University of Alberta

Collective Robotics: From Local Perception to Global Action

by

Claus Ronald Kube 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Spring 1997

Go to the ant, you lazybones:
consider its ways, and be wise.
Without having any chief
or officer or ruler,
it prepares its food in summer,
and gathers its sustenance in harvest.
PROVERBS, 6:6-8

To my wife Carole
and my children Veronica and Thomas
for your love and patience
as I chase my dream.

Abstract

Does coherent behaviour require an *explicit* mechanism of cooperation? In this dissertation, the relationship between local perception and global action in a system of multiple mobile robots was examined for a collective box-pushing task. The problem investigated was how local sensing could be used to coordinate the individual motor responses of a system of robots in a coherent manner, using only implicit communication through the task. The task was to move a large box from an initially unknown position to a specified goal location. The central thesis put forward, is that for the box-pushing task a coherent behaviour is possible, without an explicit mechanism of cooperation, by using the mass effect of a system of redundant robots.

Preliminary work in collective robotics appeared to lend weight to the hypothesis that collective tasks, by multi-robot systems, are possible without centralized control or explicit inter-robot communications, two common control mechanisms used for cooperation. The goal was to propose and verify a framework for modelling a multi-robot task, such that the system displayed both coherent and coordinated behaviour without centralized control. The result is a coordinated global action by the system similar to group transport behaviour by ants. The result is achieved using the mass effect of a system of redundant robots. The approach to connecting perception and action is through a task description, specified as changes in the environment, and a task decomposition, which describes how a system will achieve those changes.

Demonstrated is a framework using a multi-robot box-pushing task and its extension to a directed box-transport task. Steps in the task are modelled as states, and implemented as subtask controllers, with state transitions determined by binary sensing predicates called *perceptual cues*. A perceptual cue (Q), whose computation is independent from the operation of the controller, is used by a finite state controller, called a *Q-machine*, to produce an action. Results are presented for a redundant system of physical robots capable of moving a heavy object collectively to arbitrarily specified goal positions.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to those that have supported my research, for without them the work presented here would not have been possible. I have been indeed lucky to have my supervisor, Hong Zhang, as both a mentor and a friend. He has encouraged and supported my unorthodox views on research and teaching with the true spirit of an explorer in uncharted territory. To him I owe much.

A huge quantity of beer goes to my friend Rod Johnson, who never stops amazing me with his good nature, open heart and keen mind. He has always been there and we share a common enthusiasm for life, family and robots.

I am very grateful to the members of my committee, William Armstrong, Randy Goebel, Tony Marsland, and Jens Roland who took the time to read and comment on this work, but a special thanks to Ronald Arkin for taking the time to venture up to the Great White North and brave our cold weather (I hope viewing Hale-Bopp was worth it).

Finally, I am forever indebted to my loving wife, Carole, and our children Veronica and Thomas. For it is them that makes doing all things worthwhile.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Does coherent behaviour require an *explicit* mechanism of cooperation? In this dissertation, the relationship between local perception and global action in a system of multiple mobile robots is examined for a collective box-pushing task. The problem investigated was how local sensing could be used to coordinate the individual motor responses of a system of robots in a coherent manner, using only implicit communication through the task. The task was to move a large box from an initially unknown position to a specified goal location. Preliminary work in collective robotics appeared to lend weight to the hypothesis that collective tasks by multi-robot systems are possible without centralized control or explicit inter-robot communication, two common cooperation mechanisms used for control [31]. The goal was to propose and verify a framework for modelling a multi-robot task, such that the system displayed both coherent and coordinated behaviour without centralized control. The problem was explored along the dimensions of perception and action. The result is a coordinated global action by the system without resorting to explicit mechanisms of cooperation such as directed communication between robots or robot differentiation. The approach to connecting perception and action is through a task description, specified as changes in the environment, and a task decomposition, which describes how a system will achieve those changes. Demonstrated is a framework using a coordinated multi-robot box-pushing task and its extension to a directed box-transport task. The conjecture is that this approach may also be extended to other multi-step construction tasks.

1.1 Connecting Perception to Action

It has been said that the era of the industrial robot—characterized by its use in the manufacturing industry—is about to make way for the next generation "service robot," a device with a high reliance on mobility to achieve its task specific purpose [8, 21]. Along with mobility comes a need for more autonomous operation than their progenitors whose lives were spent locked in cycles of precise movement in Cartesian space. Autonomy and mobility in robotics usually leads to the problem of dealing with increasingly uncertain environments, and advances in autonomous robots are accomplished through a better understanding of the role sensors play in controlling actuators.

In robotics, the problem of connecting perception to action is usually tackled by taking a reductionist's approach. The problem is decomposed into subproblems, which are often studied in isolation with the underlying assumption that, once they are solved, someone will fit all the pieces back together into a working autonomous robot ready to fetch and deliver our next cup of coffee. However, as is often the case, this merging of solved subproblems ends up as a research problem in itself, and results in a solution brittleness that quickly rears its head as soon as the next unforeseen event or circumstance occurs. One approach to dealing with uncertainty is to circumscribe the environment in which the robot performs its designed function thereby limiting the size of the possible set of sensor stimuli. Another, is to constantly calibrate and recalibrate the sensor and actuator systems, but this does little to help solve the larger issue of system integration brought about by the original problem decomposition. An alternate view held by many is holistic in its approach to creating systems that link perception to action in a given environment, the so called *animat* approach [79].

1.1.1 The Animat Approach to AI

The animat approach models whole but simple animal-like systems and their sensory environments. In this bottom up approach to intelligence the basic hypothesis is that human level systems can eventually be built by studying complete, albeit simple, animal-like sensory/motor response systems in simulated environments. This argument aside, the holistic approach to studying the connection between perception and action has merit when applied to the difficult task of designing and constructing physical mobile robots.

By grounding the animat's internal symbols in the physical stimuli of a given real environment, a study can be made of the connection between perception and action. The

animat approach of determining the minimal machinery needed for a given animat with needs, an environment, and a sensory/motor system used to achieve those needs is also applicable to the reactive behaviour-based approach to building mobile robots. Task difficulty is increased by either increasing the complexity of the environment or the complexity of the animat's needs to some criterion [79].

1.1.2 Niche Solutions and Collective Robotics

An alternative approach to dealing with uncertainty and the brittle reliability of single robot systems, is the multi-robot system which attempts to increase system reliability through the redundancy of mass effect. Although individual robots within the system still suffer from the same spatial constraints as found in the single robot system, the mass effect of many parallel sensing and actuation operations increases the probability the system can complete a given task. An analogy is often made with the task-achieving systems of social insects capable of complex tasks with well defined global results and all without the aid of centralized control. The effects of such systems are coordinated, but not necessarily cooperative since antagonistic forces are present.

Sudd [68] provides us the example of a two-meter-high mushroom-shaped termite's nest. Like the great pyramids of Giza, whose existence results from the effort of many thousand workers over several lifetimes, the end result is a predictable shape, as if planned by some master architect who then directs its step by step completion. In the case of the termite's nest, where is the master architect who orchestrates this collective effort? And what is the nature of the intelligence controlling this collective system in such a coordinated manner? Closer examination of the activity in nest construction reveals many antagonistic actions. As termites pile pellets into columns other termites undo the work by removing the same pellets. Despite these antagonistic agents system reliability, measured in terms of task completion, remains high. How then, is such coherent behaviour achieved?

1.2 Collective Behaviour

Robotics research in general is task driven. Three typical tasks used to study and evaluate theories in multi-robot control have emerged in the field: foraging, box-pushing, and formation marching. Foraging involves the search and retrieval, by a multi-robot system, of target items distributed in an environment. Targets are usually small enough to be handled by a single robot, and task completion is therefore possible by one robot given enough time. Box-pushing requires at least two robots to move an object in a common direction: this

needs more coordination than foraging since the task is not possible with a single robot. Formation marching needs a minimum of two robots moving in a given geometric pattern along a desired trajectory. Most multi-robot research studies measure system utility in reference to one of the above tasks.

In a recent review of the field, Cao et al. [16] defined multi-robot systems in which "there is an increase in the total utility of the system" as exhibiting *cooperative behaviour* provided it is "due to some underlying mechanism of cooperation." Thus it is often assumed, either explicitly or implicitly, that cooperation involves direct communication and the ability to distinguish robots from objects as the mechanism which increases total system utility.

In Distributed Artificial Intelligence (DAI) the distinction of cooperation as a separate concept is not made. Rather, cooperation is seen "as a special case of coordination among nonantagonistic actors" [12]. Since the simulated domains of interest to DAI do not contain the same level of uncertainty as is found in the physical domains of robotics, DAI has had a limited influence on collective robotics [16]. However, many of the problems in collective robotics are analogous to those found in DAI. Of particular relevance is the area of multiagent systems in which problems are solved by coordinating intelligent behaviour among a collection of autonomous agents. Here coherence and coordination are analytical concepts in wide use [12]. In DAI the term *coherence* refers to system behaviour in terms of an evaluation criterion, while *coordination* describes interaction among agents.

Coordination implies a predictable system level result with minimum interference. The more inter-robot interference the less coordinated the system behaves. According to Bond and Gasser, coherence and coordination are somewhat related since greater coordination can lead to more efficient coherence by minimizing the degree of interference [12]. In collective robotics, like DAI, the problem is achieving coherence and coordination without centralized control or a global viewpoint.

In this dissertation, focus is on the relationship between local perception and global action. The central thesis is that:

> *Coherent collective behaviour, in some tasks, does not require an explicit mechanism of cooperation.*

The support for this proposition is based on the many examples of collective behaviour among social insects. To demonstrate coherent behaviour without centralized control or explicit inter-robot communication, a directed box-pushing task was studied in which a box was moved from an initially unknown position to an arbitrary goal location. Complimenting

hypotheses on perception and action state both can be computed separately from the decision process linking them together. Next, the perception to action connection is examined by considering how local perception and global action can result in coherent behaviour.

1.2.1 Taxis-based Discrete Action

Can predefined actions be an appropriate perceptual response? Could you and a friend, each applying a unit-force, pick up an arbitrary table? Probably not, since the amount of force needed to pick up the table would depend on its weight which you would gauge using perception. But what if a group of your unit-force friends came to help and, since in typical cases a table's weight is a function of its circumference, enough friends joined in until the table was lifted. The force needed to lift tables would then be a function of group size. Using this algorithm, your group could lift several different size tables without varying the amount of effort each applied. A hypothesis regarding action within a group might then be stated as:

> *A motor response or action can be computed separately from its stimulus without regard to either its modality or magnitude.*

The advantage this modularization holds for designing a robot's control system, lies in the ability to change parts of the perception architecture without affecting the corresponding motor actions. Nature provides several examples in support of this approach.

Vowles determined that ants were able to substitute the perceptual stimuli used in orientation motor responses [72]. Using an artificial light as stimulus ants maintain a constant orientation angle with respect to the stimulus while traversing a horizontal surface. The light source was then removed and the surface tilted vertically. The ants changed their direction and maintained the same angle with respect to the stimulus, but switched input stimulus to reference gravity instead of light. Taxis mechanisms are reflex translational or orientational movements by a freely motile agent in relation to a source of stimulation, and in ants form a connection between sensory and locomotory mechanisms. Vowles hypothesized that both sensory and locomotory mechanisms functioned independently of the taxis mechanism.

A fixed sequence of actions used to accomplish a specific task is also found to occur in animals. Referred to as *fixed-action patterns* these sequences of behaviours have been observed in the greylag goose while executing an egg retrieval behaviour [44]. When an egg rolls from its nest, the goose will complete a sequence of movements starting with extending

its head to reach the egg, then pulling back until its head is between its legs. This sequence is repeated even if contact with the egg is lost; however, small side to side adjustments are made to keep the egg in place during the pulling phase of the behaviour. Thus, predefined actions in the form of stored behavioural programs could be independent from the stimulus which triggers them.

These stored behavioural programs can be invoked by researchers using appropriate stimuli. In ants, corpse removal is a collective behaviour invoked by chemical odour. Workers dispose of dead ants by carrying them from the nest to a refuse pile. Wilson *et al.* [77] were able to invoke the same behaviour in ants by treating bits of paper with acetone extracts of ant corpses. In fact, by daubing small amounts of acetone extract on live ants, they too were carried away by nestmates and dumped on the refuse pile.

Operations to Demonstrate Motor Action Modularity

Reactive control is an approach that connects perception to action without creating an internal model of the world on which to formulate a plan of action [5]. In order to show the use of discrete motor action in a reactive controller, a set of motion primitives was designed for the box-pushing robots from which all motor actions would be composed. The underlying motor behaviour of a robot would then be a mapping from perceptual cues, as outlined in the sequel, to individual motion primitives or sequences. The hypothesis implied in such an approach is that continuous motion of a mobile robot can be approximated from a finite set of small discrete motions using perception as the element selector. Global action occurs when many redundant robots provide a mass effect while working towards a solution to a shared problem. Next, we consider the opposite end of the perception to action mapping.

1.2.2 Local Perception

Can a complex decision making process be reduced to simple sensor preprocessing? The hypothesis implicit in this question is that:

> *The perceptual process used to trigger a response in reactive control can be computed using selective perception from the environment for the action which controls the robot.*

Here *control* refers to the decision process involved in mapping perception to action. Wehner argues that an animal's solution to perceptual tasks "is often restricted to a narrow range of stimuli and situations" found in its environment [75]. Insects are cited as a prime example

of this "matched filter" approach to perception, in which sensing receptors are spatially arranged to match some environmentally specific stimulus. This, Wehner conjectures, relieves the insect from any heavy computational task by solving the decision making process at the sensory level. Further, he speculates that although this makes the system less general in its ability to handle a variety of sensing input, the information processing is easier and suitable for the "narrow ecological niches" insects occupy.

Two examples Wehner cites, in support of this hypothesis, are the *visual streaks* found in the eyes of both desert ants [74] and crabs [80]. Visual streaks refer to the close spacing of photoreceptors near the center of the eye, the area in which most retinal images are formed because of the horizon dominated world which the animals inhabit. In crabs this spatial arrangement allows a constant number of receptors to be stimulated by objects of the same size regardless of the distance of the object to the eye. Wehner speculates that this mechanism is a simple solution to the animal's problem of determining the size of the object, when the retinal images appear in a predictable way due to the predominantly flat visual environment [75]. It is also speculated that such a mechanism may be used in part as a visual cue and subsequent response to predators [80].

Another example of behaviour triggering by stimulus cues is found in social insects. Both bees and ants use dawn and dusk to start and stop their foraging behaviour. In fact, bees have a special sensor system consisting of three ocelli used to detect light level intensity and manipulating these sensors affects when the foraging behaviour begins and ends. The light-level threshold at which foraging is triggered can be varied by blinding one. two or three of the ocelli [60].

Behavioural sequences may be invoked by one or more stimulus cues. A single stimulus which triggers corpse removal behaviour in ants was found and tested by daubing bits of paper with the acetone extracts from ant corpses, causing them to be removed from the nest and dumped on the same refuse pile as dead ants [77]. Downing and Jeanne found that multiple cues are used to trigger nonlinear building behaviour in nest construction by paper wasps [20]. McFarland and Bösser cite the work of Baerends and Kruijt [9] on egg retrieval behaviour found in herring gulls, in which several cues are used to recognize an egg rolled from its nest, and note that this mechanism of adding cues is an application of the *law of heterogeneous summation*—in which diverse and independent stimuli have an additive effect on behaviour [44]—proposed by Seitz [62].

Perception in Robotics

These examples found in natural systems are also supported by recent changes in approach to control within the field of robotics. Control systems for mobile robots typically follow one of two approaches. Traditionally the connection between sensors and actuators was made through a linear model of perception, representation, reasoning and action. These systems tended to be somewhat brittle due to the way information was processed sequentially. Creating behaviour in machines from sets of stimulus-response pairs, as advocated by Braitenberg as an interesting way to study mind [13], is similar to an alternative approach to robot control proposed by Brooks [14] which made use of a more direct coupling of sensors to actuators mediated by behaviour. These stimulus-response behaviours are used in a parallel decomposition of a task into a set of behaviours, effectively precompiling both the representation and reasoning into task-achieving modules.

Fundamental to the reactive, or behaviour-based, control technique in robotics is this tight coupling between stimulus and response, called action-oriented perception [5]. Rather than creating an internal representation of the environment, to be used by a planning system as is found in traditional AI, the needs of a motor-action are specified in terms of its perceptual requirements.

Pragmatically, several issues concerning sensors must be considered before a study of their relationship to actuation can be made. Sensors provide a robot with a window into its environment that usually carves the world into a number of discrete perceptual spaces. The size of these spaces is dependent on a sensor's modality, resolution, features in the form of perceptual cues, how information is fused, and how many sensors are used. These are the issues of perception, which in a robot amount to sensing through a selection of sensors with widely varying parameters. Choosing the size of this window varies the amount of information, available to the robot, on which to make decisions.

Sensor modality Selecting which type of sensor to use should depend on the perceptual tasks to be accomplished by the robot. For example, in order to successfully navigate, mobile robots require obstacle sensors. Several different types of sensors exist to detect obstacles depending on the range, accuracy, repeatability and reliability required. Each modality has different processing requirements that vary from a single bit contact switch, to over 500,000 bits in a digital camera. Each sensor has limitations which may be overcome by combining multiple sensors of the same or different type.

Sensor resolution In perceptual tasks, how well a specific quantity can be determined or two quantities differentiated, depends on the sensor's resolution. The finer a sensor's resolution, the greater the amount of data it may produce and potentially the greater the processing requirement. The resolution requirement is also task specific. If a task requires the robot to detect one-millimeter-wide cracks in concrete surfaces, then the sensor chosen must be able to resolve distances of less than a millimeter.

Sensor cues Sensor cues, often referred to as perceptual cues or triggers, are features in the robot's perceptual space deemed important by the system's designer. Cues signify an event has occurred and can be used to mark a transition in a control process. Cues may be binary in nature like the illumination of a light indicating the start of a process, or as subtle as a change in shading indicating the movement of a light source. A cue may be defined as a feature in a specific sensor's output space or as a combination of one or more different sensor features. The question of how much information (sensor output) is needed to define a sensor cue is an important one.

Sensor fusion The issue of deciding how information about one stimulus from multiple sensors is to be used is called sensor fusion. Sensor fusion must consider the amount of information from a given sensor as well as its accuracy, modality, and resolution. When the information about a source from different sensors is conflicting, a method to resolve inconsistencies must be included in the fusion algorithm. Choosing which sensors to include in a fusion process is as difficult as deciding what information from those sensors will be used.

Sensor quantity Deciding on the number of sensors to be used for a given perceptual task is usually a pragmatic choice. Cost is often the limiting factor. Sensors of a fixed spatial range can provide more information by using larger quantities. For example, ten contact switches, each capable of detecting contact with a one centimeter squared area, can be used to detect contact with a ten centimeter squared area. Redundant sensors are often used to reduce uncertainty by making use of multiple readings. Often the sensor quantity can not be determined if a method for using the data is not known. Stereo vision employs two cameras and knowledge of their geometric arrangement in order to compute depth information from stereo images of a scene, but how could ten cameras be used and what additional information would they provide?

Operations to Demonstrate Perceptual Modularity

In order to demonstrate that perception can be computed separately from the control decision process, the perceptual cues used in the box-pushing task were specified by measuring sensor output for a given set of stimulus conditions within the task's environment. This data provided a two dimensional view of stimuli for a given sensor and environment. The hypothesis was that the stimulus data sufficiently encodes the correct control decision for an appropriate motor-response action, in a manner similar to the evolved 'matched filter' response found in nature. Next, we consider how perception and action processes are linked together in a cohesive and coordinated manner.

1.2.3 Coherent Behaviour

Does coherent behaviour require direct communication or robot differentiation? Previous studies of coordinated multi-robot box-pushing have often employed direct communication as the explicit mechanism of cooperation [59, 18, 43]. Coordination is achieved by implementing pushing protocols which require the robots to have spatial knowledge and unique functional roles in the pushing task. For example, in [18] a pushing protocol (Protocol I) was developed for a pair of mobile robots uniquely identified as Left and Right. Force information was communicated between the two robots which allowed each to calculate the net torque about a point halfway between the robots. In [59] a similar left/right knowledged strategy was used by two dissimilar robots which used broadcast communication to indicate "pushed-at-left" or "pushed-at-right" actions by each robot. A token passing "my-turn" communication protocol was used in [43] to achieve "careful coordination between the robots" in a cooperative box-pushing task. Although in these examples only two robots were used in the task, each with a unique left/right functional role, it is not clear how these approaches would scale to larger systems, as communication costs escalate as a function of the number of robots.

However, communication as a mechanism of cooperation can improve the performance of some tasks. Balch and Arkin performed simulation studies of three tasks with varying degrees of communication complexity [10]. The tasks studied were *forage, consume*, and *graze* all of which involved, to varying degrees, the spatial coverage of the environment by a multi-robot system. In tasks such as these, performance improvements were made using broadcast communication when alternate communication channels, such as the environment, were unavailable. Furthermore, chemical communication among members of an ant colony is a well understood mechanism for releasing behaviour and thus can help direct a collective

response in many tasks. Thus we acknowledge the benefit direct communication can have in certain tasks, but question whether it is a required component in multi-robot cooperation.

Robot differentiation, the ability to distinguish robots from other objects in the environment, has been postulated by some to be a *necessary condition* for coherent behaviour in multi-robot systems [59, 42]. Also referred to as "robot awareness" or "kin recognition" the ability to discern other robots presupposes that the behaviour of a robot in a group should be different from its singular behaviour. Although this may be true for specific robot architectures, it is not axiomatic in the more general case of decentralized control. Social insects such as ants are a good example of decentralized control that exhibit coherent behaviour in accomplishing several well defined collective tasks, yet no evidence has been found to support the claim that the behaviour of a single ant engaged in an activity is any different when found within its own colony [78]. Thus, it is not clear whether explicit mechanisms like directed communication or robot differentiation are *required* in cooperative control.

To further our understanding of coherent collective behaviour the following primary hypothesis was investigated:

> *Coherent collective behaviour, in some tasks, does not require an explicit mechanism of cooperation.*

The motivation stems from the many well documented cases of coherent behaviour found in social insects which result without use of explicit mechanisms of cooperation.

Social insects exhibit some of their most coherent behaviour during nest building and group transport activities. The activity may be described as a well defined series of steps or behavioural acts, with transitions between steps specified as unique stimulus cues. Nest construction by paper wasps begins by first building a stem, which holds the nest to the bottom of a horizontal surface, to which walls are added, thus forming the first cell. Downing and Jeanne identified the stimulus cues used to cause transition between building acts, and determined that cues may be composed of more than one stimulus [19]. For example, the transition between the stem and roof construction step is specified by stem length, while to determine stem perpendicularity wasps measure both sides of the stem as well as using its reference to gravity [20]. In nest construction, coherent behaviour would seem to have resulted from an evolutionary derived building program whose execution is governed by a common set of perceptual cues.

Group transport behaviour is the cooperative movement of an object by two or more ants. The behaviour is an efficient way for a small workforce to retrieve food items to

the nest [48]. Detailed study of the movement patterns during transport indicated that coordinated movement usually resulted, after a period of antagonistic actions, in response to transport difficulty [67]. Transport items are carried at standard retrieval speeds and a constant relationship exists between the dry weight of the group and the weight of the transported item. Transport is initially started by one ant with others joining in until the standard retrieval speed is reached, after which the group size remains constant. These observations have implied that individuals within the group can assess their performance [24]. Thus, a coherent transport behaviour is a result of simple rules of interaction governing the formation of groups used for food retrieval. With these examples in mind, if explicit mechanisms of cooperation in multi-robot systems are not used, how then is coherence and coordination accomplished?

Task Description and Decomposition

Problems inherent to the design and implementation of multi-robot systems are task description and decomposition, also found in DAI. Of relevance, are some dimensions used for problem description and decomposition [12].

In a problem description, we are interested in capturing both information about the environment and the task to be accomplished, as well potential solution paths that a given multi-robot system can take. The description must also identify potential pitfalls the system may encounter and allow the designer to take these into account during task decomposition. An approach proposed by Wilson [79] to modelling simulated environments describes them as *Sensory State Machines*. Inputs for these machines take the form of actions by a robot with the resulting output as changes in stimulus observable with sensors in the environment. This approach will be modified to create state graphs which describe tasks as stimulus changes to be accomplished while also indicating potential deadlock situations. Arcs in these state graphs will represent possible collective actions taken by the system and its finite set of motion primitives. These problem descriptions will affect the way in which the task is decomposed making description and decomposition iterative tasks [12].

Bond and Gasser [12] have suggested several dimensions along which a problem may be decomposed, a subset of which we translate to the collective robotics domain. In the transport task four dimensions were used for problem decomposition: abstraction levels, control and temporal dependencies, and redundancy.

Abstraction Levels. A task may be viewed from three levels of abstraction. The task-level, which describes *what* is to be accomplished. The behaviour-level, which describes

how the task is to be accomplished. The action-level, which are the primitive actions the system performs for a given task. The task-level description is specified as changes in the environment observable by an external agent with a global perspective. At the behaviour-level, a task is described as a series of steps or subtasks that the system must take in order to accomplish the task. Transitions between these steps are specified as cues from a robot's sensors, which represent local perceptual changes. At the action-level, the above subtasks are further subdivided into task-achieving reactive actions, which accomplish the specified function of the controller using stimulus-response motion primitives.

Control Dependencies. A task may be decomposed by reducing robot control dependencies. This will impact such design decisions as the amount of resources available for a given system size; reducing robot interaction through the use of noninterference protocols; restricting inter-robot communication; and limiting autonomous action based on local perception.

Temporal Dependencies. Task decomposition is also considered in the temporal dimension by encoding time constraints in the environment/task state diagrams. Tasks which require a repetitive sequence of actions with only spatial changes in their parameters are encoded temporally. For example, a wall of bricks is built with a repetitive sequence of actions by changing one spatial parameter, the location of the next brick.

Redundancy. If uncertainty is high in the perception and actions of individual robots, then tasks are decomposed by creating redundancy in the multi-robot system. Thus, systems must be homogeneous in a robot's ability to carry out all steps of a task, or if heterogeneous in composition, then each subtask must still have redundant homogeneous robots.

These guiding principles must next be reduced to research operations that will allow us to gather data in support of our hypothesis on coherent behaviour.

Operations to Demonstrate Coherent Behaviour

In order to demonstrate a coherent group behaviour without using direct communication or robot differentiation, a multi-robot box-pushing system was created under the following assumptions:

- The *transport* task of pushing a box from an unknown initial position to a known goal position will require the net force of at least two robots pushing in the same direction.
- The robots are autonomous, and control is therefore decentralized, with local perception from onboard sensors as the only means of observing the environment.

- No direct communication between robots is possible.
- Robots do not distinguish between objects to be avoided and other robots. The world, from the robot's viewpoint, consists of boxes to be pushed, obstacles to be avoided, and goal destinations.

Coherence, which describes how well the system behaves as a unit, will be measured in terms of the percentage of times the system successfully performs the transport task in a given timeframe. Coordination, which describes how well the system synchronizes its collective actions, will be measured by comparing the time it takes for different numbers of robots to complete the transport task. In performing the task we will vary the number of robots that comprise the system, the size and shape of the object being transported and the goals towards which the system must transport the object.

1.3 Thesis Outline

In Chapter 2, some of the issues that have motivated the field of collective robotics are discussed, concentrating on the typical tasks that have recently emerged and on which our synthetic systems are tested. The conjecture is made that the key to understanding coherent collective behaviour lies in the many well researched examples of decentralized control found in the field of social insects.

In Chapter 3, a taxis-based model of action is introduced. A model is presented for discrete actions composed from three classes of stimulus response behaviours. Primitive actuation is derived from either the class of goal, avoidance or kinesthetically driven behaviours. Examples of action sequences are then used to demonstrate use of the model.

In Chapter 4, a model of perception is presented, called the *perceptual cue* framework, and its approach to compiling control decision information into binary sensing predicates. The model is demonstrated by creating perceptual cues for the box-pushing task used throughout our research to study the connection between local perception and global action.

In Chapter 5, the connection between local perception and global action is made explicit through the use of task description and decomposition. *Q-machines* are introduced to model the process as a three level hierarchy of finite state automata whose execution is controlled by the previously defined perceptual cues. The steps of the transport task, the running example of a collective task, are modelled as Q-machines and the results of testing the individual step-controllers are presented.

In Chapter 6, the results from a number of experiments, both in simulation and on our system of physical mobile robots are presented, which integrate the subtask controllers presented in the previous chapter.

In Chapter 7, the results are discussed for the main question under investigation, namely does coherent behaviour of multiple robots require an explicit mechanism of cooperation? The research contribution is summarized and areas for further study are discussed.

Chapter 2

Motivation

In this dissertation, accomplishing tasks collectively with a set of mobile robots is the area of interest. The specific problem investigated is how to coordinate the actions of several autonomous mobile robots engaged in a directed box-pushing task without using direct communication between robots. As in any new field, this style of experimental robotics is staking out its territory. Often a study will concentrate on one aspect of the problem, leaving the results to be implemented by those conducting research farther on down the line. Some projects, like the one here, take a holistic approach and attempt to incorporate the multidisciplinary results into their systems. This is a breadth-first rather than depth-first approach to the problem.

In this chapter, an overview of work related to the approach taken in collective robotics is presented. A brief survey of multi-robot systems and the collective tasks used in their study is discussed. Described are both physical and simulated systems. Descriptions are brief and concentrate on the functional (task) aspect of the system, while referring to some of the attributes that characterize the approach taken along the dimensions discussed previously. Since task modelling and perception are so closely related in this style of work, they are discussed together in Section 2.3. Finally, in Section 2.4 we consider some of nature's solutions to our problems of interest.

