it with four infrared beams. Each agent then constructs a discrimination tree for the topic. If a set of discriminating features can be found, the speaker will translate it into words, which the hearer will interpret. If no feature discrimination is possible, the discrimination tree is refined, and the game ends in failure. If the speaker does not yet have a word to express the feature set, there is a 5% chance a new word is created. If the hearer does not understand the word, it can add the word to its lexicon and create a hypothesis about the meaning of the word based on the discriminations it was able to make from the topic. If both the speaker and hearer have a word for the feature set, and agree on which word should be used, the language game ends in success. If both agents have a word, but disagree on which word should be used, the language game ends in failure. Experiments found that discrimination success improved over time, which lead to an increase in language game success. Vogt [2001] later improved on the work of Steels.

Steels' work is part of the emerging field of evolutionary linguistics [Steels, 2003], which considers the origin of language and meaning. While it occurs in a very different environment from that of this thesis, it does show that it is possible to allow agents to share information with one another in coherent ways in order to develop shared symbol associations. Steels' work differs from my research in that there is a limited number of objects in Steels' environment, and that the entire domain exists only to perform symbol associations. I intend to have symbol groundings be acquired over the course of activity, not as the sole goal of the activity.

2.2 Agent Control

The purpose of all agents is to interact with their environment in some way. Agents must produce actions, but how are those actions generated? Architectures for agent control have changed significantly from early designs to present day techniques. This section will provide an overview of the development of agent control paradigms.

Agent control architectures are generally grouped into one of three categories: planning (or classic), reactive, or hybrid. Classic agent control techniques relied heavily on the physical symbol hypothesis [Newell and Simon, 1976] outlined above. Intelligent systems were approached from the standpoint of symbol manipulation. Agents contained an explicit world model, in a symbolically rich environment, which allowed detailed plans for the future to be constructed [Brooks, 1991a]. These agents were said to be *deliberative*. The processing in a classic systems could usually be subdivided into three steps [Arkin, 1998]. First, robots would sense their environment, often through some kind of vision system, and update the symbolic world model. Next a planner would process the sensed data and perform symbolic reasoning to form a plan of action in order to solve the problem for which it was designed. Then the plan would be executed by the robot. Running the above cycle frequently was intended to produce the desired robot behaviour. Examples of planning agents can be found in Crowley [1985] and Moravec [1990].

While classical systems could form plans for the future, there were some limitations which curtailed the real world success of this model. These limitations would lead researchers to suggest new approaches. One problem was the length of time it took to execute the sense-plan-act cycle. Heuristics were sometimes employed to speed up the search, but this often sacrificed accuracy [Chapman, 1989]. Updating the world model was also a time consuming task. The sense, plan, act cycle could take minutes to complete [Brooks, 1991a]: this meant that a planning agent was often creating plans based on old data [Matarić, 1997]. Besides using old data, the noisy nature of sensor readings made maintaining an elaborate world model accurately extremely difficult [Brooks, 1990; Agre and Chapman, 1991; Chapman, 1989]. As agents spent more time in the environment, their internal world model would drift further and further out of sync with the real world. The symbols inside the planning component were not properly *grounded* in the robot's perceptions. Since the symbols used in plans were out of sync with the real world, execution of these plans would often fail. Failed plan execution requires a new plan be created, further hampering the robot's real time performance.

In response to the limitations of planning techniques, *reactive* agent control was proposed [Brooks, 1990]. Reactive agents are radically different from planning agents, and able to achieve good real time performance by avoiding an elaborate symbolic world model and the search that goes along with it [Matarić, 1997; Brooks, 1990]. Instead, reactive agents employ a much simpler stimulus response mechanism with minimal, if any, state information. In opposition to planning systems, reactive systems operated under the directive "the world is its own best model" [Brooks, 1990]. *Purely* reactive agents are significantly restricted in their design: for example, they contain no world model at all [Arkin, 1998]. The *Physical Grounding Hypothesis* forms the basis for reactive systems. This hypothesis states that "to build a system that is intelligent it is necessary to have its representations grounded in the physical world." [Brooks, 1990]. An agent built using the physical grounding hypothesis is viewed as being connected to the world via its sensors and actuators. Agents of this type are said to be *situated* and *embodied* [Brooks, 1991a]. Being situated refers to the agent's actions in the real world: the agent does not operate on abstract entities, but rather real physical objects. Embodiment emphasizes the agent's physical presence in the world, and the consequences of that presence.

Many reactive systems contain the notion of a *behaviour*. A behaviour is "a stimulus/response pair for a given environmental setting that is modulated by attention and determined by intention" [Arkin, 1998]. A robot's attention causes certain behaviours to be active at the correct time, depending on the environmental context. Intention regulates the perceptual resources of the agent, which in turn influences which behaviours are active. The system is constructed from behaviour modules, where each module is a relatively simple embodiment of a sensor/actuator connection. The combination of several of these modules produces the overall *emergent* behaviour of the system [Brooks, 1990; Matarić, 1997; Arkin, 1998].

2.2.1 Subsumption

One of the best known reactive approaches is the *subsumption architecture*, [Brooks, 1985, 1989]. The subsumption architecture uses *task-achieving behaviours* [Brooks, 1989] to react to the environment. Systems are built in a bottom up manner. The problem to be solved is split into layers, where each layer represents a level of competence. A complete agent is then constructed by layering task-achieving behaviours on top of one another. Higher level components can monitor and influence lower level components, but lower level components have no knowledge of the layers above them. This makes it easy to extend an already working agent by adding more complex behaviours on top of already functioning lower level behaviours, since lower level functions are left unchanged. Lower competency levels are usually dedicated to such tasks as obstacle avoidance and keeping the agent safe. Higher levels of behaviour direct the agent to perform some useful task in the domain. Examples of agents built using the subsumption architecture include [Horswill, 1993; Matarić, 1992; Brooks and Flynn, 1994].

The biggest advantage of reactive control over planning agents is the speed at which it allows agents to respond to the environment. The lack of world model precludes the time consuming update and search cycle of classic agents. Reactive agents can deal more robustly with noisy sensor readings than classic systems, as there is very little world model in which error can accumulate. Avoiding a symbolic world model also avoids the problem of ungrounded symbols within the agent. The impoverished representational abilities of reactive agents are also limiting. It is hard to build complex behaviours when there is no world model about which to reason. Even simple tasks can require complicated interactions between layers of the system. This leads to questions about the scalability of reactive approaches [Brooks, 1990]. *Local minima* are also a problem for reactive agents, a situation where the actions executed by an agent lead back to the same state in which it started, the agent has become stuck. This problem is not experienced by planning agents with a more complete world model.

2.2.2 Schema-based Approaches

Schema-based agents were proposed soon after the subsumption architecture. They retain the fast acting properties of their reactive forerunners, while loosening representational constraints. These agents are based on schema theory [Arbib, 1981, 1992], where each schema encodes mapping from a set of precepts to an action. Each behaviour contains both a *perceptual* schema and a *motor* schema [Arkin, 1998]. Perceptual schemas define the conditions under which the behaviour produces an action, whereas motor schema define what that action is. Schema-based control does away with the strict behaviour hierarchy of the subsumption architecture. Behaviours in these systems operate in parallel, each producing an action vector [Arkin, 1998]. These individual action vectors are summed to produce a final action to be executed by the robot. Actions produced from some schemas may have higher priority than others, causing them to contribute more to the robot's final action. Schema-based agents stress simple and efficient world models and maintain that reactivity should be used before planning. Schema-based agents can avoid local minima by adding some noise to sensor readings, which causes the robot to (potentially) make different choices when presented with the same input. This eventually allows the robot to break out of the loop. Another local minima avoidance technique is to add a behaviour which directs the robot away from recently visited locations [Arkin, 1998]. Examples of agents built with the schema architectures can be found in [Arkin et al., 1993; Cameron et al., 1993; Balch and Arkin, 1995]. While schema-based agents allowed for more flexible behaviour configurations than reactive agents, they contained a minimal world model, and still lacked the ability to reason about the future.

2.2.3 Hybrid Approaches

Hybrid architectures are a combination of reactive control and the symbolic reasoning of planning agents, in an attempt to realize the benefits of both. These systems are often built upon a reactive base which handles low level navigation, such as obstacle avoidance. A conventional planner sits on top of this base, giving the agent the ability to form plans and reason about the future. Since low level navigation is handled by the reactive component, the planning portion of the agent can engage in lengthy processing without the agent suffering any ill effects.

Not all system designers agree on how to combine reactive control and planning. There are two high level approaches for *layering* planning and reactivity, *horizontal* and *vertical* [Weiss, 1999]. Horizontal layering gives each layer of the system direct access to sensory data, as if many individual agents were implemented along side each other within a single robot. This has the advantage of conceptual simplicity: if an agent needs to produce many different behaviours, then each of those behaviours can by implemented in a separate horizontal layer. However, if too many horizontal layers are present, it becomes difficult to mediate between them and maintain coherent agent behaviour [Weiss, 1999]. An example of a hybrid system with a horizontal layering can be found in [Ferguson, 1992].

The alternative to horizontal layering is vertical layering. Vertical layerings do not allow all layers to access sensors directly, information flows from one layer to the next. Within vertical layerings there are *one pass* and *two pass* architectures. One pass architectures have control flowing from layer to layer, where the final layer generates the behaviour for the agent. Two pass architectures have control flowing up through each layer, then back down to generate the final action. Vertical architectures are not as susceptible to mediation difficulties as horizontal model, at the cost of decreased flexibility [Weiss, 1999]. An example of a vertically layered agent can be found in [Müller, 1994]. Since both types of layered solutions reason using symbols, they are susceptible to the same potential problems with ungrounded symbols as classic planning agents.

2.2.4 Mapping & Path Planning

Mapping and path planning are not central topics of this research, but both are necessary for most any mobile robotic task. They will ultimately be necessary for my evaluation as well, since agents will have to plan paths to demonstrate groundings, as well as to get to the goal location when it becomes known, and maps are necessary in order to support this path planning.

Mapping is the process of generating an internal representation of the environment based on sensor data in such a fashion that it can be used to traverse the environment or give directions to others. Path planning is the process of finding a way to get from an initial location to a goal location in an environment without colliding with any obstacles. Another problem that goes hand in hand with mapping and path planning is localization. Localization is the process of an agent finding its own location in the real world, and translating that location into the internal representation of the world. Since mapping and localization are so closely related, there have been efforts to integrate the two [Thrun et al., 1998]. If agents are not able to localize effectively, path planning is useless as they do not know their start position. Maps provide agents with knowledge about parts of the environment which they can not currently sense, which is essential for path planning, and can reduce the frequency of sensor sampling [Arkin, 1998].

Mapping is a difficult problem that has received much research attention over the past two decades [Thrun, 2001]. The difficulty in mapping derives from errors in sensors and actuators. Sensor data is noisy, making it difficult to create an accurate map. Current sensors also have limited range, requiring the robot to traverse much of the environment it wishes to map. Robots also have difficulty recording how much distance they have traversed. Wheel encoders sometimes slip, leading the robot to believe it has moved when it has not, introducing error. Such error is cumulative, meaning the more time the robot spends in this environment the more error accumulates [Thrun, 1998]. Agents relying on vision systems to measure traversed distances also have problems, seeing a repeated visual pattern may make the robot think it has moved when it has not.

There are currently two dominant mapping paradigms: grid-based [Elfes, 1987] (or metric) and topological [Thrun, 1998]. Grid based methods generally model the world by breaking it down into regular, fixed-sized cells at some resolution and tracking if those cells are free space or occupied by an obstacle. Localization in grid-based maps can by done by using position values from their odometry hardware (so long as it provides sufficient accuracy). Grid-based methods with high spatial resolution are hampered by large memory requirements, and a correspondingly large amount of time to processes all the data when planning a path.

Topological methods use a graph representation, where nodes in the graph are formed from landmarks of interest. Landmarks between which the robot is capable of navigating are connected by edges. An agent using a topological map generally localize themselves with respect to known, nearby landmarks. However, this can be difficult when landmarks look similar, especially if a landmark is being approached from a new direction, causing that landmark to look different to the agent. Further, it can be difficult for topological methods to recognize the spatial relationship between landmarks [Thrun, 1998]. Topological approaches have the advantage of using less memory than grid-based methods, making path planning a more efficient endeavour. The efficiencies in path planning can be lost as the agent attempts to follow the planned path. Since topological plans lack the precise detail of metric maps, agents must perform more processing to actually traverse the path. The symbolic nature of topological approaches also provides a more natural representation for symbolic reasoning (but care must be taken to ensure those symbols are properly grounded). Grid-based maps and topological maps need not be used in isolation, many authors have chose to combine grid-based and topological approaches [Jia et al.; Thrun, 1998]. For example, by generating topological maps on top of grid-based maps [Thrun and Bücken, 1996].

Although it is possible to use *a priori* maps (a map created by some other entity, and encoded for the agent's use) [Payton, 1991], this is a grossly anthropocentric categorization. This method sufferers from problems when translating the map for the agent to use [Arkin, 1998] for the same reasons that classical planning agents have problems, the symbols they use are not properly grounded.

While mapping may be considered an independent task, some representation of the world around the agent is generally required for path planning. This representation may or may not be similar to what we traditionally think of as a map. Based on the choice of representation, path planning approaches can be classified in one of three ways; skeletonization, decomposition, or local [Baltes and Anderson, 2003].

Of the skeletonization methods, visibility graphs [Thompson, 1977] are most commonly used [Baltes and Anderson, 2003]. In this approach, a graph is constructed from the agent's representation of the environment. Edges in the graph correspond to the boundaries of obstacles and open space, while vertices are constructed at the intersection of edges. Finding a path is then a matter of adding edges from the initial and goal locations to the agent's representation of nearby obstacles, ultimately connecting them and forming a path from the edges between the start and end points.

One of the drawbacks of using a visibility graph is that the paths produced tend to follow close to the boundaries of obstacles. Even if the initial locations for obstacles are acutely known, small odometry and localization error make it likely the agent will have trouble following the planned path. Examples of systems built using visibility graphs include [Arikan et al., 2001] and [Little and Esterline, 1999].

Decomposition path planning methods break down the agent's environmental representation into discrete chunks. The decomposition may be either *approximate* or *exact*. Approximate methods break down the environment into predefined shapes, without regard for the boundaries of obstacles and free space. Exact methods break down the environment along free space / obstacle boundaries, with the end result being the union of all free space.

Quadtrees [Andresen et al., 1985] are an approximate method, and the technique used for path planning by the agents in my research. A quadtree decomposition starts by dividing the environment into four quadrants. If a quadrant is completely open, that cell is marked as open. If a quadrant consists entirely of a blocked area, it is marked as blocked. If a quadrant contains a mixture of free space and blocked areas it is marked as mixed, and that cells is then recursively decomposed into four child cells. This recursive process repeats until the entire environment has been decomposed. As the tree is constructed, node adjacency lists between open cells are maintained. Planning a path is then a matter of finding the least cost path from the start location, through adjacent free space, to the goal location. *Framed* quadtrees [Chen et al., 1995] are a variation on quadtrees which add small cells around the perimeter of large cells, in order to plan better paths [Yahja et al., 1998].

While the quadtree approach breaks up the environment in a regular manner, flexible binary space partitioning [Baltes and Anderson, 2003] is a decomposition method that attempts to decompose the environment in a more intelligent way, by employing an entropy metric to find partitions between free space and blocked areas. This method was successful in significantly reducing the number of leaf nodes required to represent the environment compared to decomposing the environment using arbitrarily-sized partitions.

As the name suggests, local approaches to path planning make use of only local information, as opposed to the broader representations maintained by the other approaches outlined here. Potential fields are often used to implement this approach. In a potential field, goals exercise an attractive force on the agent, while obstacles repel the agent [Arkin, 1998]. A potential field function can then sum these attractive and repulsive forces to produce a suitable vector for the agent to follow. Any environment for which a potential field function can be defined can employ this approach. No global information is required: only entities which the agent can directly perceive exercise an effect on its path. Since this method employs no global knowledge, it is susceptible to the same local minima as reactive agent architectures described above.

The work presented in this thesis uses grid based mapping due to its ease of implementation, and ease of localization. The grid based map is also a natural fit for the quadtree-based path planning used in my research. Further, one of the techniques I use to decide when a location is worth grounding is motivated by the entropy-based technique of [Baltes and Anderson, 2003] described above. This technique requires a grid based representation of the environment.

2.3 Grounded Communication

The previous sections in this chapter have acquainted the reader with the basic issues in mobile robotics relevant to this thesis. The following two subsections will discuss the two previous research activities which relate closely to the research presented in this thesis.

2.3.1 Billard & Dautenhahn

Billard and Dautenhahn [1999] studied the benefits of social skills to learning in heterogeneous multi-agent systems. In their environment, agents learned shared groundings for word signal pairs and how to describe a location in polar coordinates relative to a fixed predefined point. They found that giving the agents a social behaviour, in this case a *follow* behaviour (causing one agent to want to follow another), sped up the transmission of a vocabulary through the system. Using a connectionist approach, agents learned to identify the colour of a patch of floor by matching a word broadcast by the teacher with the signal for the colour of the floor. In order to do this, a teacher robot would lead a group of learners around the environment, and signal the name of the colour and the polar coordinates for the patch of floor where it was positioned. Since the learner agents were in close proximity to the teacher (due to the follow behaviour), they had similar inputs to the teacher for floor colour and had travelled roughly the same distance as the teacher. This allowed the learners to associate the teacher-provided attributes *colour* and *position*, for a given patch of floor, with their own sensory inputs. The teacher agent in this approach did not learn its groundings or alter them over time, making the language static. When a learner agent became confident enough in its own groundings for a particular concept, it became a teacher. In a further experiment, agents wandered the environment freely, broadcasting both their current location and the colour of the patch of the floor at the location, either to all other agents or only to a single nearby agent. Billard and Dautenhahn found that using the follow behaviour increased the rate at which agents were able to learn the vocabulary, as well as the number of times the vocabulary was successfully learned. The focus of Billard and Dautenhahn's work is very different from this work, in that their main research goal was to illustrate learning from a non-human teacher. Rather than having a static language (all points are already known by the teacher, in addition to colour) that is being spread among agents, my work has an initially undefined language that is being learned in parallel by all agents, gradually coming to agreement over time. That is, in my approach all agents are both teachers and learners all of the time - no teacher with any human-provided groundings (let alone all potential groundings) will exist. Individual agents develop shared labels rather than simply learning them from a single agent with all the answers. While these groundings will certainly not be initially consistent there are mechanisms in place for agents to resolve these inconsistencies. A consistent language develops over time. This is much more flexible across different environments, where there may not be a teacher which has already mapped the environment.

2.3.2 Jung & Zelinsky

Jung and Zelinsky [Jung and Zelinsky, 2000] describe the implementation of a heterogeneous cooperative robotic cleaning task which benefits from the use of a

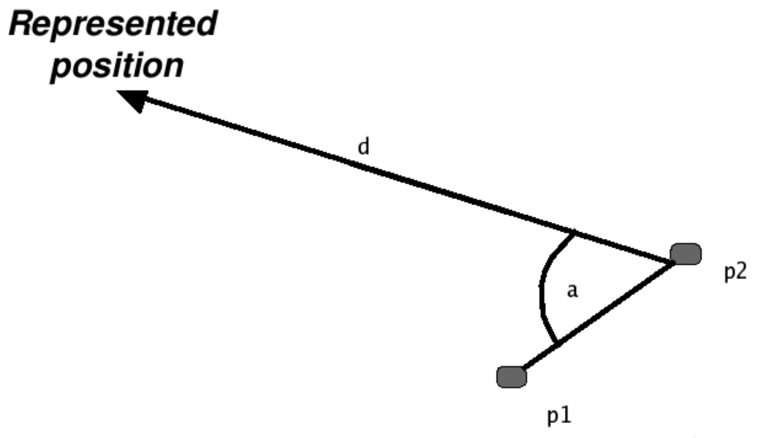

Figure 2.1: The symbolic reference used by Jung and Zelinsky [2000]. Adapted from [Jung and Zelinsky, 2000].

symbolic references for communication. Two robots were required to vacuum a laboratory floor. One robot sweeps litter away from walls and other obstacles (*Flo*), while the other vacuums up the litter (*Joh*). The system presented is divided into four layers, with each layer adding capabilities and increasing performance. The first level provides robots with basic cleaning behaviour but no ability to communicate or cooperate. The second layer provides Joh with the ability to track Flo visually: this is beneficial to overall performance, as Flo is likely to be near a pile of litter. The third layer introduces communication: when Flo sweeps a pile of litter, Flo signals the location of the litter to Joh, in coordinates relative to itself. Joh and Flo use the same wheel encoders, which provides a pre-defined shared grounding for the communicated coordinates, so Joh can make sense of Flo's communications. The fourth layer introduces communication using a symbolic reference. A symbolic reference is provided (not learned) that indicates a location in the environment in terms of two known points, $p1$ and $p2$, an angle a, and distance d. The position is indicated by drawing a reference line from $p1$ to $p2$, then travelling distance d, at an angle of a to the reference line. A diagram of this symbolic is shown if figure 2.1. Since the chosen symbol requires reference points, an explicit location labelling phase is required by the fourth layer. Their experiments found that performance increased as more layers were added to the system, although the fourth layer fell behind the others at first, since time was taken to label the environment.

Jung and Zelinsky make two central contributions in this work. First, they show that a human centred coordinate system is unnecessary for robot navigation, and communicating about locations. Second, they demonstrate the utility of a symbolic reference. Although the symbolic reference was not learned, it demonstrated the motivation for using a symbolic reference, by allowing Joh and Flo to communicate about new points in the environment, by referencing points they both already knew.

The work of Jung and Zelinsky is the closest prior work the research presented in this thesis. There is much room for improvement in their approach. The most obvious flaw is the flooding of the environment with labelled points. In a 3.45m x 5.10m office room, well over 500 points were labelled (the diagram in [Jung and Zelinsky, 2000] figure 15, was so cluttered it was hard to count). This is far in excess of what is required, and orders of magnitude more than any human would consider worth labelling for sweeping an office floor. A second problem is that their approach has only been demonstrated with two agents. This significantly simplifies the problem, since the number of potential differences in symbol grounding across the domain is limited, and each agent receives information only from one other agent. Finally, there is no allowance for a single agent to label a new point independent of the other, then share the new point later. Both agents must be present for the entire location labelling step - that is the only way in which agents acquire labels. This is unrealistic compared to most situations, where individuals are likely to take note of points that are interesting to them, and only share them with others later.

This section has presented the background necessary for the reader to understand the remainder of this thesis. It has outlined the history of symbols, agent architectures, mapping and path planning. It has also provided summaries of work by other authors which closely relate to this thesis. With the necessary background in place to understand the remainder of this thesis I will present the design and implementation of my system intended to satisfy the goals set out in Chapter 1.

Chapter 3

Developing Grounded Communication

This chapter will describe my approach to developing grounded communication. I will also present a high level view of my agent design, while leaving implementation details to the next chapter. The chapter begins with a description of the problem that the agents will solve using grounded communication in order to demonstrate my research goals. After describing the communication abilities of the agents, this chapter describes the circumstances under which individual agents can create and maintain groundings for locations, and the approach used to ultimately achieve a heuristic consistency between such groundings. The chapter ends with a high level description of the behaviours required in an agent.

3.1 Overview

3.1.1 Experimental Task

Before discussing the agent design, it is helpful to provide more detail concerning the task these agents are intended to perform. The task used as an evaluation of the approach described in this chapter is for agents to find a goal placed randomly in the environment, while developing grounded communication over time to improve overall agent performance. There is nothing special about this particular task: any task in which performance can be improved by communicating about locations could be used as a vehicle for studying grounded communication.