2.1 Introduction

Research in micromachine technology—robots too small to see with the unaided eye—is driven by applications for multi-robot systems from a diverse number of areas including

aerospace, environmental, industrial, marine, and medicine to name a few. In aerospace technology it is envisioned that teams of flying robots may effect satellite repair, and aircraft engine maintenance could be performed by thousands of robots built into the engine eliminating the need for costly disassembly for routine preventative maintenance. Environmental robots are to be used in pipe inspection and pest eradication. While industrial applications include waste disposal and micro cleaners. Ship maintenance and ocean cleaning could be performed by hundreds of underwater robots designed to remove debris from hulls and ocean floors. Fantastic as it may seem, some researchers envision microsurgical robots that could be injected into the body by the hundreds designed to perform specific manipulation tasks without the need for conventional surgical techniques [47]. In order to realize this diverse array of applications, techniques in synergistic control for collective tasks will be required.

2.2 Collective Tasks for Multi-Robots

Recently, three typical tasks used to study multi-robot control have emerged. Tasks include foraging, which involves searching and retrieving a target from a given area, box-pushing, which moves an object between two locations, and formation marching, where robots move while maintaining a fixed pattern. These collective tasks have been studied using both physical robots and simulation. The questions under investigation often ask, should the composition be from a homogeneous or heterogeneous set of robots? What is the size of the system and how many robots are needed to accomplish the task? Does communication help improve task execution speed? What is the most effective control structure for a multi-robot system? And it is through the above three tasks that these questions are often explored.

2.2.1 Collective Foraging

In a collective foraging task robots search and retrieve a target item distributed in their environment. Matarić used a homogeneous system of seven mobile robots used to find and collect randomly distributed pucks [42]. The system makes no use of explicit communication between robots to accomplish the task. The primary objective in use of the foraging task was to investigate the design of collective behavior from a set of basic behaviours based on simple local rules of interaction. Examples of these basic behaviours include obstacle avoidance dispersion, aggregation, following, homing, and flocking [41].

In order to investigate whether communication could improve task execution time, Altenburg used a similar task of collecting targets in an enclosed area using six homogeneous

robots. Task performance, measured in the time taken to complete the task. improved when a simple broadcast type recruitment signal was used. Once a robot located a target it broadcast a signal causing nearby robots to move towards the location. Control was achieved using prioritized rules [2]. Balch and Arkin conducted a simulation study of the foraging task using a homogeneous group of one to five robots, with varying amounts of inter-robot communication. They found that an average improvement of 19 percent, over a noncommunicating system, was to be had when the goal location was broadcast. Control of their system was achieved using motor-schemas and finite state acceptors (explained in the sequel) [10]. Another modified foraging task was studied by Parker using either two or three robots in a communicating heterogeneous team. The task involved locating a cluster of pucks and moving them to a second location. The issue under investigation was fault tolerance in a cooperating heterogeneous team [59].

2.2.2 Multi-Robot Box-Pushing

The above examples of foraging are noncooperative collective tasks, in that they could be performed by one robot given enough time. On the other hand, cooperative tasks such as box-pushing and formation marching, require at least two robots to complete the task. In box-pushing, both traditional AI and reactive approaches have been employed. Box-pushing requires a cooperative effort from at least two robots to move a box along some trajectory.

Traditional approaches decompose the box-pushing task into subtasks to be allocated to individual robots for execution. Caloud, Choi, Latombe, Le Pape and Yim make use of a centralized task planner to communicate with three robots executing a box-pushing task. To coordinate planning and scheduling activities a blackboard system is used [15]. Noreils describes work on a three level decentralized architecture with functional, control and planning levels designed to decompose and allocate the task to a group of robots. An experiment in box-pushing using two robots, one to push and one to supervise, is presented by [53]. Donald, Jennings and Rus describe a box-pushing protocol (Protocol I) for a pair of identical robots identified as Left and Right with explicit communication between the pair to coordinate their actions [18]. In a similar approach by Matarić *et al.* a token passing protocol was used between two robots to coordinate box-pushing actions. In each of these approaches planning is performed either globally with plans communicated to single robots or some combination of global and local planning with conflict resolution being handled centrally.

An alternate approach makes use of a reactive system in which planning has been pre-

compiled into the task description itself. Kube and Zhang report on a undirected box-pushing experiment using a decentralized noncommunicating homogeneous system of five mobile robots. Their system uses reactive control and noninterference as a simple form of cooperation [33]. Parker also describes a box-pushing experiment using a heterogeneous pair of robots in which state information is communicated between the robots as a means of coordinating their actions [58]. Thus, by designing the system for a well specified task and compiling this knowledge into its control system, the more generalized planning stage is avoided.

An interesting approach for a material transport system has been proposed by Stilwell and Bay, in which decentralized control of a group of homogeneous mobile robots equipped with force sensors collectively move a single pallet [65]. Without using communication a leader robot moves the pallet towards a destination with the direction sensed by the other robots using a single force sensor in contact with the load. The distributed control law is tested using computer simulation.

2.2.3 Formation Marching

Formation marching is another common task being investigated. In formation marching a group of robots are to move while maintaining a desired formation. Both Noreils [54] and the ACTRESS project [56] have reported on experiments using two mobile robots moving in tandem. Other formation marching studies using simulation have examined motion based on nearest neighbour tracking [73], local path planning [63], virtual impedance [55], and combinations of local and global information [57].

From the above we may see that most tasks in collective robotics, that have physical implementations, employ fewer than eight robots. Tasks that involve manipulation use either a noncooperative search and retrieval type behaviours (eg. foraging), or cooperative behaviours (eg. box-pushing, formation marching). Both homogeneous and heterogeneous systems have been used. Explicit communication has been shown to improve task performance in tasks where implicit communication through the environment is not possible (eg. foraging); however, no improvement was shown for tasks where implicit communication through the task was possible [10]. The approaches to coordination in box-pushing have primarily relied on direct communication between a pair of robots often uniquely identified in the role they play in task execution.

2.3 Collective Task Modelling and Perception

Few studies have explicitly considered the problem of modelling the task for a collective robotic system. The approach taken can be roughly divided into systems that perform the task collectively as a group with each robot executing the same control program, or systems that perform the task individually using task planning and allocation, with a single robot assigned to each subtask. In most cases, where reactive behaviour is employed in the control system, a set of task-achieving behaviours are designed for the task (i.e. the planning is part of the system design). Input to these systems takes the form of *signals* and *cues*. Signals are usually received through the use of explicit communication between robots. Cues are received by implicit communication using local perception through the robot's onboard sensors. Reactive systems may also make use of memory allowing for state, versus those systems which rely solely on their inputs. Traditional approaches that use task planning and allocation typically use a centralized planning and allocation system which sends messages or protocols to individual robots for execution. This approach requires the use of an explicit communication system to send and receive messages between the global planner and robots.

Matarić has modelled a foraging task using a set of basic behaviours and a modified type of finite state machine [42]. The perceptual system makes use of both signals and cues for transitions between states. Sensors are used as cues to detect the location of targets to be picked up, causing a transition to an acquisition behaviour. Inter-robot communication is used as a signal causing behavioural transition. This type of broadcast communication is similar to alarm communication used in ants [76].

Altenburg has used a set of prioritized rules to describe a foraging task [2]. Both signals and cues are used in the perceptual system. Cues are implemented using sensors to detect obstacles and targets, and can be combined to form transitions between rules. Rules are also triggered by timeouts and signals can be broadcast which cause rules to trigger in other robots. Goal location is determined by light intensity with rules describing necessary preconditions for triggering.

Kube and Zhang have modelled a box-pushing task using a reactive controller and fixed priority behaviours [33]. Perception makes use of cues only with implicit communication by passive sensing. Behaviours are triggered by directly connected sensors with the control system implemented in combinational logic.

Parker has extended the behaviour-based architecture [14] to allow for selection among

task-achieving behaviours [58]. Both signals and cues are employed in behaviour selection. A task in this architecture is composed of loosely coupled independent subtasks which may not be ordered. Subtasks are implemented using behaviour sets whose activation is controlled by a motivational behaviour. Only one behaviour set may be active at a time, and suppression of other behaviour sets is performed by the active behaviour set. Motivational behaviours decide which behaviour set is active based on received input from sensors, inter-robot communication, inhibit lines from other behaviour sets, and two internal state variables, impatience and acquiescence. When a preset threshold is reached the motivational behaviour activates its associated behaviour set. This method of selecting the active behaviour set can not be used if the task requires an ordered sequence of behaviours.

An alternate approach to modelling tasks that does not suffer from ordered behaviour sequencing has been recently proposed by Arkin and MacKenzie for perceptual processes [6]. Their approach controls sequences of perceptual algorithms using a finite state model with transitions between states triggered by either elapsed time, algorithm completion, algorithm failure, or termination of a motor activity. A priority based mechanism is used to handle the simultaneous triggering of several perceptual processes. This state based approach has been applied to the control of a single robot in a docking task and allows for a systematic way of temporally sequencing behaviours as the task proceeds.

The more traditional individual-based task allocation systems are represented by the work of Noreils [53], and of Asama (ACTRESS) [7]. Noreils describes a box-pushing task modelled as coordinated protocols implemented as predicate/transition nets in a three level architecture. At the highest level is a global planner responsible for coordination between local plans and collaboration used for task decomposition and allocation. Protocols which describe the task are composed of requests to the lower functional level and monitors which handle cues from sensors. Tasks are accomplished individually rather than using a collection, by allocating subtasks to individual robots. The ACTRESS architecture is characterized in a similar manner with both centralized and decentralized task planning and allocation. A global model is kept of the environment and message protocols are used to instruct individual robots in subtask execution.

2.4 Collective Biology

Examples abound in nature supporting the conjecture that locally sensed stimulus and reflexive behaviour can produce a predictable global effect. An example is the well defined mushroom shaped termite nest that often stands more than two meters high and one meter

in diameter at its base [68]. Its construction, through a linear series of building steps, is hypothesized to be the result of a building program and stimulus cues used to switch between construction steps, and forms the basis of Grassé's Stigmergy Theory [26]. Can the many examples of perceptual cues, used to trigger behaviour sequences in biological systems, be used to design a similar mechanism for multi-robot control? And can these same cues also be used to govern transitions between task steps in robotic systems the same way they regulate building acts in nest construction? In this section we examine a number of examples with the above two questions in mind.

2.4.1 Nest Construction by Wasps

Nest construction by social insects is a collective task involving a well defined sequence of construction steps. Construction by paper wasps takes place in two stages. In the first stage, a linear series of building acts, or behaviours, are used to construct a petiole, or stem, which holds the nest to the bottom of a horizontal surface, to which walls are added forming the first nest cell. In the second stage, a nonlinear series of building behaviours follow in which either the stem is reinforced, the first cell lengthened, or an additional cell is built [19].

The linear sequence of building acts are:

1. Substrate preparation;
2. Stem construction;
3. Flat sheet construction;
4. First cell construction.

Downing and Jeanne identified the stimulus cues that influenced the transitions between steps and noted that the cues remained consistent within an individual but varied between individuals. For example, they cite the transition between stem and flat sheet construction to be the length of the stem, and that although this length varied from wasp to wasp, an individual wasp would consistently build stems of the same length.

The decisions in the second nonlinear phase of nest construction are more complex since they involve a choice between any one of several building behaviours. Cues used in this phase were composed of more than one sensing stimulus. For example, when constructing the stem of the nest which holds it to a horizontal surface, the wasp measures both sides of the stem to determine its perpendicularity as well as using its reference to gravity [20].

2.4.2 Group Transport by Ants

Nature has graciously provided us, by way of the social insects, an example multi-agent system whose decentralized control is based solely on locally sensed information. Moreover, ants exhibit a group transport behaviour, used in both food and prey retrieval tasks, in which stagnation problems arise and are solved using simple recovery strategies.

Group transport is the cooperative movement of a load by two or more ants. Very few studies have examined this behaviour found almost exclusively in ants, but those that have shown group transport to be an efficient way of moving a load with a small workforce [48, 24, 66, 67]. Food is generally consumed within the nest and must be first torn apart before consumption. Ants must either transport the food item as a whole from its location or dismantle it into small enough pieces to be carried back to the nest by an individual. The efficiency of group transport is evident in Moffett's experiment using a large piece of cereal carried by 14 ants, a food item which would have required 498 ants had individual pieces been carried solitarily [48]. Franks has also determined the efficiency of group transport with ants capable of moving items which are more than the sum of pieces carried by the individual ants in a group [24]. Since items are always carried at a standard retrieval speed, Franks hypothesizes that this superefficiency is obtained by a group's ability to overcome the rotational forces necessary to balance a food item.

A detailed study of the movement patterns involved in group transport was carried out by Sudd in which it was concluded that although the behaviour of ants in a group transport was similar to that of single ants, group transport showed cooperative features [66, 67]. When an ant attempts to move a food item it first tries to carry it. If the item is restrained in any way the ant will next attempt to drag it. Sudd suggests that the resistance to transport determines whether to carry or drag the item. After some seconds are spent on resistance testing, the ant will try to realign the orientation of its body without releasing the item [67]. This has the effect of altering the direction of applied force and may be sufficient to move the food item. If the item still cannot be moved the ant will release its grasp and reposition itself by grasping at another spot. If this final attempt does not result in movement the ant will recruit other ants to the food site. The lighter the load the longer an ant will attempt to move it. Sudd cites an ant will spend up to four minutes before recruitment takes place for items less than 100mg, and up to one minute for items greater than 300mg.

The strategies of realigning, and repositioning are used by ants in the group if during

transport the item gets stuck, and therefore movement stagnates. Once movement begins, the rate of transport increases as time passes due to the increase in frequency of spatial rearrangement, which Sudd suggests results from the ants' response to the reactive forces communicated through the item being transported [67]. Although no numerical data was gathered, Sudd suggests that realignment occurred more frequently than repositioning, which suggests a priority might exist between the two behaviours although sensitivity to increased frictional forces would also explain this observation [66].

2.5 Summary

From the above it can be seen that the study of insect behaviour has much to offer in motivating control mechanisms for multi-robot control. It would seem that nature has evolved a successful approach to the stimulus plethora on which task specific behaviours make their control decisions. How then do the examples presented relate to the design of multi-robot systems?

The problem of a decision process based on locally sensed stimuli can be seen as one of sensor aliasing. In other words, how do you control the perceptual problem of unique stimuli equating to the correct decision? It would seem, from the above examples, that nature has evolved at least four guiding principles useful in limiting sensor aliasing:

- **Environment specific.** By understanding or controlling the stimulus present in the environment, unique behaviour-specific sensors can be designed for the multi-robot system. This means that the environment characterized by its stimulus output is part of the overall solution, which results in an environment-specific robot system.

- **Task decomposition.** Stimuli need only be unique within a subtask, resulting in context dependent meaning. An example found in nest construction is the meaning of light intensity. While the ant is building enclosed walls, light represents a hole to be patched, while the ant is foraging, it governs the starting and stopping of activity. For multi-robot systems this means sensor cues only need to be mutually exclusive to each subtask controller.

- **Orthogonal stimulus.** Combining nonconflicting stimuli into decisions that govern the transition between behavioural acts reduces sensor aliasing. Multiple cues such as the use of both gravity and stem perpendicularity in wasp nest construction make the cue unique.

- **Mass effect.** Since individual behavioural acts are often found to be antagonistic towards progression of a task in nature's reactive systems, successful task completion must rely on mass effect to accomplish its goal. In homogeneous multi-robot systems this means using redundancy to increase the probability of successful task execution.

It remains to be seen, however, whether these perceptual cues can in fact be used to control transitions between task-achieving behaviours in our artificial systems of robots.

Chapter 3

Taxis-based Action

Jander defines insect orientation as "the capacity and activity of controlling location and attitude in space and time with the help of external and internal references i.e. stimuli." [30]. In insects the behavioural act of orientation is controlled either externally, and results in a directional orientation using a taxis mechanism, or internally under kinesthetic control. In this chapter, a model of action is developed based on the taxis mechanisms used in insect orientation. Taxis is defined by Webster's as a reflex translational or orientational movement by a freely motile organism in relation to a source of stimulation [45]. In the model presented, robot actions are based primarily on taxis orientation or kinesthetic orientation as fixed motion patterns. The resulting action model is used to create motor behaviours to be used in a reactive controller. In the model presented, the only required knowledge about the perception side of the robot is that it corresponds to the left and right division of the mobility system used to produce actions. In other words, the input to the action model is the result of the perception of a stimulus, but does not depend on either the stimuli's modality or magnitude. Instead, a boolean decision is made by the perceptual system, presented in the next chapter, which detects the presence of a given stimulus.[1]

3.1 Introduction

In the presented model for action, motion is restricted to translation and rotation in two dimensions. Within the box-pushing environment all robot motor actions, therefore, result

[1] Portions of this chapter have been published. C. Ronald Kube and Hong Zhang 1993. Adaptive Behavior, 2(2):189-219 [34].

in changes in position and orientation with respect to a given coordinate frame. To facilitate a quick response to changes in sensor data, a reactive control system is used for motor actions.

A robot mobility base was designed and built that used differential steering as its means for achieving changes in translation and rotation. Discrete motion primitives were developed to be used as the underlying mechanism for all actions taken by the system. Perceptual processes presented in Chapter 4 are designed independently, but rely on the taxis model and its differential steering method for mobility.

Using the motion primitives, motor behaviours, called primitive actuation behaviours, are developed and form the basis of the task-achieving behaviours presented in Chapter 5. Primitive actuation behaviours are classified into three groups: positive taxis or goal driven, which provide a change in orientation or translation *towards* a stimulus; negative taxis or avoidance driven, which effect a change in orientation or translation *away* from a stimulus: and kinesthetically driven, which execute a fixed action sequence in response to stagnating or deadlock conditions.

3.2 Reactive Control: Insects and Robots

A fast response by a robot to changes in its environment is a necessary criterion when designing any robot that appears to exhibit intelligent behaviour. Controlling robots using a tighter coupling between perception and action was formalized by Brooks in 1986 [14] and since has been referred to as behaviour-based or reactive control. Real-time responsiveness was the prime motivation for the new robot control architecture. Previous mobile robot builders created systems which tried to model the robot's environment internally. Uncertainty in perception and action was dealt with by either engineering it away, the approach taken in SHAKEY at Stanford in the late 60's [52], or by a constant recalibration of the perceptual and actuation systems, the approach taken in the Stanford CART [49]. These robots were slow to respond to changes in sensory conditions. For example, the Stanford CART built by Moravec in the late 70's moved in its environment at four meters an hour. To achieve a fast real-time response to a changing environment, the model that links perception to action should be simple to compute. Nature offers a fast and simple perception to action model in insects.

The title of "fastest recorded movements" in the animal kingdom belongs to the *trap-jaw* ant. The trap-jaw worker ant opens his mandibles 180 degrees and two sensitive trigger hairs project forward so that when they come in contact with an object the mandibles close

in under one millisecond. The spiked tip at the end of the 1.8 millimeter long mandible moves at a velocity of 8.5 meters a second [28]. Hölldobler and Wilson provide the following analogy:

> If the ant were human, its response would be the equivalent of swinging the fist at about 3 kilometers a second—faster than a rifle bullet.

Insect behavioural acts are also triggered by odour sensing. Concern for the detrimental effects of pest control through the use of pesticides motivated the investigation of insect behaviour control by natural products. As the body of knowledge grew on how single behavioural acts in insects could be elicited by a simple *pheromone*—a chemical substance produced by an animal which serves as a stimulus to behavioural response—many scientists conjectured that insect behaviour could be controlled by pheromonal manipulation [25].

Of the two types of pheromones, releaser pheromones invoke an immediate behavioural response [51]. Many programmed behavioural acts can be triggered, with the most understood compounds invoking behaviours of mating, alarm, trail following, attraction and repulsion. Moser demonstrated that the same alarm pheromone serves to attract when found in low concentrations and repel in high concentrations [50]. Behaviours which control direction are referred to as orientation or taxis behaviours. This same dual response to the magnitude of a stimulus is also present in the *phototaxis* orientation behaviours. Phototaxis can either manifest itself as an increase or decrease in turning tendency as the optical stimulus increases in intensity [30].

In insects a taxis mechanism is defined as an externally controlled directional orientation which causes turning movements or changes in position [30]. The taxis mechanisms are classified by sensory modalities. The ability of ants to detect and follow odour gradients is called *osmotropotaxis*. Odour gradients are sensed, possibly exclusively, by the sense organs located on the antennae on the left and right side of an ant's head [76]. This taxis mechanism was proved by Martin to cause orientation in bees towards an attractive ordourant and is achieved by estimating the differential stimulation of the two antennae on either side of the head ([40] cited by Wilson in [76]).

This simple reflex action in which perception is reduced to a taxis was explored in synthetic robots by Braitenberg in his creation of hypothetical vehicles [13]. Attractive behaviours were created in the vehicles by cross connecting left and right side sensors to the opposite right and left side wheel motors respectively. A stimulus presented on the left side would cause the right wheel motor to turn, propelling the vehicle forward and to the

left in a manner similar to the taxis behaviours found in insects. Repulsive behaviours were created in a like manner by connecting same side sensors and wheel motors, thereby causing the robot to move away from the stimulating source. Braitenberg conjectured that more complex behaviours could thus be fabricated by various combinations of stimulus specific sensors and motor connections. The primitive actuation behaviours presented in the sequel are fashioned on these simple taxis mechanisms.

3.3 Achieving Mobility: Getting Around

To achieve changes in position and orientation in an indoor mobile robot a common configuration is differential steering. In differential steering two wheel motors on either side of the robot are driven either forward or reverse with casters placed in the front and rear for stability on smooth surfaces. Other configurations include ackerman steering, synchro-drive, tricycle drive and omni-directional drive [22].

Once the physical drive system and platform configuration is chosen a set of motion primitives is designed and used to cause changes in position and orientation. These discrete actions form the basis for designing the required perceptual processes in an action-oriented model of perception.

3.3.1 Platform Configuration

By configuring two wheel motors, as shown in Figure 3.1, rotation about a point is possible by driving the motors in opposite directions. Both orientation and position, in an x, y coordinate system, can be controlled separately or simultaneously. The same differential steering model used in the physical robot design is also used in the simulation model to calculate changes in position and orientation.

3.3.2 Discrete Actions

A wheel motor is controlled using two parameters: speed and direction of rotation. Speed is proportional to the applied input voltage and a fixed speed setting is used in all motion commands except while applying a pushing force. Continuous motion is accomplished by issuing a series of discrete motion commands, each of which moves the robot a small incremental amount. The commands have the general form: *begin(action), wait* Δt*, end(action).* The motion commands are: *stop, forward, backward, left-turn, right-turn, left-rotate, right-rotate, back-left* and *back-right* as shown in Figure 3.2 and *null* which produces no motor action. Using a fixed speed the motion commands may be specified as a direction of rotation

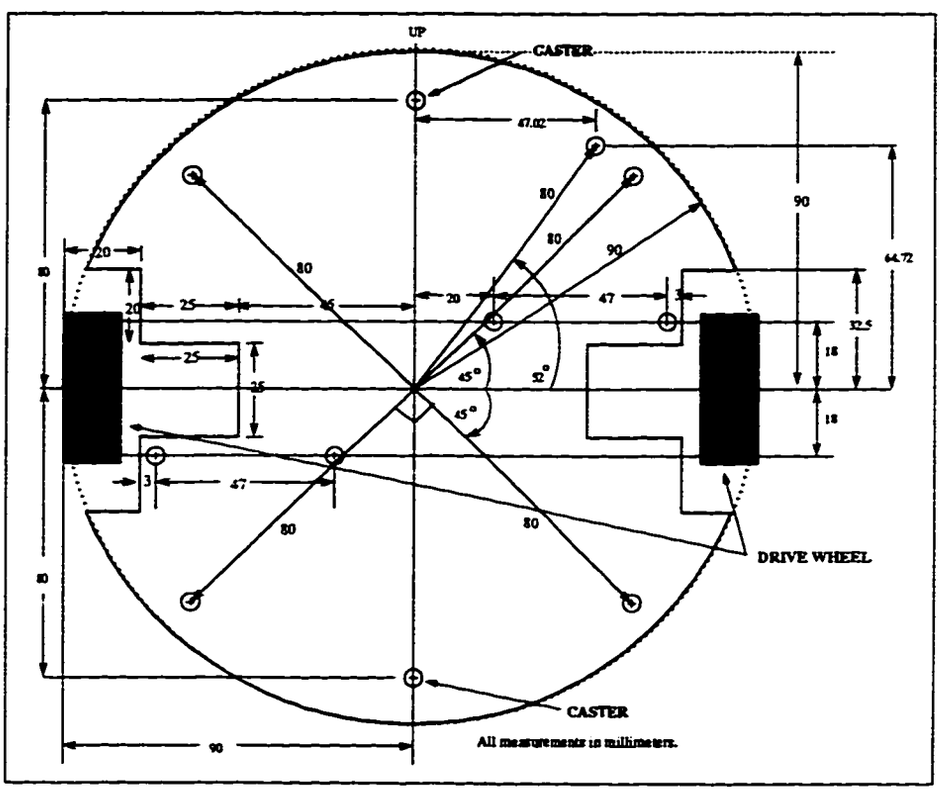


Figure 3.1: A top view of the mobility base with left and right wheel motors used in a differential steering configuration.

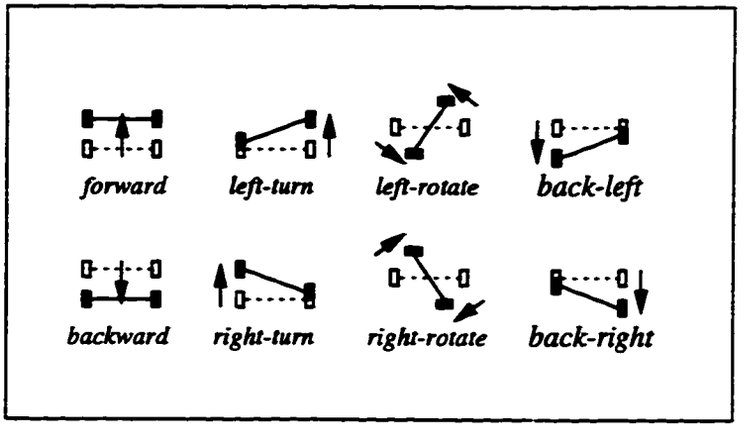


Figure 3.2: The discrete motions possible by issuing several motion commands. Initial positions are shown as dotted lines.

command	*Left Motor*	*Right Motor*
stop	0	0
forward	+1	+1
backward	−1	−1
left-turn	0	+1
right-turn	+1	0
left-rotate	−1	+1
right-rotate	+1	−1
back-left	−1	0
back-right	0	−1

Table 3.1: Using a fixed speed discrete motions are specified as a direction of rotation for the left and right wheel motors.

for the left and right wheel motors using +1 for forward rotation, −1 for backward rotation and 0 for no rotation as listed in Table 3.1. The size or resolution of the discrete action is task-dependent and in the box-pushing task discrete actions are limited to translations of approximately one centimeter and rotations of two degrees.

3.4 Primitive Actuation Behaviours

Reactive behaviours which control a specific set of actuators, are referred to here as primitive actuation (PA) behaviours. In box-pushing the only action a robot is capable of is movement in a plane. As a result, PA behaviours control direction and speed. These motor behaviours are based on a taxis model of orientation in which motor actions are under either external or internal (kinesthetic) control.

Under external control a stimulus can either attract or repel the robot resulting in a positive orientation towards the stimulus, or a negative orientation away from the stimulus. PA behaviours simply map their inputs to one of the robot's motion commands. Behaviours that cause a positive orientation towards the stimulus are said to be *goal driven*, whereas those causing a negative orientation are said to be ***avoidance driven.***

Under internal control, also referred to as ***kinesthetic orientation*** [30], both position and orientation are the result of a fixed sequence of motion primitives. Jander provides several examples in which an insect is capable of returning to the nest "by remembering and kinesthetically controlling its movements" [30]. In the sequel a detailed example of this type of orientation that occurs in ant prey transport will be discussed. Behaviours that execute a fixed sequence of actions are used to recover from stagnating or deadlock conditions and are said to be ***kinesthetically driven.***

Since each motion primitive controls a left and right wheel motor, PA behaviours that change direction use left and right stimulus pairs. In short, for motions used in box-pushing PA behaviours can be divided into three classes corresponding to the type of orientation employed:

Positive Taxis Goal driven behaviours used to attract the robot towards a given stimulus.

Negative Taxis Avoidance driven behaviours which repel the robot from a given stimulus.

Kinesthetic Orientation Behaviours used to recover from stagnating conditions by executing fixed patterns of motion primitives.

In the sections that follow, motor behaviours are developed for the box-pushing task (and later extended to the transport task) based on the taxis model of orientation found in insects. Goal and avoidance driven behaviours receive external input from sensors and correspond to the positive and negative orientation taxis mechanisms previously mentioned. Kinesthetically driven behaviours do not have external input, but rather are internally controlled using kinesthetic orientation.

3.4.1 Positive and Negative Taxis Orientation

A positive taxis or goal driven behaviour moves the robot towards a given external stimulus. Input to the behaviour takes the form of a left and right divided stimulus pair which may correspond to left and right sensors on the robot. The input variables to the behaviour are boolean and indicate the presence or absence of the stimulus within a given range

```
: SEEK-BOX  ( left_flag, right_flag --- )
     IF  !left_flag AND !right_flag  THEN  NULL-MOTION
     IF  !left_flag AND  right_flag  THEN  RIGHT-TURN
     IF   left_flag AND !right_flag  THEN  LEFT-TURN
     IF   left_flag AND  right_flag  THEN  FORWARD
;
```

Figure 3.3: Shown is the pseudo-code for the SEEK-BOX motor behaviour. Input is from left and right stimulus pairs used to determine the direction of the stimulus. The behaviour *turns* the robot towards the stimulus.

```
: PUSH-BOX  ( flag --- )
     IF  flag  THEN
        HIGH-SPEED FORWARD
;
```

Figure 3.4: Shown is the pseudo-code for the PUSH-BOX motor behaviour. The behaviour once triggered increases the wheel motor speed and moves the robot forward.

and field-of-view. Output from the behaviour is a motion command selected from a set of four commands representing the possible number of input combinations. In the case of a behaviour with a single input variable, 0 is mapped to the *null* motion command and 1 is mapped to the *forward* command. For the box-pushing task two goal driven behaviours shown in Figure 3.3 and Figure 3.4 are specified:

- SEEK-BOX - moves the robot towards a box.
- PUSH-BOX - pushes the box by increasing motor voltage.