All agents initially have an empty set of grounded locations. As an agent wanders the environment, it creates groundings for locations it deems useful enough to refer back to in the future. A number of different strategies for deciding when a grounding might be useful are employed, and will be discussed in Section 3.2.3. These groundings are purely internal, and are not shared with others (except for one of the grounding strategies that grounds points where agents meet, necessitating that two agents initially share the grounding). Irrespective of strategy employed, agents will encounter one another from time to time. These encounters are used as opportunities for one agent to demonstrate a nearby already internally grounded location to another. That is, each encounter provides an opportunity to potentially develop a shared grounding and ultimately the ability to communicate to another agent about this particular location in such a way that it has meaning to the other agent. While previous research has required an explicit phase where grounded locations are created

[Jung and Zelinsky, 2000; Billard and Dautenhahn, 1999], this work develops shared groundings *as agents are performing work in the domain,* not at contrived times. This chapter will describe in detail the method by which such individual encounters can be expanded to a larger, shared set of groundings amongst a population of agents.

When an agent finds the goal, it signals the goal's location to the other agents. Two different techniques are used to convey this information. When using the first technique, the agent simply broadcasts the name for the grounding it has which is nearest to the goal. The subset of agents having this grounding in common will then be able to plan a path and navigate to the indicated location near the goal. While the broadcast location is not necessarily at the goal location – since the exact goal location will not been previously demonstrated to agents as a location worthy of grounding – it is still valuable in that it will allow those aware of the grounding to move close enough to the goal that further exploration will likely be more fruitful than their current efforts. When using the second technique to indicate the goal's location, the agent will specify the goal location using two grounded locations and a symbolic reference. These two grounded locations form the basis of a shared coordinate system, which allows an agent to specify a goal location even if it has no grounding near the actual goal. This flexibility comes at the cost of requiring two shared groundings instead of one.

3.2 Agent Design

Agents are implemented using a hybrid approach (Section 2.2.3). A pure schema-based approach is not appropriate for this task, as the agents are required to maintain a world map to be able to plan paths and navigate from their current position to the broadcast location of the goal, requiring deliberative as well as reactive reasoning. An Agent must also maintain a list of known grounded locations with respect to the map, as well as other minor pieces of state information, such as whether a message the agent is trying to broadcast has been successfully sent. Basic robot navigation to allow exploration of the environment is implemented in a reactive manner.

3.2.1 Agent Control

Hybrid agent control models allow an agent to perform both reactive and proactive processing. Both types of processing can be captured within a behaviour. Behaviours are abstractions which partition agent control into discrete modules. Each behaviour in the system generates an action, in order to control the robot hardware. When the agent has an opportunity to act, each behaviour in the system is given the chance to generate an action. An action represents a desired speed and heading for the robot. Additionally, actions can contain a short text message for the robot to broadcast across the environment, allowing agents to communicate. A behaviour is not required to generate an action if there is no useful work for it to do at the time. For example, the behaviour that implements the *drive-to-goal* functionality cannot perform any useful work when the goal location is not known.

Associated with each behaviour is a priority for the actions it generates. The priority is the relative strength of an action compared to the actions generated by all other behaviours. The higher an action's priority, the more influential it is in determining the action which is ultimately executed by the robot. Behaviours need not generate an action containing all of a speed, heading and text message: any combination is acceptable. For example the *wiggle* behaviour described in Section 4.5.5 adds a small amount of noise to the robot's path. It does not alter the forward speed, but simply directs the robot to turn slightly to the left or right.

In order to produce a final action for the robot to execute, a weighted average (by priority) is computed for both speed and heading. While movement can easily be viewed as a summed vector, communicative actions cannot be considered this way: we cannot simply merge different messages and have coherent meaning emerge from their sum. To deal with priority in communicative actions, the behaviour with the highest priority is allowed to send its text message. Lower priority behaviours which have their text messages overridden by higher priority behaviours are forced to wait to try to rebroadcast their message.

3.2.2 Communication

While it is desirable to avoid human-provided facilities as much as possible, and therefore create a weaker technique with broader applicability, there are some behaviours which are simply not practical to learn, and which are peripheral to the issue of symbol grounding *per se.* Agents need low level communication mechanisms for encoding information in a format (syntax) that others can understand. This thesis does not aim to develop a mechanism sufficiently general to allow agents to learn the equivalent of an ASCII table or a low level communication protocol before being allowed to communicate any information at all. Rather, the lowest level communication mechanisms - those which allow messages to be exchanged without regard to the meaningful content of those messages - are assumed to be in place. The focus will be on the content of the data transported by the low level communication mechanisms. This thesis only uses *fewer* human-provided elements than previous research: it does not avoid them entirely.

The work in this thesis focuses on developing consistent shared groundings between agents, thereby improving communication. Agents need to be to given the ability to first create their own groundings, which are not shared. Then agents need opportunities to share groundings with each other. Finally, there needs to be some mechanism in place to reconcile differences between agents when their groundings do not match. The following subsections describe how each of these components are dealt with in my approach.

3.2.3 Grounding Locations

In my approach to developing grounded communication, locations are grounded to the coordinate values maintained internally by each agent. The agent maintains its localization in physical space (that is, an indication of where it is in its coordinate system) by tracking motion using the robot's motion hardware (wheel encoders).

However, the techniques for developing grounded communication presented in this thesis do not rely on any particular choice of representation for coordinates. Individual agents are free to maintain a coordinate system using any means they wish, and no shared representation for coordinates is assumed. All that is required is for an agent to be able to navigate to previously grounded locations. The agents in this thesis do not share a coordinate system, either with each other or with the simulator providing their environment.

While this may seem primarily a design constraint, it also creates intricacies from an implementation perspective. For example, if agents are moved during an experiment, in order to place them in new, random locations to see if they can make use of the locations they have already grounded, the agents must be somehow re-localized in their own coordinate systems. That is, they must obtain information as to which coordinate in their own representation they currently reside. They cannot simply be told this information, since other agents do not use same coordinate system. The solution of problems arising from the use of different coordinate systems is part of the implementation of this work, and as such will be dealt with in Chapter 4. For the purposes of explaining the grounding techniques I have developed, it is sufficient to be able to assume that each agent has whatever means its designer has chosen to keep track of the coordinates of physical locations in the world around it.

This thesis uses three different methods for an agent to decide when a location is worth grounding. These methods are *label-at-meeting*, *label-spatial-entropy*, and *label-environment-feature*. Only one of these labelling methods is active at a time.

Label at Meeting

The label-at-meeting grounding strategy creates a shared grounding between two agents when they meet. The intent here is to have a basic strategy that simply takes advantage of the agent encounters that will inevitably occur in a multi-agent setting, without any domain-specific knowledge as to whether these locations are particularly useful. The dynamics of the environment may cause more meetings to occur in some areas than others. For example, in situations where agents tend to spend the most time, there will be more encounters, such as in difficult-to-navigate portions of the environments. There is nothing in my approach, however, that arranges such meetings in any way. When one robot encounters another, it stops and invites the other to ground a symbol. If the other is amenable (since it may be busy with other things), it requests a name for the robot's current location. The original robot names the location and the new name is subsequently used by both. Mechanisms are in place to stop this from occurring too frequently so agents are not taken away from their primary task in this environment. If an agent invites another to ground a symbol and is refused, it will wait a specified amount of time before inviting a third agent to ground a new symbol.

Label Spatial Entropy

While label-at-meeting is a domain independent technique, it is also limiting. An agent is restricted to creating label only when another agent is present. Label-spatial-entropy is a more general, while still a domain independent technique that tries to predict which locations in the environment will be useful. To make this prediction heuristically, without relying on information about any particular domain, we note a previous use of information theory to make judgements regarding spatial layout.

The *entropy of a spatial area* [Baltes and Anderson, 2003] is a metric which measures how mixed a given portion of the environment is, in terms of free and open space. It is based on Shannon's previous work on information theory [Shannon and Weaver, 1949], but applied to grid-based maps. I refer to this metric as *spatial entropy.* Spatial entropy is highest where there is the largest mixture of open and blocked areas (where the environment contains the most information). The spatial entropy for a given area A is calculated as:

Entropy(A) $= -p_f log_2(p_f) - p_b log_2(p_b)$

where:

$p_f = \frac{|f|}{|A|}$ and $p_b = \frac{|b|}{|A|}$

$|b|$ is the number of blocked cells in the area

$|f|$ is the number of open cells in the area

$|A|$ is the total number of cells in the area

Spatial entropy is a real value between 0 and 1. It is 0 when the environment is totally uniform, consisting of only open cells or only blocked cells. It increases as the environment becomes more and more mixed, reaching 1 when there are equal numbers of open and blocked cells.

Baltes and Anderson [2003] used this measure to make informed decisions about where to partition the environment in a cell-decomposition path-planning method (Section 2.2.4). I adapted this measure to this setting after noting that a similar informed decision about space could be useful in developing groundings as well, at least insofar as these are intended to be useful for agents navigating the environment. In this approach, as an agent wanders the environment, it regularly computes the spatial entropy for an area around its current position. If the spatial entropy is high enough (i.e. exceeds a threshold defined in the agent's implementation), the robot creates a new grounded location for that point.

Label Environment Feature

Two techniques for creating groundings for locations have been presented so far. Label-at-meeting requires only that agents encounter one another in the environment. The second technique, label-spatial-entropy is an attempt to identify useful locations to ground in a domain independent way. These techniques should be compared to a third, domain dependent, technique. In any domain there are features which stand out as useful locations to ground, like doors or hallways. The label-environment-feature grounding strategy identifies these locations and creates groundings at these points. Actually locating which features are useful is a matter for implementation, and the alternative chosen for the purposes of my evaluation will be explained in Section 4.3.

3.3 Sharing Groundings

Once agents have a strategy to ground locations individually, a strategy must be in place to determine when to share groundings with others. As stated above, each agent in the system uses its own private coordinate system. Because of this, groundings cannot simply be shared via communicating coordinates: the coordinates themselves are meaningless to another agent. My approach instead revolves around the idea of physically demonstrating useful locations in order to allow one agent to share its knowledge with another in a grounded fashion.

Since groundings are shared by physical demonstration, agents have the freedom to ground names to locations in any way they wish. Although all the agents in this thesis use Player's tracking subsystem, which uses movement values reported by wheel encoders to infer motion, this need not be so. For example, an agent could use a vision system and differentiate locations by recognizing features of the environment, or use visual motion to detect movement (ego-motion detection) and extrapolate the agent's current coordinates. My approach does not impose any restrictions on an agent's internal representation of the groundings. As long as agents are able to navigate to existing groundings, in order to demonstrate them to others, the technique this thesis presents for developing shared groundings will be applicable.

Associated with each grounding is a reference count. A reference count is a heuristic measuring how many agents share a particular grounding. Groundings with a reference count of one are known to only a single agent, groundings with a reference count of two are known to two agents, and so on. A reference count is not an exact measure of how many agents share a particular grounding, only an indication. For example, if an agent is told a location has a reference count of five, this means that to the best of that agent's knowledge, five agents know this location under this name. Other agents may have spread the information further since that agent obtained it, and still other agents may have lost this grounding by using its name somewhere else. The details of how groundings can change are provided in the remainder of this section. When a new grounding is created the initial reference count is one, except in the case of label-at-meeting, which creates groundings with a reference count of two, since the grounding is shared between two agents.

There are two main conditions which must be met before an agent will attempt to demonstrate a grounding to another. First, there must be a second agent nearby, as groundings are only shared when agents happen to encounter one another. Second, the agent must have a grounding for a location nearby. With these conditions met, an agent will send a message offering to demonstrate a grounding to the agent it has encountered, in a similar fashion to label-at-meeting. If the second agents accepts the offer, it will signal the offering agent, and begin to follow the offering agent to the location to be demonstrated. The first agent will move to the location of the grounding it wishes to demonstrate, then sends a message to the second containing the name under which it knows this location, and the reference count. If this location is novel to the second agent, that agent will begin to use the broadcast name (that is, the name being suggested by the broadcasting agent) for the demonstrated location, and will signal the first agent that is has done so. Each agent will also increment the reference count associated with the just demonstrated grounding, to reflect that there is now one more agent which knows the demonstrated location by the broadcast name.

While this describes the general concept, in practise there are a number of cases that arise that are more complex than this. What if the second agent already knows the location the first agent demonstrated? What if the second agent already uses the given name for a different demonstrated location? Does the second agent already know about this location under this name, or does the second agent already use that name for a different location? In order to resolve these potential conflicts, the second agent must ask itself two questions: *do I know this demonstrated location already?* and *do I know this broadcast name already?* Answering these questions uncovers potential conflicts and these can then be resolved, through one or the other agent altering a grounding. When a conflict does occur, something must be done to resolve it, necessitating that one agent alter its groundings. To determine which agent should maintain its current grounding and which should change, reference counts are used. The principle underlying all conflict resolution is that the more agents that know a particular location by a particular name (as indicated by the reference count for that location), the more valuable the location is to keep under its current grounding.

Given the two binary determinations in the paragraph above, there are four possible outcomes. The case where the answer to each question is no (that is, the second agent knows neither the demonstrated name or the demonstrated location) indicates no conflicts and has already been discussed, leaving three more possibilities, each discussed in their own subsection:

Case 2: The second agent knows the demonstrated location, but not the broadcast name.

Case 3: The second agent does not know the demonstrated location, but knows the broadcast name.

Case 4: The second agent knows both the demonstrated location, and the broadcast name.

3.3.1 Case 2

If the second agent knows the demonstrated location, but not the broadcast name, it must determine which name is better to use for that location: the broadcast name, or the name it is already using. If the reference count associated with the broadcast name is higher than its own reference count for the demonstrated location, the broadcast name is better to keep, as more agents would appear to know the location by that name. In this case, the second agent signals the first that it has adopted the broadcast name for the demonstrated location and the reference count is incremented.

If the second agent's reference count for the demonstrated location is higher than the reference count for the broadcast name, it is better to keep the second agent's name for the demonstrated location. In this case the second agent tells the first its name for that location, the first forgets the broadcast name and begins to use the second agent's name for the demonstrated location, and the reference count is incremented.

3.3.2 Case 3

If the second agent does not know the demonstrated location, but does know the broadcast name, this means that this name is being used by the second agent to ground some other location. Here, the process to deal with the conflict is analogous to the previous case. If the demonstrated location is more valuable (indicated by a higher reference count), the second agent forgets its current grounding for the broadcast name (i.e. forgetting about the existence of the other grounding), and adopts the broadcast name as a grounding for the demonstrated location. Both agents also increment the broadcast reference count, since in this case one more agent will be using the new location grounding.

If the second agent's current grounding for the broadcast name (i.e. another location) has a higher reference count, the second agent tells the first that it should forget that the broadcast name refers to the demonstrated location, and adopt the second agent's grounding instead. That is, the other agent's grounding is different from, and better known than, the location being demonstrated. The second agent increments its reference count for its grounding for the broadcast name before sending it to the first agent, as there is now one more agent using the second agent's name to refer to the demonstrated location. In this situation, no new grounding has been made (the demonstration was not used), but a stronger grounding (one already used by more agents) has been spread further instead. This also shows why reference counts can be inaccurate, as in either situation here, one less agent is using a grounding that may still be used by others. Over time, this and the other conflict resolution methods should cause consistent groundings to emerge - that is, weaker groundings will gradually become extinct through this process.

3.3.3 Case 4

If the second agent knows *both* the demonstrated location and the broadcast name, the most obvious thing to consider is that this location is already known to both agents.

In order to do this the agent must determine if this is the case. Both agents may have grounded the same useful location using similar strategies, but in that case there could still be some variation in the *exact* location that was grounded. Similarly, both agents could have received a demonstration of this, and again, due to differences such as odometry errors, the location is unlikely to be *exactly* the same. In my implementation (Chapter 4), I set a tolerance value within which locations are considered to be the same, and this is used to decide whether a grounding can be considered to be the same location.

If it does (broadcast name refers to demonstrated location in both agents), then all that need be done is to adjust reference counts. Since the grounding is already shared, this encounter will be used to ensure that agents share the same reference count for this grounding, and thus spread the information about the grounding more accurately to others in future. If the second agent has a higher reference count than the first, it will broadcast this reference count to the first. If the broadcast reference count is higher than the second agent's reference count, it will adopt the broadcast reference count, and neither agent will increment the reference count (since both agents already knew the demonstrated location by the broadcast name).

If the second agent does not know the demonstrated location under the broadcast name, this means that it (and likely others as well) has grounded this location previously, but under a different name. In this case, the conflict can be resolved similarly to the others already described, and the agent determines, based on reference counts, which grounding is more valuable to keep. If the broadcast reference count is higher than the second agent's reference count the second agent forgets its current information for the broadcast name (i.e. it will forget the grounding elsewhere currently associated with this) and adopts the broadcast name for the demonstrated location (causing that agent to lose the grounding for the current location, since that grounding is less valuable to the group of agents as a whole). In both these situations, agents lose groundings, but walk away from the encounter with a more globally consistent set of groundings, which is ultimately what is most desirable. Each agent also increments the broadcast reference count, as the second agent now knows the demonstrated location as the broadcast name. If the second agent's reference count for the broadcast name is higher than the broadcast reference count, the second agent tells the first agent to forget that broadcast name refers to the demonstrated location. It is a grounding which is less valuable to the group of agents as a whole.

This section has described the way in which groundings become shared between agents. The mechanisms above describes how two agents interact in order to share a grounding, and as well as how conflicts are resolved between them. As agents explore the environment, and encounter different agents, these groundings will spread from agent to agent, eventually becoming shared across the entire agent population. The implementation of these strategies will be described in Chapter 4.

3.4 Agent Behaviours

Where the previous section described the mechanisms for sharing groundings between two agents, this section will outline the major behaviours which are present in the system to explore, find the goal, create individual groundings, and share those groundings. In order that the reader not be overwhelmed with implementation-level details here, such details are presented separately in Section 4.5.

Go Straight causes the robot to drive straight ahead, at full speed. It does not perform any obstacle avoidance. This is the basic behaviour which causes the robot to explore the environment and search for the goal when it would otherwise be stopped.

Don't Crash prevents the robot from hitting any obstacles by monitoring the distance values reported by the sonars. Most of the time this behaviour contributes very little to the robot's movement: it only becomes significant if the robot comes close to an obstacle. When an obstacle is first detected, Don't Crash gently directs the robot away from it, becoming more forceful as the obstacle becomes nearer. If an obstacle becomes too close and there is potential for a collision to occur, Don't Crash enters *bail out* mode which directs the robot to back up slightly and turn randomly to either the left or the right for a short period of time. Choosing a random turning direction for a variable amount of time helps the robot avoid getting stuck, retrying the same actions over and over, with no change in the outcome.

Random Wander causes the robot to follow a random heading for a short period of time, after which a new heading is chosen. It does not do any obstacle avoidance. If Don't Crash enters bail out mode, Random Wander gives up on its current heading and does not contribute to the robot's actions until it is time to choose a new heading.

Wiggle adds a small amount of noise to the robot's heading to help get better sonar converge for mapping. If a robot drives straight down a constant heading, it will take longer to fill in the gaps which occur between the sonars (i.e. further passes will be necessary) than if there are slight variations in heading. Small objects which fit between the sonar beams are also better detected when using Wiggle.

Self Pose Tracker tracks the robot's current position and makes it available to other behaviours. It informs other interested behaviours when the robot's position changes. Finding the robot's current position is not as straightforward as querying the hardware, because the robot does not use the full position resolution made available by the (simulated) hardware. Having a single behaviour perform this translation ensures that all behaviours maintain consistent position information.

World Resizer keeps track of the largest and smallest robot positions reported by

Self Pose Tracker. This behaviour is necessary because the robot does not assume a fixed world size. Initially the environment is assumed to be small, when a new maximum or minimum position value is reported, on either the x or y axis, World Resizer reports this information to all interested behaviours, allowing them to make the necessary adjustments.

Map Environment maintains an occupancy grid of the environment based on the robot's sonar readings. If a sonar pulse passes through a given portion of the environment (a *cell*) that cell must not be blocked, and if a sonar pulse does not pass through a cell that cell must be blocked. Once a given cell has had a certain threshold of sonar pulses pass through it it is marked as open. If enough sonar pulses are reflected from a cell that cell is marked as blocked. Cells are permitted to change states from open to blocked or vice versa, based on new sonar information. This is necessary, as without this other robots would otherwise be perceived as permanent obstacles when they are only temporary. Map Environment does not make any assumptions about size of the environment. It pays attention to events from World Resizer and responds to them by allocating more cells in the occupancy grid to accommodate the newly expanded world size. The maintained occupancy grid is used when the robot needs to plan a path to another point in the environment, but of course could serve to support many applications.

Say Message is used by other behaviours to communicate with other robots. To simulate the unreliability of real world communications, the Stage simulator does not guarantee that messages robots attempt to say will actually make it into the environment. Robots must listen to verify that communications actually made it into the environment. If a robot says a message, and it is not later heard by the robot, the message must be resent. Say Message takes care of all the message tracking and restransmission so other behaviours do not need to worry about it.

Goto Fixed Location receives a set of coordinates in the environment as input, in the robot's local coordinate system, and directs the robot to this point. A behaviour can ask Goto Fixed Location to plan and follow a path on its behalf, freeing it from details of how these are implemented. The behaviour is informed when Goto Fixed Location has navigated the robot to the destination, or a failure occurs.

Goto Moving Target is similar in purpose to Goto Fixed Location above, but instead of directing the robot to a fixed point, it tracks a moving target. A behaviour provides a location relative to the robot and Goto Moving Target turns and heads straight for it. Goto Moving Target does not do any path planning, as it assumes that the target is already in view, since callers provide goal information relative to the robot. The goal location can be updated whenever the controlling behaviour wishes.

Goal Seeker continuously scans the environment to see if the goal is in view. If it is, Goal Seeker broadcasts a message using a Say Message behaviour to all other robots describing the goal's location and uses the Goto Fixed Location behaviour to drive to the goal. The broadcast goal location can be in one of two forms. When using the first form, the name of the robot's nearest grounded location with a reference count of a least two is broadcast. When using the second form, the goal's precise location is broadcast by using a symbolic reference based the robot's two groundings with the highest reference counts (see Figure 2.1).

Goal Seeker also listens for goal locations broadcast by other robots. When goal information is heard in the form of the nearest grounding, the robot looks up the location and plans a path to the announced goal. If the goal location is not known by the robot, or path planning fails, the information is ignored. If path planning succeeds the robot starts to move toward the goal. It is also possible that an agent hears a broadcast goal location and has a grounding for the broadcast name, but it is a different grounding from the sending agent. In this case, the agent hearing the broadcast grounding will drive to the wrong part of the environment, hindering its efforts to find the goal. If goal information is broadcast in terms of a symbolic reference, the robot resolves the goal location and attempts to move to that location in the same manner described above. If goal information is received while the robot is already driving towards an announced goal it is placed in a queue, and will be used if the current goal information does not allow the robot to find the goal.

Label At Meeting implements the label-at-meeting location labelling strategy. This labelling strategy is described in Section 3.2.3, and implementation details are presented in Section 4.3.

Label Spatial Entropy implements the label-spatial-entropy labelling strategy. This labelling strategy is described in Section 3.2.3, and implementation details are presented in Section 4.3.

Label Environment Feature implements the label-environment-feature labelling strategy. This labelling strategy is described in Section 3.2.3, and implementation details are presented in Section 4.3.

Location Exchanger is the most important behaviour for agents in my research. It allows one robot to demonstrate a new grounding to another, creating a shared grounding between the two. It is by this mechanism that groundings created by an individual robot are spread throughout the population. When this behaviour detects another robot nearby, and knows a nearby grounding, it starts a conversation with the second robot. If the second robot accepts the conversation invitation, the first will drive to the grounding to be demonstrated and tell the second the name for that grounding. The second robot will then known that location by the same name as the first robot, and both robots will increase the reference count for that location. Issues of conflict in groundings and their resolution are dealt with as described in Section 3.3. The implementation of the location exchanger in Section 4.4.

This chapter has outlined my approach, at a high level, to building agents capable of developing a consistent set of shared groundings for locations. The next chapter will provide the specifics of how this approach was implemented.

Chapter 4

Implementation

This chapter describes the implementation of the design presented in Chapter 3. It begins with a discussion of the Stage simulator, the programming language and testing hardware used in my implementation. Following this, the communication and mapping abilities are detailed. The heart of the chapter provides implementation details on each of the three labelling strategies: label-at-meeting, label-spatial-entropy, and label-environment-feature. This leads into a discussion of how locations are exchanged, how conflicts can occur, and how these conflicts are resolved. The final section of the chapter presents the agents' behaviours and related concepts from an implementation perspective.

4.1 Simulation

While running experimental trials with real robots is desirable, it is not always practical. Simulations allow experimental trials to be run more quickly than in the real world [Hanks et al., 1993]. Simulation also assists in dealing with issues of reliability and reproducibility in the real world. Software simulations provide the ability to start agents in precisely the same state, and experience precisely the same inputs during an experimental run. This is extremely difficult to accomplish in the real world [Cohen et al., 1989]. Beyond the ability of a single researcher to reproduce their own results, reproducibility also applies to separate researchers. If multiple researchers use the same simulation package, their work can easily be compared, while the inevitable differences between the experimental domains of researchers using physical agents hampers comparison. Problems with physical robot reliability also add difficulty to running trials on real robots [Etzioni et al., 1992]. If an agent experiences a mechanical malfunction the trial is invalid, and will need to be rerun. This is not necessarily a significant issue with a single robot, but becomes a large problem when a team of any size is involved. Beyond these issues, I did not have access to the 16 robots required to evaluate my approach in the real world: limited equipment and affordability is another reason that simulation is often used in multi-agent systems research.

The software simulator used for my implementation was Player/Stage [Gerkey et al., 2001], which while theoretically useful for any robotic system, has been validated as accurately simulating the behaviour of Pioneer robots. Code developed under Player/Stage will run under Pioneer robots directly, as well as in simulation. Stage is a program that simulates the environment, while Player is a program which acts as a proxy between either the Stage simulator (when used in simulation), or the actual Pioneer robot hardware (when used on physical robots). Player receives sensor readings from the environment (real, or simulated via Stage) and passes them to the robot, as well as executing the motor commands received from the agent.

My robot control program is written in Java 1.4. I used the publicly available Java Client for Player/Stage [Batalin, 2004] to interface with Player/Stage. Simulations were carried out on six Pentium III 850MHz computers with 256MB RAM running the Fedora Core 2 Linux distribution. These computers were used in three pairs. In each pair the first computer was used to run the Stage simulator, while the second was used to run all the of the simulated robots. Since Stage has the ability to run faster than real time, I was able to run some simulations more quickly that would be possible with physical robots. I found that the fewer robots that were simulated, the faster I could run Stage while maintaining some CPU idle time, which ensures that Stage has enough processing time to perform an accurate simulation.

4.2 Agent Description

Although the agents in this work are simulated in software, the simulated hardware they are equipped with is the same as a Pioneer DX2 by ActiveMedia Robotics. The Pioneer DX2 has been superseded by the DX3, but is equivalent for the purposes of this thesis. The Pioneer has eight forward facing sonars spanning 180 degrees. The sonars are positioned at 90, 50, 30, 10, -10, -30, -50, and -90 degrees with respect to the middle of the front of the robot. Pioneers are equipped with wheel encoders which provide an estimate of the robot's current heading and position, with respect to the heading and position of the robot when it was first turned on. Pioneer robots can travel at a maximum velocity of 1.6m/s. Examples of research performed using physical Pioneer robots can be found in [Drysdale and Lyons, 2004; Shell and

Figure 4.1: A Pioneer DX2 Robot by ActiveMedia

Matarić, 2005]. In addition to the standard Pioneer hardware, my simulated robots are equipped with a laser range finder and a broadcast device. While laser range finders are much more accurate for mapping purposes than sonar, in this research they are only used for their ability to identify particular markers in the environment. All mapping is performed exclusively with sonar. Each agent is given a unique numeric marker detectable by an agent's laser range finder hardware. Agents use this numeric identifier to uniquely identify each other. This is necessary to allow agents to specify an intended recipient for a message as the simulation software does not provide any direct agent-to-agent messaging options.

4.2.1 Communication

Agents are able to communicate with each other by broadcasting messages through the environment. In my approach, messages are of the form:

R_{source},R_{dest}:*message number*,*message specific data*;

where:

R_{source} is the numerical identifier of the sender of the message

R_{dest} is the numerical identifier of the intended recipient of the message

message number is the type of the message (the meaning) *message specific data* varies according to the message

4.2.2 Mapping & Path Planning

At the beginning of experimental trial (details of the experiments performed will appear in Section 5.1.1), each agent is placed in the environment with a random location, with a random heading. While the Stage simulator defines the origin in the bottom left hand corner of the simulated physical environment, each agent perceives its starting location at the coordinates 0,0, with a heading of 0. This means that each agent has its own private coordinate system, which is both offset and rotated with respect to the Stage coordinate system. An agent has no knowledge of the relationship between its own coordinate system and the coordinate system of others: it is thus not able to communicate meaningful spatial information by using raw coordinates and headings, and must use names for commonly grounded locations.

As an agent explores the environment searching for the goal, it maintains an occupancy grid using sonar data. The occupancy grid is represented as a 2-dimensional array of *cells*. Each cell in my implementation was 10x10cm. Cells which have sonar pass through them are considered to be open, while cells which reflect sonar are considered to be blocked. Since sonar data is noisy, determining a cell's open or blocked status is more complicated than taking a single sonar reading concerning that cell. To account for this noise, a record of the number of sonar pass-throughs and reflections is kept. All cells start out with a score of zero. A pass-through increments a cell's score by one, while a reflection decrements the cells score by three. Cells can have a maximum score of 12 and a minimum score of -6. A cell with a score of 6 or larger is considered open, while a cell with a score of -3 or lower is considered blocked, and values in between are considered unknown. This pessimistic scheme is biased to conclude that cells are blocked more quickly than open, in order to help ensure collision-free path planning.

Agents also have no information about the size of the environment at the beginning of the simulation. Initially, each agent assumes the environment extends 20 grid cells in each direction from their starting position of 0,0. As an agent explores the environment, it may find itself moving beyond its initial estimated environment size. When this happens, the agent expands its world size by 5%, along the appropriate axis, to accommodate a larger environment. This world expansion scheme allows the agent to represent large environments while not wasting memory on small environments. The flexibility comes at the cost of some runtime efficiency, as portions of the occupancy grid sometimes need to be copied when the world is expanded.

As the agent is searching for the goal it may hear information about the location of the goal being broadcast through the environment. When this occurs, a quadtree (Section 2.2.4) is constructed to represent the free space and obstacles in the environment. A path is then planned using the quadtree representation. After the path to the goal has been constructed, the path is simplified by removing unnecessary nodes, using the technique of Davis [2000]. This simplification is based on the realization that some nodes along a path planned through a quadtree decomposition are unnecessary. For example, if a planned path goes from node a to node b to node c, it may be possible to remove the middle node, node b. This is possible when there is an obstacle-free straight line path from node a to node c, then node b is unnecessary and may be eliminated. This simplification results in shorter paths.

4.3 Location Grounding Behaviours

This section describes the implementation of the location grounding behaviours described in Section 3.2.3. Only one of these grounding strategies is active at a time. Regardless of the grounding strategy in use, when a new location is grounded, any existing locations within a 1.5m radius with a reference count of one are forgotten. This is to prevent the environment from becoming flooded with groundings which have a reference count of one. Groundings with a reference count of one are expendable, as they are not shared with any other agents. All grounding strategies name new groundings with a randomly generated integer in the range 0 to 10000. If the new randomly generated name is already in use to ground a location within the agent a new name is selected.

Label At Meeting

Implementing the label-at-meeting strategy involves the appropriate use of the communication mechanisms described above. These are summarized in Figure 4.2.

When one robot (r_1) senses another robot (r_2) within 4m, r_1 stops in place, and sends a LABEL_MEETING_START message to r_2. This is an invitation to begin the grounding creation process. If r_2 is receptive, the two agents will create a new shared grounding. If r_2 is receptive, it stops in place, turns to face r_1, and sends a LABEL_MEETING_LOC_NAME message. Included in this message is a name for the new grounding (l_p). l_p is randomly generated by r_2, but r_2 ensures that it

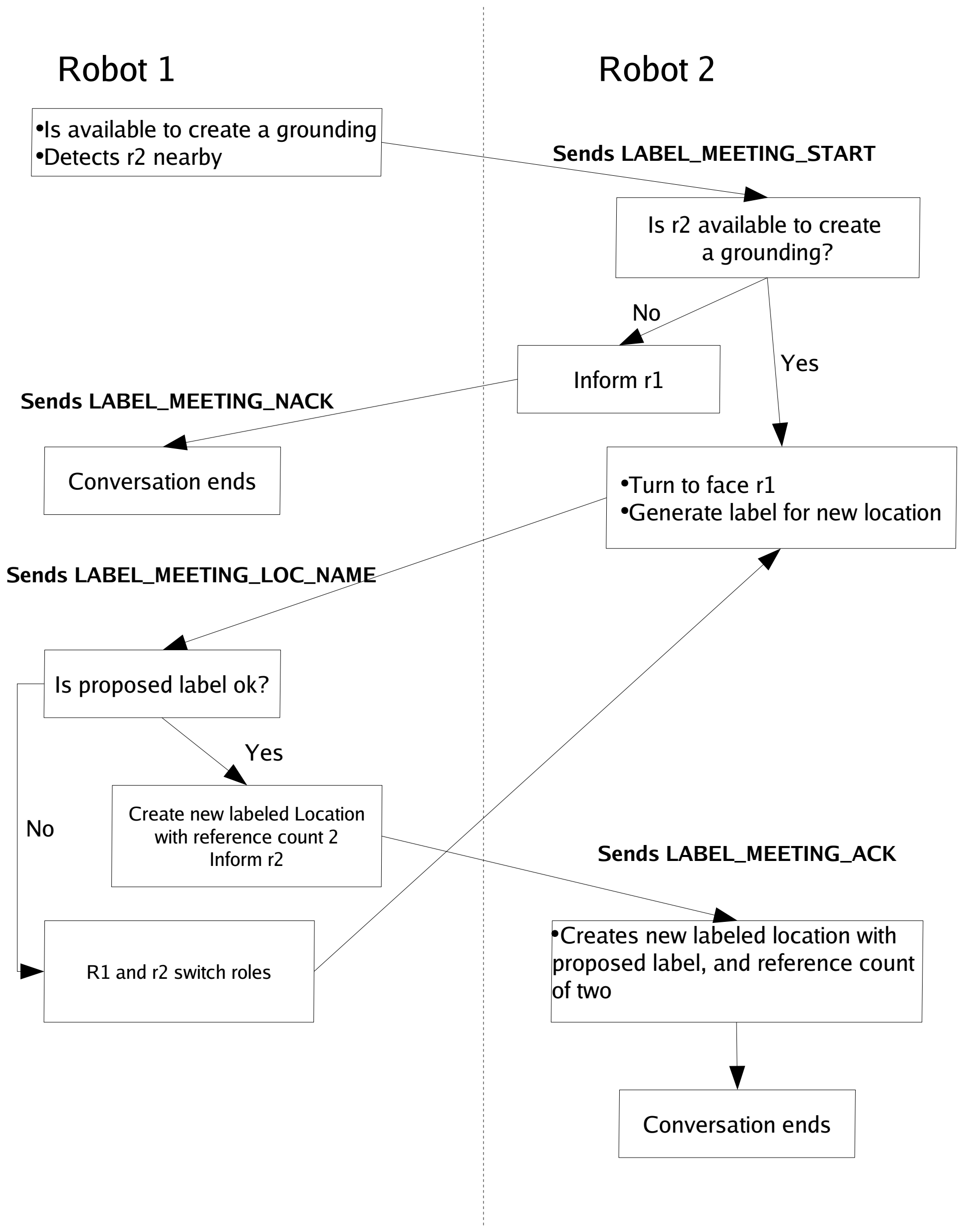
Robot 1
Robot 2
•Is available to create a grounding
•Detects r2 nearby
Sends LABEL_MEETING_START
Is r2 available to create a grounding?
No
Yes
Inform r1
Sends LABEL_MEETING_NACK
Conversation ends
•Turn to face r1
•Generate label for new location
Sends LABEL_MEETING_LOC_NAME
Is proposed label ok?
Yes
No
Create new labeled Location with reference count 2 Inform r2
Sends LABEL_MEETING_ACK
R1 and r2 switch roles
•Creates new labeled location with proposed label, and reference count of two
Conversation ends

Figure 4.2: Label-at-meeting agent interaction.

does not already use l_p to name another grounded location. If r_2 is not receptive, it responds to r_1 with a LABEL_MEETING_NACK and both agents continue to explore the environment, searching for the goal. r_1 will not try to initiate another label-at-meeting conversation for 20 seconds after receiving a LABEL_MEETING_NACK message. Agents are required to wait 30 seconds between creating label-at-meeting groundings in order that they can spend time searching for the goal, not just creating new grounded locations with each other. This is especially important when there are a large numbers of robots in a small environment.

Assuming that r_2 was receptive, and responded to r_1 with a message of type LABEL_MEETING_LOC_NAME, this message is processed by r_1. It evaluates the proposed location name. If the proposed name l_p is not already in use in r_1, then l_p is accepted. r_1 signals this to r_2 with a LABEL_MEETING_ACK message. Both agents then know *the location halfway between them* as l_p. l_p starts with a reference count of two, since there are two agents which know the location as l_p. In the event that r_1 already has a grounded location by the name of l_p, it sends a LABEL_MEETING_LOC_NAME message back to r_2, with a new name for the location. r_2 processes the LABEL_MEETING_LOC_NAME message in the same manner that r_1 did. This process goes back and forth until a novel name in both r_1 and r_2 is found, or the conversation times out after 15 seconds.

Label Spatial Entropy

The label-spatial-entropy labelling strategy labels locations in the environment based on spatial entropy (described in Section 3.2.3). The agent computes the spatial entropy for a 15x15 cell box centred around its current location every ten seconds. If the spatial entropy is ≥ 0.75, the location is grounded. If the agent is near the edge of the environment, so the box extends outside the environment, it is moved to be within the environment. Note that Chapter 5 describes some experiments where values other than 0.75 were used as the minimum amount of spatial entropy to make a location worth grounding.

Label Environment Feature

Label-environment-feature creates groundings for locations based on features in the environment. This is an anthropocentric attempt to capture useful domain-specific properties, such as doorways and hallways. Because the focus of this thesis is not on robotic perception, instead of going through the more complicated procedure of recognizing doorways and hallways using sonar, I placed markers in the environment at these locations. Each marker was given a unique identifier in the same way that agents are uniquely identified – these thus allow the unique identification of environmental features when they are discovered. An agent using this strategy creates a grounding when it perceives a marker within 2m. The agent remembers the unique identifier of each marker they create a grounding for, and grounds the location of each marker only once per simulation trial.

While using unique markers for locations that are worth grounding instead of having to detect the features themselves speeds the implementation, it is not entirely accurate. A full detection implementation would be subject to both false positives and false negatives when attempting to identify environment features worth grounding.

Also, since each marker is given a unique identifier, agents could potentially use these markers to artificially aid in localization. If an agent perceives a *unique* marker that it has perceived before, it will know it is in the exact same location it was before. This is as opposed to using a the perception of a landmark for localization, where perceiving (for example) a doorway would lead to at least some uncertainty of what specific doorway it was. Agents in this thesis do not use the markers to aid in localization, localization is performed solely with the agent's wheel encoder hardware.

4.4 Location Exchanger

This section discusses in detail the implementation of the mechanism by which groundings are shared, described in Section 3.3. When sharing groundings, the question will arise as to whether a location being demonstrated or referred to by one agent is "the same as" that used by another. Due to errors in both perception and odometry, determining the spatial equality of two locations is more complex than simply comparing numeric coordinates for equality, even if disparate coordinate systems could be reconciled. For the purposes of implementing the approach for sharing groundings detailed in Section3.3, an tolerance factor ϵ is employed. If two locations are physically within a limited range (50 cm here, though some experiments in Chapter 5 varied this factor in order to examine its effect on successful grounded communication), they are considered the same.

Figure 4.3 illustrates the detailed interaction between two agents attempting to develop a shared grounding. When an agent r_1 detects another agent r_2 nearby (by default 5m, but Chapter 5 describes experiments where other values are used), and has

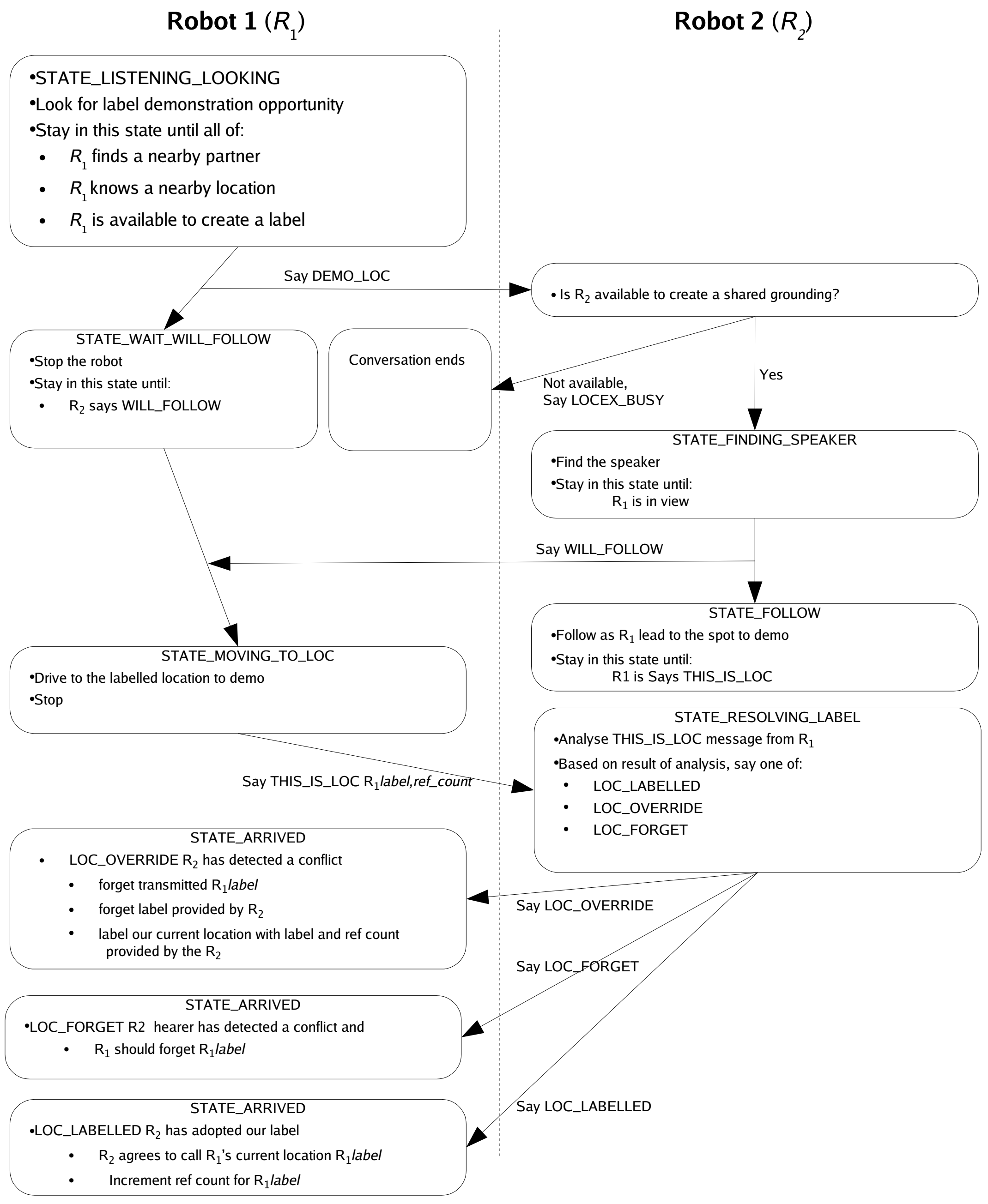
Robot 1 (R_1)
Robot 2 (R_2)
•STATE_LISTENING_LOOKING
•Look for label demonstration opportunity
•Stay in this state until all of:
• R_1 finds a nearby partner
• R_1 knows a nearby location
• R_1 is available to create a label
Say DEMO_LOC
• Is R_2 available to create a shared grounding?
STATE_WAIT_WILL_FOLLOW
•Stop the robot
•Stay in this state until:
• R_2 says WILL_FOLLOW
Conversation ends
Not available,
Say LOCEX_BUSY
Yes
STATE_FINDING_SPEAKER
•Find the speaker
•Stay in this state until:
R_1 is in view
Say WILL_FOLLOW
STATE_FOLLOW
•Follow as R_1 lead to the spot to demo
•Stay in this state until:
R1 is Says THIS_IS_LOC
STATE_MOVING_TO_LOC
•Drive to the labelled location to demo
•Stop
STATE_RESOLVING_LABEL
•Analyse THIS_IS_LOC message from R_1
•Based on result of analysis, say one of:
• LOC_LABELLED
• LOC_OVERRIDE
• LOC_FORGET
Say THIS_IS_LOC R_1*label,ref_count*
STATE_ARRIVED
• LOC_OVERRIDE R_2 has detected a conflict
• forget transmitted R_1*label*
• forget label provided by R_2
• label our current location with label and ref count provided by the R_2
Say LOC_OVERRIDE
Say LOC_FORGET
STATE_ARRIVED
•LOC_FORGET R2 hearer has detected a conflict and
• R_1 should forget R_1*label*
Say LOC_LABELLED
STATE_ARRIVED
•LOC_LABELLED R_2 has adopted our label
• R_2 agrees to call R_1's current location R_1*label*
• Increment ref count for R_1*label*

Figure 4.3: Diagram of the agent interaction implemented by the Location Exchanger behaviour.

grounded a nearby location (within 2.5m of its current position), r_1 may attempt to share a grounding with r_2. The conversation is initiated by r_1 sending a DEMO_LOC message to r_2. If r_2 is available it turns to face r_1 and replies with a WILL_FOLLOW message, indicating that it will follow r_1 to the location to be demonstrated. r_2 may not be available because it is already creating a shared grounding with another agent, or it has recently been involved in creating a shared grounding. An agent must wait 30 seconds between creating shared groundings.

After r_2 sends a WILL_FOLLOW message, it begins to track r_1. Once r_1 has arrived at the location to be demonstrated, it stops and sends a THIS_IS_LOC message to r_2. This message also contains name of the grounding r_1 is demonstrating and r_1's reference count for that location. r_2 now has the following information:

$r_{1location}$	r_1's current location
r_{1name}	the name r_1 broadcast for its current location (the name of the grounding)
$r_{2labelled\_location}$	the grounding which r_2 knows which is within ϵ of $r_{1location}$ (may be null)
r_{2name}	the grounding which r_2 knows by the name r_{1name}

r_2 now knows the coordinates (r_1's location) and r_1's name for this grounding. r_2 then compares this information to its own list of groundings. r_2 needs to make two important Boolean determinations, which determine its response. The first decision is whether or not r_2 knows the coordinates of the location r_1 is demonstrating, called $demo_{coordinates}$. The second is whether or not r_2 knows the name that r_1 has provided for the coordinates, called $demo_{name}$. Note r_2 records $demo_{coordinates}$ in its own coordinate system, as it has no knowledge of $demo_{coordinates}$'s coordinates in r_1's coordinate system. The cases that result from these two decisions have already been explained in Section 3.3. Here, I explain what occurs in each case from an implementation perspective.

If f_2 knows neither $demo_{coordinates}$ nor $demo_{name}$, there is no possible conflict. r_2 adopts $demo_{name}$ for $demo_{coordinates}$, and replies to r_1 with a LOC_LABELLED message. Now r_1 and r_2 know r_1's current location as $demo_{name}$. Since there is now one more agent referring to $demo_{coordinates}$ as $demo_{name}$, each agent increments the reference count which r_1 originally broadcast for the location in the THIS_IS_LOC message. The conversation then ends. The case where there are no conflicts is the one trivial case in this problem. The other three cases cover possible conflicts, and are dealt with in the following subsections.