In the same manner negative taxis or avoidance driven behaviour repels a robot from a given stimulus. Inputs of two binary values correspond to a left and right stimulus pair, whereas single value inputs are mapped to the *null* motion for an input of binary 0 and a *backward* motion command for an input of binary 1. For the box-pushing task the two avoidance driven behaviours shown in Figure 3.5 and Figure 3.6 are specified:

- AVOID - *turns* the robot away from obstacles.
- CONTACT - *rotates* the robot away from obstacles.

The motor behaviours which cause changes in orientation are summarized in Table 3.2.

In Chapter 5 the PA behaviours listed here will form the basis of task driven controllers. As an example, a simple box-pushing controller for use in an environment in which box

```
: AVOID  ( left_flag, right_flag --- )
     IF  !left_flag AND !right_flag  THEN  NULL-MOTION
     IF  !left_flag AND  right_flag  THEN  LEFT-TURN
     IF   left_flag AND !right_flag  THEN  RIGHT-TURN
     IF   left_flag AND  right_flag  THEN  RIGHT-TURN
;
```

Figure 3.5: Shown is the pseudo-code for the AVOID motor behaviour. Input is from left and right stimulus pairs used to determine the direction of the stimulus. The behaviour *turns* the robot away from the stimulus.

```
: CONTACT  ( left_flag, right_flag --- )
     IF  !left_flag AND !right_flag  THEN  NULL-MOTION
     IF  !left_flag AND  right_flag  THEN  LEFT-ROTATE
     IF   left_flag AND !right_flag  THEN  RIGHT-ROTATE
     IF   left_flag AND  right_flag  THEN  RIGHT-ROTATE
;
```

Figure 3.6: Shown is the pseudo-code for the CONTACT motor behaviour. Input is from left and right stimulus pairs used to determine the direction of the stimulus. The behaviour *rotates* the robot away from the stimulus.

Positive and Negative Taxis Mappings				
Stimulus		Negative Taxis		Positive Taxis
L	R	AVOID	CONTACT	SEEK-BOX
0	0	*null*	*null*	*null*
0	1	*left-turn*	*left-rotate*	*right-turn*
1	0	*right-turn*	*right-rotate*	*left-turn*
1	1	*right-turn*	*right-rotate*	*forward*

Table 3.2: The positive and negative taxis behaviour mappings. Behaviours that cause directional changes based on external stimuli expect a stimulus from the left and right sides of the robot similar to stimulus sensing found in insects. The "null" output means the behaviour doesn't produce a motion command.

sensing is not limited in range or direction consists of the SEEK-BOX and AVOID behaviours. This controller produces collision free navigation in the simulated environment shown in Figure 3.7.

A problem occurs when the robots surround the box applying an equal pushing force. In cases where an insufficient net force is applied to move the box, a deadlock or stagnating condition arises with no means of resolution possible using the given controller. The problem is common to reactive controllers and is analogous to finding a local maximum when using a hill climbing strategy in AI search problems. One possible solution to this type of stagnating condition involves kinesthetically driven behaviours. The stagnating condition is detected and a fixed sequence of action is performed. The approach is presented next.

3.4.2 Kinesthetic Orientation

Kinesthetic orientation serves two purposes here: motion in the absence of external stimuli and stagnation recovery movements. In the case of both positive and negative taxis, orientation of the robot is under control of external stimuli.[2] At any time the motor behaviour relies on an external stimulus to decide the correct response in orientation. However, many behavioural acts in both insects and robots lack the external stimulus needed to guide the orientation mechanism. Rather a correct behavioural response might simply be a fixed pattern of motor activity stored in memory and released under suitable conditions. For example, a spider can return to a given location by "remembering and kinesthetically controlling its movements," a skill also found in bees and ants [30].

In the absence of stimuli, a fixed pattern of motor activity can serve as a strategy while foraging for food or searching for a goal. For instance, when an ant leaves its nest to search for food it leaves in a straight line until it encounters either food or an odour trail which it then follows using a positive odour-taxis mechanism [78]. In box-pushing, a search strategy called RANDOM-WALK is used which keeps the robot moving in a forward direction by issuing a sequence of motion primitives (see Figure 3.8). Continuous motion by the robot in the absence of any external stimulus is thus accomplished.

Recovery from deadlock or stagnation is the second use of kinesthetic orientation. During the execution of a task by robots using reactive control strategies, the absence of a plan can result in a condition in which the execution of the task gets stuck or is said to stagnate. For example, a dead end is reached by a robot trying to navigate to a given goal as in Arkin's

[2]Portions of this section have been published. C. Ronald Kube and Hong Zhang 1994. IEEE IROS, 3:1883-1890 [35].

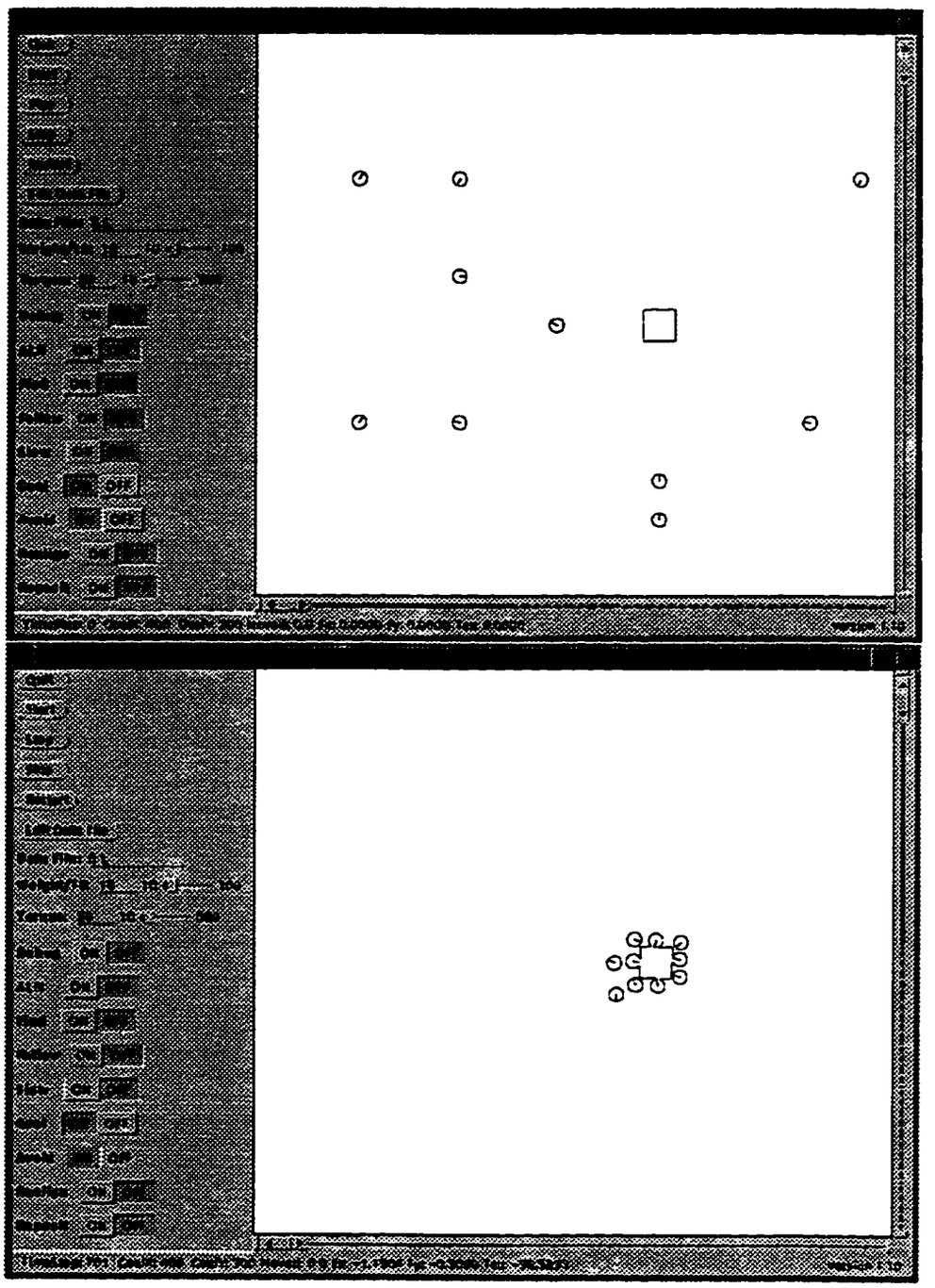


Figure 3.7: Shown are the initial and final configuration of 10 simulated robots (circles) under the control of two motor behaviours, SEEK-BOX and AVOID. The output from the AVOID behaviour has priority over SEEK-BOX if obstacles are detected. With this initial configuration the system reaches stagnation without the box changing position.

```
: RANDOM-WALK  ( --- )
    IF turn_counter < max_turns  THEN
        IF turn_flag  THEN
            LEFT-ROTATE
        ELSE
            RIGHT-ROTATE
    ELSE
        FORWARD
        INCREMENT turn_counter
;
```

Figure 3.8: The pseudo-code for the RANDOM-WALK motor behaviour. The behaviour causes the robot to swagger while constantly moving forward.

box canyon problem [17]. The problem is similar to finding a local maximum, encountered by hill-climbing algorithms, when the goal is to find the global maximum.

The problem of stagnation also occurs in nondirected box-pushing where the goal is to push the box in an unspecified direction. For example, in a box-pushing task the net force applied by the robots may equal zero if the robots are evenly distributed around the perimeter of the box as shown in Figure 3.7. In such a case, a robot might attempt indefinitely to push the box unsuccessfully. An equivalent problem can be found in nature among ants displaying a group transport behaviour [48]. How do ants equipped with simple sensory-response behaviours deal with the stagnation that results when the item they are transporting becomes stuck?

Group Transport by Ants

As was discussed in Chapter 2, group transport is the cooperative movement of a load by two or more ants. The strategies of realigning, and repositioning are used by ants in the group if during transport the item gets stuck, and therefore movement stagnates. Similar stagnation recovery strategies are designed here for box-pushing and illustrated in Figures 3.9, 3.11 and Figure 3.10.

The strategies employed by ants to handle task stagnation—a condition that occurs when an item being carried gets stuck during a group transport task—can be viewed as stored behaviours designed to overcome difficulty. Activated as a response to increased frictional forces, the behaviours are used by ants both in group transport and during individual transport of food items. These behaviours appear to be ordered in their application. For example, Sudd notes realignment seemed to occur more frequently than repositioning

```
: BACK-OFF  ( --- )
     FOR i=1 to i=8 DO
         BACKWARD
     FOR i=1 to i=6 DO
         RIGHT-ROTATE
;
```

Figure 3.9: The pseudo-code for the BACK-OFF motor behaviour. Once triggered it causes the robot to backup and rotate towards the right.

```
: REALIGNMENT  ( --- )
     IF  rand > 0.5 THEN
         RIGHT-ROTATE
     ELSE
         LEFT-ROTATE
;
```

Figure 3.10: The pseudo-code for the REALIGNMENT motor behavior. The behaviour causes a robot to change its pushing angle by a small random amount.

with the former being applied as the first response to the increase in frictional forces [66]. Simulation was used to compare the strategies of realignment and repositioning using our simulation environment[34] and nondirected box-pushing [35] with the results presented in Chapter 5.

The recovery behaviours increase the task success rate by providing a strategy for dealing with deadlock situations. These results motivated the use of the kinesthetically driven recovery behaviours used in directed box-pushing. For box-pushing the kinesthetically driven behaviours are:

- RANDOM-WALK - causes the robot to move forward in an "S" pattern.
- BACK-OFF - causes the robot to back away from objects contacted by its touch sensor.
- REPOSITION - moves the robot in a backward arc.
- REALIGNMENT - changes the pushing angle when in contact with the box.

3.5 Summary

In insects, translation and orientation is accomplished by way of a taxis mechanism and is a fast and simple response to external and internal stimuli. In robots, that need to have

```
: REPOSITION  ( --- )
     FOR i=1 to i=6 DO
         BACKWARD
     FOR i=1 to i=3 DO
         RIGHT-TURN
     FOR i=1 to i=4 DO
         FORWARD
;
```

Figure 3.11: The pseudo-code for the REPOSITION motor behavior. The behaviour causes a robot to move a fixed distance counterclockwise.

a real-time response to changing environmental stimuli, a reactive model for action based on taxis is simple to compute. And by defining the input stimulus to both positive and negative taxes as binary values, the robot's action is decoupled from both the modality and magnitude of the stimulus, resulting in a clean abstraction suitable for component change or replacement.

The three forms of orientation: positive and negative taxes, under external control of stimuli, and kinesthetic orientation, under the internal control of fixed action sequences. form the basis of reactive motor behaviours referred to here as primitive actuation behaviours. Along with the material on perception developed in Chapter 4, PA behaviours form the building blocks of finite state machines, called *Q-machines*, discussed in Chapter 5.

However, taxis by itself suffers the same problems of stagnation (local minima/maxima) as do other reactive control methods. Kinesthetic orientation, in the form of fixed action sequences and triggered by either the presence or absence of a controlling stimulus, is one solution suitable to the stagnating conditions in the box-pushing task.

Processing the input stimuli to the PA behaviours is the role of perception and the subject of the next chapter, in which a model for local perception is developed called *perceptual cues*. Can perception be reduced to the yes/no type of binary inputs required by the motor behaviours developed in this chapter? And can the process of perception be computed for a reactive controller independent of the action primitive it is used with? Perceptual cues provide such a model and are based largely on further examples motivated by the social insects.

Chapter 4

Local Perception

Given a robot with predefined motor actions, the role of perception is to determine what actions take place and when. In this style of action-oriented perception, the action defines the form of the perception in terms of what information is needed for the action to make its control decision. In this Chapter, an assumption is made that local sensing can be used to decode information present in the environment that signifies the appropriate motor action. This presupposes that the environment and its stimuli are part of the system design process, and can be characterized through a given set of physical sensors. In the next chapter, perceptual cues are combined with primitive actuation behaviours to produce subtask controllers that implement the control system for a single robot. At the task-level, the subtask controller represents task state information with perceptual cues used to control transitions between states.[1]

4.1 Introduction

Perception is used to help decide what action the robot should take and when it should be performed. In the perceptual cue framework, the task environment encodes information in its stimulus output on what actions the robot should take and when to take them. In this manner, perceptual cues simply decode this information. Since errors occur in decoding this information locally, the mass effect of distributed sensing increases the probability of at least some of the robots correctly decoding the stimulus. Two feature extraction techniques,

[1] Portions of this chapter have been submitted for publication. C. Ronald Kube and Hong Zhang 1995. Robotica - special issue on Interacting Robots, 9 pages [36]

presented in the sequel, simplify the decoding process. The approach is demonstrated by designing the perceptual cues for a multi-robot box-pushing task. Therefore, deciding when a robot should act in a given task involves designing a system with both the environment and the robot as part of the solution.

For an environment to encode information, Wilson suggests it be considered as a type of machine with inputs and outputs [79]. On the input side are the actions performed by the robots on the environment, which responds with changes in stimuli as its output. The model for the robots considers stimulus changes on its input side with motor actions as output. As a task progresses in execution, changes in stimulus from the environment serve to guide the robot's action selection process. How the environment can be modelled to encode information about task execution is covered in more detail in Chapter 5. For now let us assume that it is, and the question is how the information is decoded and used in action selection. Action selection is what a robot does when a motor behaviour is activated. When that action is performed depends on the current state in task execution.

Perceptual cues are used to decode the information in an environment and decide which motor behaviour is activated. Behaviour activation decides what action is performed, but depends on the state of the robot's task controller. The relationship between local sensing, behavioural state and the output actions of the robot will be fully explained in the next chapter. State changes of a task controller are also controlled by perceptual cues that uniquely extract features from sensor data. However, since perception is local to a robot, global action by the system depends on mass effect.

The connection between local perception and global action is through the mass effect of a redundant system of mobile robots. For any one robot, its locally derived perception may not decode the environment completely, due to limitations imposed by the robot's position within the environment (a spatial constraint). Nevertheless, since sensing in such a spatially distributed system increases the probability that some of the robots correctly respond to the environment, then the actions performed on the environment by those robots may allow others to sense stimulus changes. For example, a robot unable to sense an object that is outside the range of its sensors, may sense the object as it is pushed into its sensor range by other robots. How local perception decodes stimuli depends on two approaches to integrating sensor data.

The first approach to sensor integration involves two orthogonal sensing strategies: spatially and modally orthogonal sensors. Spatially orthogonal refers to a geometric arrangement of like sensors which carves the robot's perceptual field-of-view into discrete non-

overlapping regions. Modally orthogonal is the integration of sensor data from dissimilar sensor types. By combining sensor data using these approaches, features from the environment's stimulus output are extracted and used by the robot's motor decision process.

The second method of integrating sensor data uses previously defined perceptual cues *additively* by concatenating binary decisions (cue outputs) into vectors that can be used to control state transitions in the robot's task model. Box-pushing is the task used to demonstrate the feasibility of the perceptual cue framework, by defining the cues used in the task independently from the actions performed by the robots. In this manner, what the system of robots is to do is defined by a robot's task controller, but how the robots accomplish the task is not explicitly defined. The end result is a predictable task completion, but the solution path taken and performed by the intermediate steps is not unique.

The remainder of this chapter presents the details of the model by describing both the definition and function of perceptual cues. An example task is then presented in detail which outlines the steps involved in specifying the cues for a given environment. Finally, a brief summary of the framework and how it fits into the remaining chapters follows.

4.2 Perceptual Cue Framework

Defining and specifying perceptual cues involves three techniques for cue creation: feature extraction using threshold logic; orthogonal sensing as a means for integrating physical sensors; and additive cue construction specified as clauses in predicate calculus. The result is cues that answer yes/no type questions about what can be sensed in the robot's immediate vicinity. Functionally, perceptual cues are used for either activating motor behaviours or for causing state transitions among the robot's subtask controllers. Consequently, the framework can be summarized as a way of determining the "what and when" of robot action sequences.

4.2.1 Perceptual Cue Definition

A perceptual cue is a boolean value which indicates either the presence or absence of a pattern of stimuli. Perceptual cues (PCs) are context dependent features in sensor data which indicate a perceived event. Context is determined by the current state in task execution space. States in task execution space are specified as steps in the task and implemented as subtask controllers explained fully in the next chapter. At the level of task description, PCs are used to determine which step of the task is being executed. Each subtask controller consists of a finite number of states, where each state is associated with a certain motor

action and implemented as a primitive actuation behaviour. Within the PA behaviours. sensor features detected by a perceptual cue map directly to motor actions. Features are obtained by processing sensor data to produce a binary output. Sensor data is acquired from single or multiple sensors and is processed using simple threshold logic. Cues can be created by using data from different sensor types combined using boolean operators. Cues are context dependent in that they are specified for a specific task and a given environment. Sensor features which are not unique can be combined *orthogonally* or *additively*, as explained in the sequel, to produce a unique feature. Perceptual cues are binary vectors created by combining features extracted from sensor data using three techniques: threshold logic, orthogonal sensing and additive cue construction.

Feature Extraction Using Threshold Logic

A crude form of feature extraction is a threshold function provided that a monotonic relationship exists between the sensor's analog output and the parameter of interest. For example, the output of a light sensor is a function of the intensity of the light source. If the radiant energy from a light source falls normal to the surface of the sensor, then a correlation can be made between the magnitude of the signal from the sensor and the distance to the light source. If a cue is to be created which detects when the robot is within a certain distance of the light source, then a threshold is specified which corresponds to the magnitude of the signal at the desired distance. Exceeding this threshold triggers the cue and produces a "1" as its output bit. The assumption, for such a mechanism to work, is that the robot is working in a known environment. Thus, when using threshold logic for feature extraction, both the environment and the sensor's response to it are considered by the system designer as part of the solution.

When a nonmonotonic relationship between sensor output and a parameter exists the data is partitioned into linear segments.[2] Cues are then created using threshold logic as before on each linear segment with the results combined using the boolean operators AND, OR, and NOT. For example, a signal peak-detection cue, described in Section 4.3.4, is created by combining a cue that detects a rise in signal magnitude greater than a threshold followed by a cue that detects a fall in signal magnitude within a fixed period of time. The two cues are combined using the AND operator. The signal peak-detection cue is true if a fall is detected after a rise in signal magnitude which is greater than a given threshold. When perceptual cues are created from two or more sensors the resulting binary features

[2]This does not preclude the use of other statistical functions for feature extraction.

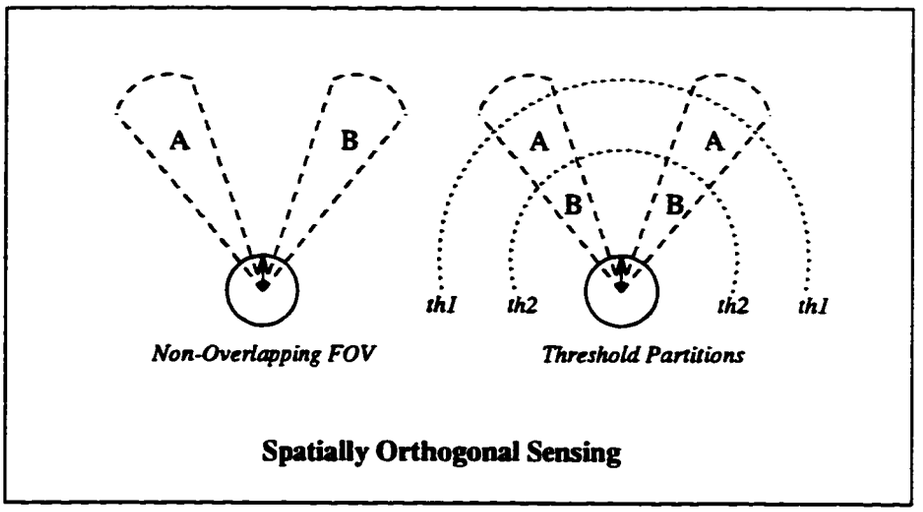


Figure 4.1: Sensing can be made spatially orthogonal by either arranging the same type of sensors geometrically with nonoverlapping fields-of-view or by partitioning the field-of-view with thresholds.

are combined using orthogonal sensing strategies.

Orthogonal Sensing

In order to simplify sensor processing, binary cues created using threshold logic can be integrated by employing either spatially or modally orthogonal sensing strategies. The result integrates multiple sensors of the same type geometrically, by spatially partitioning the robot's perceptual field-of-view. Sensors of different type are combined to create cues in which all bit positions in the output vector are from dissimilar stimulus modalities. Their combination makes the extracted sensor feature temporally unique.

Sensors of the same type can be made *spatially orthogonal* either by geometric arrangement, with nonoverlapping fields of view, or by partitioning the sensor's range discretely using different threshold values as shown in Figure 4.1. As an example of a spatially orthogonal sensor, consider a ring of eight sensors, each with a 45 degree field-of-view and equally spaced on a circle.[3] The perceptual space is divided into eight discrete zones in which stimuli may be detected. If obstacle sensors were used, then each bit of an 8-bit vector could represent the presence of an obstacle within the assigned zone. Thus, 256 possible combinations are available for mapping to motor actions used in obstacle avoidance.

A perceptual cue that is *modally orthogonal* is specified by taking the binary outputs of sensors with incompatible outputs like temperature and contact sensing, or range and

[3] A common configuration found in commercial mobile platforms.

odometry data. The outputs are combined using boolean operators resulting in a unique feature in the sensor's output space from sensors of different modalities. For example, to detect the side of a brightly lit box both touch and light intensity are used since their combination is unique in the box-pushing environment.

Additive Cue Construction

Perceptual cues can also be defined by combining cues additively as a Horn clause. In predicate calculus a *Horn clause* is any disjunction of the form:

$$\neg A \vee \neg B \vee \cdots \vee \neg C \vee D$$

Each Horn clause has at most one positive literal, and can be rewritten as an equivalent implicational formula:

$$A \wedge B \wedge \cdots \wedge C \rightarrow D$$

The above formula is a notational variant of Horn clauses used in Logic Programming. The newly defined cue is the consequent (variable D) of previously defined cues (variables $A, B, \ldots, C$) which are the antecedents of the general form:

antecedent *1*, $\cdots$, antecedent *n*. $\rightarrow$ consequent.

Cues defined in this manner define the state in the task model. Cues are also used for behaviour activation as explained next.

4.2.2 Perceptual Cue Function

A perceptual cue is a control decision used to trigger a motor behaviour and to control the transition among states in a task model. Motor behaviours remain active for a fixed period of time, at the end of which the cue's truth value is reevaluated. Either the same cue is applicable or the stimulus conditions have changed, thereby activating another cue. In executing a robot task, defined as a multistep procedure, stimulus conditions may also change sufficiently to indicate a state transition, or change in activity, where "state" represents a separate motion controller designed to accomplish one step in a task description. Using cues to trigger a behavioural response is a common mechanism for action in social insects [51] and for governing different phases of activity in tasks such as nest building [19].

The advantage of reducing motor behaviour control decisions to binary values is in the cue's functional abstraction. In this manner, activation of a motor behaviour is not

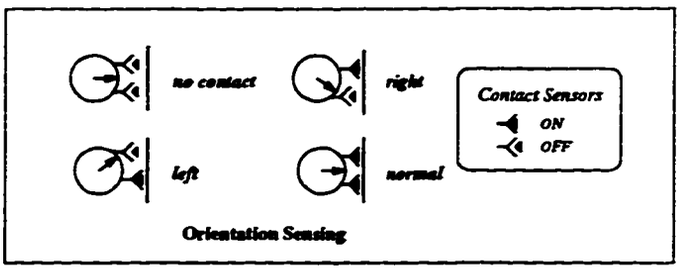


Figure 4.2: To maintain a normal orientation with respect to a surface, left and right contact information is used. The information can be provided by either touch sensors as illustrated or another sensor modality such as in phototaxis.

dependent on a specific perceptual cue, but rather on the decision that results from sensor processing. For example, a motor behaviour created to keep a robot in perpendicular contact with the surface of an object, relies on sensor information from either side of the point of contact. If touch sensors are used then perpendicularity about the point of contact could be specified when both touch sensors are in contact with the surface as illustrated in Figure 4.2, and return a binary '11' value. Contact with the left side only would be represented as binary '10' and contact with the right as a '01' value, with the no contact condition specified as a '00' value. The same contact information using light sensors and phototaxis could be specified by determining the sensor's threshold value when in contact with a surface and creating cues that return '11' when the robot is in perpendicular contact with the surface in a similar manner. The advantage is that the design of the motor behaviour does not change when different sensor types or alternate feature extraction techniques are used since the information needed by the motor behaviour is the same binary vector in both cases.

In short, the function of perceptual cues is to control behaviour activation and state transitions in a manner that allows for changes in perception design and implementation without affecting the control architecture's connection to motor action.

Controlling Behaviour Activation

Behaviour activation refers to the process of deciding which behaviour is to become active in the current context (i.e. in the currently executing controller). The decision as to which action the robot performs is made by the primitive actuation behaviours described previously in Chapter 3. Each behaviour has an associated perceptual cue that *activates* the behaviour to produce a motion command as output. More than one behaviour may become active during the control loop. A priority scheme among the behaviours within the current executing controller determines which action is executed by the robot. Thus, in a

known environment a robot's action is based on a perceptual process that uses local sensing to look for specific features in sensor data.

Controlling Task State Transitions

Perceptual cues used to control state transitions in task execution are specified as predicates with perceptual preconditions that must be satisfied. Each task is decomposed into subtasks and a controller is designed for each subtask. Control system processing is handled in discrete steps, with control either remaining within the current subtask controller or passing on to the next one, as specified in the task model digraph using a forward (FL) or repeat (RL) edge. The cue used for transition in each subtask controller, or step i, is related to its predecessor by:

$$FL_i = FL_{i-1} \wedge c_i \tag{4.1}$$

for $i = 1, 2, \ldots, n$ where $n =$ the number of subtasks and c_i is a new perceptual cue for step i. How each perceptual cue, c_i, is computed for the box-pushing task is explained in the next section.

$$RL_i = FL_{i-1} \wedge \neg c_i \tag{4.2}$$

Specified in this manner the forward edge, illustrated in Figure 4.3, is the cue signalling step transition and signifies that a locally detectable event has occurred indicating step completion. The repeat edge indicates the current action is to be repeated since the specified change in stimulus (the detectable event) has not occurred.

When a task is modelled as a multistep procedure, with each step represented as a state in a task digraph, explained in chapter 5, then the current state (ST) is specified as a logical AND of the perceptual cues, for $i = 1, 2, \ldots, n$ where $n =$ the number of subtasks:

$$\begin{aligned}
ST_1 &\leftarrow \neg c_1 \\
ST_2 &\leftarrow c_1 \wedge \neg c_2 \\
ST_3 &\leftarrow c_1 \wedge c_2 \wedge \neg c_3 \\
&\vdots \quad\quad \vdots \\
ST_{n-1} &\leftarrow c_1 \wedge c_2 \wedge c_3 \wedge \cdots \wedge \neg c_{n-1} \\
ST_n &\leftarrow c_1 \wedge c_2 \wedge c_3 \wedge \cdots \wedge c_{n-1}
\end{aligned}$$

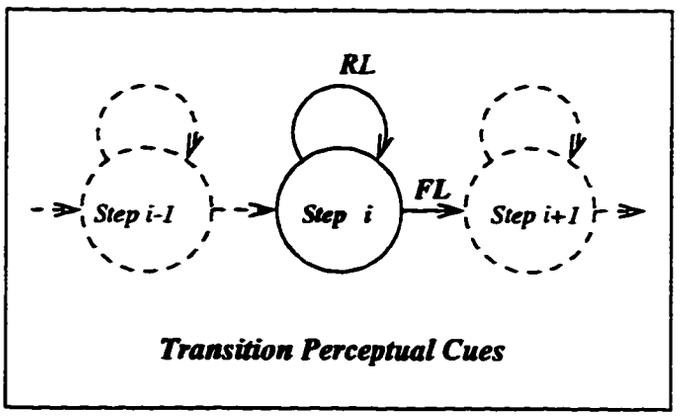


Figure 4.3: Each step in the task is modelled as a state in a finite state machine with perceptual cues used for state transition. The perceptual cue causing a forward transition (FL) is simply a concatenation of another boolean variable to the previous step's forward perceptual cue.