4.4.1 Case 2

If r_2 knows $demo_{coordinates}$, but not $demo_{name}$, r_2 must determine which name is best to use for the proposed location, its own or r_1's. If the name that r_2 is currently using for $demo_{coordinates}$ has a higher reference count than the reference count r_1 broadcast in the initial THIS_IS_LOC message, it is better for both agents to use r_2's name for the location, as described in Section 3.3.1. In this case r_2 broadcasts a LOC_OVERRIDE message, and tells r_1 its name and reference count for r_1's current location, which r_1 adopts.

If r_1's proposed location has a higher reference count than r_2's other name for that location, r_2 adopts the $demo_{name}$ for $demo_{coordinates}$, and adds one to the reference count (since it now also calls $demo_{coordinates}$ by $demo_{name}$). Then r_2 sends a LOC_LABELLED message back to r_1, causing r_1 to increment its reference count for $demo_{name}$. The conversation then ends.

4.4.2 Case 3

If r_2 knows $demo_{name}$, but not $demo_{coordinates}$, r_2 must determine whether the location currently grounded by $demo_{name}$ is more useful than this new proposed grounding, as described in Section 3.3.2. If r_2's name has a higher reference count than r_1's, r_2's location is more valuable. However, since it's impractical for r_2 to demonstrate this location to r_1 (it may be far away), the best option is for r_1 to simply forget about $demo_{name}$. To effect this, r_2 sends a LOC_FORGET message to r_1, and r_1 forgets all knowledge of $demo_{name}$

If r_1's name has a higher reference count, r_2 forgets its current information about $demo_{name}$ and grounds $demo_{name}$ to $demo_{coordinates}$. r_2 then sends a LOC_LABELLED message to r_1 and both agents increment the reference count for $demo_{name}$ which r_1 broadcast in the THIS_IS_LOC message.

4.4.3 Case 4

There is one more possibility to consider, where r_2 knows both $demo_{name}$ and $demo_{coordinates}$. In this case r_2 must first determine if it has $demo_{name}$ grounded to $demo_{coordinates}$. If so, r_1 and r_2 already share a grounding for this location. The only thing to do in this case is ensure the reference counts are the same, as described in Section 3.3.3. If r_1 has a higher reference count, then r_2 adopts that reference count for $demo_{name}$, responds to r_1 with a LOC_OVERRIDE message, and provides the same reference count r1 provided in the THIS_IS_LOC message. r_2 must send a LOC_OVERRIDE message instead of LOC_LABELLED, because a LOC_LABELLED message would cause r_1 to incorrectly increase reference count for

$demo_{name}$.

If r_2 determines that it does not share a grounding for $demo_{name}$ with r_1 ($demo_{name}$ refers to different location in the two agents), then r_2 must determine which location should be kept based on the reference counts. If the reference count r_1 provided in the THIS_IS_LOC message is higher than the reference count for r_2's grounding of the name $demo_{name}$, r_2 adopts r_1's label and reference count. Then r_2 sends a LOC_LABELLED message back to r_1, and both agents increment their reference counts.

If r_2 has a higher higher reference count than r_1, r_1 should forget $demo_{name}$. r_2 sends a LOC_FORGET message causing r_1 to forget about its lower reference count grounding for $demo_{name}$.

4.5 Agent Implementation

Having presented the details of how locations are grounded and then exchanged, this section moves on to describe the major components in the implementation of the agents used in my research. Is starts by introducing the concept of an Action, and then discusses how Actions are generated and executed. Next the Experiment Coordinator is discussed, the entity which coordinates the experimental trials. Finally, the behaviours which generate the Actions are discussed.

4.5.1 Actions

All of the behaviours used in my agents (the behaviour-based model of control used in my agents was described in Section 3.2.1) control the robot by generating

Actions. An Action is the combination of: a speed, a turn rate, a text message to be broadcast, and a priority. The speed of an Action represents how fast the agent should be travelling, straight ahead in meters per second. A negative speed means the agent should be driving backwards. The turn rate specifies how fast an agent should be turning. A turn rate of zero means the agent should not turn at all. A positive turn rate means the agent is to turn left at the specified number of degrees per second. A negative turn rate means the agent is to turn to the right at the specified rate. The text message component of an Action is a text message which the agent is to broadcast through the environment. Finally, priority determines the strength of this Action relative to other Actions. The manner in which Actions are combined has already been described in Section 3.2.1.

4.5.2 RobotMain

The class RobotMain represents a single agent and contains the main sense-act loop. Upon startup, RobotMain configures its behaviours, connects to the Player server, and begins the sense-act cycle. RobotMain reads its configuration from a properties file, which contains information about which behaviours are present, and attributes for the behaviours. The only top level behaviour contained in RobotMain is BehaviourShell. As the name suggests, BehaviourShell does not generate any actions on its own: it serves only to hold the actual behaviours which control the robot. When the robot has an opportunity to act, RobotMain calls a single method in BehaviourShell. BehaviourShell then consults all the behaviours contained in it, and returns an Action to RobotMain. This action is normalized by ensuring that the speed and turn rate contain reasonable values. RobotMain then executes that Action by sending the appropriate commands to the Player server.

4.5.3 Experiment Coordinator

In order to effectively run experimental trials, I found it necessary to have an *Experiment Coordinator* oversee each trial. Each trial consists of many iterations. The Experiment Coordinator is necessary to synchronize the robots from one iteration of a trial to the next. It also serves to collect various statistics about the trials, much like as human would need to do if running experiments using physical robots. When an experimental trial begins, all agents open a TCP/IP connection to the Experiment Coordinator, and when all robots have connected, the trial can begin. At the beginning of each iteration, the Coordinator must find random starting positions and orientations for each robot and a random position for the goal.

When the Experiment Coordinator finds a random starting position for an agent, it must make sure the location is suitable for the agent. Without this, an agent may end up inside a wall, causing it to stay stuck for the entire iteration. To find a suitable location, the Experiment Coordinator tests possible locations by using a small object equipped with eight sonars spaced evenly around the outside of the device (spanning 360 degrees), in order that the distance between the robot placement point and any obstacles in the simulated world can be measured. If the location testing device is at least 45cm away from the nearest obstacle, the location is considered to suitable. This artificial device is used rather than the robot itself because the Pioneer robots have sonars spanning only 180 degrees, leading to the possibility of an impossible placement (e.g. the back portion of the robot, not covered by sonar, being embedded in a wall).

With a suitable location found, the Experiment Coordinator transmits the coordinates of the chosen location, in Stage's coordinate system (which uses 0,0 in the bottom left hand corner of the world), and a random heading to the agent. At this point, the robot has been informed of its new location, and has been moved to that location in the simulated environment. However, these new coordinates have no direct meaning to the agent, because as discussed above, agents do not share the same coordinate system as Stage. The agent must thus update its own odometry to reflect its new position and heading, in its own coordinate system. It does so by rotating and then translating the coordinates and heading received from Stage into its own coordinate system.

In my implementation this is a reasonably straightforward operation, as all agents use an x,y coordinate system to record information about their worlds (the origin and units will differ across agents, but the same basic representation scheme is used). Agents may have more widely disparate representations for coordinates (e.g. a radian angle and a straight line distance from some origin): whatever the implementation, a transformation routine must be provided. If coordinate transformation is not done, the agent will not be able to localize correctly, making all existing grounded locations useless. When this process is complete, the agent signals the Experiment Coordinator that it is in position, and the Experiment Coordinator can then move on to the next agent. When all agents have been positioned for the iteration, the Experiment Coordinator finds a random position for the goal. As with the agents, the goal must be at least 45cm away from all obstacles in order to ensure it is accessible to the agents. After this is done, the Experiment Coordinator sends a message to all agents, telling them to begin the iteration.

While the iteration is running the Coordinator is passive: it only monitors messages which are broadcast in the environment (for statistical purposes). An agent completes an iteration by either finding the goal, or by giving up after ten minutes of searching. The goal is a special marker known to the agents. When an agent detects the goal within 75cm of its position it considers the goal found, and signals the Experiment Coordinator that it has completed the iteration, along with its list of labelled locations. After an agent has finished an iteration, it asks the simulator to move it outside the environment, where it awaits the next iteration.

4.5.4 Behaviour Characteristics

The superclass for all agent behaviours is called Behaviour. This class defines things which all behaviours have in common. The central method of this class is *getAction(Action)*, which returns an *Action* (see Section 4.5.1) which the behaviour wishes to perform. The getAction() method is called for each behaviour every acting cycle. There is a priority associated with each behaviour, in order that some can be given more weight than others. Also, there are methods to access data from the robot's hardware: the sonar, laser range finder, and wheel encoders.

In addition to the core functionality above, there are various utility methods. For example, there is a *log()* method which allows behaviours to log diagnostic information. The Behaviour class also has a name variable, and the log method prefixes its messages with the behaviour's name, which is useful for debugging.

The Behaviour superclass contains one more important ability. It permits one behaviour to register an interest in events generated by another behaviour. A behaviour generates an event whenever another behaviour may be interested in something it has noticed. For example, the collision-avoidance behaviour generates an event when it has to take extreme measures to prevent a collision. The wander behaviour receives this event, concludes the current heading is not useful, and does not attempt to return the robot to the heading which nearly caused a collision.

4.5.5 Behaviours

Section 3.4 has provided an overview of the behaviours necessary to develop grounded communication, including behaviours for navigation, mapping, and agent interaction. Several of these behaviours require some explanation at the implementation level, and those behaviours are elaborated upon here.

DontCrash prevents the robot from hitting obstacles by monitoring the distance values reported by the sonars. If the robot gets within 3m of an obstacle, DontCrash begins to gently turn the robot away from the obstacle. The turn rate is computed by first computing how fast the left sonars would like to turn the agent away from obstacles on the left side, and combining that with how fast the right sonars would like to turn away from obstacles they detect. This causes the agent to naturally drive down the middle of hallways, and pass halfway between two obstacles. If this behaviour detects an obstacle within 20cm, it decides that drastic action is necessary, and enters bail out mode. It signals this by generating a EVENT_BAIL_START event. The agent randomly decides if it should turn left or right while bailing out, and for how long it should bail out. These randomizations serve to keep the agent from becoming stuck in areas of the environment where it is difficult to navigate. While the agent is bailing out, it generates actions which cause the agent to turn sharply and backup slightly. When the bail out cycle is complete, the agent sends out a EVENT_BAIL_END event and resumes normal operation.

SelfPoseTracker is responsible for maintaining information about the robot's position, orientation, and velocity. Whenever any of these change, SelfPoseTracker generates an EVENT_POSE_CHANGED event containing the updated information. All of the information this behaviour tracks is read from the robot's wheel encoders. The robot's wheel encoders offer position resolution down to the millimetre level. Since maintaining an occupancy grid at this level of detail would consume too much memory, SelfPoseTracker divides all x and y coordinates by 100 before reporting them to others.

WorldResizer is responsible for maintaining information about the size of the environment explored so far, as explained in Section 4.2.2. Agents start with a small world (20x20), and expand that notion as SelfPoseTracker reports values bigger or smaller than has been seen before. If SelfPoseTracker reports an x coordinate value that is bigger than anything seen before, WorldResizer records this value and sends out an event, notifying others that the world size has changed. As an efficiency measure, the world size is not expanded by a minimal amount to accommodate the new minimum or maximum. Rather, the world size is increased by 5% along the axis on which SelfPoseTracker has reported a new minimum or maximum. The event which WorldResizer generates depends on which axis needed to be extended, and in which direction. It is one of: EVENT_WORLD_RESIZE_NORTH, EVENT_WORLD_RESIZE_SOUTH, EVENT_WORLD_RESIZE_EAST, or EVENT_WORLD_RESIZE_WEST.

SayMessage is used by other behaviours to broadcast messages into the environment. A message is broadcast by sending it to the Stage simulator, which later broadcasts it into the environment for all agents to hear. Since Stage does not guarantee messages that robots attempt to say will make it into the environment due to interference, robots must listen to make sure they actually hear messages that they have attempted to say. Other behaviours that wish to broadcast a message into the environment can instantiate SayMessage, and let it take care of the details of message verification and retransmission. When SayMessage is given some text to broadcast, it attempts to broadcast it on the next acting cycle. Then it waits a random amount of time, between 250ms and 750ms, to hear that message broadcast into the environment. If it does not hear the message it attempted to broadcast, it will try again. SayMessage continues this cycle until is hears the message or ten broadcast attempts have been made. If ten attempts are made, and the message still has not been heard to confirm its broadcast, SayMessage gives up and returns failure.

GoalSeeker is able to detect the goal in the environment and direct the agent towards it. Upon detecting the goal, it also informs other agents of the goal's location. The goal takes the form of a fiducial marker, which is detectable by the robot's laser range finder. Recall that there are two different formats for broadcasting goal information. In the first form, the agent broadcasts the name of the grounding which is closest to the detected goal location, that has a reference count of at least two.

There is no point in broadcasting the name of a grounding that has a reference count of only one, since no other agent knows that grounding. Agents that hear the message and know the broadcast grounding are then able to plan a path and drive to the grounding's location. In the second form, a symbolic reference is used to indicate the goal's location in terms of two existing groundings. The symbolic reference is specified as (p_1, p_2, a, d). The indicated position is found by first drawing a reference line from p_1 to p_2, then starting at p_2 travel d times the distance from p_1 to p_2, at angle $a \times 360^o$ to the reference line. This is similar to the symbolic reference used by Jung and Zelinsky [2000], shown in Figure 2.1.

When goal seeker first detects the goal, or detects the goal at a different location than it has previously, it sends out a EVENT_GOAL_INFO_CHANGED event along with information about the goal's location. When an agent gets within 75cm of the goal, it considers itself to have arrived at the goal, and signals this with an EVENT-_ITERATION_OVER event.

TrialRunner is the agent behaviour which oversees the running of the experimental trial. When it detects an EVENT_ITERATION_OVER event from GoalSeeker, or the agent has failed to find find the goal after ten minutes of searching, TrialRunner ends the iteration. TrialRunner then transmits the time it took for the agent to complete the iteration and its set of grounded locations to the Experiment Coordinator. After this transmission, TrialRunner removes the agent from the environment to await the beginning of the next iteration. When the Experiment Coordinator signals the agent with starting coordinates for the next iteration, TrialRunner moves the agent back into the environment, at the specified coordinates. It then updates the robot's odometry to reflect its new position, resulting in perfect localization at the beginning of each iteration. Since the coordinates received from the Experiment Coordinator are in the Stage coordinate system, they must be converted into the agent's coordinate system. To support this, TrialRunner records the initial rotation and translation of the agent's coordinate system when the simulation begins. This information can then be used to rotate and translate the coordinates provided by the Experiment Coordinator into the agent's coordinate system.

GotoFixedLocation is the behaviour which encapsulates path planning and following. When another behaviour needs a path planned, it provides the coordinates of the destination as input, and plans a path from the robot's current position to those coordinates as described in Section 4.2.2.

GotoLocation also has the ability to follow paths it has planned. Since all nodes are connected by straight lines, driving from one to the next is a fairly straight forward task. When an agent navigates within 25cm of a waypoint, it begins to drive to the next. However, due to error in sonar used to create the occupancy grid, accumulated error in the robot's position, and interference from other robots, obstacle avoidance from DontCrash may become active. If DontCrash needs to enter bail out mode to prevent crashes this distance is increased in increments of 5cm until it reaches 50cm. If an agent can not get within 50cm it has failed to follow the path and abandons the path.

GotoFixedLocation generates many events that are convenient for GUI support. A EVENT_QUADTREE_NEW is generated when a new quadtree is under construction. When a new path is planned through the quadtree an EVENT_QUADTREE_ROUTE is generated. When this behaviour receives instructions to begin following the last path it has planned it generates an EVENT_GOING_TO_LOC. Each time the robot reaches a node (waypoint) on the way to its destination, it generates an EVENT-_ROUTE_WAYPOINT. If path following fails an EVENT_DEST_CANCELLED is generated. When the agent successfully arrives at its destination, it sends out an EVENT_ARRIVED_AT_LOC event.

MapEnvironment is the behaviour which maintains the occupancy grid using sonar data (described in Section 4.2.2). MapEnvironment makes no assumptions about the world size. When this behaviour is initialized, it gets the initial world size from WorldResizer, and creates is 2-dimensional array of cells accordingly. When WorldResizer generates an event saying the world size has changed, MapEnvironment responds by allocating more memory to contain the expanded occupancy grid in the appropriate direction. When a cell's status changes, MapEnvironment generates an EVENT_CELL_STATE_CHANGED: this is necessary for a GUI to monitor the robot's exploration.

GotoMovingTarget is used by the LocationExchanger behaviour. When one agent is demonstrating a grounding to a second agent, the second agent must track the first while it drives to the location to be demonstrated. GotoMovingTarget target handles this tracking.

RandomWander picks a random heading and directs the agent to follow it for 20 to 40 seconds. When time is up a new heading is selected for the agent to follow. If the selected heading directs the agent into an obstacle, causing DontCrash to invoke its bail out mode, the current heading is abandoned, and the robot runs strictly on obstacle avoidance until it is time to select a new heading.

LabelAtMeeting implements the label-at-meeting labelling strategy described in Section 4.3.

LabelSpatialEntropy implements the label-spatial-entropy labelling strategy described in Section 4.3.

LabelEnvironmentFeature implements the label-environment-feature labelling strategy described in Section 4.3.

This chapter has outlined the implementation of my approach to developing grounded communication described in Chapter 3. It discussed the Player/Stage simulator, the location grounding behaviours, how locations are shared between agents, and how the conflicts that arise as a result of that exchange are resolved. This chapter has also discussed the implementation of the agent behaviours. The following chapter will present the experimental results and the analysis of those results.

Chapter 5

Evaluation

This chapter presents an evaluation of my approach to developing grounded communication. It begins by describing the experimental domains and the various trial configurations for my experiments. It then describes the metrics used to evaluate my results, followed by the results themselves. The chapter ends with an analysis of the results.

Since the purpose of the evaluation is to examine the effect of grounded communication as it is developed over time, the experimental scenarios must have some element that the agents can communicate about. In this evaluation, this element is a goal location that, once found, can be communicated to other agents. Each agent starts at a random location, with a random heading, and has no information about the goal's location. An agent must find the goal by exploring the environment. While an agent is searching for the goal, it creates groundings for what it deems to be worthwhile locations, using the strategies described in Chapters 3 and 4. When one agent encounters another, it can demonstrate a grounding, in order to create a shared grounding between the two agents. When the agent does find the goal, it signals the goal's location to other agents using the grounding(s) it has established. The standard way of doing this, described in Section 3.1.1, involves broadcasting the name of the location it has grounded nearest to the goal. Other agents that share this grounding can use this information to aid in their search for the goal. If an agent has not found the goal after 10 minutes of searching, it gives up. After all agents have found the goal (or given up), one iteration of a trial is complete. During this iteration, agents collectively created a set of groundings which should be useful to improve their performance in this environment in future. Moreover, these groundings should become more extensive and more consistent as agents gather more experience in this environment. In order to examine this effect, a full experimental trial consists of a series of 200 of these iterations, where agents maintain their groundings between iterations and are placed in new random locations (with a new, randomly located goal) for each iteration. Experimental trials were conducted in many different configurations, as discussed in the following section.

5.1 Experiments

5.1.1 Experimental Configurations

As described in Chapter 3 and Chapter 4, there are a number of different possible variations in strategy that need to be examined to evaluate my approach. There are also a number of parameters to my grounding strategies that can affect results (e.g the epsilon distance for deciding location similarity), as well as a desire to examine the effects of this approach in domains of different sizes and structures, and with different numbers of agents. This section outlines the combinations of factors were used to direct my experimentation. These trials varied the strategy individual agents used to decide when a location was worth grounding, the number of agents, and the size and configuration of the environment. Beyond these trials to establish baseline results, additional trials were run that varied how eager agents were to share groundings with one another, and how much distance can be between two locations while still being considered equal by an agent (ϵ). Each experimental trial consists of 200 iterations.

Grounding Strategies

I tested three different grounding strategies, described in Sections 3.2.3 and 4.3: label-at-meeting, label-spatial-entropy,and label-environment-feature. The purpose of testing different grounding strategies was to determine whether the conditions under which individual groundings were created influenced the performance of the general technique.

Number of Agents

This evaluation also examined the development of grounded communication using different numbers of agents. Trials were run with 2, 4, 8, and 16 agents. This was intended to examine the scalability of the technique, and in conjunction with varying the environment size, the effect of overcrowding and resulting agent interference in an environment.

Experimental Environments

The experimental domains I tested can be divided into two broad categories based on their size: 8x8m and 11x11m. The purpose of using two different environment sizes is to determine the scalability of the technique in terms of the size of the space involved, as well as under varying agent populations. In addition to the size of the environment, there is also the issue of environmental complexity: a smaller environment with many interesting features would likely result in completely different performance than a large open environment. To examine this, I set up four basic environment configurations, each with an 8x8m and 11x11m version. The four basic environment configurations are: *office* (intended to represent a sample office environment with a few simple walls, as shown in Figures 5.1 and 5.2), *hallway* (a partitioned environment with hallway openings, as depicted in Figures 5.3 and 5.4), *split* (an environment with barriers that more strongly restrict travel from one end to the other, shown in Figures 5.5 and 5.6), and *open* (a purely open environment for purposes containing no obstacles other than the robots themselves, shown in Figures 5.7 and 5.8). The small circles visible in the figures are the markers for use by the label-environment-feature grounding strategy (see Section 4.3), to provide an indication of the domain-specific features that were decided to be worth grounding. Experimenting with different environmental configurations allows testing across a broader range of conditions, ensuring the technique or some portion of the robot's implementation is not biased toward one particular type of environment. It also serves to uncover situations where communication may be more useful than others. For example, the results of my experiments show that communication is more valuable in environments

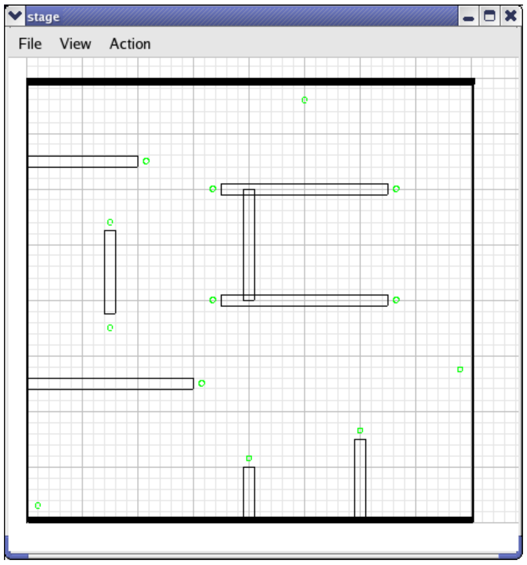

Figure 5.1: The office 8x8m environment.

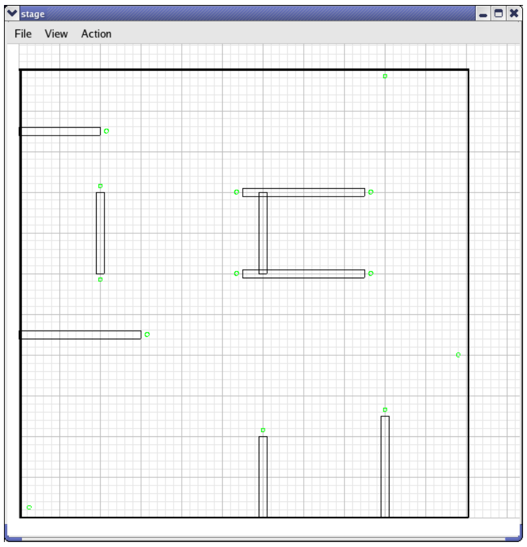

Figure 5.2: The office 11x11m environment.

where the goal is harder to find with random searching. In the open environments, which contain no obstacles, it is much easier to find the goal by random exploration than in environments that contain significant obstacles.

Baseline Experiments

Each combination of the variations described above was tested in order to gather baseline results. That is, all eight environments (Section 5.1.1) were examined with agent populations of of 2, 4, 8, and 16 agents using each of the three different individual grounding strategies. The process of gathering this baseline data was both time-consuming and computation-intensive. One full trial of 16 agents in one environment, for example, took six days using 2 computers. Because of the substantial time required for data-gathering, additional experiments varying baseline parameters (described in the next subsection) were confined to a subset of these domains.

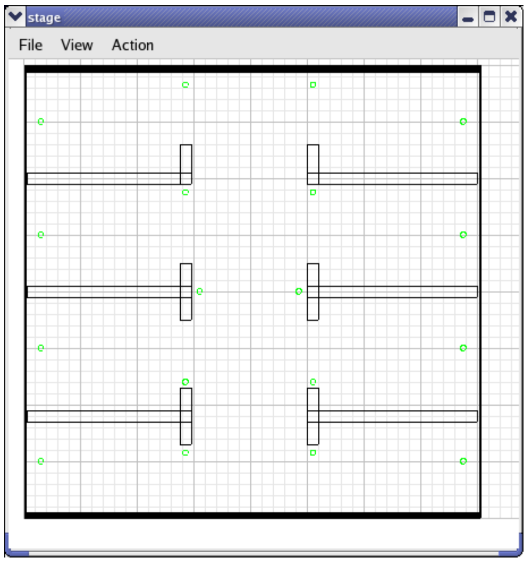

Figure 5.3: The hallway 8x8m environment.

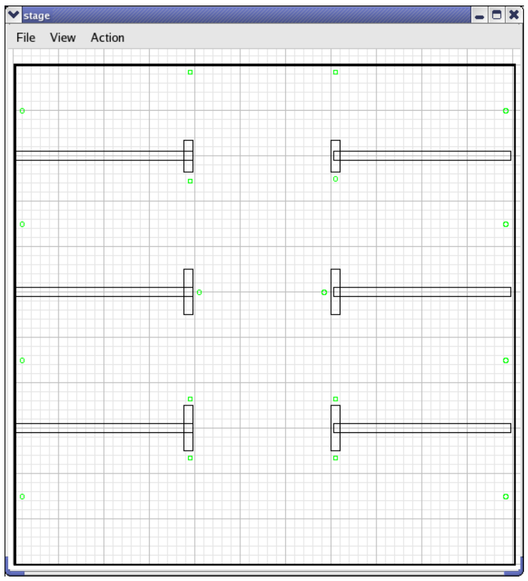

Figure 5.4: The hallway 11x11m environment.

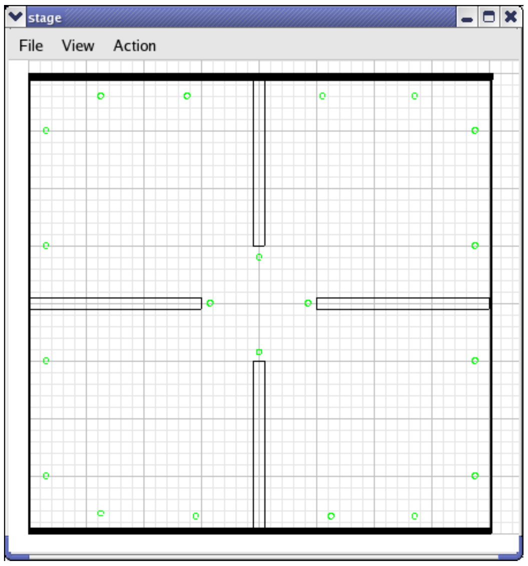

Figure 5.5: The split 8x8m environment.

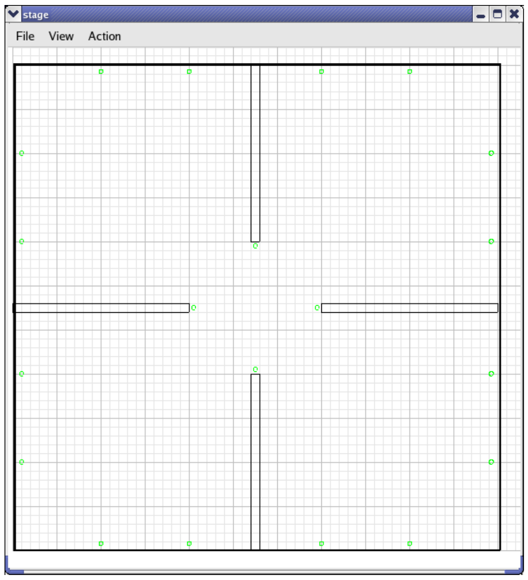

Figure 5.6: The split 11x11m environment.

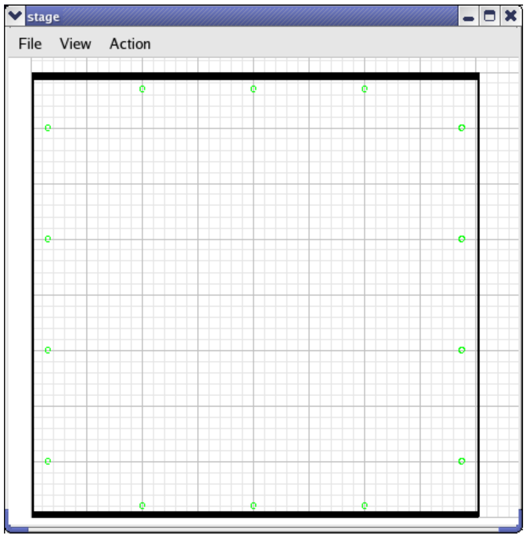

Figure 5.7: The open 8x8m environment.

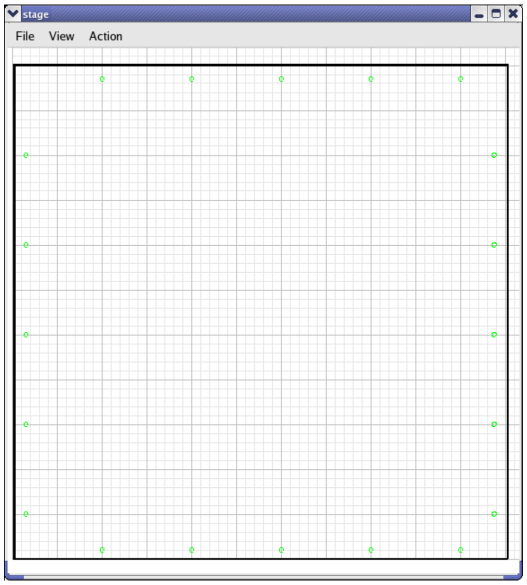

Figure 5.8: The open 11x11m environment.

5.1.2 Further Experiments

Beyond the basic experiments described above, I ran some additional experimental trials to examine the effects of changing factors that could affect the utility of these grounding strategies. In the first of these, I examined the effect of changing the distance at which one agent will attempt to demonstrate a grounding to another. In the experiments described earlier, a value of 5m was used, while in these further experiments I tested a lower value of 2m. The second variation tested was the distance at which an agent considers two locations to be equal: the ϵ parameter. Initial experiments used a value of 50cm, subsequent experiments used a low value of 20cm, and a high value of 100cm.

The third variant tested in the additional experiments involved attempting to explore a more complex use of grounded communication, through symbolic references.

The symbolic reference I used specifies the goal's location in terms of two other grounded locations. If the two groundings used to specify the goal's location match exactly, the goal's location will be specified exactly, since the distance and orientation from two different grounded points will allow a precise extrapolation. If the two groundings do not match exactly, there will be some error. Jung and Zelinsky [2000] found that a symbolic reference improved the performance of their system, and I wanted to see if the same was true for my technique. The implementation of this technique involves the use of the GoalSeeker behaviour, described in Section 4.5.5.

5.2 Metrics

In any evaluation, some standard metric must be used to form a basis for comparison. In my work, the most important metric for the purposes of comparison is time. I define the performance of the approach during a trial as the average amount of time it takes for an agent to find the goal across all iterations. Since each trial involves 200 iterations of up to 10 minutes each, a trial thus amounts to a reasonably long-term average performance for a set of agents developing grounded communication. In early iterations, there will be few grounded points to which agents can refer, and most agents will have to discover the goal through their own exploration rather than through useful communication, or will fail to discover the goal at all. In later trials, after a set of shared locations has been established, more useful grounded communication can take place, and the agents should be able to find the goal through communication and path planning much of the time. The long-term average thus covers both the successful use of grounded communication and the learning curve during which these groundings are developed.

All of the experimental configurations described in Section 5.1.1 are compared against a control group using the same configuration but with the ability to communicate turned off. This prevents agents in the control group from creating shared groundings, and also prevents them from communicating the goal location (which would be useless anyway, since there are no shared groundings through which to understand this communication). The measure by which the performance of the experimental configuration exceeds the respective control group is then expressed as a percentage *improvement*.

In addition to the improvement in performance measured by time, I also wished to measure how consistent the set of groundings between all agents ultimately becomes under these techniques. I calculated a *global grounding consistency* metric to measure this. This metric is computed as follows.

The *global grounding set* is an aggregate set of all groundings across all agents in the system. Associated with each *global grounding* in the global grounding set is a reference count, rc_{real}. rc_{real} is the actual reference count for a particular grounding. That is, it is the number of agents that actually know a particular location by the same name. rc_{real} is different from the reference count of an individual agent, as an individual agent's reference count for a grounding is subjective and error prone (see Section 3.3).

The global grounding set and the rc_{real} associated with each of the individual global groundings are built by processing each agent's set of groundings one at a time. For each agent, each grounding is examined in turn. If the name of the current grounding is not used by any existing global grounding, a new global grounding is created with an rc_{real} of 1: that is, this new grounding is stored, and its use by other agents will be tallied as other agents' groundings are processed. If the name for the current grounding has already been encountered, and the current grounding is within 80cm[1] of the stored global grounding (i.e. the groundings match), then the rc_{real} associated with the matching global grounding is incremented. However, if the name for the current grounding is the same as an existing global grounding, but the current grounding is more than 80cm away from that global grounding, it is not considered the same for the purposes of examining consistency. In this case, a new global location with the same name is created, with a reference count of 1.

When all groundings in all agents have been processed, it is possible there may be multiple global groundings with the same name in the global grounding set. Such a name cannot be used to specify an unambiguous location in the environment. This a state of conflict, as it makes no sense for there to be groundings at different locations with the same name. To resolve this conflict, only the best global grounding is kept, (i.e. the one with the highest rc_{real}), and all other global groundings with that name are discarded.

With a consistent set of global groundings constructed, the global grounding consistency can now be computed. The grounding consistency for an individual agent can be computed by finding how many of that agent's groundings match with those in the global grounding set. An individual's grounding matches a global grounding if it has the same name as the global grounding, and it is within 80cm of that global

[1]The value of 80cm used to determine the global grounding consistency is constant. It is different from the variable ϵ parameter individual agents use to determine if grounds are equal.

grounding (this is the same value used for comparison purposes when compiling the set of global groundings). The number of matches is divided by the total number of global groundings to find the grounding consistency for that agent. The *global grounding consistency* is then the average of the individual global grounding consistencies across all agents, expressed as a percentage.

A global grounding consistency of 100% means that all agents have the same set of names grounded to the same locations (within 80cm). Other values than 80cm are possible for determining if two locations are equal. I chose a value of 80cm because it is twice the length of an agent.

Due to the way the metric is calculated, achieving a high global grounding consistency is much more difficult with large numbers of agents than with small populations. To illustrate this, consider a group of two agents vs a group of 16 agents, when there is only one global grounding in each group. The two agent set will have one agent with a 100% grounding consitency, and one agent with a 0% grounding consistency, producing an overall global grounding consistency of 50% ($\frac{100+0}{2}$). The 16 agent set will have single agent with a grounding consisteny of 100%, with the remaining 15 agents having a grounding consistency of 0%, producing a global grounding consistency of only 6.25% ($\frac{100+0+0+...+0}{16}$). This illustrates how a location that is grounded in only a single agent hampers global location consistency much more when there are many agents. Moreover, global grounding consistency is also a pessimistic measure in general because it throws away groundings with duplicate names, other than the most common use of that grounding. There may still be some consistency between agents (e.g. 50% of the population may use one particular grounding with a given name, while another 25% use a different grounding), but this consistency is not reflected in the final measure.

I also wished to measure the number of groundings created by each grounding strategy as a function of the size of the environment, to determine its influence on performance. To measure this, I defined a *grounding density* metric which is the number of groundings per square metre.

5.3 Results & Analysis

The graphs presented in this section are an aggregate of the four environment configurations of each size. That is, the bar in Figure 5.9 for the 2 agent label-at-meeting percentage improvement is the average for 2 agents using the label-at-meeting grounding strategy in the office 8x8m, split 8x8m, hallway 8x8m, and open 8x8m environments. The interested reader may consult appendix A to view the individual times and percentage improvement for each individual configuration.

While there is some variation between each environment configuration, the environments all exhibit the same pattern when viewed by number of agents. The 8x8m environments show an improvement from 2 to 4 agents, then again from 4 to 8 agents, and then a drop in performance from 8 to 16 agents. The reasons behind this and the other factors in this general pattern will be discussed in the sections that follow. The 11x11m environments show the same improvement as the 8x8m environments from 2 to 4 to 8 agents, but differ in the 16 agent case. The 11x11m environments generally show about the same level improvement when examining 16 agents as 8 agents.

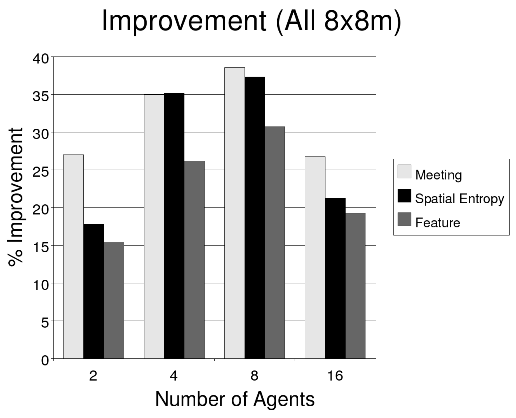

Figure 5.9: Average improvement by grounding strategy in 8x8m environments.

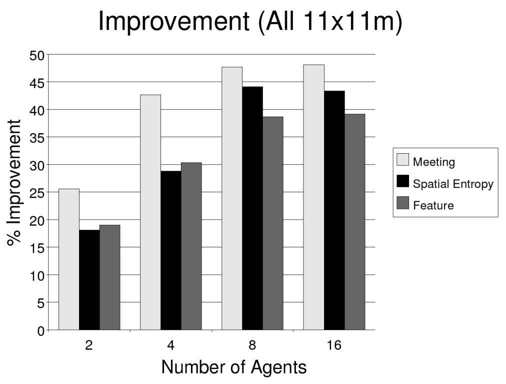

Figure 5.10: Average percentage improvement by grounding strategy in 11x11m environments.

5.3.1 Baseline Results

Analysis by Grounding Strategy

Figures 5.9 and 5.10 illustrate the performance increase resulting from the use of each of the grounding strategies described in Section 3.2.3. From these figures, we can see that all grounding strategies exhibit an observable performance gain over no communication. This is to be expected, since adding communication often improves performance in multi-agent systems. In this case, communication is providing information about the location of the goal. It is possible the signalled goal location is not actually near the goal, since an agent communicates its closest grounded location without regard to its distance from the goal. If agents do not share the grounding used to indicate the goal's position, the communication will not be helpful, and may even harm performance. The latter could occur when agents have, by chance, happened to independently create different groundings with the same name, and thus an agent may be directed away from the goal rather than toward it. This is unlikely to occur however, as agents randomly select an integer in the range 0 to 10000 as the name for a new grounding.

Also from Figures 5.9 and 5.10, it is obvious that the label-at-meeting strategy has the largest performance improvement, especially in the 11x11m environments. I attribute this mainly to the implicit sharing between two agents when a grounding is first created. This leads to a high global location consistency, which makes communication more effective: agents are more likely to share groundings, which allows for more accurate communication about the location of the goal. This is especially relevant with two agents, where a grounding is shared across the entire population

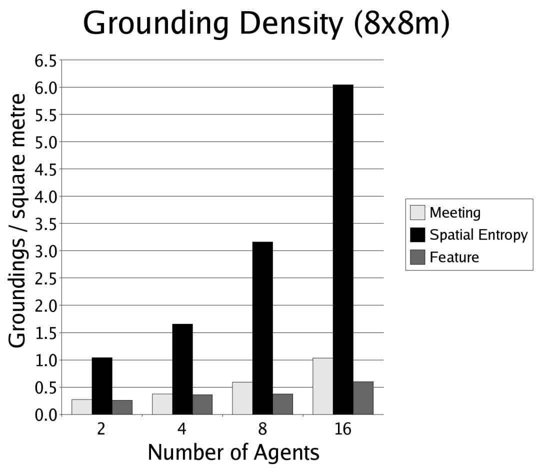

Figure 5.11: Average grounding density by grounding strategy in 8x8m environments.

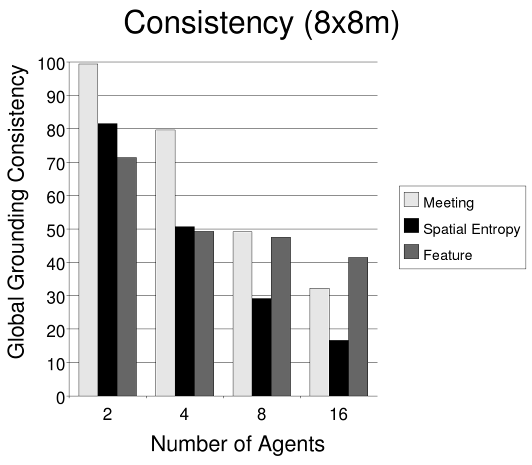

Figure 5.12: Average global grounding consistency by grounding strategy in 8x8m environments.

when it is created. With more agents this becomes less of an advantage, as a newly created grounding is shared across a much smaller percentage of the population. Label-at-meeting grounds locations where agents happen to meet. This leads to a set of grounded locations that are naturally distributed across the environment. For the task used to evaluate agent performance, good coverage of the environment is important. Full coverage allows an agent to more accurately indicate the location of the goal to other agents. The label-at-meeting grounding strategy may have not have been as effective if the grounded locations were used for another task, such as common waypoints for navigation.

The technique which showed the second highest performance improvement was label-spatial-entropy. This is despite the fact that it grounded far more locations than the other two strategies, especially with 16 agents (Figures 5.11 and 5.13). Despite the increase in grounded points, this technique still grounds far fewer than that

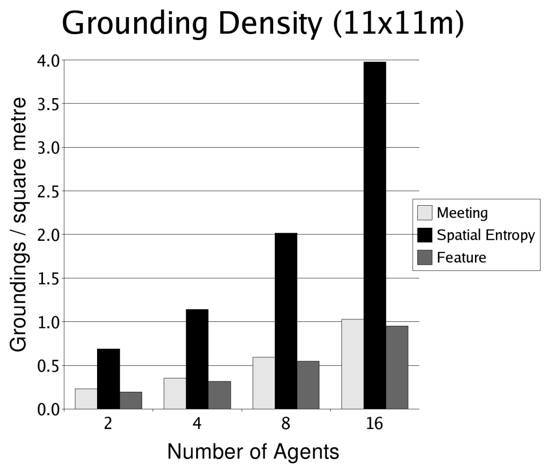

Figure 5.13: Average grounding density by grounding strategy in 11x11m environments.

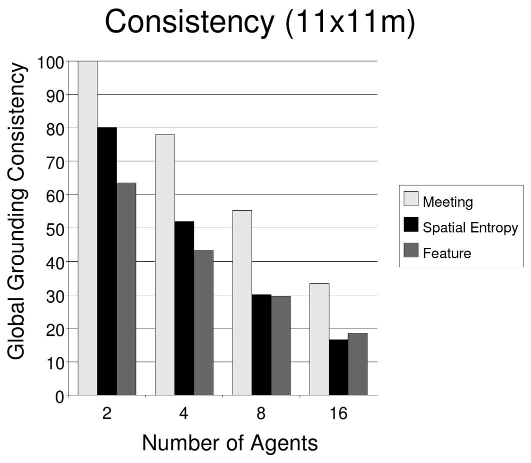

Figure 5.14: Average global grounding consistency by grounding strategy in 11x11m environments.

of Jung and Zelinsky [2000], especially when one compares the number of groundings for only two agents (Jung and Zelinsky [2000] only demonstrated their approach with two agents). Also note that when using label-spatial-entropy, the number of global locations grows by a larger factor as the population of agents increases, compared to the other two grounding strategies (Figure 5.11 and 5.13). For 2 and 4 agent populations, agents are able to exchange enough locations to maintain a reasonable level of consistency, relative to the other grounding strategies. However, with 8 and 16 agents the global grounding consistency begins to suffer relative to the other techniques (Figures 5.12 and 5.14). This is due to the larger number of locations grounded by label-spatial-entropy: there are simply too many groundings for the agents to reconcile, which leads to fewer shared groundings, and a lower global grounding consistency. This could possibly be corrected using a higher threshold for spatial entropy when deciding to make a new grounding.

It at first seems to be of some concern that random encounters should do better than an approach designed to mark useful locations. However, not only is there greater sharing built into the label-at-meeting strategy, there are some important considerations about the points that are being grounded as well. Spatial entropy causes the agents to want to label points that are interesting from the standpoint of navigation - those where the environment is a mix of free and blocked space. The grounded communication here, however, consists only of the goal location. In that respect, these points are no better than random locations. I anticipate that if an environment were to be set up where the grounded communication was more navigationally-oriented, this approach would be more successful. An example of this would be an environment where information about waypoints to a goal were communicated instead of the goal itself.

Label-environment-feature had the lowest overall improvement. This is a reflection of the fact that it has too few locations available for it to ground. The grounding density for label-environment-feature grows the slowest as the number of agents increases. While having few locations to ground seems like it should encourage a high global location consistency, it also limits opportunities for one agent to demonstrate an already grounded location to another agent. When one agent does meet another in a sparsely grounded environment, it is less likely that there is a nearby grounding which one of the agents can demonstrate to the other.

The performance for label-environment-feature drops off with larger numbers of agents compared to label-spatial-entropy. This is likely due to label-environment-feature having a harder time demonstrating groundings than label-spatial-entropy.

The markers used to represent features were placed near walls and corners in order not to be in the way of navigation in open areas (the markers themselves are physical and agents cannot pass through them). Placing these near corners and walls makes it difficult to navigate near them (because these areas are themselves hazards to navigation). Preliminary trials had these markers placed in the walls, but the obstacle avoidance behaviour made it even more difficult for an agent to navigate to these locations when demonstrating a grounding to another agent. The markers were thus moved slightly out from the walls for the final tests. Because the agent cannot occupy the same space as the marker, it also cannot demonstrate the precise location of the marker, so an agent will not be able to demonstrate the precise location of an individual grounding it has created.