In Chapter 5 task modelling will be presented with examples of the above cues used for state transition. Next is presented an implementation of the perceptual cue model in a multi-robot box-pushing task.

4.3 Perceptual Cues for Box-Pushing

Transporting a box from an unknown initial position towards a final goal destination was modelled using three types of perceptual cues. Obstacle avoidance cues were used to detect an obstacle and trigger avoidance behaviours. Box detection cues were used to locate and track a moving box, as well as, to control state transitions among the task step controllers. And a goal detection cue was used to indicate proper robot orientation, with respect to the goal, for a pushing behaviour. The cues are designed with a given set of motor actions in mind. The design and implementation of each perceptual cue involve the following steps:

1. **Sensor Placement** Given a sensor type, determine the position, orientation and number of sensors to be used in the sensor system.

2. **Data Collection** For a given environment, collect data from the sensor that represents the condition under which the task is performed.

3. **Data Analysis** Determine what features of the data may be used to meet the perceptual cue's specification.

4. **Algorithm Design** Design an algorithm to extract the desired feature.

5. **Algorithm Verification** Specify the tests to verify that the cue performs as designed.

In the next chapter, these perceptual cues will be used to control the states of a robot as it executes the transport task.

4.3.1 Physical Sensors for Transporting a Box

In choosing a minimal set of sensors for the transport task, the robot's activities of avoiding obstacles, locating the box to be moved, and pushing it to a goal location, are considered. In Chapter 3 the actions that each behaviour could take were enumerated, with inputs to the behaviours specified as binary input variables. This establishes the minimal number of binary variables that each behaviour uses in mapping perception to actions. For example, the possible actions of the AVOID behaviour are *idle, left-turn, right-turn* therefore requiring two binary input variables allowing for a maximum of four actions. In a similar manner, the box locating behaviours use two input variables and the pushing behaviours use one. Although several types of sensors are available for mobile robots,[4] optical sensors were used in each of three activities mentioned above. To implement obstacle avoidance infrared emitter/detectors, whose output is dependent on the magnitude of the reflected energy, along with contact sensors were used. Box tracking behaviours made use of photocells, which vary in resistance as a function of light intensity, and contact sensors. Goal direction behaviours used phototransistors whose output current is a function of light intensity. The fact that the sensors are commonly available, inexpensive, and easy to use guided our decision. What follows is a brief discussion of their characteristics.

Infrared Photo Emitters and Detectors[5]

Infrared radiation is electromagnetic energy with a wavelength longer than visible red (i.e. in the range from 770 to 1500 nanometers). Two optoelectronic devices that make use of this energy are the Infrared-Emitting Diode (IRED), and the Infrared Phototransistor. Two types of IRED radiant-energy sources are Gallium Arsenide (GaAs) and Gallium Aluminum Arsenide (GaAlAs), which emit in the 940 nm and 820 nm portion of the near-infrared spectrum respectively. Infrared phototransistors are simply transistors designed to be responsive to this radiant energy. Figure 4.4 shows the relative spectral characteristics of the human eye, a Silicon Phototransistor, two types of Infrared Light Emitting Diodes, and a Tungsten light source.

[4] See H. R. Everett's book [22] for a recent survey.

[5] Portions of this section have been published. C. Ronald Kube 1996. The Robotics Practitioner, 2(2):15-20. [32].

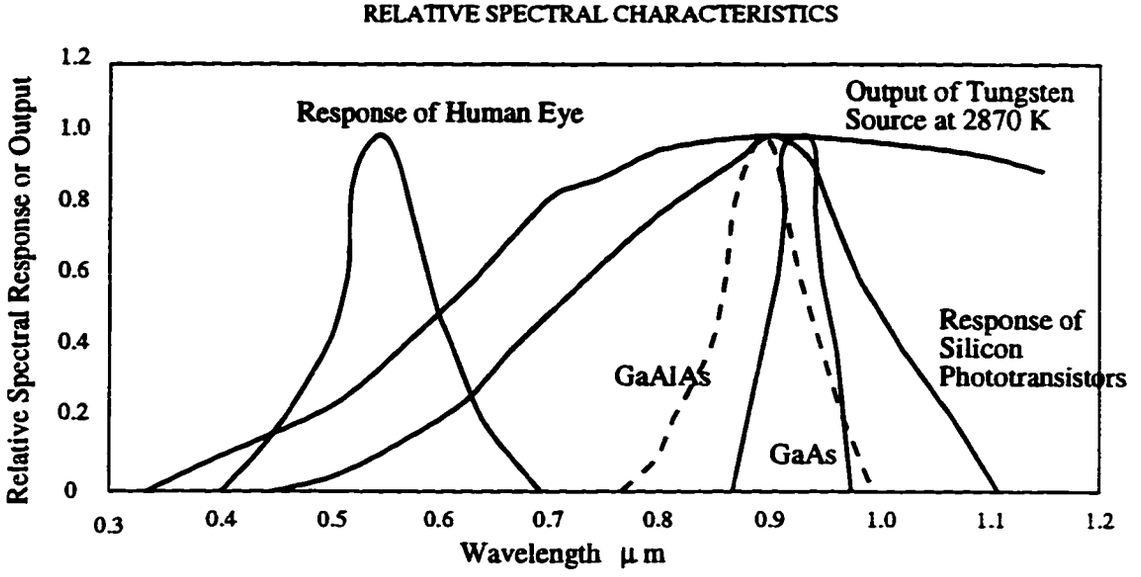


Figure 4.4: The relative spectral characteristics of the human eye, a tungsten light source and a silicon phototransitor (adapted from [69]).

Infrared emitters and detectors can be configured as retroreflective sensors by placing the emitter alongside the detector. In this configuration the output of the phototransistor detector is a function of the reflected infrared energy. For a given object surface the reflected energy can be calibrated as a function of distance to the object as shown in Figure 4.5. Different phototransistor detectors can vary in their response to reflected infrared energy shown in Figure 4.6.

Cadmium-Sulfide Photocells

Cadmium-sulfide photocells are sensitive only to visible light and have a number of applications in detecting changes in lighting conditions. These sensors whose output resistance varies as a function of light intensity respond slower than the phototransistors. Response times vary up to one second before the change in resistance stabilizes. Although slow, these sensors are very sensitive to changes in light intensity. Since these devices do not respond to infrared radiation they may be used in parallel with the infrared emitters, discussed in the previous section, without concern for optical interference. The output of a cadmium-sulfide photocell is measured as a function of distance to a 100 watt light bulb shown in Figure 4.7.

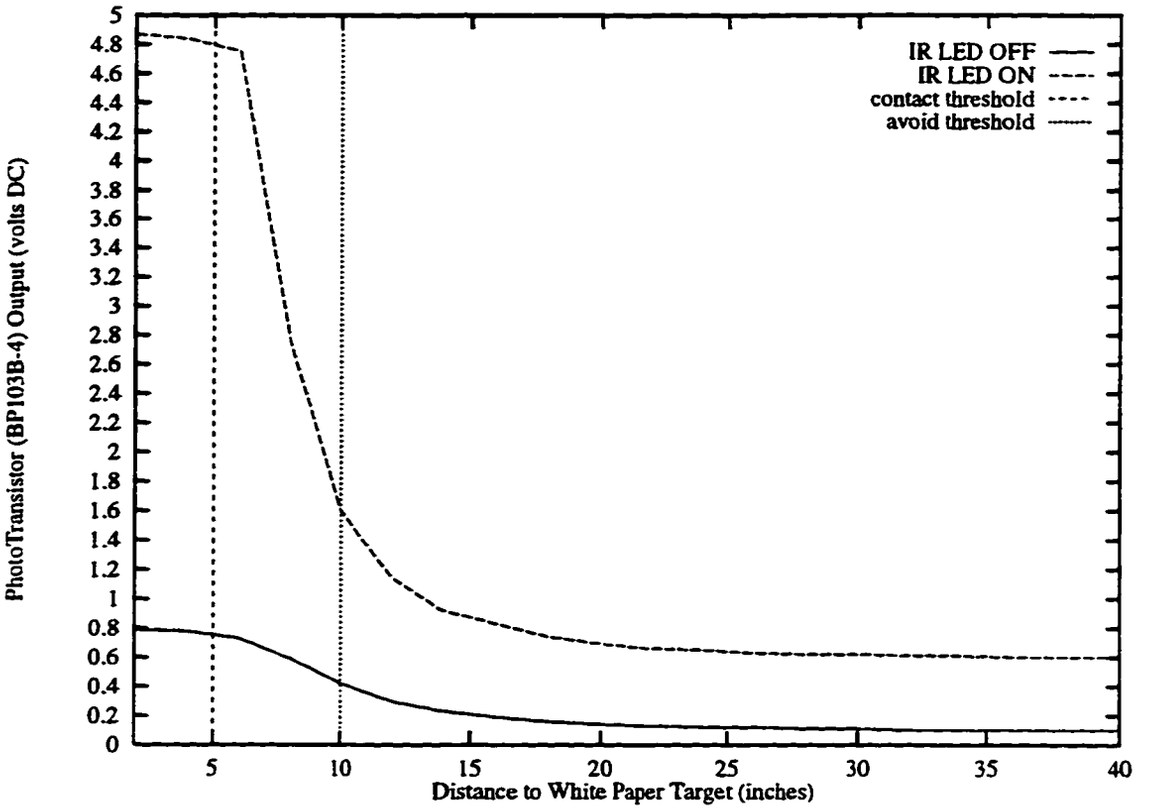


Figure 4.5: A plot showing the output of an infrared obstacle sensor as a function of distance to a white paper target.

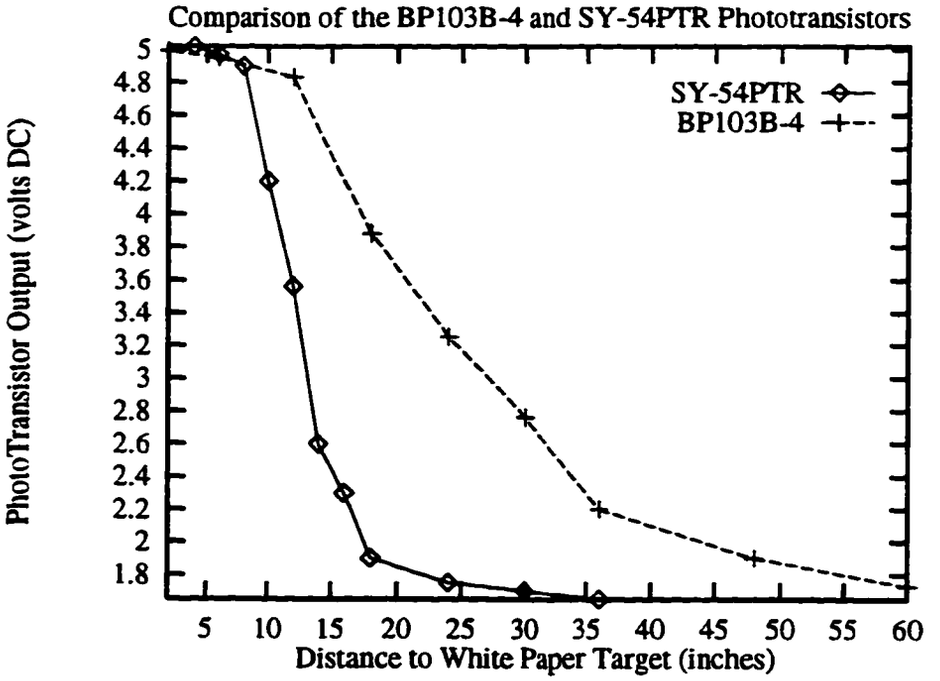


Figure 4.6: Compared are the output voltages of a SY-54PTR and a BP103B-4 phototransistor as a function of distance to a white target.

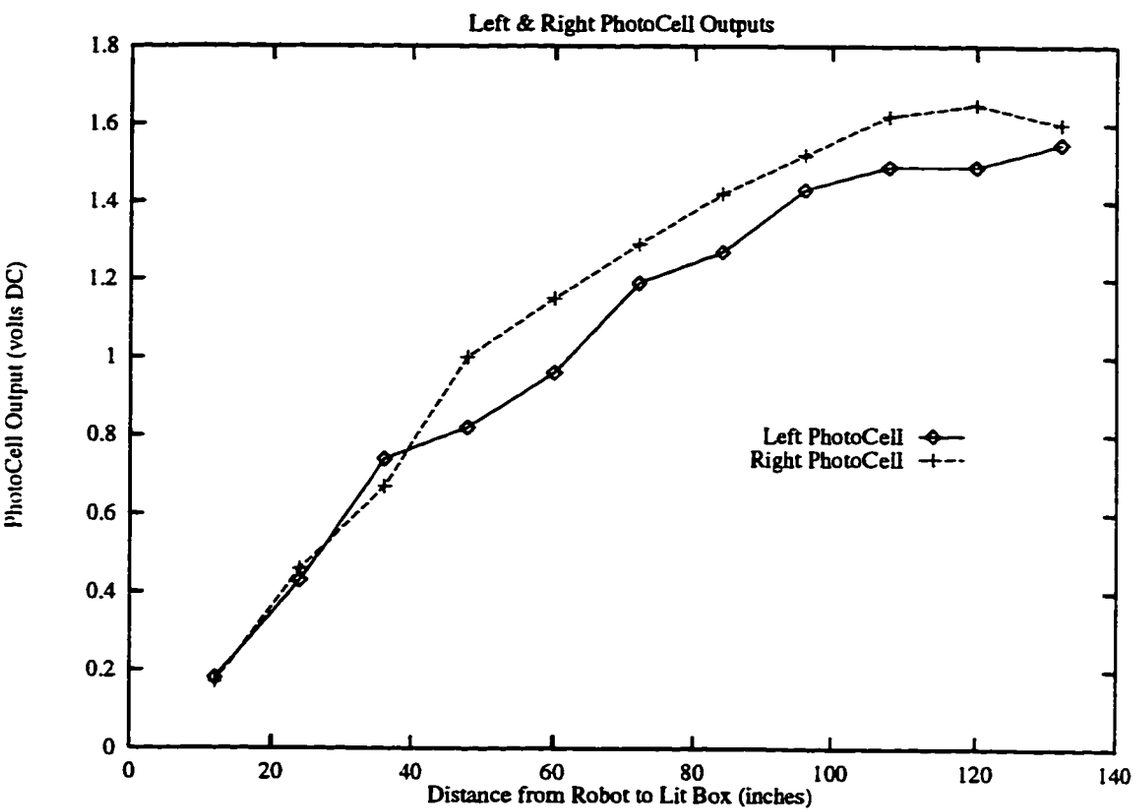


Figure 4.7: To locate the brightly lit box, left and right photocells, pointing forward at 20 degrees off center, whose output varies as a function of light intensity are used. Shown are the output voltages of the left and right photocell as a function of the distance to the box.

Silicon Phototransistors

Silicon phototransistors are fast devices sensitive to changes in light intensity with response times of a few microseconds. Typically used in electronic flash units their fast response time makes the phototransistor ideal when used in a moving sensor system. The output current of a phototransistor varies as a function of light intensity incident on the semiconductor surface.

The above sensors will be used individually and in combination to form the basis of the robot's perception system. In implementing perceptual cues from these sensors the problem of perception has been simplified to recognizing features in the sensor data for a specific task environment. The sensor features are reduced to boolean vectors and used as input to the primitive actuation behaviours which map them to corresponding motion primitives. How sensor data is used to create these boolean vectors is described next and the complete perception to action mapping is explained in the following chapter.

4.3.2 Obstacle Detection Cues

The purpose of the obstacle detection cues are to provide obstacle distance information to the robot. Three discrete thresholds are used corresponding to the distances of: less than 25 cm, less than 12.5 cm, and in physical contact with the robot. Active infrared emitter/detector pairs are used to provide non-contact obstacle information for the left and right front of the robot. Contact obstacle detection is determined using a single bit contact switch. The obstacle detection cues are defined as:[6]

?OBSTACLE Return right and left true flags indicating the corresponding obstacle sensor has exceeded the input threshold.

?TOUCH Return a true flag if the front contact switch is pressed.

Sensor Placement

Obstacle proximity detection is accomplished by configuring the infrared emitter/detector pair as a *retroreflective sensor*. The object to be detected reflects the radiant energy from the emitter back to an adjacent detector. Two retroreflective sensors are placed facing outward on both left and right sides of the robot's centerline. Each infrared detector has a 50 degree field-of-view, also termed acceptance angle, and is paired with an infrared emitter with a 16 degree beam angle, the total angle between the half intensity points. The sensors

[6] Perceptual cues will be identified by their leading question mark.

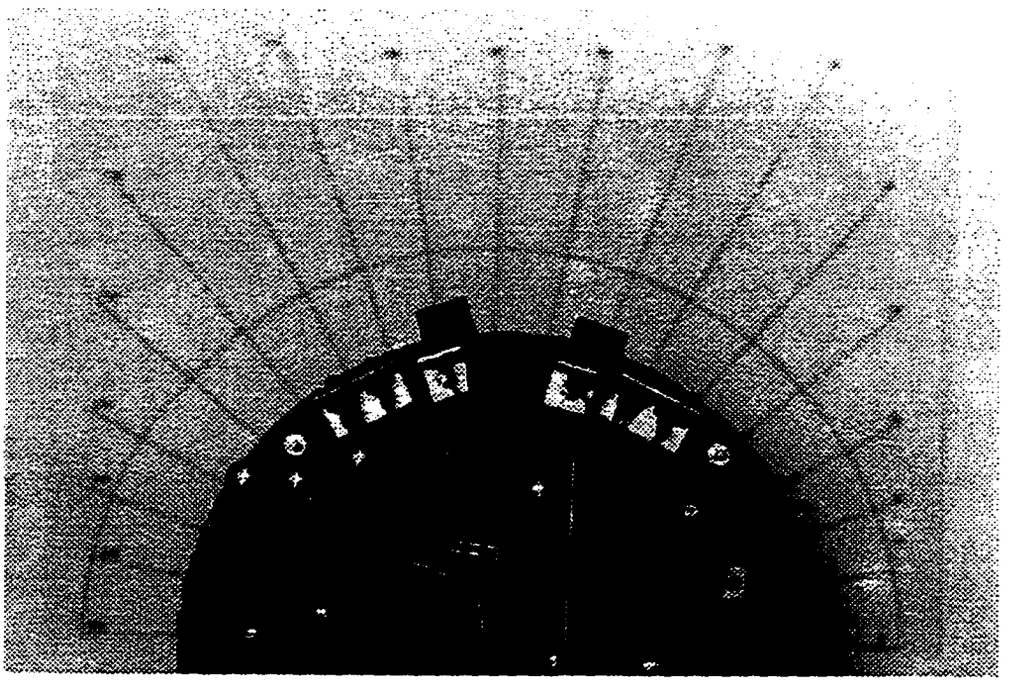


Figure 4.8: Infrared emitter/detector pairs are placed on the circumference pointing outward at 30 degrees both left and right of center.

are placed on the robot's circumference, at 30 degrees both left and right of the centerline as shown in Figure 4.8.

Sensor Data Collection

To determine the sensor's output response as a function of distance to an obstacle, a white paper target was moved toward the sensor with output the voltage and distance recorded. To measure the sensor's response as a function of obstacle angle, a small 2.5 centimeter square white target was moved in an arc in front of the sensor with readings taken every 10 degrees as shown in Figure 4.9.

Sensor Data Analysis

The energy reflected back to the detector will depend on the magnitude of the energy radiated, the surface properties of the object upon which the energy is incident, the distance to the object and the angle of reflection [22]. In the approach presented here, the environment and its objects are known quantities. The measured reflected infrared energy is a function of the distance to the object as shown in Figure 4.10.

As an object passes in front of a retroreflective sensor it enters and then leaves the field-of-view, or acceptance angle of the detector. The typical output of such a sensor

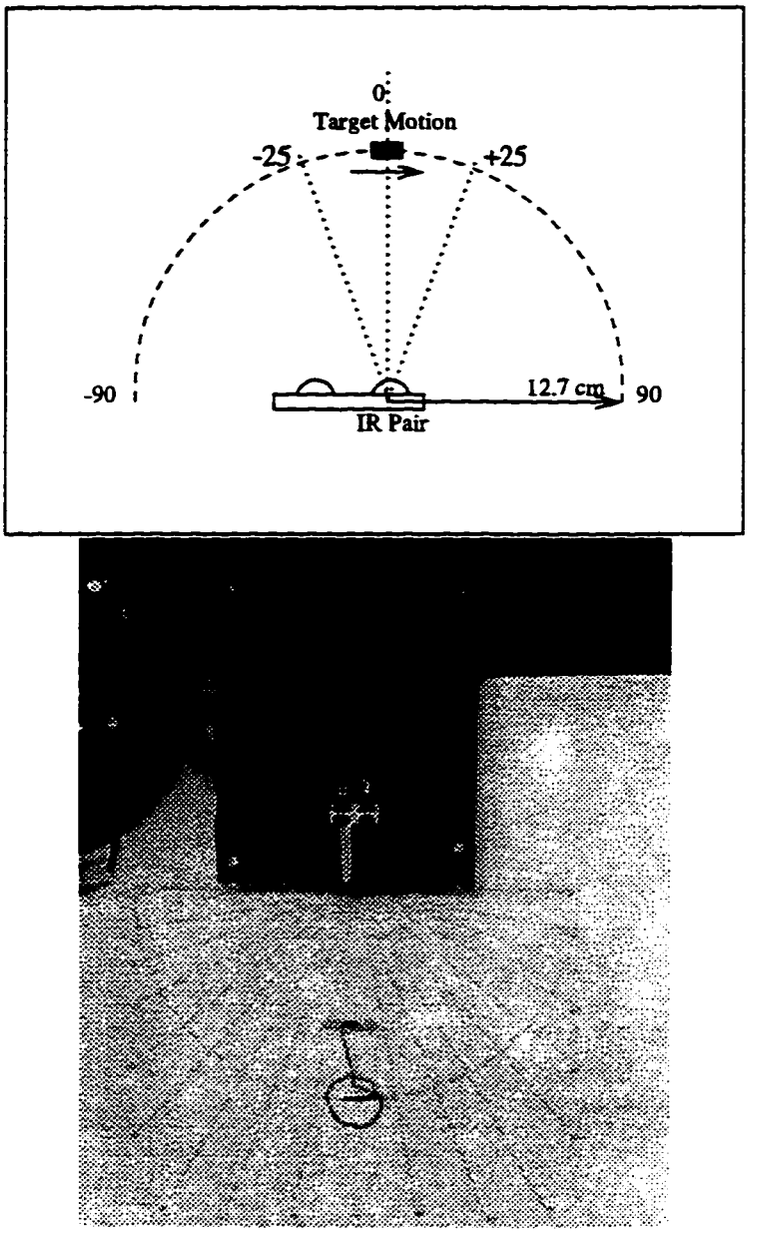


Figure 4.9: Data from the sensor was collected by taking voltage readings as a function of angle to a small 2.5 cm square target at the sensor's height. The target was moved in 10 degree increments on a 12.7 cm semicircle.

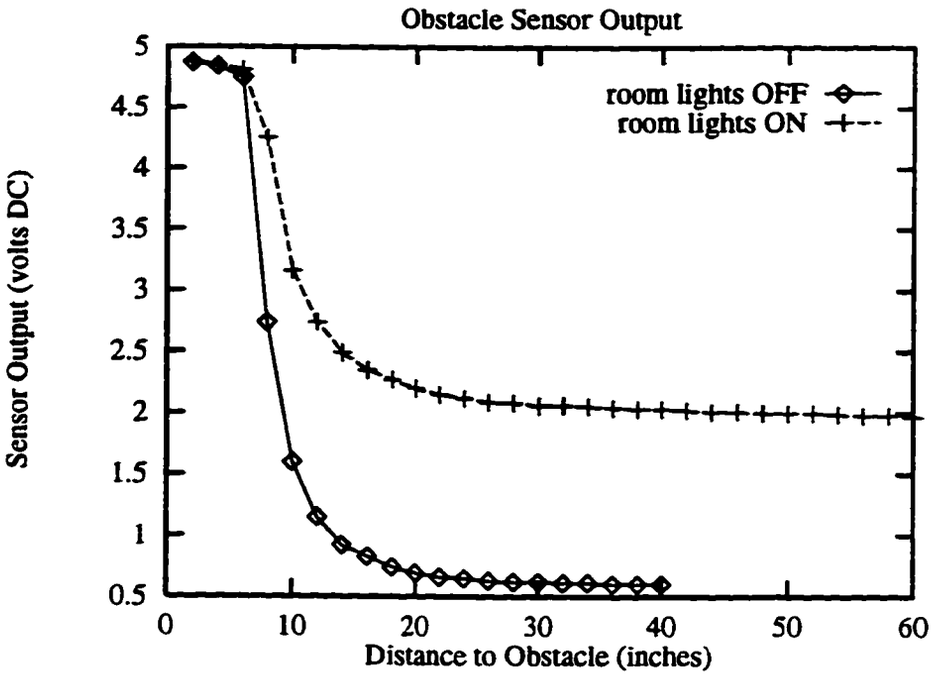


Figure 4.10: Data from the infrared emitter/detector obstacle sensor. A white target is moved towards the sensor and output voltage readings are taken as a function of target distance. Readings are repeated with the room lights ON for comparison.

as a function of target angle is shown in Figure 4.11. If we design a minimal obstacle detection system using a pair of retroreflective sensors, positioned on the left and right side of the robot, then the range of the sensors can be partitioned spatially, as explained in the previous section, by using a threshold function that corresponds to a desired detection distance. Thus, the type of sensing information returned is of the form "there is an obstacle less than 15 cm on your right," or "there is an obstacle less than 22 cm directly in front" etc.

Obstacle Detection Algorithm

The function of the obstacle detection cue is to provide left, right, and center obstacle detection for a fixed distance. The distance is determined as an input parameter called *threshold*, with two output parameters indicating obstacles detected by the left or right sensor. For the box-pushing task two thresholds corresponding to obstacles detected within the range of 12.5 cm and 25 cm are used. Pseudo-code for the obstacle detection cue is shown in Figure 4.12. The obstacle detection cue can be used to create cues specific to the range (specified as a threshold) as shown in Figure 4.14. For detecting obstacles in physical contact with the front of the robot the algorithm shown in Figure 4.13 is used to read the

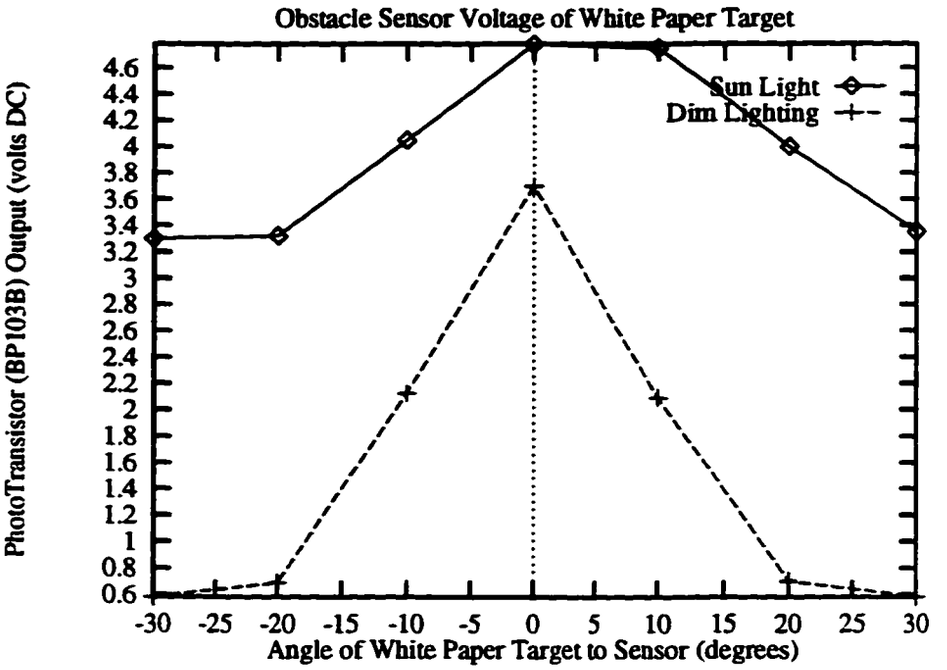


Figure 4.11: The output voltage of a Siemens BP103B-4 phototransistor as a function of angle to a small 2.5 by 2.5 cm white target. Shown are two room lighting conditions: Sunlight (which contains lots of IR noise) and a dimly lit room with low ambient IR noise.

```
: ?OBSTACLE  ( threshold --- left_flag, right_flag )
    FOR the right and left obstacle sensor DO
       take a reading with the obstacle sensor_ON
       take a reading with the obstacle sensor_OFF
       IF  sensor_ON - sensor_OFF > threshold  THEN
          SET the flag = TRUE
       ELSE
          SET the flag = FALSE
    RETURN { left_flag, right_flag }
;
```

Figure 4.12: Shown is the pseudo-code obstacle detection algorithm with input and output parameters on the left and right of the --- symbol respectively. Obstacles are detected when the reflected infrared energy is greater than a given threshold value. To detect reflected energy, ambient infrared readings are subtracted before comparison with the threshold value.

```
: ?TOUCH  ( --- touch_flag )
     take a reading of the front contact switch
     IF  sensor_ON THEN
        SET touch_flag = TRUE
     ELSE
        SET touch_flag = FALSE
     RETURN { touch_flag }
;
```

Figure 4.13: The pseudo-code for the touch obstacle detection cue returns a true flag if the forward contact switch is depressed.

binary value of the front contact switch.

Obstacle Detection Verification

To test the obstacle detection cues a simple motion controller is created using the RANDOM-WALK, AVOID and CONTACT behaviours defined in Chapter 3. The controller consists of a processing loop which calls each behaviour in order. The output of a left and right obstacle sensor is mapped by the AVOID and CONTACT behaviours to left and right motion commands. The possible outputs of AVOID are { *idle, left-turn, right-turn* } while CONTACT maps sensor output to { *idle, left-rotate, right-rotate* }. Ambient infrared light is accounted for by taking a detector reading while the emitter is off. Currently both behaviours make use of the same infrared obstacle sensors, but with different threshold functions on the outputs shown in Figure 4.5. The robot wanders in a room with other static robots and the sensor positions are adjusted until collision free movement is achieved. Although in this example both behaviours are using the same sensor output this separation in sensor processing allows for sensor replacement.[7]

4.3.3 Box Detection Cues

Three perceptual cues are used for box detection:

?BOX-DIRECTION Return right and left true flags indicating the corresponding box sensor has exceeded the input threshold.

?BOX-DETECT Return a true flag if either left or right box sensors exceed a given input threshold.

[7] CONTACT's sensors could be replaced with left and right tactile switches.

```
: ?AVOID-DETECT ( threshold-1 --- avoid_flag )
     ?OBSTACLE( threshold-1, right_flag, left_flag )
     IF the right_flag OR the left_flag THEN
        SET avoid_flag = TRUE
     ELSE
        SET avoid_flag = FALSE
     RETURN { avoid_flag }
;
: ?CONTACT-DETECT ( threshold-2 --- contact_flag )
     ?OBSTACLE( threshold-2, right_flag, left_flag )
     IF the right_flag OR the left_flag THEN
        SET contact_flag = TRUE
     ELSE
        SET contact_flag = FALSE
     RETURN { contact_flag }
;
```

Figure 4.14: The avoid and contact detection algorithm sets a flag true if either right or left sensor thresholds are exceeded.

?BOX-CONTACT Return a true flag if ?TOUCH is true AND either right or left box sensors exceed a given input threshold.

Box detection is simplified by using a bright light placed at the center of the box. The box detection cue asks the question: Can the robot see the box-light? The answer depends on the robot's distance from the box and the orientation of its two forward pointing sensors with respect to the box. An adjustable threshold varies the range at which the box-light is detectable and is dynamically determined as a function of ambient light. Recognizing physical contact with the box is a combination of two different types of sensing, touch and light intensity. This combination of stimulus is unique in the task's environment simplifying box recognition. The perceptual cues are designed using the following five step procedure.

Sensor Placement

Two forward pointing sensors whose output is a function of light intensity are placed on the robot pointing 20 degrees off center. The sensor's field-of-view is restricted to a narrow band in the horizontal plane at the same height as the box-light as depicted in Figure 4.15. This minimizes interference from other light sources in the environment placed at different heights. The field of view of the box light sensors is fixed at 80 degrees. Individually each sensor's field-of-view is approximately 40 degrees.

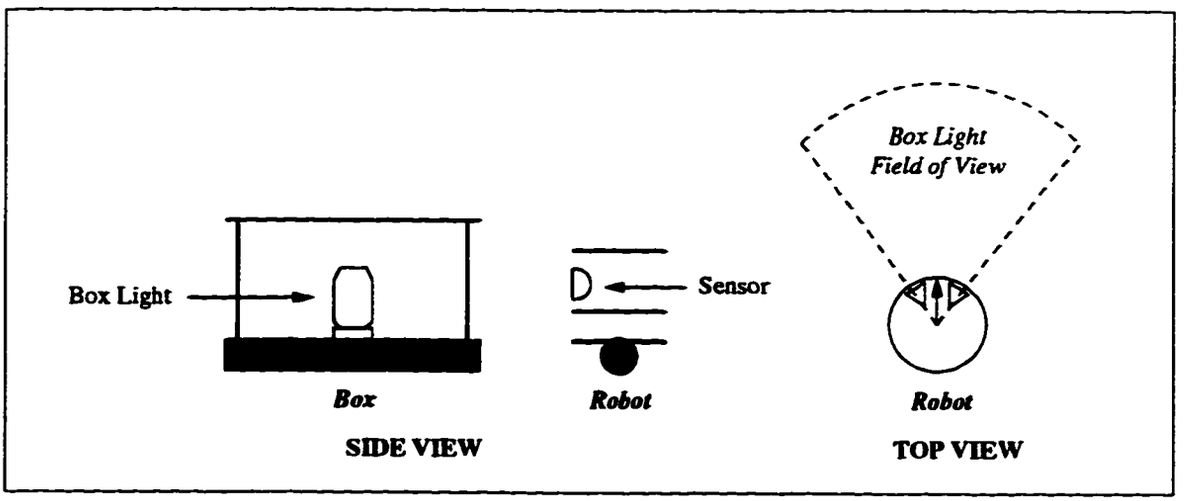


Figure 4.15: Shown is the placement of the box-light sensors on the robot with respect to the height of the box. The two sensors see in a narrow forward pointing cone of roughly 80 degrees. By restricting both the sensor's field-of-view and the box-light stimulus to a narrow horizontal band, box recognition is simplified.

Sensor Data Collection

Sensor data was collected for the box direction and contact cues. Box direction sensor data was collected using the setup illustrated in Figure 4.16. A robot is positioned facing forward with center being 90 degrees and right and left sides corresponding to 0 and 180 degrees. A brightly lit box is moved on a 1.8 meter arc from 50 to 130 degrees with sensor readings recorded for each 10 degree increment. Data for the box contact cue was recorded by measuring the box sensor's output voltage while a robot was in contact with a box.

Sensor Data Analysis

Figure 4.17 shows the output voltage of the left and right photocell box sensor as a function of direction angle with respect to the robot. Lower voltages correspond to brighter light intensities. As the box enters the sensor's field-of-view the voltage decreases with a minimum value at a box angle normal to the sensor and then rises as the box angle increases. Note the minimum should occur at 110 degrees for the left photocell and 70 degrees for the right photocell since each sensor is pointing 20 degrees off center. Since the minimum reading of the right sensor is at 80 degrees, the sensor's direction is adjusted. The left and right sensor plots cross when the box is directly in front of the robot, approximately at the 90 degree position. This information provides a rough estimate of box direction with respect to the robot.

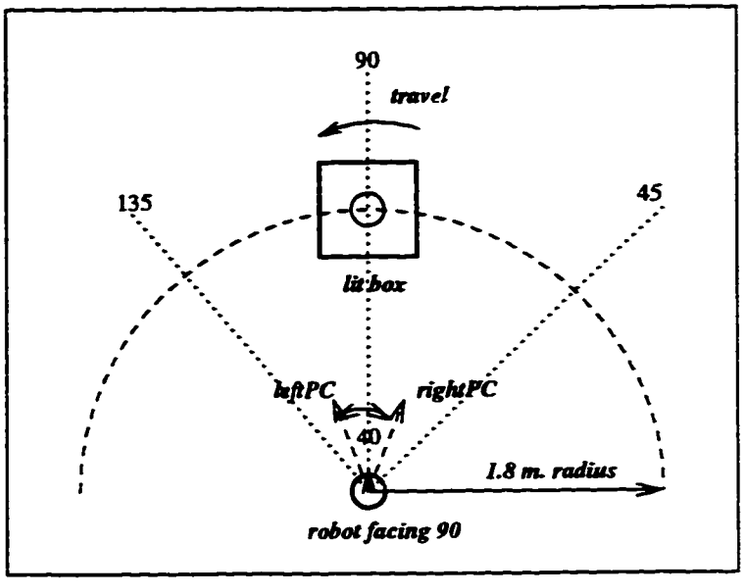


Figure 4.16: To gather box tracking data from the forward facing photocells the robot was positioned pointing at the 90 degree mark while the lit box was moved along a 1.8 meter 80 degree arc from 50 to 130 degrees in 10 degree increments.

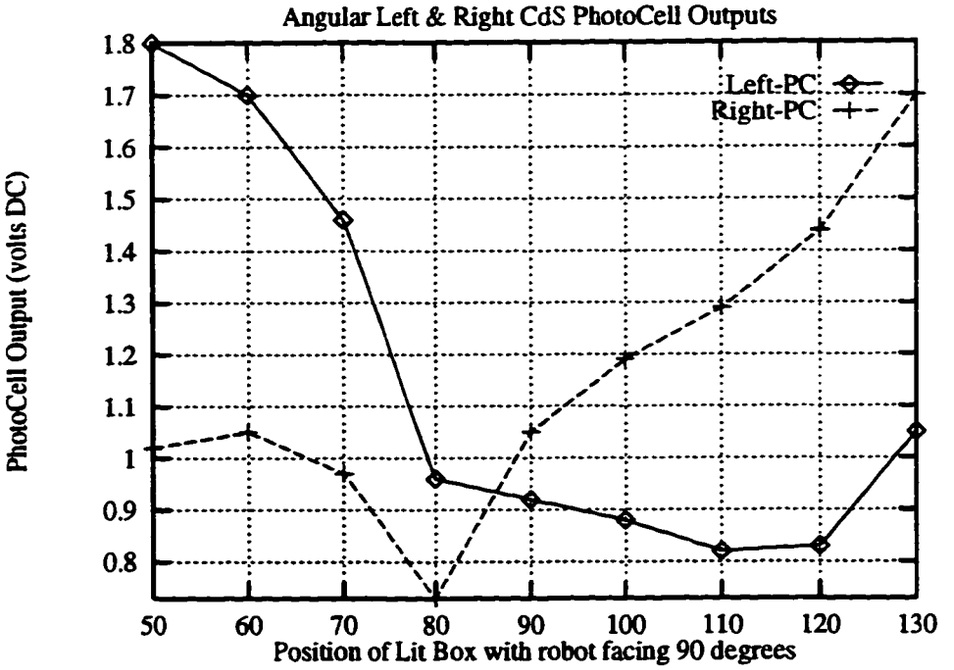


Figure 4.17: To locate the lit box two forward pointing photocells measure light intensity in a horizontal plane. Shown are the left and right box-sensor outputs as the box is moved along a 1.8 meter arc from 50 to 130 degrees with the robot facing 90 degrees.

```
: ?BOX-DIRECTION  ( threshold --- right_flag, left_flag )
     FOR the right and left box sensor DO
        take a reading from the box sensor
        IF  sensor reading  > threshold  THEN
           SET the flag = TRUE
        ELSE
           SET the flag = FALSE
     RETURN { right_flag, left_flag }
;
```

Figure 4.18: The box direction algorithm takes readings from two forward pointing left and right photocells and sets their respective flags if the readings are greater than a given threshold.

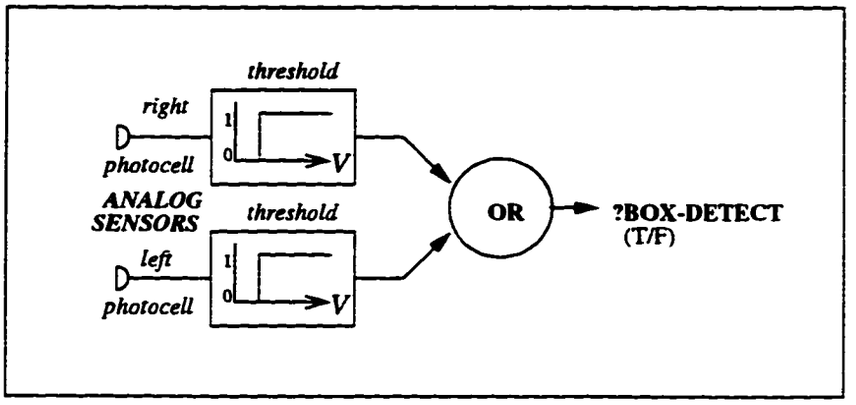


Figure 4.19: The ?BOX-DETECT cue is true if either the left or right sensor thresholds are exceeded. The threshold value correspond to an approximate distance of 1.5 meters between the robot and the box.

Box Detection Algorithms

The function of the box direction cue is to provide an approximate direction towards the box based on the intersection of the left and right box sensors' field-of-views. If the box is directly ahead both left and right sensor thresholds are activated. The value of the threshold determines the range at which the box is detected. Pseudo code for box direction is shown in Figure 4.18.

The box detection and contact cues are used to control state transitions between the step controllers explained in the next chapter. Figure 4.19 and Figure 4.20 summarize the algorithms presented next.

The box detection cue, shown in Figure 4.21, is based on the box direction cue and returns a true flag if either left or right box sensor thresholds are exceeded.

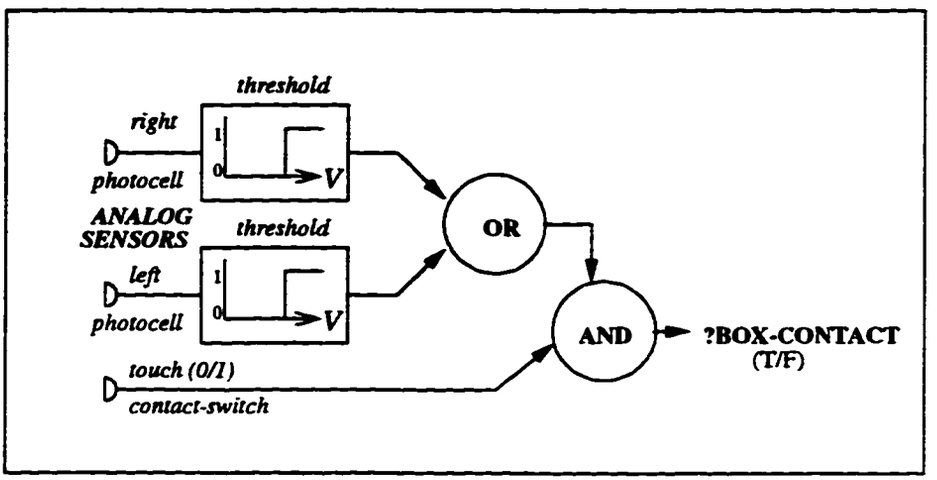


Figure 4.20: The ?BOX-CONTACT cue is true if either the left or right sensor threshold is exceeded AND the contact switch is closed. The threshold used corresponds to the box-light intensity found at the side of the box. Contact with the box is therefore determined by a combination of bright light and touch.

```
: ?BOX-DETECT ( threshold-1 --- box_flag )
     ?BOX-DIRECTION( threshold-1, right_flag, left_flag )
     IF the right_flag OR the left_flag THEN
        SET box_flag = TRUE
     ELSE
        SET box_flag = FALSE
     RETURN { box_flag }
;
```

Figure 4.21: The box detection algorithm sets the box flag true if either right or left box sensor thresholds are exceeded.

```
: ?BOX-CONTACT ( threshold-2 --- box-contact_flag )
     ?BOX-DETECT( threshold-2, box_flag )
     ?TOUCH( touch_flag )
     IF box_flag AND touch_flag THEN
        SET box-contact_flag = TRUE
     ELSE
        SET box-contact_flag = FALSE
     RETURN { box-contact_flag }
;
```

Figure 4.22: The box contact algorithm sets the box-contact flag true if the robot is touching the side of a box.

The box contact cue, shown in Figure 4.22, also uses the box detection cue, but a different value for the threshold, corresponding to the higher light intensity found at a boxside, is used. The box contact cue combines the box detection cue and the touch cue and returns a true flag if both are true.

Box Detection Verification

The three box detection cues are tested using a static robot tethered to a workstation to display output. The threshold used to detect the box-light is set to be twice the ambient room light and is determined dynamically on power up. The effect of bright ambient light readings is to reduce the distance at which the box-light is detected. The box direction cue is tested using the same procedure as for data collection. The outputs from both cues are displayed on the workstation as the box is moved from 0 to 180 degrees. The box contact cue is tested by putting the robot in contact with a side of the lit box. Although these tests are preliminary, the cues are retested when integrated with the primitive actuation behaviours.

4.3.4 Goal Detection Cue

The goal direction cue asks the question: Can the robot see the goal? The answer is a function of the robot's orientation with respect to the goal indicator, which in this instance is a spotlight placed near the ceiling. The goal detection cue is defined as:

?SEE-GOAL Return a true flag if a signal peak greater than the input threshold is detected within the user defined field-of-view.

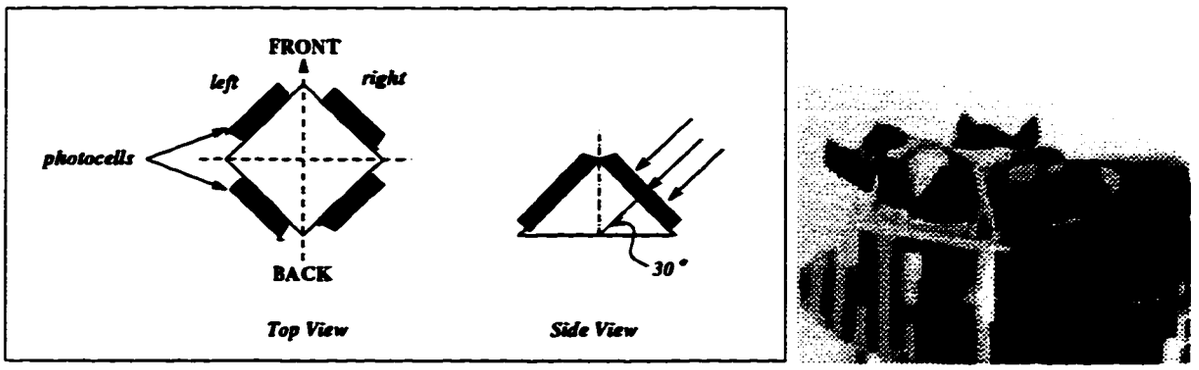


Figure 4.23: The first design for the goal direction sensor consisted of four photocells mounted on a square and pointed upward at 30 degrees elevation. The fixed sensor positions proved inflexible and the second version mounted a single sensor on a rotating motor.

The first design of this sensor system was not successful and will be discussed in the sequel. The final design consists of a narrow field of view sensor which is swept by a motor in an upward pointing arc. If a signal peak occurs, caused by the spotlight, within an adjustable window the goal is detected. The box detection sensors which face horizontally are shielded from light sources above the robot, while the goal detection sensors face upward and therefore the goal-light does not interfere with the box-light. The design of the goal detection cue involves five steps: sensor placement, data collection, data analysis, algorithm design and verification.

Sensor Placement

A preliminary design tried to use the same photocells used in box tracking as shown in Figure 4.23. If two sensors could be used to track a brightly lit box in the horizontal plane, then the same approach should work in an elevated plane. The sensors were arranged on four sides forming a pyramid pointing upward at an elevation of 30 degrees. An omni-directional view is available by considering any two pairs of sensors.

Figure 4.24 shows the output from the two forward facing sensors as a function of the goal-light angle. The readings were taken by moving the goal-light in a 180 degree 4.6 meter arc in front of a robot. Figure 4.24 shows the sensor readings are not symmetric about the 90 degree position, because the fixed position of each photocell makes the sensor difficult to align and the wide field-of-view results in the asymmetric sensor output. The fixed photocell position does not allow the field-of-view to be changed. Although the method was abandoned it provided the motivation for a design using a rotating sensor.

The alternate design allowed the sensor to be swept in an arc using a small positional

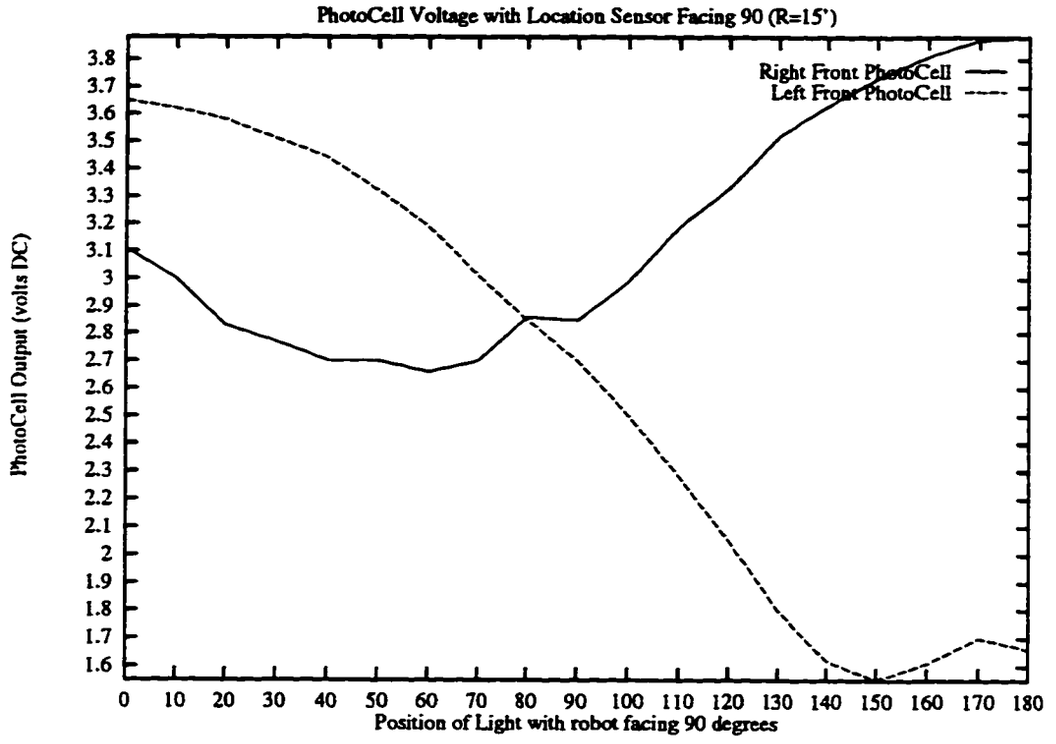


Figure 4.24: The output from a preliminary goal direction sensor design. The reading were taken with the sensor mounted in a fixed position and orientation. Shown are the outputs of two photocells whose voltage vary as a function of light intensity. A lower output voltage indicates a brighter stimulus. The goal indicator, a downward pointing spotlight, was moved along a 4.6 meter circular arc in front of the robot at 10 degree increments. As can be seen, the data from the two photocells that comprise the sensor are not symetric about the 90 degree position. The sensor design was abandoned for one based on rotating the sensor to gather readings.

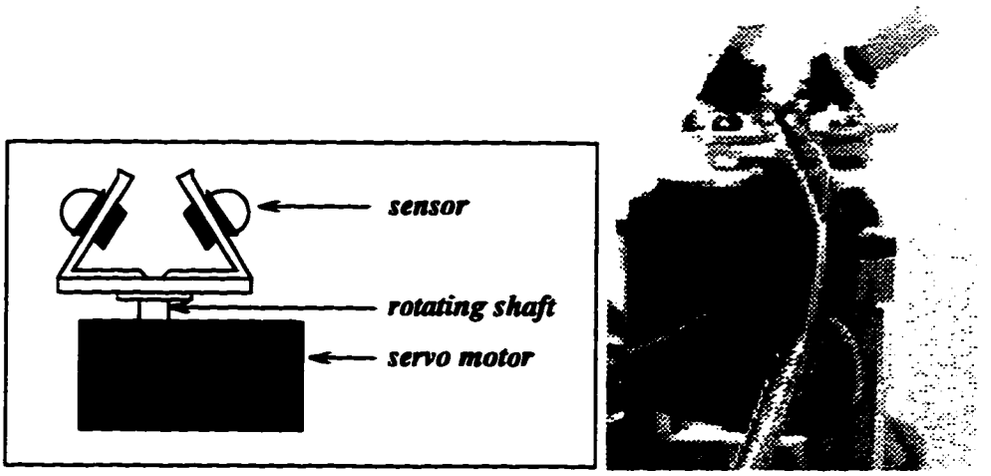


Figure 4.25: The omni-directional goal sensor design consists of a forward and rear facing phototransistor which is swept in a 180 degree arc from left to right using a servo motor. Readings are taken every five degrees once the robot has made contact with the box.

servo motor. In this manner the sensor system's field-of-view could be made variable from the sensor's fixed field-of-view of 10 degree to a maximum of 180 degrees determined by the length of the arc swept by the motor. An omni-directional view was obtained by using two opposing sensors. The sensor's elevation angle was calculated for the lab environment used. which had a maximum distance of 4.2 meters from a goal indicator placed at height of 2.4 meters. The elevation angle θ is equal to $\arctan(y/x)$ where y is the height of the goal indicator, and x is the maximum distance at which the goal is to be visible. The final design of the sensor system is shown in Figure 4.25.

Sensor Data Collection

The next step is to collect data from the sensor system in the intended environment and under similar dynamic conditions as the transport task. The goal-direction sensor was mounted on top of a robot which was then placed pointing towards the goal-light. Sensor readings were taken for positions between 1.75 and 4.75 meters from the goal-light as shown in Figure 4.26. Data from these positions were to represent the conditions while the box was moving toward the goal.

Sensor Data Analysis

The purpose in the data analysis step is for the designer of the system to get a feel for the sensor system's output as the task executes. Shown in Figure 4.27 is the sensor's output as a function of sweep angle. The goal-light is positioned directly in front of the robot and forms

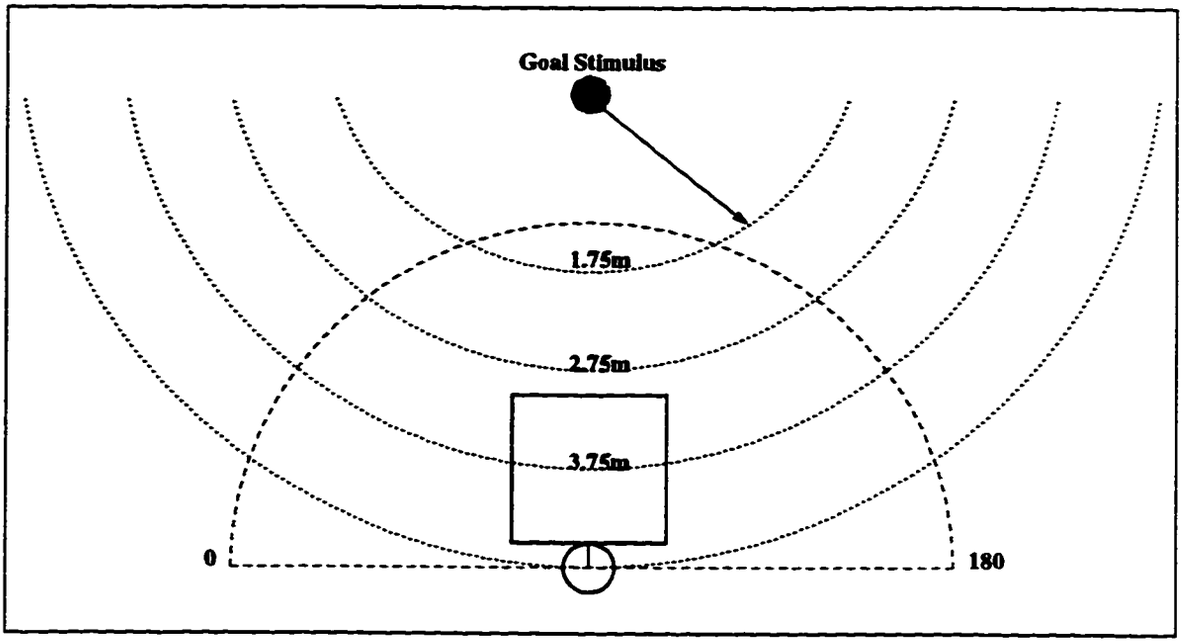


Figure 4.26: The laboratory setup used to gather goal direction sensor data. The robot is positioned facing the goal stimulus, an overhead spotlight, with the box between the robot and goal. Sensor readings were taken as the sensor is swept from 0 to 180 degrees. The distance between the goal and robot is then reduced and the measurements repeated. In this way a spatial stimulus map is produced for each perceptual cue.

a signal peak at 90 degrees. A simple threshold function will not detect the goal location since light reflecting off adjacent walls causes a higher than baseline reading as evident at the 0 and 180 degree positions. Since the width of the signal peak increases as the distance to the goal decreases, signal width can not be used as the cue for goal detection. However, as Figure 4.27 shows, the sharp rise and fall of the signal can be used to detect the goal direction with respect to sensor orientation.