While label-at-meeting and label-spatial-entropy continue creating new individual groundings for locations over an entire trial, label-environment-feature works differently. Since label-environment-feature has been implemented to ground only specific markers, and each agent grounds each marker only once, it would be expected that eventually all agents would ground all markers. Once this has occurred, no new groundings will be introduced. This should cause the global location consistency to begin to rise, as agents share and reconcile their existing groundings. As a result, the global location consistency should rise to 100%[2]. The location consistency graphs show that this does not occur. This is due to an error in the implementation of my approach, which affects all grounding strategies, but is most noticeable with label-

[2]Or close to 100%. While global grounding consistency is a pessimistic measure in that it throws away global groundings other than the majority in situations where identicle labels are grounded to disparate locations, this situation uses relatively few groundings. The liklihood of these being grounded to different symbols across a population is not high.

environment-feature. The error occurs when r_2 knows the name of the grounding which r_1 is demonstrating, but has the name that r_1 is using for the demonstrated location grounded to a different location than the one r_1 is demonstrating (this conflict was described in Section 4.4.3). When r_2's reference count for its grounding is higher than the reference count for the grounding r_1 is demonstrating, r_2 should respond with a LOC_FORGET message, causing r_1 to forget its current grounding for the demonstrated location, as it is inconsistent with, and less well known than r_2's grounding of the same name. However, in my implementation r_2 responds with LOC_OVERRIDE message, causing r_1 to adopt r_2's name for r_1's current location, while r_2 maintains its existing grounding for the name, at a separate location. The result is that r_1 and r_2 leave the exchange with incompatible groundings for the same name, which my approach was designed to avoid. From the standpoint of the effectiveness of my technique, this error allows inconsistencies to persist when they should be removed, and the results presented in this chapter would only be better if these studies were to be repeated with this error fixed.

Analysis by Number of Agents and Environment Size

An important factor that was expected to be demonstrated in the varying sizes of these domains and varying population sizes of agents was interference between agents. One agent interferes with another when its physical presence hinders the actions of the second agent. Other researchers have identified interference to be the most important limiting feature in multi-agent systems [Matarić, 1998; Arkin, 1998].

When there are many agents, and only a fixed number of locations that can be grounded (as in label-environment-feature) there will be competition for the physical space around those groundings, leading to increased interference. It would be interesting to see if label-spatial-entropy would also suffer from this problem if it had grounded fewer points. Where label-environment-feature can ground only certain precise points, label-spatial-entropy grounds the same kinds of locations, maybe even in the same general area, but likely not the same precise locations. I think this would decrease the amount of interference in label-spatial-entropy compared to label-environment-feature, at the same global grounding density. It would be interesting future work to attempt to confirm this.

Having many agents in a small area will cause agents to spend more time demonstrating locations to each other. While agents must wait 30 seconds between creating shared groundings, when there are several agents in a small area, as soon as those 30 seconds are up another agent is available to either demonstrate a grounding, or have a grounding demonstrated to it. This leaves comparatively little time left to actually perform exploration. If there is some portion of the environment in which it is difficult to navigate, a set of agents may become isolated, and end up demonstrating the same locations to each other repeatedly because it takes more than 30 seconds to successfully navigate out of the constricted area. This could be at least partly alleviated by giving each agent a memory of which groundings it has demonstrated to which agents. There would be some price to pay for this in scalability, since it would a significant overhead to do this for large numbers of agents. Another problem with having many agents in a small area is that it will be more difficult to physically demonstrate a grounding, as it will be harder to navigate to the position

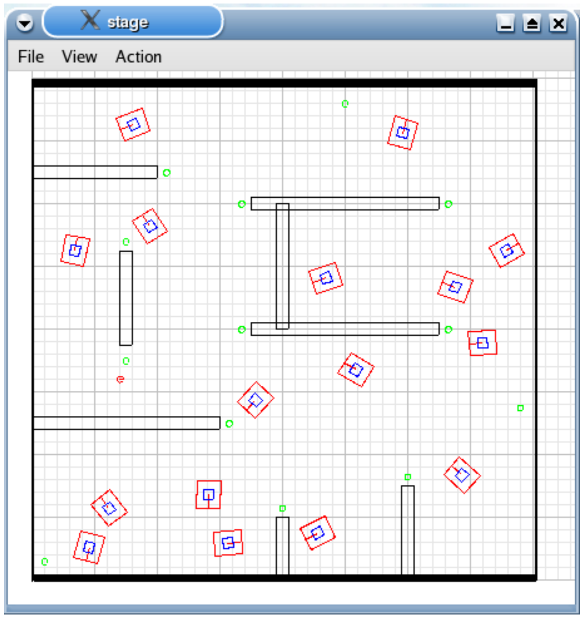

Figure 5.15: 16 agents in the Office 8x8m environment.

to be demonstrated due to interference from other agents. This occupies more of the agent's time, decreasing performance.

The most striking feature of Figures 5.9 and 5.10 is that performance increases from 2 to 4 to 8 agents, then drops off with 16. I believe this decrease in performance with a large number of agents is due to the agents interfering with one another. Referring to Figures 5.9 and 5.10, which show improvement in 8x8m and 11x11m environments respectively, a dramatic difference in improvement with 16 agents when going from 8x8m to 11x11m environments can been seen. I believe this indicates that an 8x8m environment is too small for 16 agents to operate in effectively. This is also intuitive from a screen shot (Figure 5.15) of 16 agents operating in such an environment. Further, the tiny performance improvement from 8 to 16 agents in 11x11m environments indicates that an even larger environment is needed in order to allow 16 agents to perform most effectively. A smaller number of agents may also show more improvement in a larger environment, due to decreased inference and the increased utility of goal information.

Performance generally increases with the number of agents, even though there is the potential for more interference, and the global grounding consistency suffers as the agent population increases (Figures 5.12 and 5.14), because agents have more opportunities to learn the goal's location from one another. When there are only two agents in the environment, at least 1/2 of the agents (i.e. 1) will have to find the goal by random wandering. When there are 16 agents, there exists the potential that only 1/16th of the agents will find the goal by random wandering. Once the first agent finds the goal by random wandering, it signals the goal's location to the other agents. Even if only some of the other agents know the broadcast grounding, they will make their way to the goal, and broadcast their nearest groundings (which may be different than the first agent's grounding) when they can directly sense the goal. This gives the agents who did not know the first agent's grounding another opportunity to hear a broadcast grounding which they know, and navigate towards the goal.

Figures 5.9 and 5.10 show a higher performance improvement for larger numbers of agents. I believe this is due to communication being more valuable in a larger environment, and reduced interference in the larger environment.

5.4 Variations

All variations were run with only 4 and 8 agent agent populations in split 8x8m and split 11x11m environments. This subset of the baseline experiments provides reasonable coverage while limiting the number of experimental trials that needed to be run. The split environment configuration was selected to test these variations because it showed a high average improvement in the baseline experiments, and it also had high absolute times. The office environment configuration showed slightly higher average improvement than split in both 8x8m (32.14% in office 8x8m vs 30.59% in split 8x8m), and 11x11m (36.90% in office 8x8m vs 36.19% in split 8x8m) configurations. The higher absolute times in the split environment make it more sensitive to the tested variations, and thus the better choice.

The results presented in the subsections that follow are also tabulated in Appendix B.

5.4.1 Varying Grounding Sharing Distance

In this experiment I varied the distance at which an agent could ask another to start a conversation in order to share a grounding. In the baseline experiments above, an agent could ask another agent to begin a grounding sharing session when it was up to 5m away. In these trials two other values were tested, 2m and 8m. For the 8m trial, the agents themselves were allowed to begin a conversation with another agent that was up to 8m away. However, the parameter within Stage defining how far away an agent could perceive another was inadvertently left at 5m. The result of this was that even though agents were free to ask for a demonstration at distances up to 8m, it was impossible for them to detect one another at distances further than 5m, and so the ultimate effect was to have a second set of data with 5m limit on demonstrations. As this data was the same as that from the 5m trials already discussed (which further

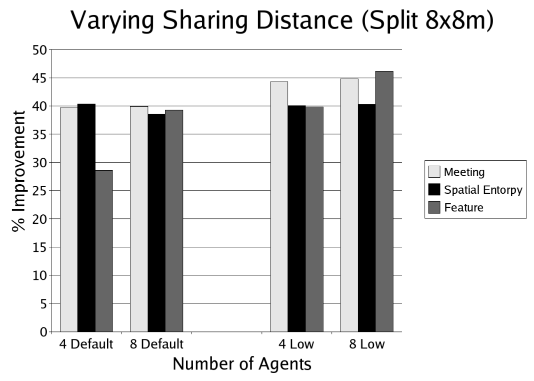

Figure 5.16: Average improvement by grounding strategy and agent populations (4 and 8) comparing the default grounding sharing distance to the low grounding sharing distance in split 8x8m.

serves to confirm these numbers), the 8m data is not presented. The remainder of this section describes the results of this experiment, looking at each category of environment separately.

Figure 5.16 shows that in the split 8x8m environment, the label-at-meeting and label-environment-feature strategies show more improvement with a low grounding demonstration distance, while label-spatial-entropy performance does not. I believe this is due to the large number of groundings created by the label-spatial-entropy grounding strategy, combined with the size and configuration of the split 8x8m environment (Figure 5.5). Agents in the same "quadrant" of the environment are likely to come close enough to encounter one another and share a grounding, as each quadrant is quite small. This will cause an agent using the label-spatial-entropy to become involved in many grounding exchanges. However, the label-at-meeting and label-environment-feature strategy have far fewer groundings available to demonstrate than label-spatial-entropy (Figures 5.18 and 5.19). Thus, even if two agents come close

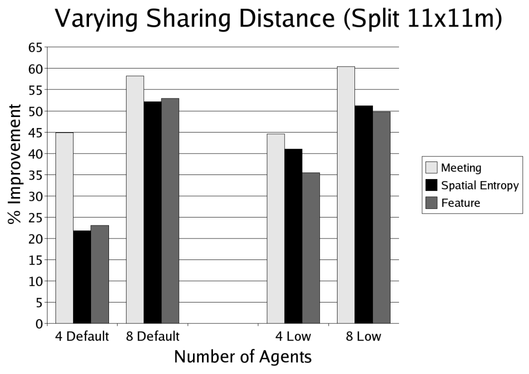

Figure 5.17: Average improvement by grounding strategy and agent populations (4 and 8) comparing the default grounding sharing distance to the low grounding sharing distance in split 11x11m.

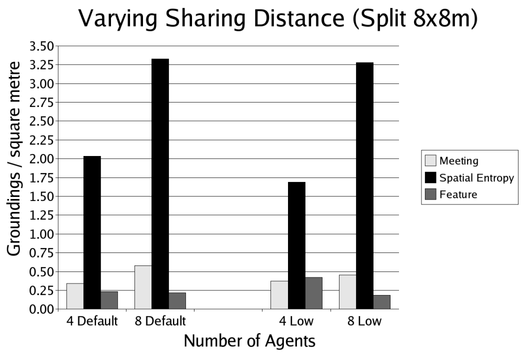

Figure 5.18: Average grounding density by grounding strategy and agent population (4 and 8) when varying the grounding demonstration distance in split 8x8m.

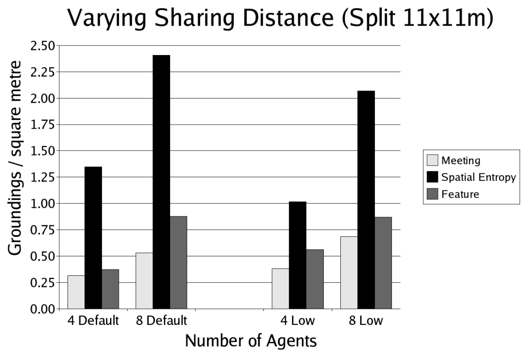

Figure 5.19: Average grounding density by grounding strategy and agent population (4 and 8) when varying the grounding demonstration distance in split 11x11m.

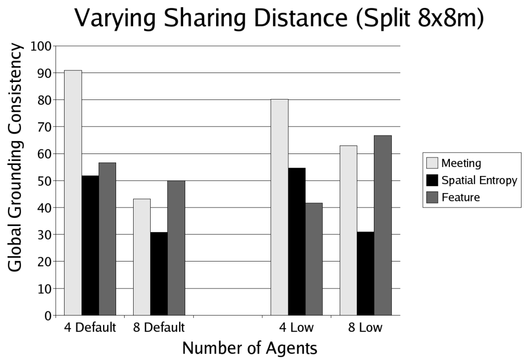

Figure 5.20: Average global grounding consistency by grounding strategy and agent population (4 and 8) when varying the grounding demonstration distance in split 8x8m.

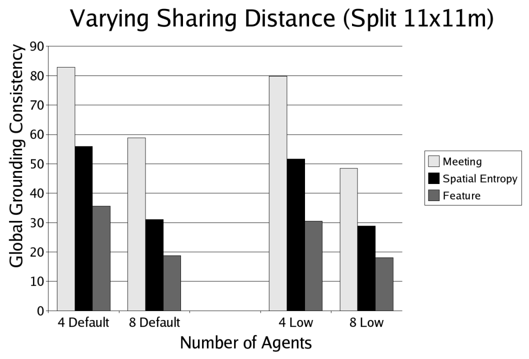

Figure 5.21: Average global grounding consistency by grounding strategy and agent population (4 and 8) when varying the grounding demonstration distance in split 11x11m.

enough to each other to begin sharing a grounding, the agents are much less likely to know a nearby and already grounded location that they can demonstrate to the other.

Figure 5.17 shows that the label-at-meeting grounding strategy is not affected by the grounding sharing distance in the split 11x11m environment, while label-spatial-entropy and label-environment-feature both show higher improvement with 4 agents, and the same improvement with 8 agents. The performance of the label-spatial-entropy and label-environment feature strategies can be explained by interference. Where interference limited the performance improvement of the lower grounding sharing distance in both the 4 and 8 agent case in the split 8x8m environment, the split 11x11m environment is large enough to decrease interference, and allow the 4 agent low grounding demonstration distance to show an observable improvement over the higher default grounding demonstration distance.

Similarly, the label-environment-feature labelling strategy showed a relative improvement when lowering the grounding sharing distance with 4 agents, but no change when examining 8 agents. Again, I believe this is due to the decreased interference in the 4 agent configuration, while the 8 agent configuration still contains enough interference to curtail any relative improvement.

Examining the global location consistency in the split 11x11m environment (Figure 5.21), it can be seen that global location consistency decreases slightly when using the low grounding sharing distance, compared to the default grounding sharing distance, for all grounding strategies and numbers of agents. This is to be expected, as the less likely agents are to reconcile their groundings, the less consistent they are likely to be. This holds for 4 agents in split 8x8m environment (Figure 5.20), but the global location consistency *increases* with a lower grounding demonstration distance for the label-at-meeting and label-environment-feature strategies. I believe that this is due to decreased interference in these circumstances. When an agent demonstrates a grounding, it employs the Goto Fixed Location behaviour (see Section 4.5.5), which may place an agent up to 50cm away from the desired location, if the agent is having trouble navigating. This causes greater error in location demonstrations, leading to a lower overall grounding consistency.

5.4.2 Varying Tolerance of Location Equality

In this experiment, I examined the effect that changing the ϵ parameter to the grounding strategies had on the performance of these techniques. This parameter represents the tolerance associated with whether two locations are considered equal: a high ϵ parameter will allow a greater physical disparity while still considering two locations equal. This ϵ parameter is referred to colloquially here as the *location*

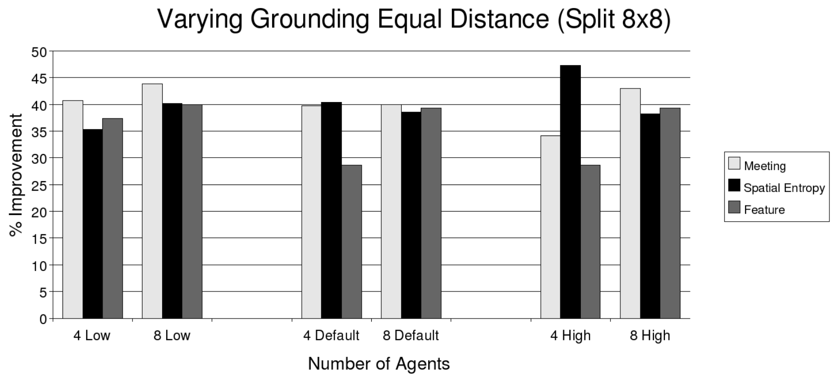

Figure 5.22: Average improvement by grounding strategy and agent population (4 and 8) when varying the tolerance of location equality inside an agent, in split 8x8m.

equality tolerance in order to remind the reader of its purpose.

In the baseline experiments this value was set to 50cm. This means that when an agent is demonstrating a grounding to another, the second agent will consider the demonstrated location as equal to an existing grounding within 50cm of the demonstrating agent's position. The variants I examined were a low value of 20cm and a high value of 100cm. Note that this does not change the value that the global location consistency metric uses for locations to be considered equal. That value is constant at 80cm to allow a direct comparison between the different location equality tolerance values being tested.

Figure 5.22 shows the improvement in the split 8x8m environment under the various ϵ values. This figure shows no appreciable change in performance, for eight agents, when the tolerance with which two locations are considered equal is varied. I believe this is due to interference. The agents interfere with each other sufficiently to make any changes in the tolerance of location equality insignificant.

When viewing the performance increase when using four agents (also Figure 5.22),

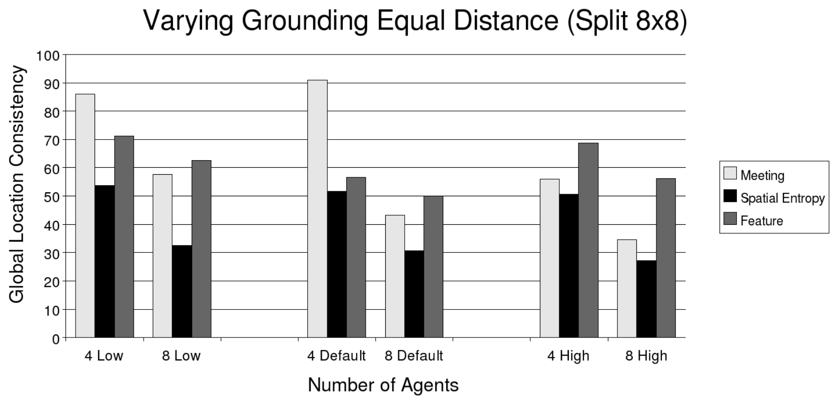

Figure 5.23: Average global grounding consistency by grounding strategy and agent population (4 and 8) when varying the tolerance of location equality inside an agent, in split 8x8m.

where interference should be much less evident, some performance variations are evident. Performance for the label-at-meeting and label-environment-feature grounding strategies decrease slightly as the tolerance for location equality increases. This is an intuitive result, as a higher location tolerance distance leaves more room for error in locations communicated between agents. This should decrease the effectiveness of communication about the goal's location, which in turn should decrease performance.

In contrast to label-at-meeting and label-environment-feature, the label-spatial-entropy grounding strategy shows a consistent *increase* in performance as the tolerance for location equality increases. This is due to the global location consistency staying about constant (Figure 5.23), with fewer global groundings (Figure 5.24). Thus, there is a smaller, but just as consistent set of global groundings. This allows for more effective communication about the location of the goal, which leads to better performance. This technique seems to benefit from a higher tolerance of location equality, and warrants further experimentation.

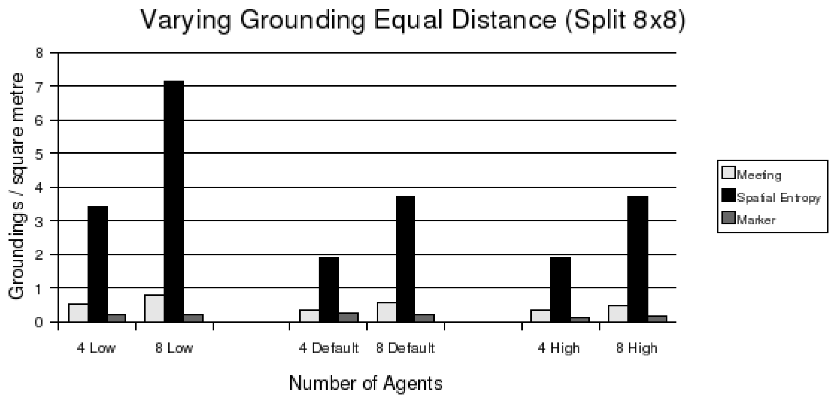

Figure 5.24: Average global grounding density by grounding strategy and agent population (4 and 8) when varying the tolerance of location equality inside an agent, in split 8x8m.

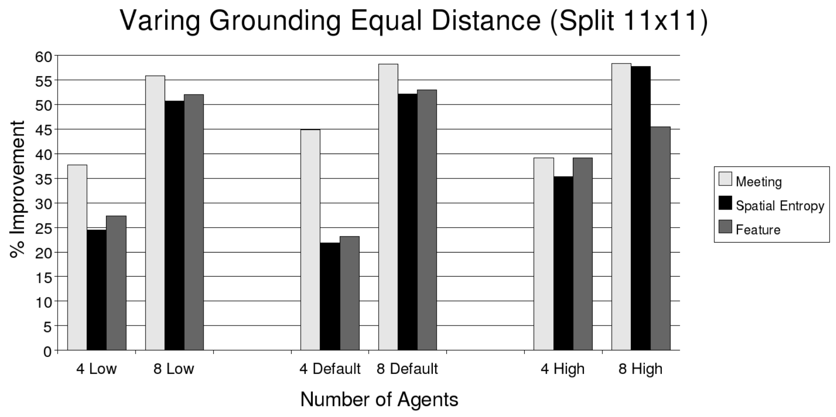

Figure 5.25: Average improvement by grounding strategy and agent population (4 and 8) when varying the tolerance of location equality in split 11x11m.

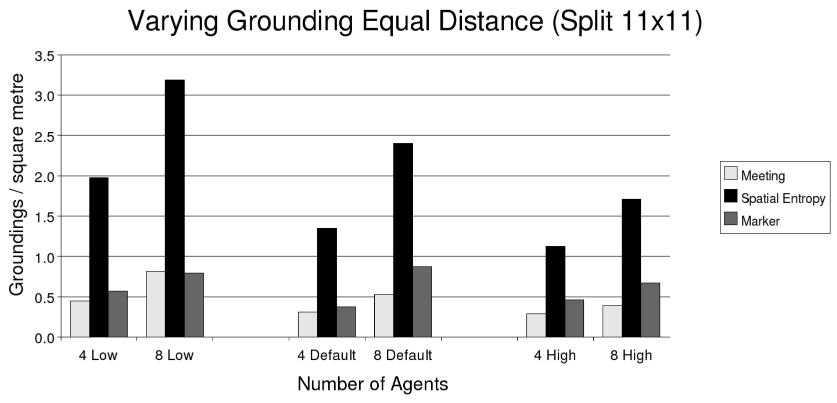

Figure 5.26: Average global grounding density by grounding strategy and agent population (4 and 8) when varying the tolerance of location equality inside an agent, in split 11x11m.

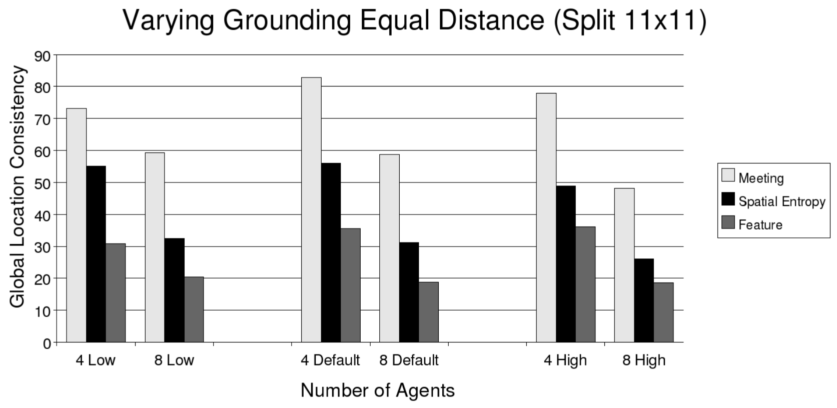

Figure 5.27: Average global grounding consistency by grounding strategy and agent population (4 and 8) when varying the tolerance of location equality inside an agent, in split 11x11m.

Figure 5.25 shows the improvement in the split 11x11m environment. The split 11x11m environment seems to be large to enough to allow the label-spatial-entropy grounding strategy to show an improvement. In both the four and eight agent case, performance rises with the tolerance for location equality. I believe this is the due to the same factors described in the previous environment. As the location equality tolerance rises, the number of grounded locations decreases (Figure 5.26), but the global location consistency remains about the same (Figure 5.27). This means there is a smaller set of grounding, with the same consistency, causing communication using that smaller set to be more effective.