Goal Detection Algorithm

The function of the goal detection cue is to determine if a robot is on the *right* side of a box to push. The "right" side of a box is any side on which a robot pushes that causes the box to move towards the goal. The wrong side is any side on which a robot pushes that moves the box away from the goal. The number of correct sides on which to push can be controlled by specifying the range of orientation angles in which the signal peak is detected. These angles can be specified by defining a window in the sensor system's field-of-view in which the signal peak must fall. Both the position and size of the window may be determined by specifying its right and left field-of-views. Also required in goal detection is the magnitude of

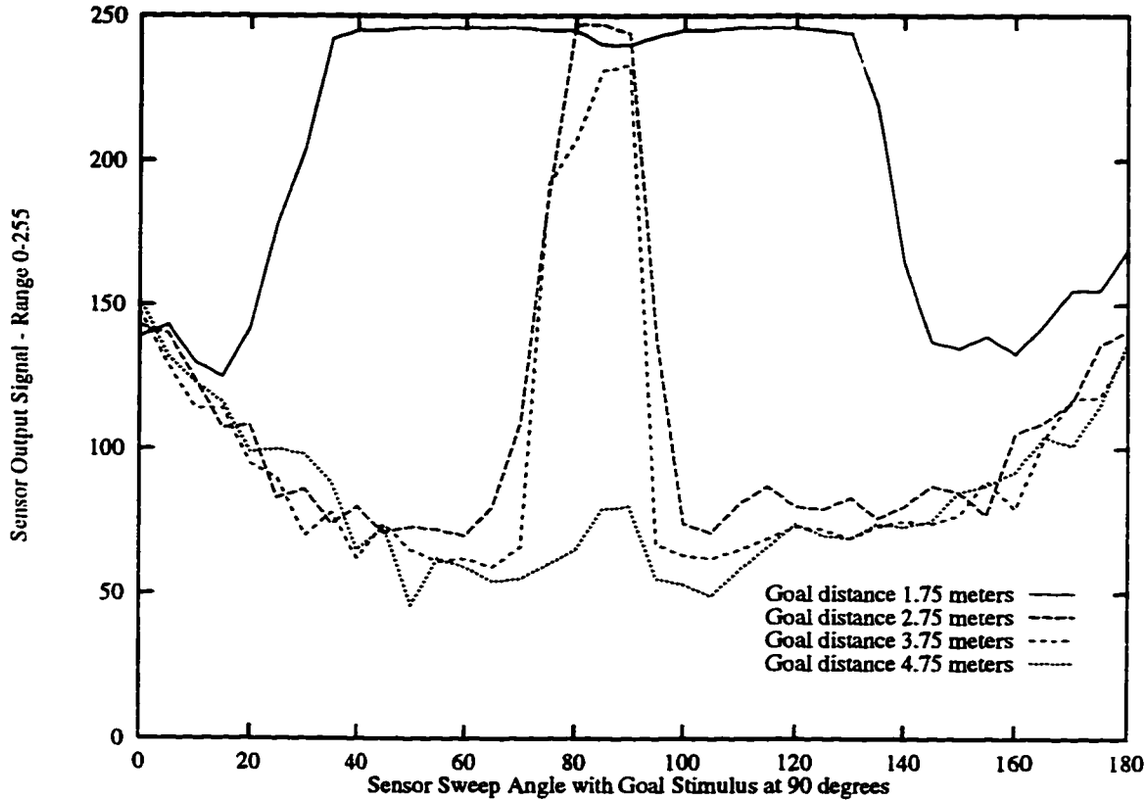


Figure 4.27: Shown is the output of the goal direction sensor as a function of angular position. The goal stimulus is located at 90 degrees. The sensor is swept from 0 to 180 degrees in front of the robot. Four plots show how the signal varies as a function of distance from the robot to the goal stimulus (1.75 - 4.75 meters).

```
: RISE?  ( current_reading, previous_reading --- rise_flag )
     IF  current_reading  > previous_reading THEN
        SET rise_flag = TRUE
     ELSE
SET rise_flag = FALSE
;
: FALL? ( current_reading, previous_reading --- fall_flag )
     IF current_reading < previous_reading THEN
        SET fall_flag = TRUE
     ELSE
        SET fall_flag = FALSE
;
: ?SEE-GOAL  ( threshold, right_FOV, left_FOV --- goal_flag )
     SET previous_reading = initial_reading
     SET goal_flag and rise_flag = FALSE
     FOR move the sensor from left_FOV to right_FOV DO
        take a reading from the goal sensor
        IF | current_reading - previous_reading | > threshold THEN
           IF  RISE?  THEN  SET rise_flag = TRUE
           IF  FALL? AND rise_flag  THEN
              SET goal_flag = TRUE
           ELSE
              SET goal_flag = FALSE
     RETURN { goal_flag }
;
```

Figure 4.28: The goal detection cue determines if the goal indicator is within the robot's allowable orientation angles. The output is true if a signal peak falls within the range of angles. The cue is used to trigger a pushing behaviour.

signal peaks, which may be specified as the minimum difference between successive sensor readings. Thus, the goal detection algorithm has three input parameters describing the magnitude of signal peaks, and the size and position of the orientation window. The result is a perceptual cue which detects whether the robot is properly oriented to push a box with respect to a goal direction indicator. The pseudo-code for the goal detection cue, ?SEE-GOAL, is shown in Figure 4.28.

Goal Detection Verification

Verification of the algorithm is performed for static positions only. The box is oriented on 45 degree line to the goal indicator as shown in Figure 4.29. The robot is tethered to a workstation so that the results of the test may be displayed. A subset of possible orientation positions are used to test the cue. The input parameters are specified for signal peaks of

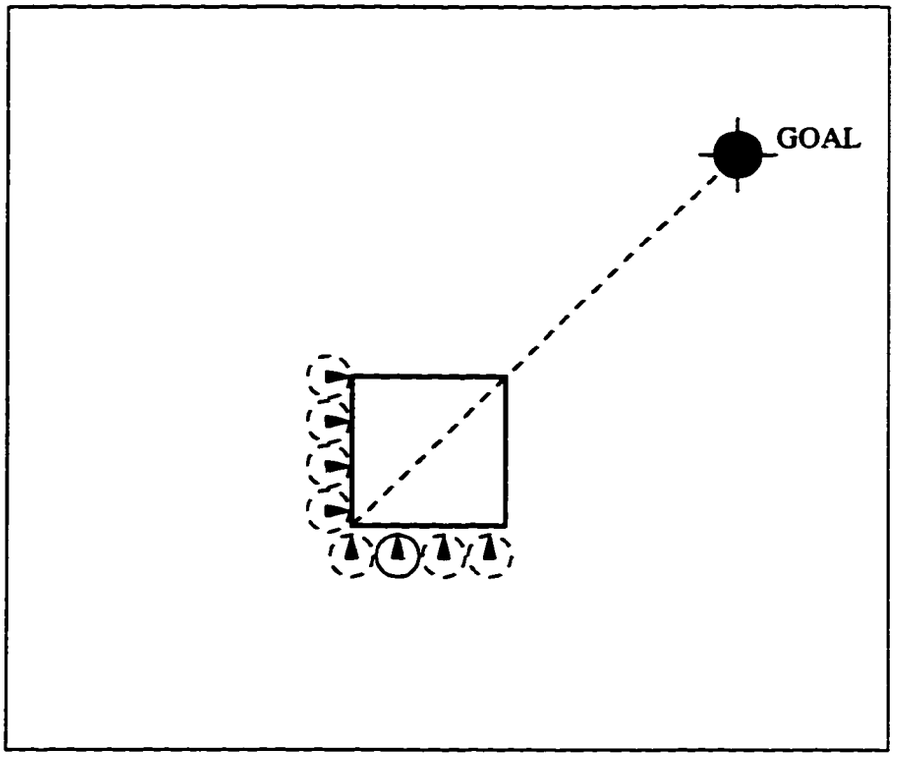


Figure 4.29: The goal detection cue is verified for a subset of possible orientations. A robot is tethered to a workstation and then placed in the indicated positions. The output of the ?SEE-GOAL algorithm is tested with `min_difference` = 20, `right_FOV` = 150 and `left_FOV` = 30 input parameters. In all the above positions the cue returns `TRUE`.

20 (0-255), and a window of 120 degrees centered at 90 degrees. For each of the positions shown in Figure 4.29 the cue returns a TRUE output value.

4.4 Summary

If the environment is considered when designing a task specific system then it can be used to encode information about the execution order of a task. The problem of what a robot should do and when, is transformed to decoding that information. Perceptual cues used for behaviour activation decode *what* action the robot takes, and cues used for state transitions decode *when* the action is performed.

Techniques to simplify the decoding process include spatially and modally orthogonal sensing and additive cue construction. These methods were used to reduce the perception problem to answering yes/no questions about conditions in a robot's task environment.

To elucidate the cue design process, the multi-robot task of box-pushing was used. The approach to cue construction involves a careful characterization of the environment, in

which the task is performed, using the chosen sensors. The result is sensor data that varies spatially and whose analysis leads to algorithms that extract features used in behaviour activation and state transitions in the task model.

Binary vectors are the interface to the primitive motor behaviours. This allows the perceptual cues to be computed independently from the implementation details of motor action. This adds weight to the hypothesis that perception serves to motivate motor behaviours, but does not have to be functionally dependent on the output signal those motor behaviours produce. The advantage of such a supposition is architectural, in that pieces of the perception to action control system can be replaced without causing a redesign.

In the next Chapter, the connection between local perception and global action is made explicit in the design of subtask controllers, or Q-machines, which produce the coherent system behaviour needed in multi-robot box-pushing.

Chapter 5

Coherent Behaviour

Do multi-robot systems requiring close coordination, as found in manipulation tasks, require an explicit "mechanism of cooperation?" Or can a system of robots display a coherent behaviour by carefully decomposing the problem into subtasks and coordinate the mass action based on local perception alone? In this chapter, it is demonstrated that certain cooperative tasks are possible without explicit communication or cooperation mechanisms. The approach relies on subtask decomposition and sensor preprocessing. A framework is described for modelling multi-robot tasks which are described as a series of steps with each step possibly consisting of substeps. Finite state automata theory is used to model steps with state transitions determined by values of binary sensing predicates called *perceptual cues*. A perceptual cue (Q), whose computation is independent from the operation of the automata, is processed by a finite state machine called a *Q-machine*. The model is based on entomological evidence that suggests local stimulus cues are used to regulate a linear series of building acts in nest construction. The approach is designed for a redundant set of homogeneous mobile robots. Both a model and its implementation are described. Results are presented, in the next chapter, for a system of physical robots capable of collectively moving a heavy object to an arbitrarily specified goal position.[1]

[1]Portions of this chapter have been published. C. Ronald Kube and Hong Zhang 1994, 1997. IEEE IROS (3):1883-1890, Autonomous Robots 4(1):53-72 [35, 38].

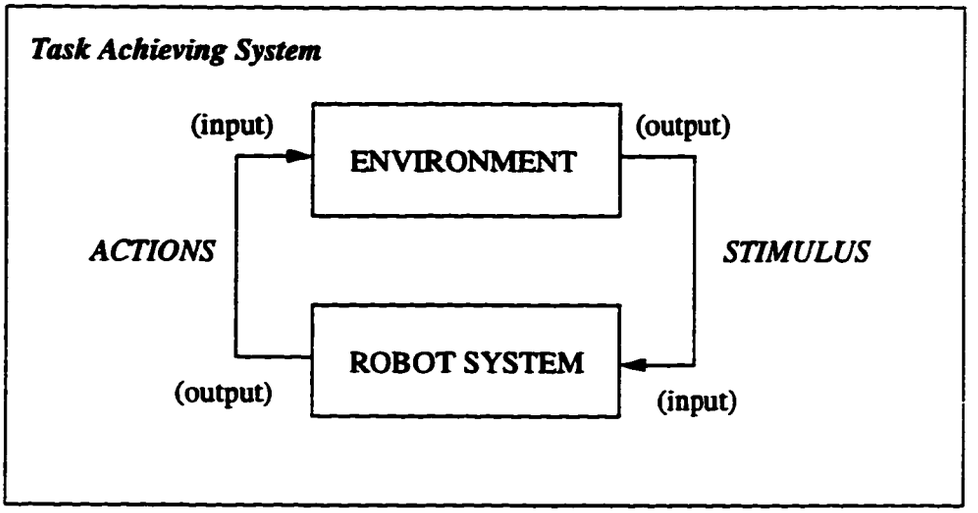


Figure 5.1: A solution to a given task is considered to consist of two parts: the environment with actions on its input and changes in stimulus as its output, and the robot system with stimulus as input and actions on the environment as output.

5.1 Introduction

A collective system that acts as a unit in a well coordinated manner is displaying a coherent system behaviour. Such a system, be it composed of people, insects or robots is thought to be more effective at achieving some goals than individuals acting alone. In robot tasks, like collective manipulation, is such a cooperative system possible without inter-robot communication or robot identification?

Coherent behaviour is accomplished by viewing the system that solves the problem as two equally important parts consisting of the environment and the robot system as shown in Figure 5.1. The environment has actions performed in it on its input side, which result in changes that may be perceived on its output side. The robot system has perception on its input and produces actions in the environment as its output.

In such a system the task to be accomplished is the desired change in the environment in response to input actions performed by the robots. The robot system is the procedural mechanism used to achieve those changes. In this synergistic system coherent behaviour becomes possible as the common task and its environment become the central coordinating mechanism.

Presented here is a model which connects local perception to global action by describing tasks as a sequence of changes in position for a given stimulus-object.[2] The task to be

[2]Stimulus-object refers to a manipulation object that has been "tagged" for easy identification.

accomplished is specified using a directed graph whose traversal describes the temporal order in which the task is completed.

Using the predefined actions described in Chapter 3 and the perceptual cues described in Chapter 4 machines are defined that accomplish the desired task in an ordered stepwise manner. The machines described as FSM controllers for each step in the task with transitions between steps specified as locally sensed perceptual cues. This type of control is analogous to phase-based control used in dextrous manipulation tasks [71].

Transporting a heavy box between two known locations by a system of many robots is a manipulation task requiring coordination to be effective. What is demonstrated here is a feasible solution which is obtained by coordinating behaviour among a collection of autonomous robots without using either explicit direct communication between robots or robot identification.

5.2 Coherent Insect Behaviour

The examples discussed in chapter 2 on nest construction [19] and prey transport [67] by some social insects are prime examples of tasks performed by a repetitive sequence of behaviours. Sensing plays a key role in triggering the transition between different task construction or transport behaviour steps. It is reasonable, therefore, to speculate that such a mechanism may also be used as a means of synchronizing several asynchronous robots in the execution of a common task.

A frequent question about social insects is how they collectively build sophisticated nests without centralized planning. Coordinating their building activities often involves simple rules applied without communicating directly with other workers as Franks *et al.* concluded after modelling the two dimensional structures built by ants using a bulldozing-building behaviour [23].

Nest building by ants that live in the flat crevices of rocks involves making perimeter walls around their colonies without the need to construct either a roof or floor. This type of two dimensional structure is highly conducive to laboratory observation and data collection, as nests could be built between two microscope glass slides separated with cardboard columns. The first stage of wall construction described involves an individual ant carrying a granule into the nest towards the cluster of nest mates. Once the ant is close it reverses its direction 180 degrees and begins to push the granule into other existing granules. This bulldozing behaviour was tested as a computer-simulation model producing a similar pattern of granules that formed perimeter walls. Thus, bulldozing behaviour is an example of how a

simple rule for building can be used to produce a predictable result without direct communication between builders. Rather, indirect communications through the environment by way of the building structure serves to coordinate collective activity [23]. In this way both the environment and behavioural act used for task completion is part of the solution.

Attempts to model the states of both the environment and its cognizant occupants is not novel. In animal behaviour McFarland and Bösser have defined a motivational state as a combination of a physiological and perceptual state, with behaviour used to change states in motivational space [44]. They extended this approach to modelling the system behaviour by assigning state variables to environmental space, behaviour space and task space. Environmental space defines the constraints imposed on the system with regards to movement and topology. Behavioural space refers to the partition of the environment made by the animal's (or robot's) sensory system. Tasks are defined by their initial and final states using state variables that are relevant to the task.

Finite state automata (FSA) have been used to model perceptual tasks [6] and motivational behaviour in animals [64, 46]. Arkin and McKenzie have used FSA to model the space time relationship in a perceptual processing task on a mobile robot. This approach allows for perceptual tasks to be sequenced in a reactive control system. In short, finite state automata used to model the steps in a task as rules of interaction along with local perception to control the application of that action is a plausible model for a collective coherent behaviour.

5.3 Task Description and Decomposition

Task description and decomposition can be divided into task-related and tool-related knowledge [44]. In other words, what is to be done and how to do it. Task-related knowledge can be described in terms of externally observable desired changes in the environment, independent of the procedural mechanism used to accomplish them. This is synonymous with Wilson's sensory state machines in which the environment is considered as a machine with the effects of robot actions considered as input and changes in observable stimulus as output [79].

Our model assumes that the task under consideration can be described as a sequence of steps. A finite state machine (FSM) will then be designed to accomplish each step with transitions between steps triggered by *perceptual cues*. Each step may, of course, be composed of substeps or subtasks to also be performed sequentially. In this manner a task may be described in as fine a detail as required by its decompositional analysis.

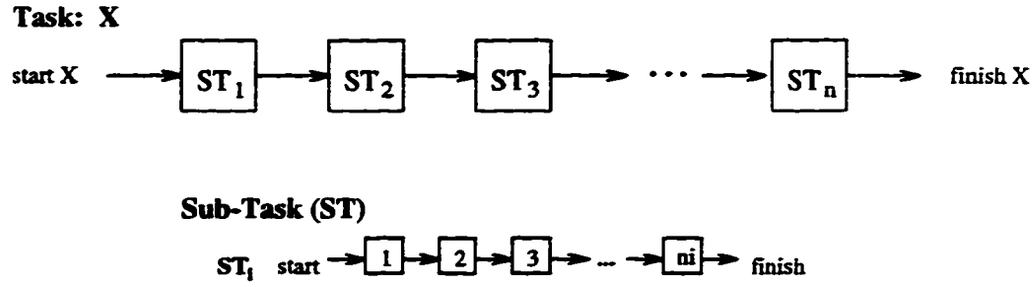


Figure 5.2: Tasks are described as a sequence of steps, with each step possibly composed of additional subtasks (ST).

This results in a task description having the hierarchical structure illustrated in Figure 5.2. In the presented model, task description is specified in a directed graph, called a task description graph (TDG), with vertices representing a stimulus-object and its position to be manipulated by the system, and edges in the graph representing possible actions that effect those manipulations.

Tool-related knowledge is specific to the mechanism employed by the system and refers to robot actions in the environment and therefore is procedural in nature. A task decomposition into finite state controllers that accomplish the desired changes specified in the task's TDG is modelled using the primitive actions previously described in Chapter 3. Both the execution of individual subtask controllers and the transitions between them are accomplished with perceptual cues. Perceptual cues and their finite state machine controllers are called *Q-machines* and together with a task description graph provide a model that considers the environment and robots together in its solution to the specified task.

5.3.1 Task Description Graphs

In the class of manipulation tasks being modelled here objects to be manipulated are described as stimulus-objects and states determined by position, time and a performance metric. Since states are vectors there are an infinite number of states in the environment. However, in the box-pushing task the states of interest are: initial, final, intermediate and stagnating. Thus the states correspond to several actual positions of the object being manipulated in an X, Y coordinate system. A task to be accomplished by the system is described by defining the initial and goal positions of the object being manipulated. As well, stagnating conditions are identified as positions in the graph requiring special actions (i.e. stagnation recovery behaviours). In the box-pushing examples that follow, two actions are used that manipulate the box: A_1 PUSH-BOX and A_2 REPOSITION. Task description is

an external global point-of-view and describes changes in the environment without regard to the mechanism that causes those same changes. The description of the task is captured in a directed graph defined as:

Definition 5.1 *A task description graph (TDG) is a directed graph G with n vertices and m edges. The vertex set $V(G) = \{v_1, \ldots, v_n\}$ describes the state uniquely determined by position, time and a performance metric, of an object (S) perceived as a stimulus to be manipulated, and the edge set $A(G) = \{a_1, \ldots, a_m\}$ describes the actions needed to manipulate the object without speaking about the actor or actors. $V(G)$ contains an initial state and a goal state, each of which can be associated with a set of positions, times and performance metrics according to the precision to which the values are known.*

Two examples are now presented of nondirected and directed box-pushing.

Nondirected Box-Pushing

Nondirected box-pushing involves pushing a box from an initial position for a fixed distance in any direction. The task is considered successful if the box S is pushed a fixed distance R in under a given amount of time T. Distance R is the radius of a circle with center at an initial position P_{t_0} as illustrated in Figure 5.3. The goal position P is any position that satisfies $R \geq |P - P_{t_0}|$ which is simply the distance between the goal and initial box positions.

Each vertex in the TDG shown in Figure 5.4 specifies a unique condition as defined by changes in radius from the initial position $\Delta r = |P - P_{t-\delta}|$ per time period δ summarized as:

Initial box position, which may take any value in the 2D position space with $\Delta r = 0$

$$v_1 : (S_1, P_t) \mid \Delta r = 0,\ t = t_0 \tag{5.1}$$

Intermediate box positions, which may take any value in the 2D position space with $\Delta r > 0$

$$v_2 : (S_1, P_t) \mid \Delta r > 0,\ t = t_0 + \delta \tag{5.2}$$

Goal box positions, which may be specified as any value in the 2D position space where $\Delta r \geq R$ and $t_n - t_0 \leq T$

$$v_3 : (S_1, P_t) \mid \Delta r \geq R,\ t = t_n \tag{5.3}$$

Stagnating box position, which describes a position that has not changed in time period δ resulting in $\Delta r = 0$

$$v_4 : (S_1, P_t) \mid \Delta r = 0,\ t = t_0 + k\delta \tag{5.4}$$

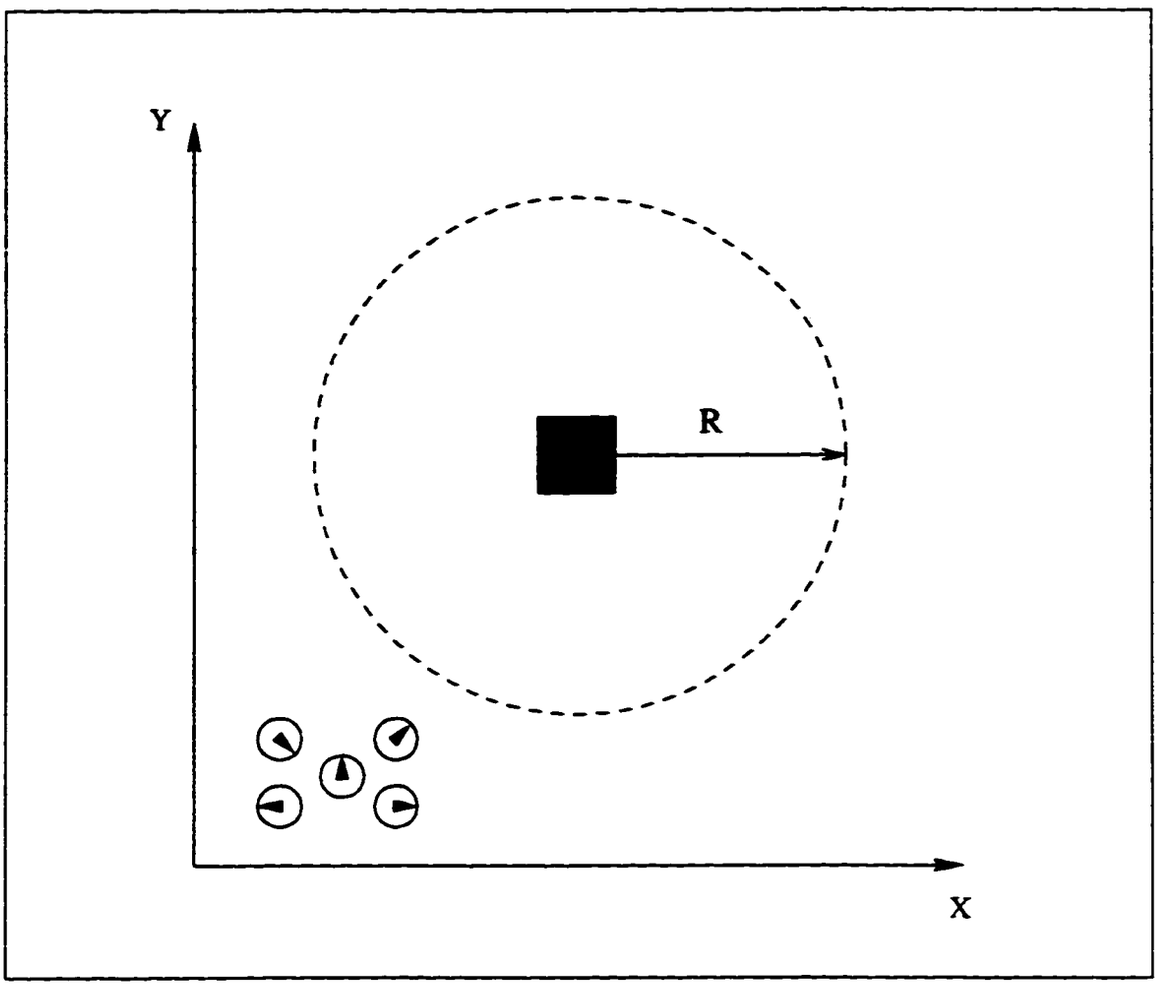


Figure 5.3: Illustrated is the nondirected box-pushing task where the robots (small circles) push the box in any direction for a fixed distance (dotted circle).

where $k\delta$ is the time period before stagnation is detected (i.e. a timeout).

The stagnating condition in equation 5.4 occurs when robots push the box in opposing directions thereby producing a resultant net force insufficient to move the box. The problem occurs due to the nondirected nature of the task. The solution is a recovery behaviour whose output is a robot action that changes the orientation of the pushing force and is labelled as A_2 in Figure 5.4.

Directed Box-pushing

The task description can be changed to directed box-pushing towards specific positions and can include a temporal sequence of positions as indicated in Figure 5.5. In this example, the box is first moved from an initial unknown position P_i to a known position P_A and

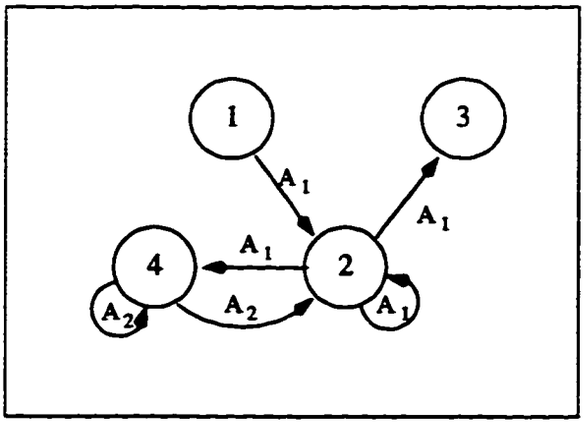


Figure 5.4: The task description graph where vertices represent an object (box) and position, and edges represent actions that effect changes in object's position.

then moved to a second position P_B. In each step the path taken is unspecified and may therefore be suboptimal. The corresponding TDG is illustrated in Figure 5.6.

5.3.2 Q-machine Controllers

In the model presented, tasks are decomposed along four dimensions: abstraction levels, control and temporal dependencies, and redundancy. Using these guiding principles controllers are then designed which accomplish each step in the task decomposition. After a brief discussion of these four dimensions, detailed examples of two approaches taken to nondirected and directed box-pushing are discussed.

Abstraction Levels

Three levels of abstraction are used to decompose any given task: the task-level which describes what is to be performed by the system and is specified by a task description graph; the behaviour-level in which a task is specified as subtasks; and the action-level in which taxis-based actions are used to accomplish each subtask.

At the task-level objects to be manipulated and their positions are specified in temporal sequences using a task description graph. In the specific example presented in the next section this level of description corresponds to statements like "first move box number 1 to goal position A; next wait one minute; then move box number 1 to goal position B."

At the behaviour-level the task is further decomposed into a number of subtasks. An example will be presented in which the task is decomposed into three subtasks, corresponding to finding the box perceptually, moving towards it until contact is made, and finally pushing the box towards position A.

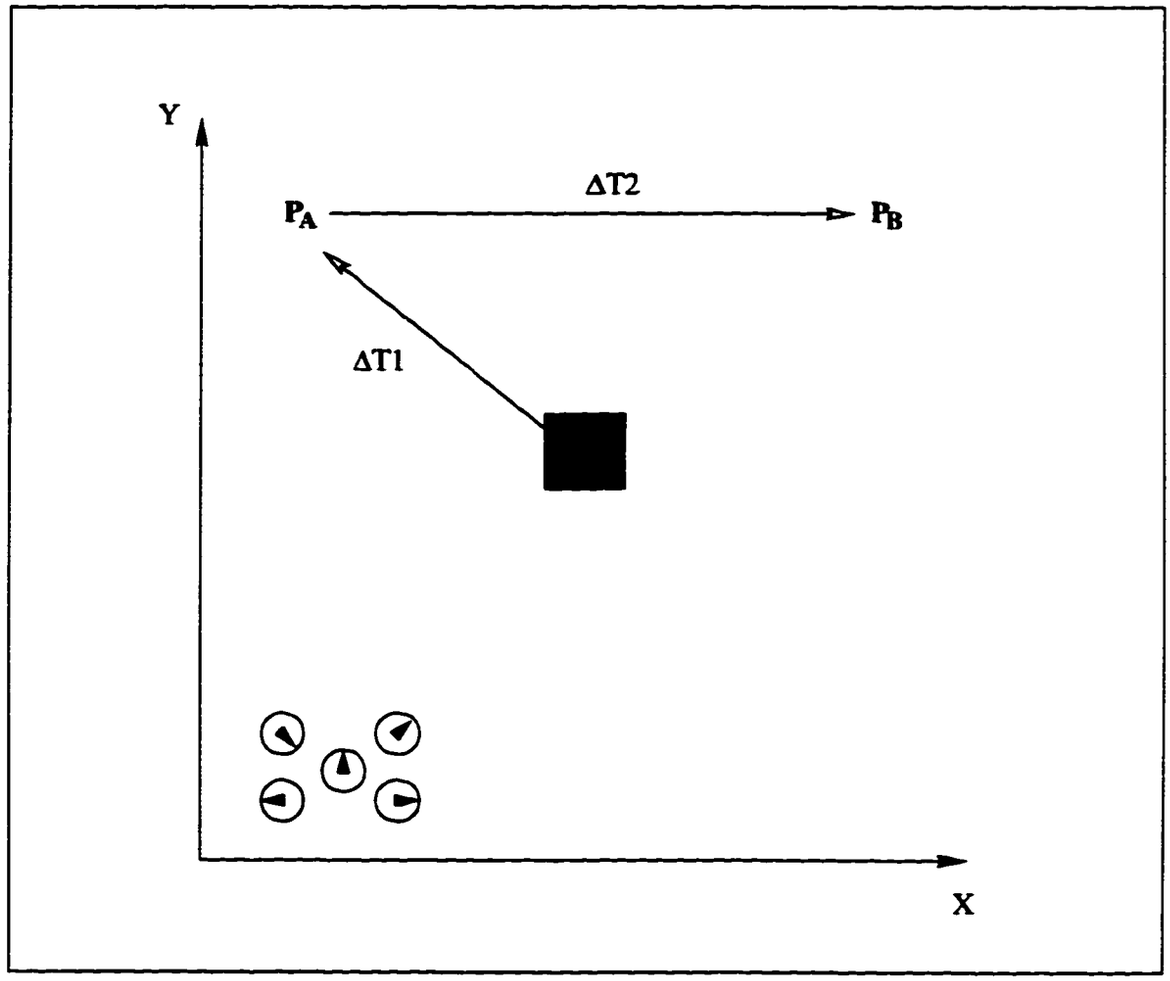


Figure 5.5: Illustrated is the directed box-pushing task where the robots (small circles) push the box first to position P_A and then to position P_B

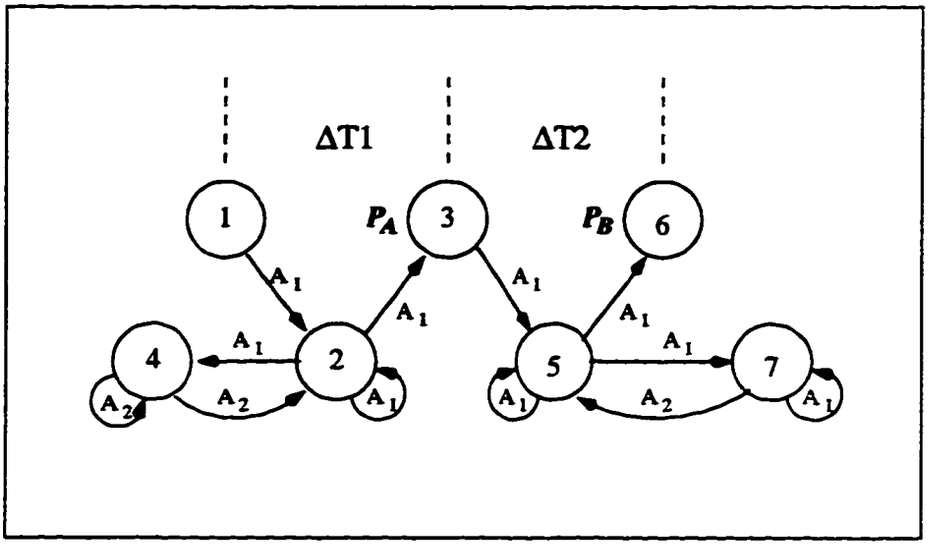


Figure 5.6: The task description graph for directed box-pushing. The box is pushed from an unknown initial position labelled as v_1 to the position described by vertex v_3 in time period $\Delta T1$ and then to position described by vertex v_6 in time period $\Delta T2$. Positions $v_{2,5}$ describe intermediate positions during execution, while position $v_{4,7}$ refer to stagnating positions from which recovery actions are required.