The label-environment-feature grounding strategy, with eight agents, shows a slight improvement from the low to the default tolerance for location equality. With four agents, the performance is about the same with the low and default tolerance, then jumps with the high tolerance of location equality. I believe that interference has overtaken any effect that an increase in the tolerance of location equality has in the eight agent case, as it shows little difference in global location consistency and grounding density. The four agent label-environment-feature configuration gets a performance boost with the high location tolerance distance because it is able to maintain the same global location consistency as the default tolerance of location equality (Figure 5.27), while grounding a greater number of locations (Figure 5.26). This means that agents know more locations, and just as consistently, when using the high tolerance of location equality as compared the default tolerance, which leads to increased performance.

The label-at-meeting grounding strategy shows about the same improvement in split 11x11m for all tolerances of grounding equality, when testing eight agents. As the tolerance of location equality increases, the number of global groundings decreases (Figure 5.26), leading to less precise goal location broadcasts. However, this is offset by a higher global location consistency (Figure 5.27). The end result is little change in improvement for label-at-meeting with eight agents, when varying the tolerance of location equality.

The four agent case, when using the label-at-meeting strategy in a split 11x11m environment, seems to show an opposite trend from a split 8x8m environment. Where performance decreased slightly with a higher tolerance of location equality in a split 8x8m environment, it shows a general improvement in in a split 11x11m environment. I believe this difference is due to the size of the environment. A natural consequence of increasing the tolerance at which locations are considered equal is that when a goal location is broadcast, this will represent a broader range of space in the physical environment. In the 8x8m environment, a less precise goal broadcast led to decreased performance, while in the 11x11m environment, having any information at all about the goal is more valuable than in 8x8m. So, despite the fact that the goal is specified with less precision, there are a smaller number of groundings (also because of the increased tolerance for equality), allowing more agents to know the goal's location with each goal location broadcast, which in turn increases performance.

5.4.3 Employing Symbolic References

As in the other experiments in this section, symbolic references were tested with 4 and 8 agents, in split 8x8m and 11x11m environments only. In order to properly

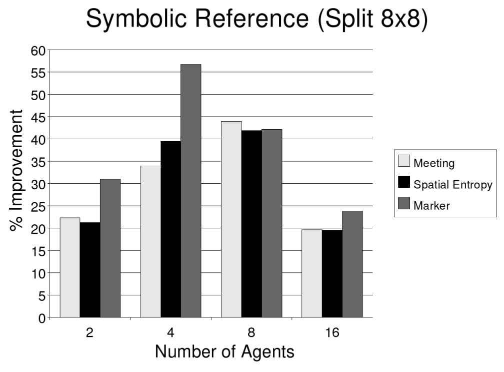

Figure 5.28: Average improvement by grounding strategy using a symbolic reference for 4 and 8 agents in the split 8x8m environment.

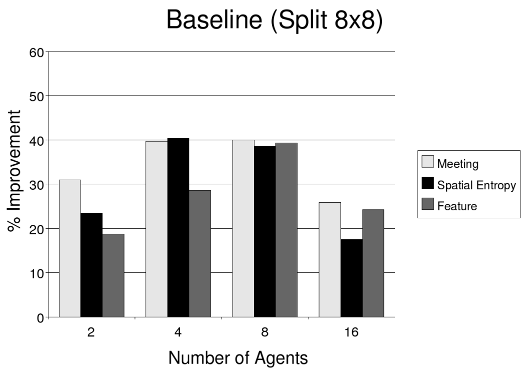

Figure 5.29: Average improvement by grounding strategy using a nearest grounding to specify the goal's location for 4 and 8 agents in the split 8x8m environment.

compare the use of symbolic references to the baseline approach, the baseline results for split 8x8m and 11x11m environments must be considered separately from the aggregated results described in Section 5.3.1. In the figures presented in this section, the baseline results are those for these two environments only (as listed in Appendix A) as opposed to the aggregated results shown in the figures in Section 5.3. In addition, the use of symbolic references affects only the manner in which the goal is described, not the decision as to when to ground a location or how to share it with others. Because of this, location density and global grounding consistency under symbolic references should be similar to those under baseline conditions, and are not particularly interesting metrics as far as the use of symbolic references is concerned. For comparison purposes, the charts showing these secondary metrics are included in Appendix B (Figures B.1 – B.8).

Looking at Figures 5.28 - 5.31, it can be seen that the label-at-meeting grounding strategy using an indexical reference (i.e. stating the closest grounded location when

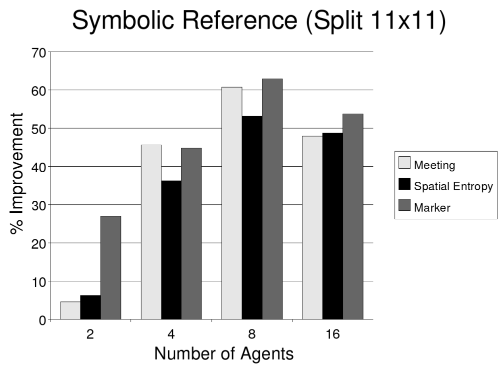

Figure 5.30: Average improvement by grounding strategy using a symbolic reference to specify the goal's location for 4 and 8 agents in the split 11x11m environment.

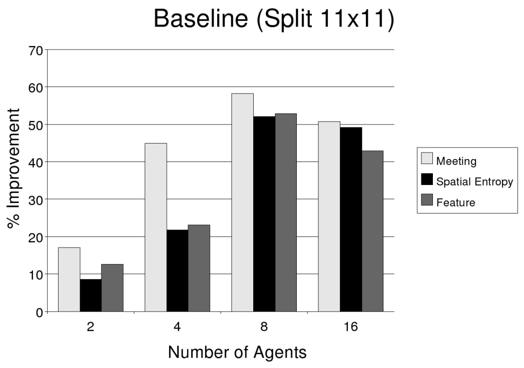

Figure 5.31: Average improvement by grounding strategy using a nearest grounding to specify the goal's location for 4 and 8 agents in the split 11x11m environment.

describing the goal) performs as well or better than when using a symbolic reference. The label-at-meeting environment provides good distribution of shared groundings over the environment, with high consistency, allowing the goal's location to be specified accurately enough with the indexical reference. This is coupled with the fact that a symbolic reference requires twice as many shared groundings to specify the goal, which leads to approximately equal performance between the two references.

There is one label-at-meeting trial which is difficult to explain. Figure 5.30 shows the improvement for label-at-meeting for 2 agents in the split 11x11m environment. The improvement is quite low, (about 5%): much less than expected. Since each grounding is implicitly shared when it is created, and there are only 2 agents in this experiment, a symbolic reference should be able to specify any location in the environment after only two groundings have been created, leading to good performance. This particular configuration should be rerun to verify the result.

Interestingly, the label-spatial-entropy grounding strategy shows little difference in performance across all agent populations, in both the split 8x8m and split 11x11m environments, when using an indexical reference as opposed to a symbolic reference to describe the goal location. The increased flexibility of a symbolic reference seems to be mitigated by the need to share twice the number of groundings in order to make use of it (recall that label-spatial-entropy grounds many locations). Future work could explore this by selecting a domain in which an indexical reference is less likely to be useful. For example, a domain in which an agent is unlikely to be able to successfully navigate to a given area by random wandering. It is unlikely to have any shared groundings in this area, as it is unlikely to have been there before to receive any demonstrations. When using a symbolic reference, locations inside this area could still be communicated accurately by using grounded locations elsewhere in the environment, which is not possible with an indexical reference. Although the split environment may seem to fit this criteria, the random placement of agents at beginning of each iteration allows shared groundings to be established across the entire environment.

The label-environment-feature grounding strategy shows a much higher improvement in the split 8x8m environments containing 2 or 4 agents, when using a symbolic reference as opposed to an indexical reference to describe the goal. However, this advantage disappears with 8 and 16 agents: these populations perform equally well using either type of reference. I believe that interference is again overshadowing all other factors for 8 and 16 agents in the split 8x8m environment, as there are a limited number of locations available to the grounding strategy to ground and demonstrate. In the split 11x11m environment, the symbolic reference outperforms the indexical reference for all agent populations. I believe this is due to the advantage of a symbolic reference over the indexical reference. Since agents using the label-environment-feature grounding strategy are limited to grounding certain locations, the indexical reference has limited ability to specify the goal's location using these locations. This is in contrast to a symbolic reference, which gives label-environment-feature the ability to specify an arbitrary point in the environment, and hence the goal's location more accurately.

5.5 Summary

This chapter has presented the results and analysis of the experiments used to evaluate my approach to developing grounded communication. The baseline experiments in Section 5.3.1 showed that all individual grounding strategies, combined with communication about the goal's location through an indexical reference, perform better than not using any communication. The amount of performance varied with the number of agents, the individual grounding strategy, and the environment size. When the environment is large enough to support the agent population without significant interference, performance increases with the number of agents. If the environment is not large enough for the agent population, the performance is limited by interference.

The best overall grounding strategy for this task was label-at-meeting. The implicit sharing when agents create a new grounded location leads to a high global location consistency, which improves the effectiveness of communication about the goal. The performance of the label-spatial-entropy grounding strategy was limited by the large number of locations it grounded. Having a large number of grounded locations makes it difficult to achieve a high global location consistency, reducing effectiveness of goal communication. Using a higher threshold for determining when a location was worth labelling would likely have benefited this grounding strategy. The label-environment-feature grounding strategy was the only labelling strategy which was limited to grounding a certain number of locations. This led to increased interference around these portions of the environment. This hampered performance improvement of the label-environment-feature with large numbers of agents relative to the other grounding strategies. In a more realistic implementation of the label-environment-feature grounding strategy, additional groundings would likely be created due to perceptual errors. This would provide more grounded locations that could be used to specify the goal's location, which would likely lead to improved performance.

Interference was an important factor when the distance at which an agent could demonstrate a grounding to another was varied. Where there were differences in improvement between the low and default distances, the low distance generally exhibited better performance than the default distance. Due to the decreased number of agent interactions, agents were able to spend more time searching for the goal, which improved performance.

When the ϵ parameter (the tolerance for location equality) was varied, none of the grounding strategies showed an observable difference in performance when examined with 8 agents, due to interference overwhelming those differences. When using 4 agents, some differences were evident. Label-at-meeting does better with a lower tolerance for location equality in split 8x8m, because the goal's location can be specified more accurately, but worse in split 11x11m due to the relative utility of any goal location information. The percentage improvement for label-environment-feature also drops as the tolerance for location equality rises, due to less accurate goal location specifications. The label-spatial-entropy grounding strategy shows an increase in performance, as the tolerance for location equality increases, as there are fewer, more consistent, global groundings with a high tolerance for location equality.

When using a symbolic reference to specify the goal's location, the label-at-meeting strategy did not show an observable performance improvement compared to using an indexical reference. I believe this is due to an indexical reference, combined the a well distributed set of consistent shared groundings, offering good performance on its own. There is simply little to be gained in this configuration. The label-spatial-entropy grounding strategy also shows little change when comparing an indexical reference to a symbolic reference. This is likely due to the increased flexibility of a symbolic reference being offset by the need to share twice as many groundings in order to use it. The label-environment-feature grounding strategy showed a marked improvement when using a symbolic reference. I think this is due to a symbolic reference's ability to specify an arbitrary point in the environment. When using an indexical reference, label-environment-feature is limited in both the number and location of the points which can be grounded, harming performance.

Chapter 6

Conclusion

This thesis has presented an approach to developing consistent shared groundings over time, and has described an implementation and evaluation of that approach. In this chapter I will outline my contributions, and list important differences between my work and similar related work. I then present some limitations of my system, and suggest future work.

6.1 Answers to Research Questions

Using a predefined coordinate system is an anthropocentric categorization, and should be avoided. Anthropocentric categorizations are a human perspective on how an agent should represent entities about which the agent must reason. Problems arise because a human system designer cannot set aside his or her human bias and objectively ground entities in the language of the agent's sensors and cognitive abilities. This leads to problems when agents needs to reason about and perform actions with the entities which have been artificially grounded.

Solutions developed without anthropocentric categorizations are also much more general than those that include them. In a multi-agent mapping situation, for example, agents that must share a coordinate system cannot function in situations where that cannot be guaranteed. Agents that do not share a coordinate system must perform more intellectual work to either develop one or infer partial maps without the help of coordinate information, but can function in a much broader range of situations.

Finally, even if a system designer could accurately ground entities in the agent, there remains a question of scale. Future systems will be required to reason about more and more entities. How many groundings can a system designer specify before there are simply too many?

One solution that deals with all of the above issues is to allow agents to learn their own groundings. However, this introduces a problem in a multi-agent environment. If individual agents in a multi-agent system are allowed to learn groundings purely individually, they will never arrive at the same groundings collectively. How then can the agents communicate using these groundings? The research presented in this thesis answers this question by demonstrating a system where agents can learn shared groundings for locations in an environment. The evaluation presented in Chapter 5 showed that that the techniques I have developed do cause effective shared groundings to be developed, and that the use of those shared improved agent performance.

Specifically, this thesis set out to answer the following questions:

1. Can agents develop consistent shared groundings for locations in an environment, despite not sharing a coordinate system, in order that agent performance

can be improved in a domain that benefits from communication about locations?

2. Does the manner in which agents create individual groundings impact the performance of the approach for sharing groundings?

While the two research questions above are central to this thesis, the approach I have developed to answer these questions allows a number of factors to be changed, leading to the following secondary research questions:

1. Does the distance at which an agent can begin a conversation with another agent affect the performance of the approach?
2. Does the maximum distance two locations can be apart, and still be considered equal (ϵ), affect the performance of the approach?
3. How does using a symbolic reference, instead of an indexical reference, to designate the goal's location affect performance?

Through my research I have found that agents can indeed develop consistent shared groundings for locations in an environment. Communication using these shared groundings empowered the set of agents to aid one another in their search for the goal, leading to an increase in performance. The manner in which agents created individual groundings, which were later shared, proved to be an important factor. This factor exhibited different performance characteristics depending on the size of the environment, and the size of the agent population.

I also found that the distance at which one agent can begin to share a grounding with another agent, and the maximum distance between two grounded locations that can still be considered equal, have an impact on the performance increase. While this impact was certainly observable in many instances, it was also sometimes overshadowed by interference between agents. This is especially true in small environments with large agent populations.

Using a symbolic reference, instead of an indexical reference, to specify the goal's location also increased performance in some circumstances. The increase was most noticeable when a symbolic reference was used in conjunction with a grounding strategy that could only ground a fixed number of predefined locations. When a symbolic reference was used with a grounding strategy that already provided good coverage of the environment, it offered approximately equivalent performance as an indexical reference.

6.2 Contributions

The primary contribution of this thesis is the development and demonstration of a practical system that allows agents to collectively develop shared groundings for locations in a physical environment over time for the purposes of communication. This is a recent research topic with few successful examples. While other approaches have been noted in the literature review, each of these has problems which are improved upon by my approach. For example, my approach is judicious in its use of groundings, (as opposed to simply blanketing the environment with groundings as others do). My approach also does not commit agents to solely working to perform groundings, as others do - it is intended that such groundings develop over time as a consequence of encounters between agents that would ordinarily occur during the performance of some task. The potential applicability of this research is very significant, in that tasks that currently require pre-stated groundings (i.e. most physical environments) could be adapted to the much more scalable solution of having the groundings be developed by the agents themselves.

My work also examined the relative utility of the initial sources of groundings. Three different strategies were compared, label-at-meeting, label-spatial-entropy, and label-environment-feature. Of these, the label-at-meeting technique was found to offer the biggest improvement in performance. This technique produced better results than the others due to the implicit sharing of the grounding, when it is initially created. The other techniques do not enjoy this advantage.

Another contribution of my work is the development of a metric for determining the global grounding consistency across a set of individual agents of individual agents. This metric was used to assess the effectiveness of the sharing and conflict resolution mechanisms present in my thesis research. I am not aware of any other research that uses such a metric.

Agents in this thesis also did not assume a fixed world size. Agents started with a small initial environment, which was expanded along the appropriate axis as the agent explored territory further from its starting position. In a memory constrained agent, this may be a valuable contribution.

6.3 Relationship to Previous Research

While elements of the work presented in this thesis, such as symbol grounding, mapping, and path planning have been well studied in isolation, there is little previous work which examines the issue of developing shared groundings for locations in an environment. The differences between my work and two recent works that are directly related are discussed in this section.

In the work of Billard and Dautenhahn [1999], described in Section 2.3.1, agents learned a static language for word / signal pairs, and a polar coordinate system, from a teacher agent. In my work, the language is initially undefined: it comes about as a consequence of the interactions between agents. There is no identifiable teacher. Further, my approach assumes to no fixed reference point in oder to define a coordinate system. Each agent has its own, private coordinate system.

Jung and Zelinsky [2000] have also provided contributions in this area. They demonstrated a system (described in Section 2.3.2) which benefited from the use of a symbolic reference. While the agents in their work did not initially share a coordinate system, their work still contains several limitations relative to the work in this thesis. For example, their system grounded locations at a *much* higher density than is required by my approach. Moreover, creating these shared groundings was an explicit and necessary phase of the agents' operation. My approach does not require an explicit phase to create shared groundings: this happens when agents encounter each other during the natural course of their work. Finally, Jung and Zelinsky [2000] demonstrated their approach with only two agents in a single environment. In Chapter 5, I presented the results of experiments with 2, 4, 8, and 16 agents, using two sizes of four different environment configurations, demonstrating its effectiveness in both simple and complex environments, and with various populations of agents.

6.4 Future Work & Limitations

While the techniques presented in this thesis remove assumptions present in previous work, there is still a great deal of future work that could be performed. For example, all agents in this thesis used the same representation of the environment, an occupancy grid. There is no aspect of my technique for creating and sharing grounded locations that requires this representation. An agent that uses a topological map should be just as effective as an agent using a grid-based internal representation. Further, agents that employ a topological map to represent the environment should be able to coexist, and exchange groundings with, agents that employ a grid based representation. All that my technique requires for demonstrating groundings is that agents be able to navigate back to a location they have previously grounded, in order to demonstrate that location to another agent. It would be valuable to verify this with experimental data.

My evaluation noted that especially in smaller environments, interference was a problem as agent populations increased. Because of this, another useful course for future work to take would be quantifying the amount of interference in a domain, and examining ways to reduce the interference between agents. Once the interference can be measured, solutions can be tested.

Measuring the amount of time that an agent spends demonstrating groundings to other agents instead of searching for the goal would also be useful, in that it would provide insight into exactly how much benefit could be gained by limiting these interactions. For example, presumably giving an agent a memory of which groundings it has demonstrated to which agents would avoid redundant demonstrations, allowing agents to perform more useful work. On the other hand, this raises significant scalability questions. How much memory is required to achieve this? What happens if a significant number of additional agents are added? Perhaps the solution to this problem is to allow each agent to remember the last few (say 10) groundings that it has demonstrated to other agents. This would reduce the memory required to a small constant amount, regardless of the number of agents in the system. Having this small, short-term memory may be enough for agents to disperse themselves, and significantly reduce repeat demonstrations.

Another useful extension to my approach would be some sort of mechanism to detect when there are a sufficient number of groundings in the environment: that is, coverage that is complete enough to support communication that is useful to the task at hand. Once sufficient coverage has been reached, agents can then concentrate more on useful work (i.e, ignore others when they encounter them, rather than using that as an opportunity to produce more shared groundings). One possible way of accomplishing this in a domain-independent manner would be to disallow agents to create any new groundings, regardless of grounding strategy, when there are existing groundings within some distance. This distance would be a key parameter in determining the final density of groundings in the system. Some time after agents have stopped creating new groundings, a set of consistent global shared locations should develop. Once this point has been reached, there is no advantage to agent's continuing to share groundings with each other. Instead, agents should spend all their time searching for the goal, as a consistent set of shared groundings is known to be in place.

In this research, the global grounding consistency was not computed by each agent. A separate entity (the Experiment Coordinator) with knowledge of each agent's groundings and coordinate system offsets computed this value. It is hard to envision empowering an individual agent with the ability to know when the set of global groundings has become consistent. Perhaps some heuristics could be employed within each agent to approximate this result. For example, agents could remember the outcomes of the last few grounding exchanges. If the grounding exchanges did *not* result in either of the agents involved learning a new grounding (i.e. both agents already knew the demonstrated grounding), then the agent could be made to be less likely to demonstrate a grounding next time it had an opportunity. Another possible method for naturally limiting the number of grounding demonstrations would be forbidding an agent from participating in a grounding demonstration when it has heard *and understood* the goal location broadcast by another agent. As the set of groundings becomes more consistent across the population, agents would be more likely to understand the broadcast goal location messages, and hence less likely to become involved in grounding demonstrations.

Note that the proposed extensions here are as domain-independent are possible. There is also the issue of tuning groundings for the agents' purpose in the domain. That is, a set of groundings useful for communicating while moving around an office is much finer than that required for giving directions around a neighbourhood. Ultimately, for a specific domain the extensions described above should be combined with knowledge about that domain to make the best decision as to what constitutes a good set of groundings.

While this thesis made several contributions, it also contains limitations. The most obvious is that it was run in software simulation, rather than on real robots. While the Stage simulator has been verified as accurately simulating the Pioneer robots used in this research, these results should still be verified using physical Pioneer robots. Agents in this work were perfectly localized at the beginning of each trial iteration. Since a single iteration lasted for a maximum of five minutes, this places an artificial upper bound on the difference between the agent's perceived position and its actual position.

This work also assumed that all agents in the system were working together, towards the common goal of creating a consistent set of shared locations. If a malicious agent was introduced into the system, it could likely wreak havoc by deliberately demonstrating incorrect groundings. The agents presented in this thesis would have no way to detecting or accounting for this. Future work could consider ways to detect and defeat such behaviour.

6.5 Summary

This thesis has presented an approach and implementation that allows agents to learn shared groundings for locations in an environment over time, without sharing a predefined coordinate system. Agents used three different strategies to decide when a location was worth grounding: label-at-meeting, label-spatial-entropy, and label-environment-feature. Each of these methods, combined with my technique for sharing groundings and reconciling conflicts, showed an observable improvement compared to when agents did not use any communication at all, for the task of finding a randomly placed goal in the environment.

Throughout my work I have tried to emphasize both the importance of this topic and the many subtleties involved in developing shared groundings. Future multi-agent systems will be required to reason about an increasing number of entities, in increasingly complex environments. Agents that are able to learn groundings individually, and later reconcile them with others, represent the way forward.

Appendix A

Baseline Experiment Result Listing

This appendix provides result listings for the baseline experiments detailed in Section 5.3.1.

In the tables below *POP* is the number of agents in the population being tested. Times are specified in milliseconds. *No comm* is the average time to complete a trial iteration in milliseconds, without communication. *LAM* is the average time for the label-at-meeting grounding strategy. *LAM %* is the percentage improvment that label-at-meeting offers over no communication. *SE* is the average time taken for the label-spatial-entropy grounding strategy. *SE %* is the percentage improvement that label-spatial-entropy offers over no communication. *FTR* is the average time for the label-environment-feature grounding strategy. *FTR %* is the percentage improvement that label-environment-feature offers over no communication. *Ave Imp* is the average percentage improvement for the given row in the table.