At the action-level, each of the above subtasks are further decomposed into primitive actuation behaviours described in Chapter 3. Taken together these PA behaviours compose a subtask controller which accomplishes the subtask, for instance "finding the box perceptually." This three-level hierarchy decomposes a given problem into subproblems each of which may use an architecturally different approach to its solution. In the model presented here reactive control is used at all three levels.

Control Dependencies

Task decomposition is performed on the basis of trying to reduce four robot control dependencies:

- *Resource conflicts.* By ensuring that a sufficient supply of resources per robot are available conflicts between robots are reduced. In the box-pushing task this translates to having enough free space on a boxside available by either limiting the number of robots used or increasing the size of the box being pushed.
- *Robot interaction.* Reducing the interaction between robots helps minimize the antagonistic forces present in trying to coordinate the actions or mass effect of a multi-robot system. Obstacle avoidance is used as a means of implementing noninterference protocols.

- *Inter-robot communication.* In the approach presented, robots do not explicitly communicate, thereby reducing the control dependency present in methods that use inter-robot communications as a means of coordination.

- *Global information.* All actions taken by a robot are based on locally perceived information only, thus eliminating the need to obtain a global view of a problem before making a decision.

Temporal Dependencies

Task decomposition is possible along the temporal dimension by specifying sequences of actions to be performed using the task description graph. In the case of directed box-pushing this amounts to sequencing the stimulus used to indicate each goal position. This allows the box to be moved between two goal positions using the same set of subtask controllers by controlling when and where the goal stimulus appears. This approach requires the control of some of the environment's stimulus to be part of the solution design

Redundancy

Since the uncertainty is high in both the perception and action of individual robots, the task is decomposed under the assumption that there will be many homogeneous and therefore redundant robots. All robots carry identical programs for solving the task. Observing the noninterference protocols mentioned above increases the probability that a specific action will be performed.

5.4 Q-machine Controller Design

Two examples to help further elucidate the presented model are now presented. In the next Chapter, the model for nondirected box-pushing will be explored in simulation and serve as a precursor for the results obtained for directed box-pushing by a physical system of robots.

Nondirected Box-Pushing

In the computer-simulation of nondirected box-pushing the forces on the box are modeled as the sum of a single robot applying a unit force at an angle to the box side. This will produce a resultant force vector and a torque applied about the box's center. If the resultant force is greater than a user-defined threshold the box will translate in an XY plane. Likewise a user-set torque threshold will cause rotation of the box (see Figure 5.9). If an equal

distribution of robots push the box the resulting net force is insufficient to overcome the weight of the box and the system stagnates.

A single controller is used for each robot. The controller is composed of the following PA behaviours: AVOID, REPOSITION, REALIGNMENT and SEEK-BOX, which have a fixed priority with AVOID being the highest and SEEK-BOX the lowest. The stagnation recovery behaviours each contain a counter that is reset each time the robot moves. Each counter has a threshold that can be set, and if reached (i.e. a timeout) the behaviour is activated. Using thresholds, behaviours can be ordered. For example, a REALIGNMENT behaviour with a time threshold $T_1 = k$ where k is some constant, and a REPOSITIONING behaviour with a time threshold of $T_2 = 4k$ will take four times as much time before being invoked. As long as the robot is moving, the behaviour does not become active since its threshold counter is constantly being reset to zero. Since the behaviours alter the robot's orientation or position by a small random amount, the robots behave asynchronously.

In the box-pushing task, stagnation refers to any configuration in which robots are in contact with the box and the box is not moving. For example, consider a minimum configuration with two robots on the same side pushing with a unit force at an orientation of 45° and the weight of the box set at 1.50. The resulting force calculations using a unit force and equations 5.5 and 5.6 are $F_x = 1.414$ and $F_y = 0.0$. Since $F_x \leq 1.50$ stagnation has occurred.

$$F_x = \sum_{i=1}^{n} f_{ix} \tag{5.5}$$

$$F_y = \sum_{i=1}^{n} f_{iy} \tag{5.6}$$

$$f_{ix} = cos(\theta_R) \tag{5.7}$$

$$f_{iy} = sin(\theta_R) \tag{5.8}$$

Once the realignment timeout threshold is reached the behaviour changes the direction of the applied force. Since the realignment is random this may increase or decrease the resultant force. For cases where realignment is not sufficient to establish box movement the repositioning behaviour becomes active. In this case the repositioning behaviour causes the robot to assume a new position on the box. Figure 5.7 compares the success rates of two controllers with and without recovery behaviours. Figure 5.8 compares two recovery behaviours as a function of the number of robots. The simulation results of nondirected box-pushing led to the model used in directed box-pushing described next.

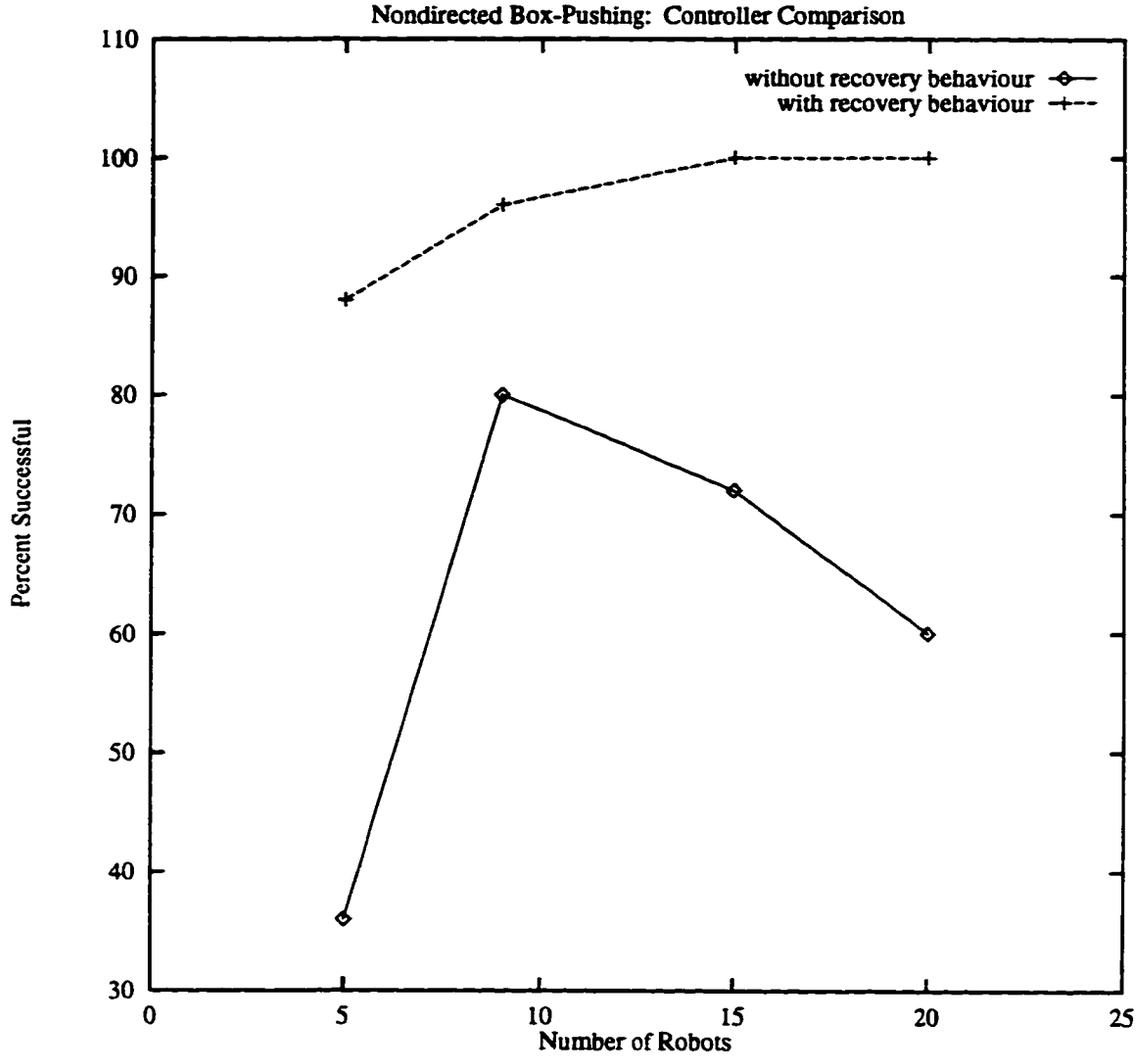


Figure 5.7: Shown is a comparison of a nondirected box-pushing controller with and without a stagnation recovery behaviour. Success rate is plotted as a function of system size, where success was defined as pushing the box 200 units from its initial position within 2000 simulation timesteps. Each data point represents 25 trials with the number of robots placed at a random initial position. The recovery behaviour tested was REALIGNMENT which randomly changes the angle of pushing force upon detecting no robot movement while pushing within a fixed time period.

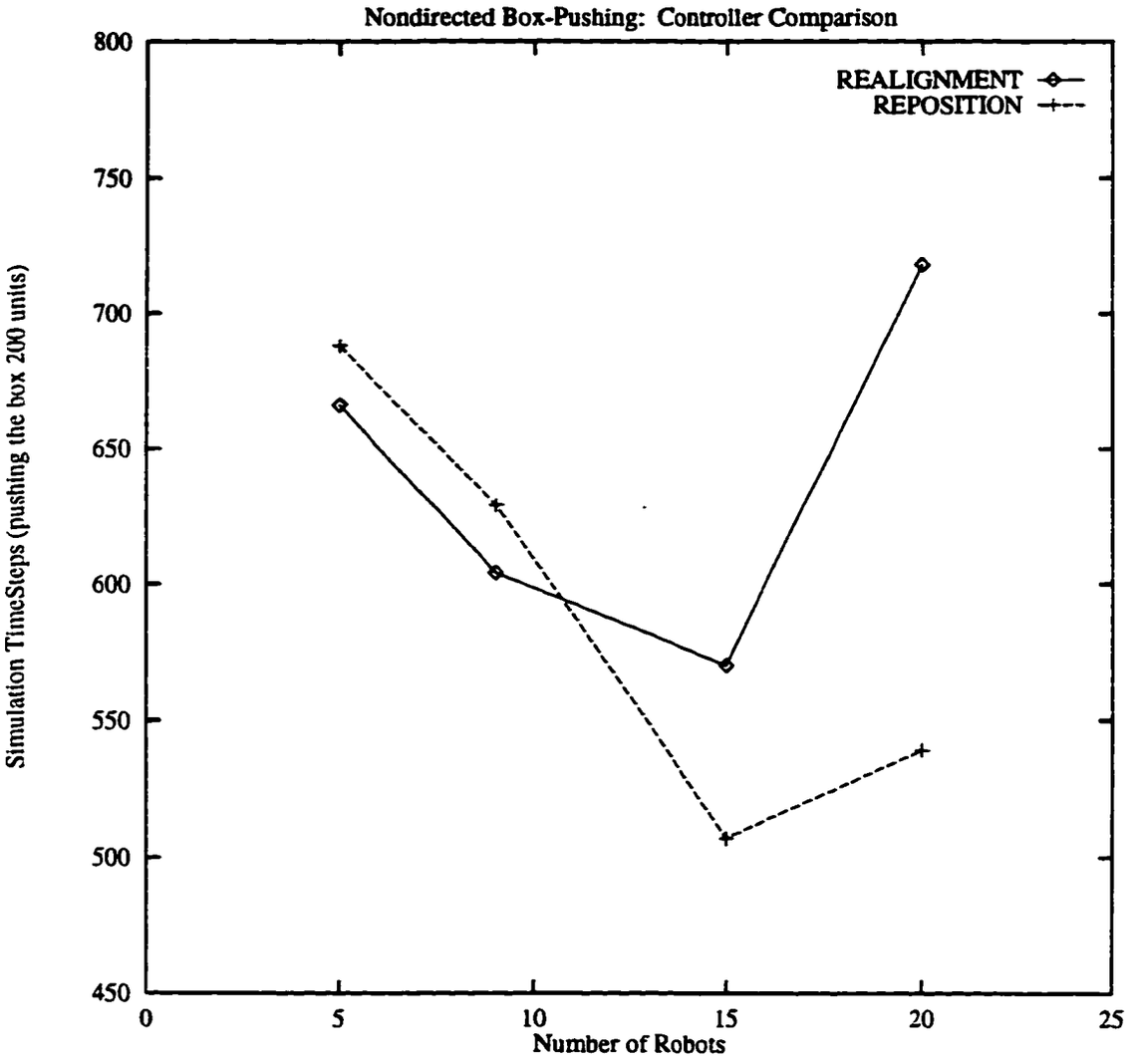


Figure 5.8: A comparison of the REALIGNMENT and REPOSITION recovery behaviours. Since REALIGNMENT does not change the position of the robot on the box, changes in force magnitude are smaller than found in REPOSITIONING.

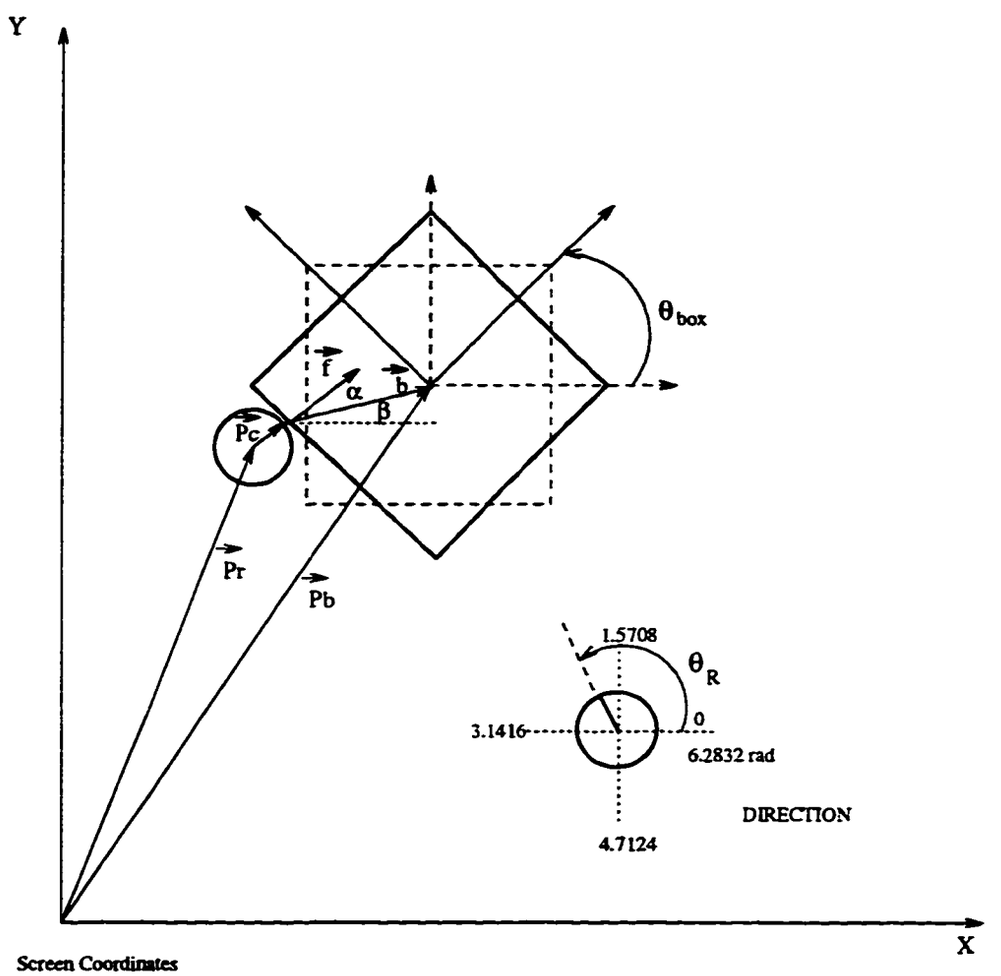


Figure 5.9: The model used to calculate the box force vector.

Directed Box-Pushing

The problem of transporting a heavy box from an unknown location, in a given environment, to a known goal destination can be divided into three subtasks:

- **Find the Box.** Since physical sensors used for perception have a limited range and field-of-view, the robot must search the environment for the box while avoiding collisions with obstacles.
- **Move to the Box.** Once the box is located within a robot's field-of-view, move towards the box while avoiding obstacles and bring the robot into contact with any side.
- **Push the Box towards the Goal.** If the box is between the robot and the goal destination then push the box; otherwise reposition the robot to another spot on the box.

The reactive controller designed for each subtask consists of a method for achieving its task (goal-driven positive taxis PA behaviour), a method for dealing with impediments (avoidance-driven negative taxis PA behaviour), and a method for recovering from stagnation or deadlock conditions (recovery-driven kinesthetic PA behaviour). A minimally designed controller must contain at least one PA behaviour from the goal or recovery classes. Controllers used for navigation will also include behaviours from the avoidance class. A subtask controller can be modeled as a FSM with each state represented by a single PA behaviour. For the transport task the three subtask controllers are **FIND-BOX**, **MOVE-TO-BOX** and **PUSH-TO-GOAL**.

In order to integrate the subtask controllers into a single three state machine, with each state represented by a single controller, a mechanism is required to control state transitions. Perceptual cues are the computationally independent mechanism used. The cues can be expressed as Horn clauses with each atom representing a stimulus needed to satisfy the truth of the clause. The integrated controller is simply a machine that processes perceptual cues (Q) to determine the state (or subtask controller) that controls the robot, hence we call them *Q-machines*. Since the number of states is few, execution is controlled by cycling through the states using a perceptual cue to determine the correct state of the transport task Q-machine shown in Figure 5.10. The perceptual cues used to determine state are:

- ?BOX-DETECT True when the forward box sensors detect a lit box.

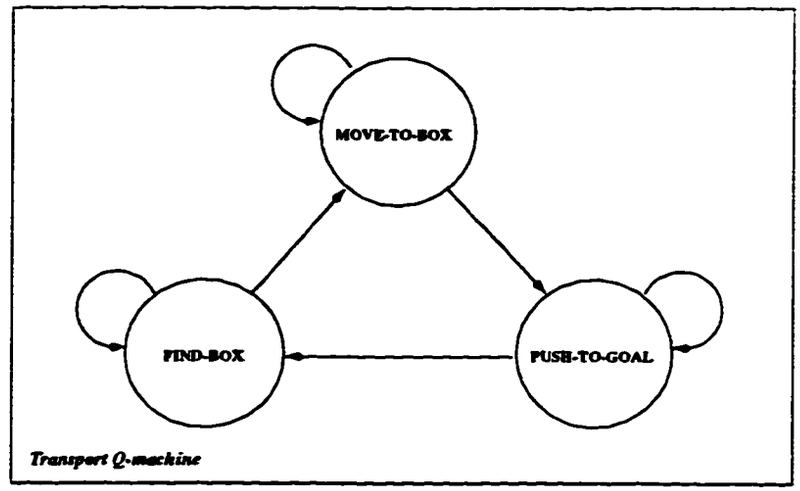


Figure 5.10: The behaviour-level description of the controller for transporting a box by pushing it from an initially unknown position to a final goal destination.

- ?BOX-CONTACT True when the robot is in contact with a lit box.

The state of the transport machine is specified by:

1. **FIND-BOX** $\Leftarrow \overline{?\text{BOX}-\text{DETECT}}$
2. **MOVE-TO-BOX** $\Leftarrow$?BOX-DETECT $\wedge$ $\overline{?\text{BOX}-\text{CONTACT}}$
3. **PUSH-TO-GOAL** $\Leftarrow$?BOX-DETECT $\wedge$?BOX-CONTACT

Each subtask controller is a FSM with state represented by a single PA behaviour described in Chapter 3. Described next are the subtask controllers and their implementation.

FIND-BOX Subtask Controller

A machine is designed that will find a brightly lit box in the robot's environment while avoiding obstacles. From the given list of PA behaviours a random search strategy (RANDOM-WALK) is chosen. Two behaviours from the avoidance class (CONTACT, AVOID) along with an additional recovery class behaviour(BACK-OFF) complete the controller whose pseudo-code algorithm is shown in Figure 5.11. An alternate design might have used a more methodical search and movement generator in place of RANDOM-WALK, or any one of the many strategies for spatial searching. The perceptual cues used to determine the state of the Q-machine are listed in Table 5.1.

Control is maintained within **FIND-BOX** until the?BOX-DETECT perceptual cue becomes true, at which point control is passed to the **MOVE-TO-BOX** subtask controller. As indicated in Table 5.1 a fixed priority is maintained between the PA behaviours as a means

FIND-BOX Subtask Controller			
Perceptual Cue (Input)			Behaviour State (Output)
?TOUCH	?CONTACT-	?AVOID-	PA Behaviour
0	0	0	RANDOM-WALK
0	0	1	AVOID
0	1	X	CONTACT
1	X	X	BACK-OFF

Table 5.1: The **FIND-BOX** Q-machine is the subtask controller used to locate the box to be manipulated. Input is from the listed perceptual cues which define the output behaviour state specified as a primitive actuation (PA) behaviour. The "X" in the input table indicates a *don't care* term. The perceptual cues corresponding to the dashed labels are: ?CONTACT- = ?CONTACT-DETECT; ?AVOID- = ?AVOID-DETECT described in Chapter 4.

```
: FIND-BOX  ( --- )
    NOT ?BOX-DETECT  ( transition perceptual cue )
    WHILE ( while you don't see the box search for it )
       IF ?TOUCH          THEN BACK-OFF  ( kinesthetic )    ELSE
       IF ?CONTACT-DETECT THEN CONTACT   ( negative taxis ) ELSE
       IF ?AVOID-DETECT   THEN AVOID     ( negative taxis ) ELSE
       RANDOM-WALK  ( kinesthetic )
;
```

Figure 5.11: Shown is the pseudo-code for the **FIND-BOX** Q-machine. Each state in the FSM is a primitive actuation behaviour described in Chapter 3. Comments are in lower case and enclosed in parentheses.

of behaviour arbitration. Using a fixed priority in behaviour arbitration is simple if the number of behaviours is few making control unambiguous. If the control choice is ambiguous then an additional subdivision of the task into another subtask controller is advised. An alternate means of behaviour arbitration among a larger number (> 5) of competing behaviours was explored and the results presented in [39].

MOVE-TO-BOX Subtask Controller

Once the box stimulus has been detected control passes to the **MOVE-TO-BOX** subtask controller responsible for guiding the robot towards the box. The controller is identical to **FIND-BOX** with the movement generator (RANDOM-WALK) replaced with a goal driven behaviour (SEEK-BOX) based on a positive phototaxis. The pseudo-code algorithm illustrated in Figure 5.12 brings the robot into contact with any side of the box so that the pushing force is normal to the point of contact.

MOVE-TO-BOX Subtask Controller				
Perceptual Cue (Input)				Behaviour State (Output)
?TOUCH	?CONTACT-	?AVOID-	?BOX-	PA Behaviour
0	0	0	1	SEEK-BOX
0	0	1	X	AVOID
0	1	X	X	CONTACT
1	X	X	X	BACK-OFF

Table 5.2: The **MOVE-TO-BOX** Q-machine is the subtask controller that moves the robot towards any side of the brightly lit box to be manipulated. Input is from the listed perceptual cues which define the output behaviour state specified as a primitive actuation (PA) behaviour. The "X" in the input table indicates a *don't care* term. The perceptual cues corresponding to the dashed labels are: ?CONTACT- = ?CONTACT-DETECT; ?AVOID- = ?AVOID-DETECT; and ?BOX- = ?BOX-DETECT described in Chapter 4.

```
: MOVE-TO-BOX  ( --- )
     ?BOX-DETECT AND NOT ?BOX-CONTACT ( transition perceptual cue )
     WHILE ( while you see the box move towards it )
        IF ?TOUCH          THEN BACK-OFF  ( kinesthetic )    ELSE
        IF ?CONTACT-DETECT THEN CONTACT   ( negative taxis ) ELSE
        IF ?AVOID-DETECT   THEN AVOID     ( negative taxis ) ELSE
        SEEK-BOX ( positive taxis )
;
```

Figure 5.12: Shown is the pseudo-code for the **MOVE-TO-BOX** Q-machine. States are primitive actuation behaviour described in Chapter 3. Comments are in lower case and enclosed in parentheses.

If during the course of navigating towards the box one of the avoidance driven behaviours causes the robot to lose sight of the box control is passed back to the **FIND-BOX** subtask controller. Control is maintained within the controller until the ?BOX-CONTACT cue is true, at which point control passes to **PUSH-TO-GOAL**.

PUSH-TO-GOAL Subtask Controller

The strategy used to move a box towards a goal position is based on positioning the robot so that the box to be pushed is between the robot and the goal position (see Figure 5.13). Once in contact with a side of the box the ?SEE-GOAL cue determines if the robot is correctly positioned. If the cue is true the robot begins to push and if the cue is false the robot executes a repositioning behaviour. Continuous execution of the kinesthetic REPOSITION behaviour causes the robot to move in a counterclockwise direction around the box. While

PUSH-TO-GOAL Subtask Controller Perceptual Cue (Input) Behaviour State (Output)	
?SEE-GOAL	PA Behaviour
0	REPOSITION
1	PUSH-BOX

Table 5.3: The **PUSH-TO-GOAL** Q-machine is the subtask controller that either pushes the box towards a goal destination or repositions the robot on another position of the box to be manipulated. Input from the ?SEE-GOAL perceptual cue which determines pushing angles can vary the acceptable pushing angles.

```
: PUSH-TO-GOAL  ( --- )
     ?BOX-DETECT AND ?BOX-CONTACT ( transition perceptual cue )
     WHILE ( while in contact with the box )
        IF ?SEE-GOAL  THEN PUSH-BOX ( kinesthetic ) ELSE
        REPOSITION  ( kinesthetic )
;
```

Figure 5.13: Shown is the pseudo-code for the **PUSH-TO-GOAL** Q-machine. Once positioned on a box side the robot determines if the goal stimulus is within a fixed field of view needed for pushing. If not, a kinesthetic behaviour repositions the robot. Comments are in lower case and enclosed in parentheses.

in contact with a boxside avoidance behaviour is not relevant and therefore not part of the controller whose states are listed in Table 5.3.

Control is maintained within the **PUSH-TO-GOAL** controller as long as the robot remains in contact with the box. However, should the robot lose sight of the goal stimulus as determined by ?SEE-GOAL the REPOSITION behaviour forces the robot to break contact with the box thereby passing control to either **FIND-BOX** or **MOVE-TO-BOX** depending on the state determined by their respective perceptual cues.

Since the movement of the box towards a goal position is not specified as an explicit path in two dimensional space, but rather as a rule of interaction between the box, the robot and the goal position, a unique solution is never obtained. Instead, like declarative programming in which *what* is specified rather than *how*, the rules of interaction (Q-machines) provide a method for achieving the solution.

```
: TRANSPORT  ( --- )
     BEGIN
        FIND-BOX
        MOVE-TO-BOX
        PUSH-TO-GOAL
     AGAIN
;
```

Figure 5.14: Shown is the Forth code for the *TRANSPORT* Q-machine. The three state machine uses transition perceptual cues to determine which state is relevant based on local sensing.

Transport Q-machine

The *TRANSPORT* machine is created by cycling through the three behaviour-level controllers as shown in Figure 5.14. The correct state is determined by the ?BOX-DETECT and ?BOX-CONTACT cues. As indicated in the task description graph (shown in Figure 5.6) different goal positions towards which the box is pushed are specified by externally controlling the time at which the goal stimulus is present. The system of robots simply pushes the box towards the active goal stimulus which indicates where the box is to be transported.

5.5 Summary

Social insects offer one of nature's most startling examples of coherent behaviour from a collective system. The well coordinated effort found in nest building results in repeatable structures that prove the activity is more than just random behavioural acts. As shown in the bulldoze-building behaviour of ants, simple rules of behaviour governed by local sensing results in a predictable global action without resorting to directly communicating building intentions between ants. Rather, indirect communication through the task itself is sufficient to produce a coherent behaviour.