A.1 Hallway 8x8

POP	No comm	LAM	LAM %	SE	SE %	FTR	FTR %	Avg Imp
2	254648	202606	20%	185576	27%	214285	16%	21%
4	252957	177479	30%	189486	25%	208791	17%	24%
8	265530	169429	36%	163212	39%	189243	29%	34%
16	259150	201324	22%	215428	17%	238752	8%	16%
Avg			27%		27%		17%	24%

A.2 Split 8x8

POP	No comm	LAM	LAM %	SE	SE %	FTR	FTR %	Avg Imp
2	324914	224313	31%	248600	23%	264114	19%	24%
4	320144	193078	40%	190986	40%	228614	29%	36%
8	326747	196217	40%	200924	39%	198351	39%	39%
16	307701	228151	26%	253765	18%	233217	24%	23%
Avg			34%		30%		28%	31%

A.3 Office 8x8

POP	No comm	LAM	LAM %	SE	SE %	FTR	FTR %	Avg Imp
2	207046	161911	22%	196772	5%	170101	18%	15%
4	206730	132571	36%	122670	41%	124371	40%	39%
8	237856	130952	45%	122706	48%	128282	46%	46%
16	233710	160908	31%	165749	29%	175099	25%	28%
Avg			33%		31%		32%	32%

A.4 Open 8x8

POP	No comm	LAM	LAM %	SE	SE %	FTR	FTR %	Avg Imp
2	67754	44193	35%	57240	16%	61583	9%	20%
4	67890	44605	34%	44426	35%	55145	19%	29%
8	58240	38979	33%	44383	24%	53169	9%	22%
16	69357	50207	28%	54514	21%	55481	20%	23%
Avg			32%		24%		14%	23%

A.5 Hallway 11x11

POP	No comm	LAM	LAM %	SE	SE %	FTR	FTR %	Avg Imp
2	279646	218448	22%	231353	17%	206407	26%	22%
4	276973	153314	45%	183683	34%	187817	32%	37%
8	265446	153393	42%	150567	43%	179289	32%	39%
16	266678	138733	48%	160622	40%	178889	33%	40%
Avg			39%		33%		31%	35%

A.6 Split 11x11

POP	No comm	LAM	LAM %	SE	SE %	FTR	FTR %	Avg Imp
2	259739	215348	17%	237361	9%	226924	13%	13%
4	263048	144973	45%	205662	22%	202271	23%	30%
8	305963	127875	58%	146496	52%	144066	53%	54%
16	292538	143952	51%	148681	49%	166887	43%	48%
Avg			43%		33%		33%	36%

A.7 Office 11x11

POP	No comm	LAM	LAM %	SE	SE %	FTR	FTR %	Avg Imp
2	198157	133513	33%	165950	16%	157744	20%	23%
4	191335	110317	42%	136806	28%	120124	37%	36%
8	191219	99196	48%	109664	43%	118602	38%	43%
16	202145	103120	49%	114046	44%	112993	44%	46%
Avg			43%		33%		35%	37%

A.8 Open 11x11

POP	No comm	LAM	LAM %	SE	SE %	FTR	FTR %	Avg Imp
2	164762	114039	31%	115115	30%	137116	17%	26%
4	136194	83624	39%	93725	31%	96954	29%	33%
8	125193	72330	42%	77266	38%	86206	31%	37%
16	135344	74892	45%	80057	41%	85901	37%	41%
Avg			39%		35%		28%	34%

Appendix B

Additional Experiments

This appendix provides result listings for the additional experiments detailed in Section 5.4.

B.1 Varying Grounding Sharing Distance

In the tables below, *POP* is the number of agents in the population being tested. Times are specified in milliseconds. *MtgLow* is the average time for an agent to complete a trial iteration when using the label-at-meeting grounding strategy with a low grounding sharing distance. *MtgLow %* is the percentage improvement that the label-at-meeting grounding strategy offers with a low grounding sharing distance offers over using no communication. *SE Low* is the average time for an agent to complete a trial iteration when using the label-spatial-entropy grounding strategy, with a low grounding sharing distance. *SE Low %* is the percentage improvement that the label-spatial-entropy grounding strategy offers with a low grounding sharing distance offers over using no communication. *FtrLow* is the average time for an agent to complete a trial iteration when using the label-environment-feature grounding strategy, with a low grounding sharing distance. *FtrLow %* is the percentage improvement that the label-environment-feature grounding strategy offers with a low grounding sharing distance offers over using no communication.

B.1.1 Split 8x8

Agents	MtgLow	MtgLow%	SE Low	SE low %	FtrLow	FtrLow %
4	178170	44%	191865	40%	192598	40%
8	180194	45%	195186	40%	176021	46%

B.1.2 Split 11x11

POP	MtgLow	MtgLow%	SE Low	SE low %	FtrLow	FtrLow %
4	145727	45%	155242	41%	169723	35%
8	120955	60%	149385	51%	153683	50%

B.2 Varying Tolerance of Location Equality

In the tables below, *POP* is the number of agents in the population being tested. Times are specified in milliseconds. *LAM LE HIGH* refers to the label-at-meeting grounding strategy when using a high tolerance for location equality. *LEH %* refers to the percentage improvement that label-at-meeting offers over no communication when using the high tolerance for location quality. *LAM LE LOW* refers to the label-at-meeting grounding strategy when using a low tolerance for location equality. *LEL %* refers to the percentage improvement that label-at-meeting offers over no communication when using the low tolerance for location quality.

SE LE HIGH refers to the label-spatial-entropy grounding strategy when using a high tolerance for location equality. *SEH %* refers to the percentage improvement that label-spatial-entropy offers over no communication when using the high tolerance for location quality. *SE LE LOW* refers to the label-spatial-entropy grounding strategy when using a low tolerance for location equality. *SEL %* refers to the percentage improvement that label-spatial-entropy offers over no communication when using the low tolerance for location quality.

FTR LE HIGH refers to the label-environment-feature grounding strategy when using a high tolerance for location equality. *FEH %* refers to the percentage improvement that label-environment-feature offers over no communication when using the high tolerance for location quality. *FTR LE LOW* refers to the label-environment-feature grounding strategy when using a low tolerance for location equality. *FEL %* refers to the percentage improvement that label-environment-feature offers over no communication when using the low tolerance for location quality.

B.2.1 Split 8x8

POP	LAM LE HIGH	LEH %	LAM LE LOW	LEL %
4	210749	34%	189854	41%
8	186170	43%	183596	44%

POP	SE LE HIGH	SEH %	SE LE LOW	SEL %
4	168835	47%	207223	35%
8	201854	38%	195385	40%

POP	FTR EQ HIGH	FEH %	FTR EQ LOW	FEL %
4	175124	45%	200627	37%
8	180290	45%	196203	40%

B.2.2 Split 11x11

POP	LAM LE HIGH	LEH %	LAM LE LOW	LEL %
4	160017	39%	163913	38%
8	127507	58%	135345	56%

POP	SE LE HIGH	SEH %	SE LE LOW	SEL %
4	170244	35%	198689	24%
8	129328	58%	151011	51%

POP	FTR EQ HIGH	FEH %	FTR EQ LOW	FEL %
4	160174	39%	191285	27%
8	166737	46%	146700	52%

B.3 Employing Symbolic References

In the tables below, *POP* is the number of agents in the population being tested. Times are specified in milliseconds. *LAM* is the average time for the label-at-meeting grounding strategy. *LAM %* is the percentage improvment that label-at-meeting offers over no communication when using the symbolic reference. *SE* is the average time taken for the label-spatial-entropy grounding strategy. *SE %* is the percentage improvement that label-spatial-entropy offers over no communication when using the symbolic reference. *FTR* is the average time for the label-environment-feature grounding strategy. *FTR %* is the percentage improvement that label-environment-feature offers over no communication when using the symbolic reference.

B.3.1 Split 8x8

POP	LAM	LAM %	SE	SE %	FTR	FTR %
2	252444	22%	255840	21%	224037	31%
4	211572	34%	193788	39%	138442	57%
8	182964	44%	189956	42%	188880	42%
16	247290	20%	247428	20%	234321	24%

B.3.2 Split 11x11

POP	LAM	LAM %	SE	SE %	FTR	FTR %
2	247767	5%	243456	6%	189626	27%
4	142969	46%	167735	36%	145277	45%
8	120008	61%	143427	53%	113545	63%
16	152462	48%	149951	49%	135230	54%

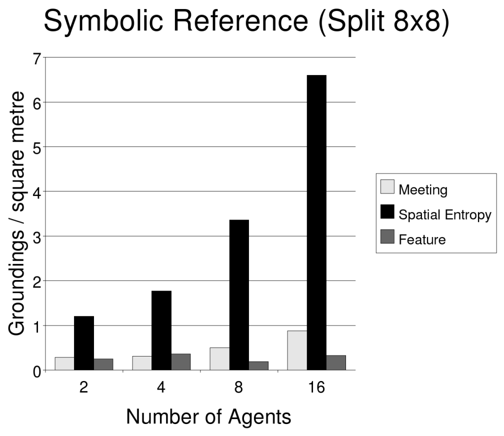

Figure B.1: Average global location density by grounding strategy using a symbolic reference to specify the goal's location for 4 and 8 agents in the split 8x8m environment.

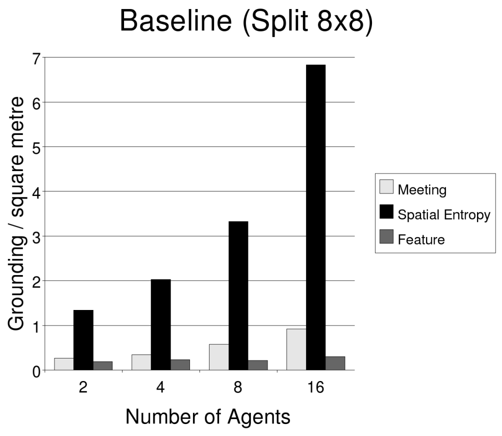

Figure B.2: Average global location density by grounding strategy using the nearest grounding to specify the goal's location for 4 and 8 agents in the split 8x8m environment.

B.3.3 Symbolic Reference Graphs

This subsection contains graphs for the global location consistency and grounding density in split 8x8m and split 11x11m environments when using a symbolic reference.

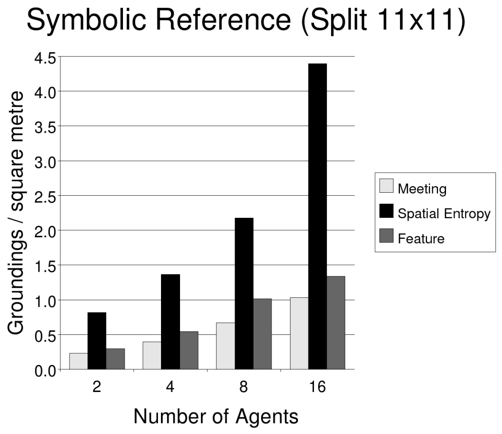

Figure B.3: Average global location density by grounding strategy using a symbolic reference to specify the goal's location for 4 and 8 agents in the split 11x11m environment.

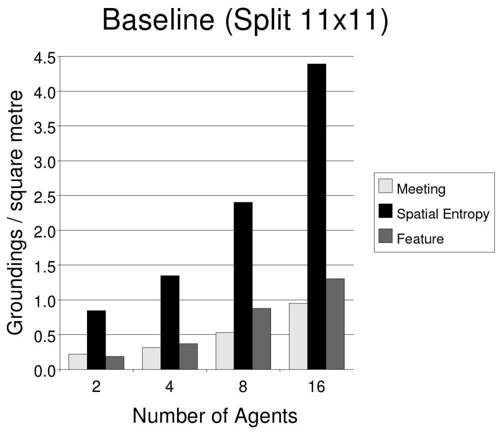

Figure B.4: Average global location density by grounding strategy using the nearest grounding to specify the goal's location for 4 and 8 agents in the split 11x11m environment.

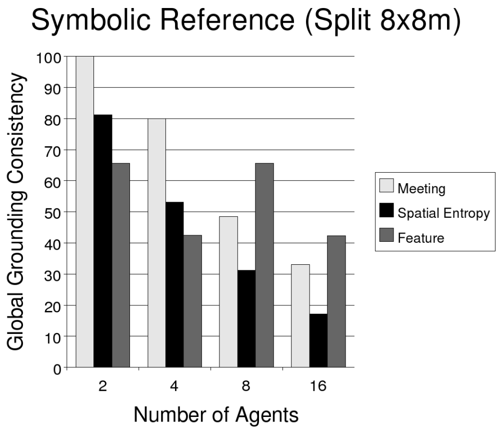

Figure B.5: Average global location consistency by grounding strategy using a symbolic reference to specify the goal's location for 4 and 8 agents in the split 8x8m environment.

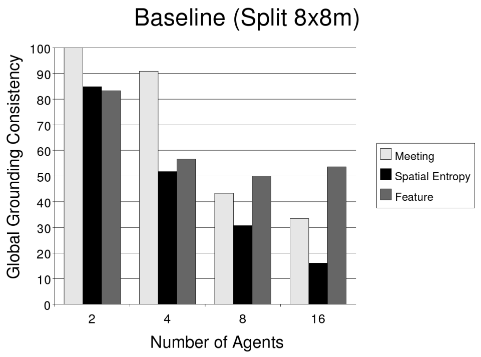

Figure B.6: Average global location consistency by grounding strategy using the nearest grounding to specify the goal's location for 4 and 8 agents in the split 8x8m environment.

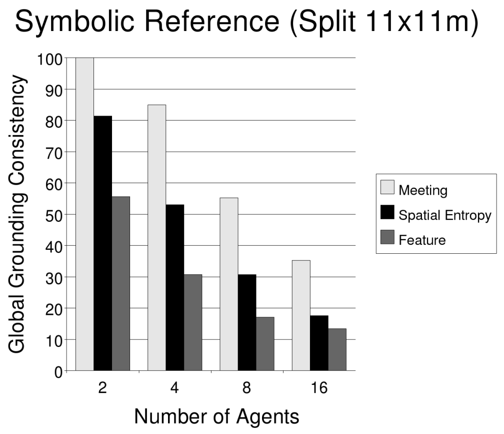

Figure B.7: Average global location consistency by grounding strategy using a symbolic reference to specify the goal's location for 4 and 8 agents in the split 11x11m environment.

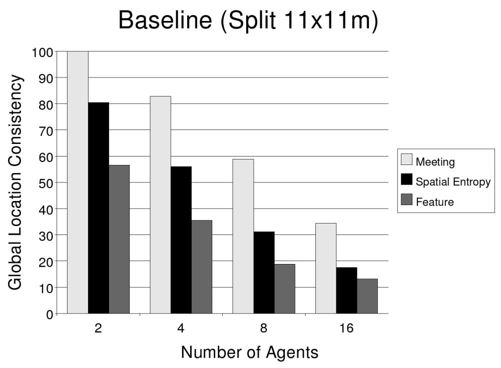

Figure B.8: Average global location consistency by grounding strategy using the nearest grounding to specify the goal's location for 4 and 8 agents in the split 11x11m environment.

Bibliography

Philip Agre and David Chapman. What are plans for? In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 17–34. MIT Press, March 1991.

F Andresen, L Davis, R Eastman, and S Kambhampati. Visual algorithms for autonomous navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 856–61. 1985.

M A Arbib. Perceptual structures and distributed motor control. In V B Brooks, editor, *Handbook of Physiology - The Nervous System II: Motor Control*, pages 1449–80. American Physiological Society, 1981.

M A Arbib. Schema theory. In S Shapiro, editor, *The Encyclopedia of Artificial Intelligence, 2nd ed.* Wiley-Interscience, 1992.

Okan Arikan, Stephen Chenney, and D. A. Forsyth. Efficient multi-agent path planning. In *Proceedings of the 2001 Eurographics Workshop on Animation and Simulation*, Sept 2001.

R Arkin, T Balch, T Collins, A Henshaw, D MacKenzie, E Nitz, R Rodriguez, and K Ward. Buzz, an instantiation of a schema-based reactive robotic system. In *Proceeding of the International Conference on Intelligent Autonomous Systems* $IAS - 3$, pages 418–427, February 1993.

Ronald C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, Ma., 1998.

T Balch and R Arkin. Motor schema-based formation control for multiagent robot teams. In *Proceedinggs of the 1995 Internation Conference on Multiagent Systems*, pages 10–16, 1995.

Tucker Balch. The impact of diversity on performance in multi-robot foraging. In Oren Etzioni, Jörg P. Müller, and Jeffrey M. Bradshaw, editors, *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 92–99. ACM Press, 1999.

Tucker Balch and Ronald C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):27–52, 1994.

Jacky Baltes and John Anderson. Flexible binary space partitioning for robotic rescue. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, October 2003.

Maxim Batalin. Java client for player/stage. Webpage, July 2004. http://www-robotics.usc.edu/∼maxim/JavaClient/jc.htm.

Aude Billard and Kerstin Dautenhahn. Experiments in learning by imitation – grounding and use of communication in robotic agents. *Adaptive Behavior*, 7(3/4): 415–438, 1999.

R Brooks and A Flynn. Building brains for bodies. *Autonomous Robots*, (1):7–25, 1994.

Rodney A. Brooks. A robot that walks: Emergent behaviors from a carefully evolved network. Technical Report AI MEMO 1091, MIT, 1989.

Rodney A. Brooks. A robust layered control system for a mobile robot. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1985.

Rodney A. Brooks. New approaches to robotics. *Science*, 253:1227–1232, September 1991a.

Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1&2):3 – 15, June 1990.

Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139 – 159, 1991b.

J Cameron, D MacKenzie, K Ward, R Arkin, and W Book. Reactive control for mobile manipulation. In *Proceedings of the International Conference on Robotics and Automation*, pages 228–235, 1993.

D. Chapman. Penguins can make cake. *AI Mag.*, 10(4):45–50, 1989.

Chen, Szczerba, and Uhran. Planning conditional shortest paths through an unknown environment: A framed- quadtree approach. In *Proceedings of the IEEE International Conference on Robotics and Automation*. May 1995.

P. R. Cohen, M. L. Greenberg, D. M. Hart, and A. E. Howe. Trial by fire: understanding the design requirements for agents in complex environments. *AI Magazine*, 10 (3):34–48, 1989.

S. Coradeschi and A. Saffiotti. Anchoring symbols to sensor data: preliminary report. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI)*, pages 129–135, Menlo Park, CA, 2000. AAAI Press.

S. Coradeschi and A. Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems*, 43(2-3):85–96, 2003.

S. Coradeschi and A. Saffiotti. Perceptual anchoring of symbols for action. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence IJCAI*, pages 407–412, Seattle, WA, 2001.

James L. Crowley. Navigation for an intelligent mobile robot. *IEEE Journal of Robotics and Automation*, 1:31–41, 1985.

Ian Davis. Warp speed: Path planning for star trek[tm]: Armada. Technical Report AAAI SS-00-02, Artificial Intelligence and Interactive Entertainment: Papers front the 2000 AAAI Spring Symposum, 2000. http://www.maddocsoftware.com/pdf/I_Davis_00.pdf.

Terrance Deacon. *The Symbolic Species: the co-evolution of language and the human brain*. Penguin Books, 1997. ISBN 0-713-99188-7.

J Drysdale and D Lyons. Learning image-based landmarks for wayfinding using a neural network. In *Proceedings of Artificial Neural Networks in Engineering*. 2004.

Alberto Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–56, 1987.

Etzioni, Oren, and Richard Segal. Softbots as testbeds for machine learning. In *Proceedings of the Machine Learning Workshop at AI/GI/VI '92*, pages v1–v8. May 1992.

Innes A. Ferguson. Touring machines: Autonomous agents with attitudes. *Computer*, 25(5):51–55, May 1992.

Brian P. Gerkey, Richard T. Vaughan, Kasper Støy, Andrew Howard, Gaurav S Sukhtame, and Maja J Matarić. Most Valuable Player: A Robot Device Server for Distributed Control. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1226–1231, Wailea, Hawaii, October 2001.

Hanks, Steve, Martha E. Pollack, and Paul R. Cohen. Benchmarks, test beds, controlled experimentation, and the design of agent architectures. *AI Magazine*, 14 (4):17–42, 1993.

Stevan Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

B. Hayes-Roth. An architecture for adaptive intelligent systems. *Artificial Intelligence: Special Issue on Agents and Interactivity*, 72:329–365, 1995.

Ian Horswill. Polly: A vision-based artificial agent. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI)*, pages 824–829, Washington, DC, July 1993.

Menglei Jia, Guangming Zhou, and Zonghai Chen. An efficient strategy integrating grid and topological information for robot exploration. In *Proceedings of 2004 IEEE Conference on Robotics, Automation and Mechatronics (RAM04)*, pages 667–672.

David Jung and Alexander Zelinsky. Grounded symbolic communication between heterogeneous cooperating robots. *Autonomous Robots*, 8(3):269–292, 2000.

Hayward Little and Albert C. Esterline. Motion planning for mobile agents. In *Proceedings of ADMI-99 Minority Institutions Computing Conference*, June 1999.

Pattie Maes. Artificial life meets entertainment: Life like autonomous agents. *Communications of the ACM*, 11:108–114, 1995.

M. Matarić. Integration of representation into goal driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992.

M. Matarić. Behavior-based control: Examples from navigation: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, (2-3):323–336, 1997.

Maja J. Matarić. Using communication to reduce locality in distributed multiagent learning. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3): 357–369, 1998.

H. P. Moravec. The stanford cart and the cmu rover. pages 407–419, 1990.

J P Müller. A conceptual model for agent interaction. In S M Deen, editor, *Proceedings of the 2nd International Working Conference on Cooperative Knowledge Based Systems (CKBS-94)*, pages 213–234. 1994.

A Newell and H Simon. Computer science as empeirical inquiry: symbols and search. *Communications of ACM*, 19(3):113–126, 1976.

D Payton. *Internalized Plans: A Representation for Action Resources*, pages 98–103. MIT Press, 1991.

ActivMedia Robotics. Pioneer 2 operations manual version 9. Technical report.

Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*. Alan Apt, Upper Saddle River, New Jersey, 1995.

A. Saffiotti. Pick-up what? In C. Bäckström and E. Sandewall, editors, *Current Trends in AI Planning*, pages 266–277. IOS Press, Amsterdam, NL, 1994. ISBN 90-5199-153-3.

John A. Sauter, Robert Matthews, H. Van Dyke Parunak, and Sven Brueckner. Evolving adaptive pheromone path planning mechanisms. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 434 – 440, Bologna, Italy, 2002.

John Searle. Minds, brains, and programs. *The Behavioral and Brain Sciences*, 3: 417–424, 1980.

C Shannon and W Weaver. *The Mathematical Theory of Communication*. Urbana: University of Illinois Press, 1949.

D. A. Shell and M. J. Matarić. Insights toward robot-assisted evacuation. *Advanced Robotics*, 19(8):797–818, 2005.

Luc Steels. Evolving grounded communication for robots. *Trends in Cognitive Sciences*, 7(7):308–312, 7 2003.

Luc Steels. Constructing and sharing perceptual distinctions. In M. van Someren and G. Widmer, editors, *Proceedings of the European Conference on Machine Learning*, pages 4–13. Springer-Verlag, 1997.

Peter Stone and Manuela M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

Ron Sun. Symbol grounding: a new look at an old idea. *Philosophical Psychology*, 13:149–172, 2000.

A Thompson. The navigation system of the jpl robot. In *5th Int. Joint Conf. on Artificial Intelligence*, page 749?757. 1977.

S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

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

S. Thrun, D. Fox, and W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998. also appeared in Autonomous Robots 5, 253–271 (joint issue).

Sebastian Thrun and Arno Bücken. Integration grid-based and topologial maps for mobile robot navigation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligent, AAAI*. August 1996.

Paul Vogt. Symbol grounding in communicative mobile robots. In *AAAI Fall Symposium on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems*. November 2001. URL `http://www.aass.oru.se/Agora/FSS01/index.html`.

Gerhard Weiss. *Multiagent Systems*. MIT Press, 1999.

Charles E. Wright. Controlling sequential motor activity. In Daniel N. Osherson, Stephen M. Kosslyn, and John M. Hollerbach, editors, *Visual Cognition and Action*, pages 285 – 316. MIT Press, Cambridge, MA, 1990.

Alfred Wurr. Robotic team navigation in complex environments using stigmergic cues. Master's thesis, Department of Computer Science, University of Manitoba, Winnipeg, MB, July 2003.

Alex Yahja, Sanjiv Singh, and Anthony Stentz. Recent results in path planning for mobile robots operating in vast outdoor environments. In *In Proc. 1998 Symposium on Image, Speech, Signal Processing and Robotics*. September 1998.