In insects, evolution provided the programming skills necessary to produce behavioural programs finely tuned to the environmental niche they inhabit. In robots, solutions that encompass both robots and their environment as part of the model may achieve nature's intention in solving complex problems using decentralized mechanisms.

In the model presented, the problem—specified as observable changes in the environment—becomes part of the solution by directing the robots' behavioural acts to converge in a desired way.

Task description graphs capture both the spatial and temporal stimulus changes that define the task to be accomplished. Together with Q-machines, which are based on finite state automata, and their context dependent behavioural state, a unique approach is taken to the control of a collective multi-robot system.

Verifying the feasibility of the model on a physical system of mobile robots presents a new challenge in itself. Designing, prototyping, testing and then replicating 11 identical mobile robots so that repeatable experiments could be performed is time consuming. However, the results as discussed in the next chapter were rewarding in themselves, but not so much in the positive confirmation of the proposed framework, rather in the questions raised, and discussed in the sequel. These open new vistas for exploration.

Chapter 6

Global Action: Results

Stigmergy, a term coined by French biologist P. Grassé, which means to incite work by the effect of previous work [26] is a principle finding its way from the field of social insects to collective robotics [11, 70]. With their limited repertoire of behavioural acts social insects display an amazing competence in building nest structures. From the simple nests produced by the blind bulldozing of ants [23] to the termite homes that stand over a meter tall [68] all of which result from common task coordination that does not appear to depend on interaction between the agents but rather on the object they act upon. In this chapter, the results are presented for the integrated models of the previous three chapters. The resulting Q-machine model takes local perception derived by the perceptual cues presented in Chapter 4 and the context sensitive decision process of Chapter 5 to produce the taxis-based actions of Chapter 3, which together form a coherent global action. This global action is demonstrated in the collective transport task and represents another step in the pursuit of knowledge already obtained by social insects.

6.1 Introduction

In the results presented, global action is the effect produced when a set of identical mobile robots execute the common task of pushing an object towards an arbitrarily specified goal position. Coordination is achieved without resort to direct inter-robot communication or robot differentiation. Instead, context sensitive subtask controllers decompose the box transport task into three phases. The phases describe *what* is to be achieved, in terms of the externally observable events described by box position, without specifying *how* the task is to be accomplished by way of a unique path.

Described is the experimental system used to gather the data followed by the primary results in support of the main hypothesis and the secondary results which make some preliminary comparisons of execution times under various experimental conditions.

6.2 Experimental System

The experimental setup used to gather the data presented in the sequel consisted of a robot environment, in which various boxes were placed along with two spotlights used to indicate final goal positions, and a set of identical mobile robots complete with sensors and Q-machine task controllers. In total over 100 box-pushing trials were run using from one to 11 robots, four different box types and in three different venues. The final set of experiments were recorded on over four hours of video tape with an individual trial lasting between 30 seconds and five minutes. Described briefly is both the robot environment and hardware used.

6.2.1 Robot Environment

The ideal test environment would be a large open space without walls leaving the robots free to push the box along any desired path. Since this environment was not available a smaller and more restrictive area defined by walls was used. A permanent space large enough in which to conduct experiments was often difficult to find, resulting in the creation of a portable testing environment consisting of: 11 robots, two spotlights on stands for goal position indicators, the box to be manipulated, and a video camera to record the results. However, the majority of the experiments were conducted in the area depicted in Figure 6.1 which became available towards the end of this study.

6.2.2 Robot Hardware

The mobility hardware designed is composed of a set of homogeneous two-wheeled robots, each weighing 1.3 kilograms and measuring approximately 18 centimeters in height and diameter as shown in Figure 6.2. A battery allows for 45 minutes of operation with a 10 minute recharge time. Control electronics are separate modules plugged into the robot, allowing for quick maintenance in the event of failure.[1] Task specific sensors are added to the generic base and can be attached onto a grid of evenly spaced holes. A Motorola 68HC11 microcontroller with 8K of RAM and programmed in Forth is used to map sensor output to one of nine motion primitives. A minimum number of sensors (6) was sought in

[1]See Appendix A for a detailed description.

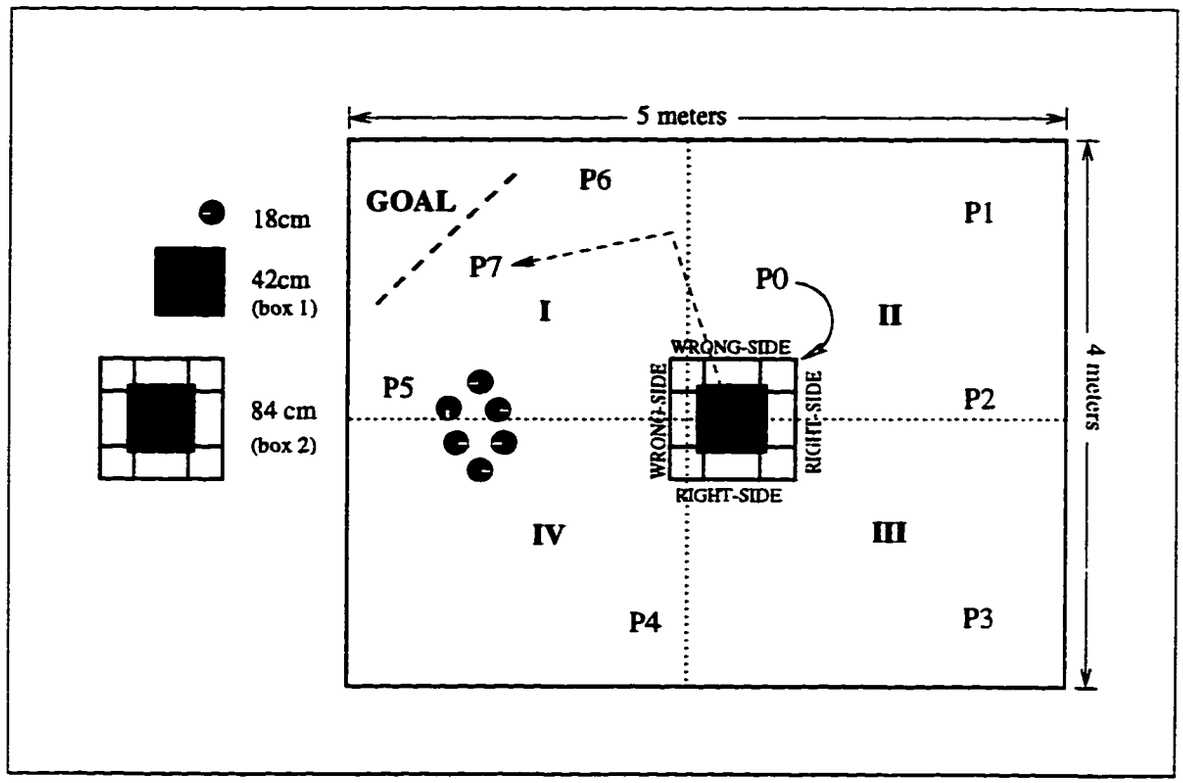


Figure 6.1: A schematic of the lab environment used to test the integrated transport controller. In each trial the box was placed at initial position three meters from the goal line and the robots were placed at one of the indicated starting positions labelled P1 - P5.

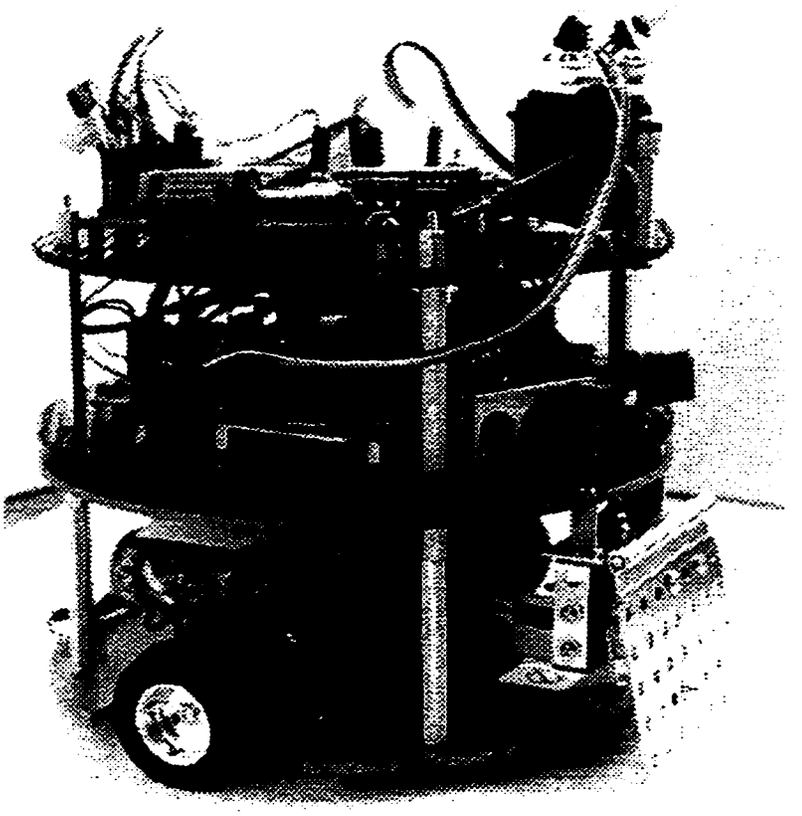


Figure 6.2: Each of the robots are equipped with two forward pointing infrared obstacle sensors, one touch sensor, two CdS box-tracking photocells, and a destination sensor, all mounted on a differentially steered base.

implementing the perceptual cues. Additional sensors would allow a more omnidirectional field-of-view in the case of obstacle and box sensing, and better pushing orientation in the case of box contact sensing, but the objective was to determine what could be accomplished with the minimal number of sensing bits. The hardware proved to be robust with few breakdowns.

6.2.3 Controller Verification

Each subtask controller was tested individually on the physical robots and then combined to form the final integrated controller used in the transport task. The three subtask controllers are **FIND-BOX**, **MOVE-TO-BOX** and **PUSH-TO-GOAL**.

First, the **FIND-BOX** controller was tested for its ability to search the robot's environment for a brightly lit box. **FIND-BOX** is a four-state machine consisting of RANDOM-WALK, AVOID, CONTACT and BACK-OFF PA behaviours. To test the controller a felt-tipped pen is

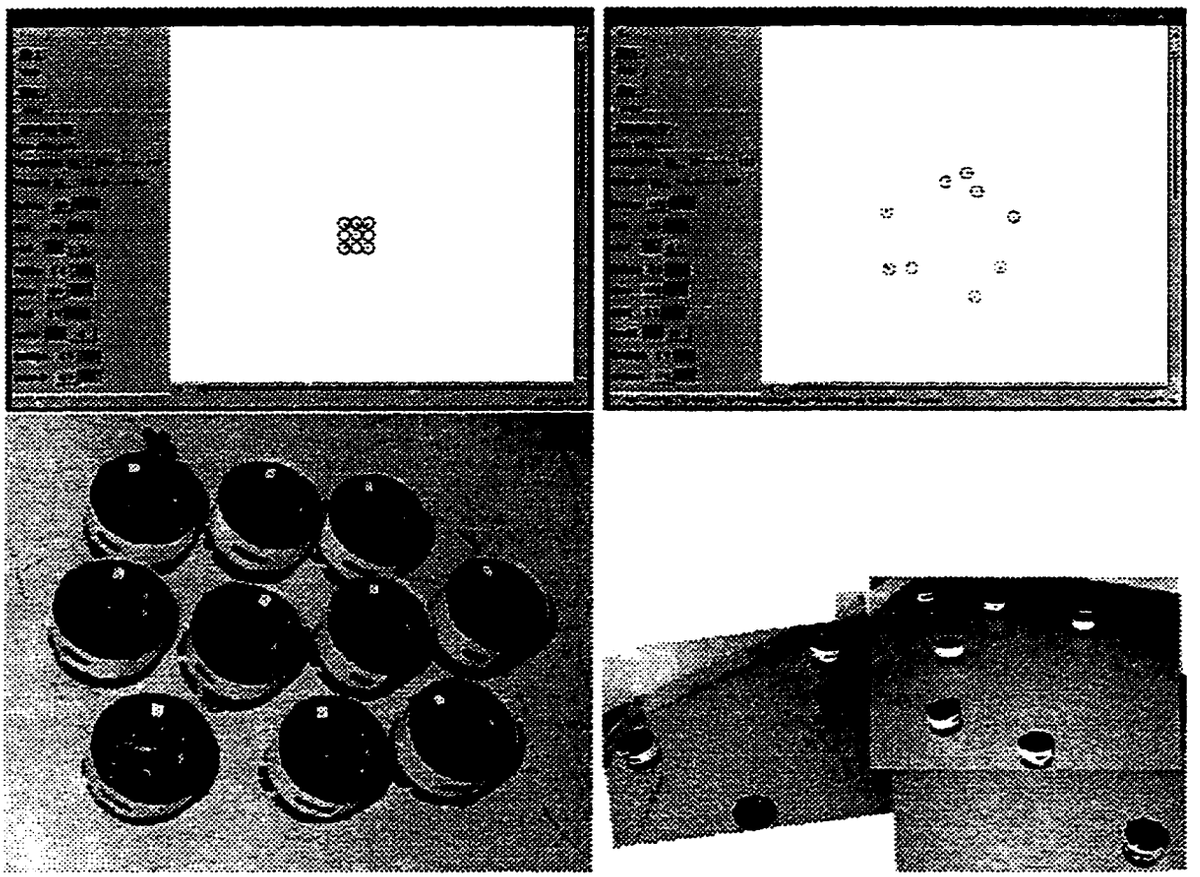


Figure 6.3: Shown are the dispersive effects of obstacle avoidance behaviours in both simulated and physical robot experiments. Starting from a close initial configuration, the **FIND-BOX** controller will disperse the robots until the obstacle sensors are inactive. Adjusting the sensor's threshold effects the inter-robot distances. Note that the obstacle sensors are not omnidirectional and point in the forward direction only.

attached to the robot marking its traveled path. A single robot was placed in a 2.7 meter square enclosed grid composed of 81 cells of which 90% were visited in three minutes. The fixed motion pattern generates a pseudo-random motion when the obstacle sensors change the path of the robot. The same controller tested with 11 robots in a five by six meter room produces continuous stagnation-free motion for 10 minutes.

An example of how the avoidance behaviours, within the **FIND-BOX** controller, cause robot dispersion is shown in Figure 6.3. If the robots are placed close to each other in an initial configuration so that obstacle sensors are active, the robots will disperse using avoidance behaviours until the obstacle sensors are inactive. The inter-robot distance is a function of the obstacle avoidance thresholds [32].

Next, the **MOVE-TO-BOX** controller was tested for its ability to detect and direct the

robot towards a box, resulting in contact with a side. **MOVE-TO-BOX** is a four-state machine composed of SEEK-BOX, AVOID, CONTACT and BACK-OFF PA behaviours. The controller was first tested using a single robot which followed a lit box as it was dragged around the room. The **FIND-BOX** controller is added and 10 robots were placed at opposite ends in a three by five meter room. All robots located the box while obstacle avoidance created an even distribution around its circumference.

In order to test the transition perceptual cue between **FIND-BOX** and **MOVE-TO-BOX** two experiments were run in two separate environments. In the first, 10 robots were placed in one corner of a five by three meter room with the box located at the opposite end. The initial and final configurations were then compared against the simulated version of the same experiment. In both versions the initial configuration of robots expanded spatially avoiding obstacles while executing the RANDOM-WALK PA behaviour. Once their forward pointing light sensors detected the box they converged and attempted to occupy a spot on a box side. Additional robots converged until obstacle sensors detected an object already present on a boxside, forcing the additional robot to find a free spot. This resulted in a distribution of robots around the box and marked the end of the second task step (see Figure 6.4) [37].

The same results were obtained in a second environment in which six robots were placed in a six by 5.5 meter room with the box located 5.8 meters from the robots. In each of the three trials the robots located the box after an initial random search in under two minutes.[2] Finally, floor level lights placed at opposite corners of the same room were used to march nine robots back and forth across the floor in a simple homing experiment shown in Figure 6.5. Interference between interacting robots is minimized by reducing the distance at which obstacles are detected.

Finally, the **PUSH-TO-GOAL** controller is tested for its ability to provide directed box-pushing or repositioning behaviours. **PUSH-TO-GOAL** is a two-state machine using the PUSH-BOX and REPOSITION PA behaviours. The machine enters the PUSH-BOX state if the perceptual cue ?SEE-GOAL is true. ?SEE-GOAL is true if the box is between the robot and the destination goal as illustrated in Figure 6.6. The cue is created using an upward

[2]One item of interest not modelled in the simulation experiment is that once the box is surrounded by robots it becomes undetectable to the remaining robots since the radiating box light is now blocked. This automatically frees the remaining robots to search for other boxes, a result that bares analogy to ant prey transport reported by Franks[24], who found that the number of ants that were involved in a group transport task was related to the available perimeter space on the item being retrieved. Thus, the number of robots directly participating in the task might be self regulating and a function of sensing rather than explicit control.

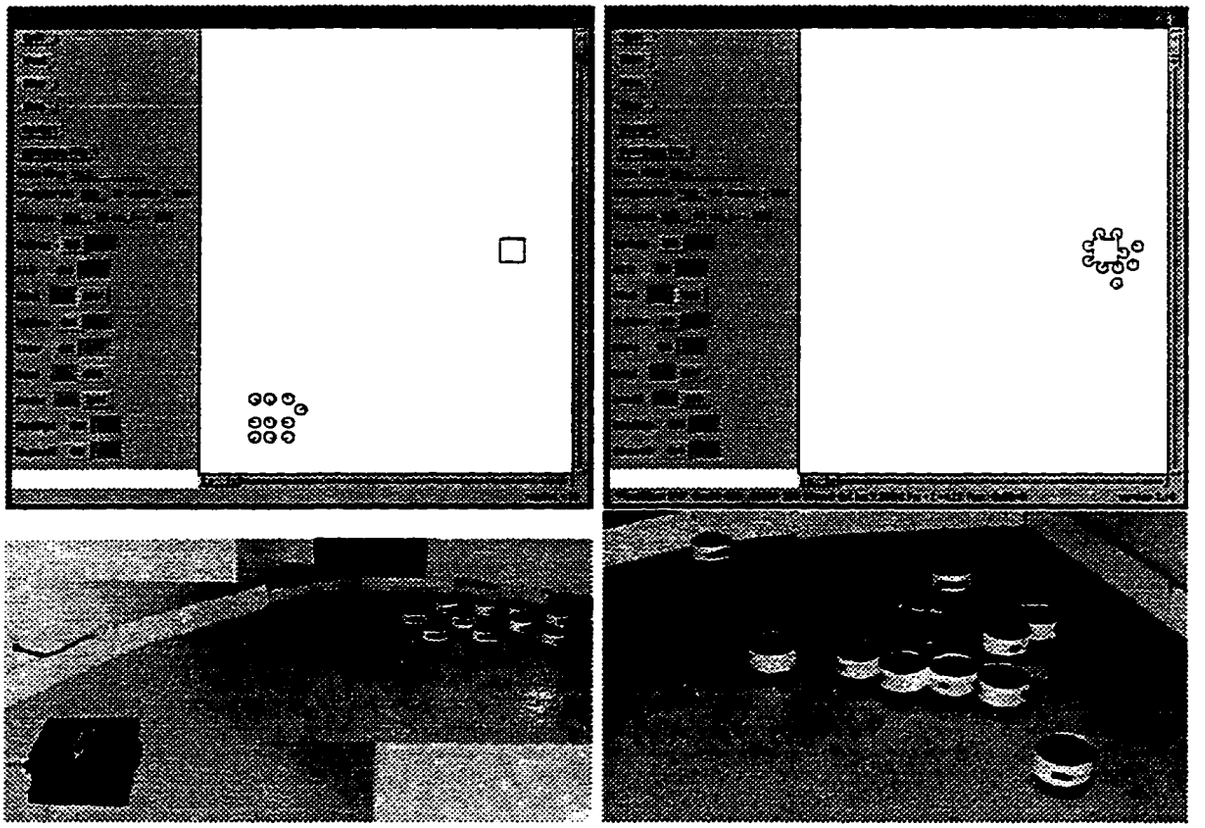


Figure 6.4: The results of both simulated and physical tests on locating a lit box. The distribution around the box results when the avoidance behaviours, AVOID and CONTACT, keep the robot away from other robots until a free spot against a boxside is found.

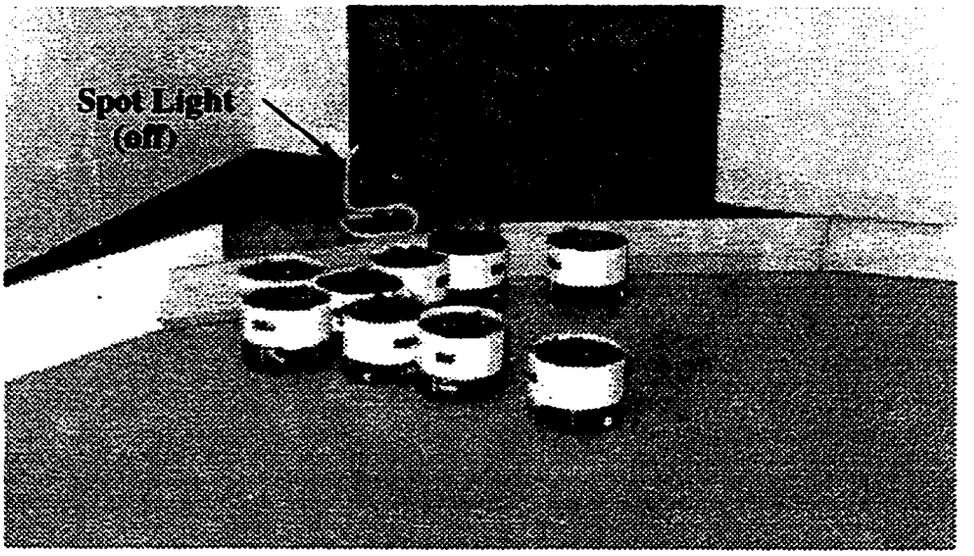


Figure 6.5: One test of the **MOVE-TO-BOX** controller involved marching nine robots back and forth between two floor level lights turned on alternately and placed at opposite corners of the room. A white shell is added to each robot so that a reflective surface is available for the obstacle sensors.

pointing rotating sensor which detects signal peaks within a specified field of view. To test the controller two robots, needed to push the box, are positioned on a boxside and facing the goal indicator. The robots successfully push the box towards the goal. Next the robots are placed on a box side facing away from the goal causing the REPOSITION state to move the robots to a new position and orientation. The final tests involve integrating the three subtask controllers into a machine that can transport the box from any initial position within the environment to an arbitrarily specified goal position.

6.3 Empirical Results: Directed Box-Pushing

The above three subtask controllers are combined to form a three state *transport Q-machine*. The integrated controller was tested in experiments using one to six robots in which four different types of boxes were transported from the same initial position to an arbitrary final goal position in the environment depicted in Figure 6.1. Task complexity was increased by changing the goal positions during transport execution, thereby requiring the robots to dynamically reconfigure their pushing orientation towards the new goal position. The results are presented in support of the primary hypothesis followed by secondary results which require additional experiments for statistical conclusions.

6.3.1 Primary Results

The primary hypothesis that coherent behaviour from a multi-robot system, in some tasks, does not *require* explicit cooperation mechanisms was examined by gathering experimental evidence on a directed box-pushing task. Over 50 successful trials were recorded of a physical system of robots in which a box, requiring at least two robots to move, was pushed from an initial starting position towards a specified goal position. The control framework tested did not make use of inter-robot communication or robot identification to coordinate the system. Rather, decentralized control was used in which the autonomous robots made use of local sensors as the only means of observing their environment. Experiments were conducted which varied the number of robots, the box size and shape, and task difficulty by requiring the robots to transport the box between a sequence of goal positions. From the experimental data the following statements can be made in support of the primary hypothesis:

- **Insensitive to System Size.** The number of robots used to transport the box to different goal positions was varied from two to six with 58 successful trials recorded.

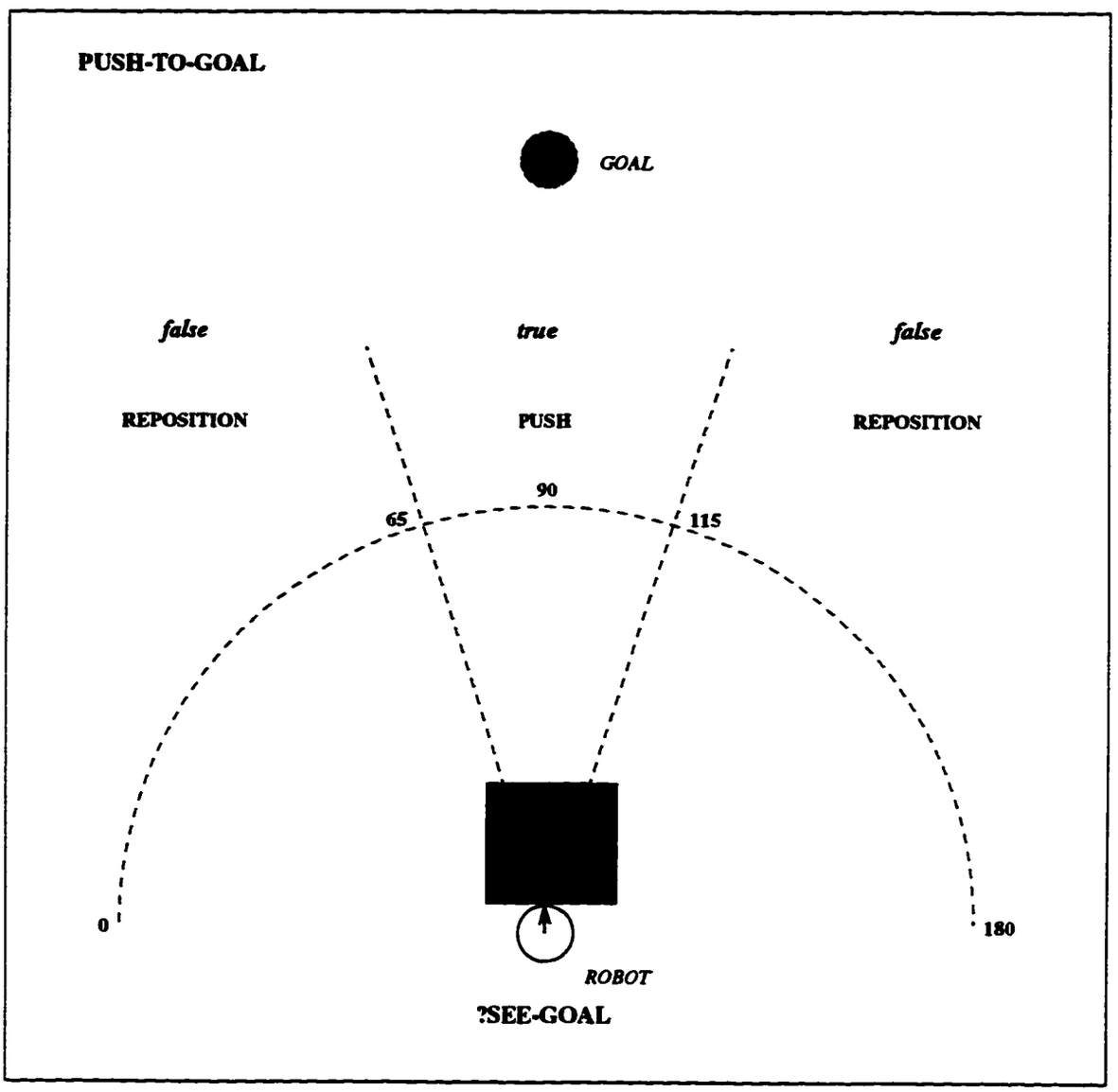


Figure 6.6: The actions taken by the **PUSH-TO-GOAL** machine depend on the ?SEE-GOAL perceptual cue. If the goal is within the sensor's field-of-view the machine is controlled by the PUSH-BOX PA behaviour; otherwise control is passed to REPOSITION which causes the robot to locate another spot on the box.

The success of the transport task was not sensitive to the number of robots used to compose the system.

- **Insensitive to Some Types of Convex Object Geometry.** Six robots and four different box types of varying size and shape, were transported with 39 successful trials recorded. The transport task success was not sensitive to the four box types used.

- **Insensitive to Changes in Goal Position.** Using between four and six robots to transport a large round box between a sequence of goal positions, eight successful trials were recorded. The transport task success was not sensitive to the increase in task complexity.

In the sections that follow, successful trials will be discussed with regard to the above three statements. The unsuccessful trials can be grouped into two: failure due to a robot system fault, or failure due to an environment restriction. A system fault included problems specific to the robots. For example, a tire or wheel fell off disabling the robot, a rundown battery leaving little pushing force, or the door to the lab was left open and the robots left the arena. Failures also occurred due to restrictions, previously mentioned, on the environment used to conduct the experiments. A failure occurred if the box was pushed against a wall leaving no space for the counterclockwise repositioning needed to bring the robots in correct alignment with the goal. Despite these restrictions the system was successful in transporting the box in over 70% of the trials conducted.

Insensitivity to System Size

Increasing the number of robots from two to six did not affect the successful outcome of the transport experiments. This is an analogous result to the simulation results (shown in Figure 6.11) in which successful task completion remained high despite an increase in the number of robots. However, no claim is being made that task completion time is not affected, since completion times were found to vary as the number of robots increased and were dependent on available resources as discussed in Chapter 7. In each of the 58 successful trials recorded the box was pushed from an initial starting position, located approximately in the center of a five by four meter area, towards the goal area indicated in Figure 6.1 and ending in quadrant I at a distance of at least 2.5 meters. The robots were started in each trial from positions one to five in quadrants II-IV shown in Figure 6.1. Successful trials would run between 32 and 214 seconds and were executed in three phases.

The first phase began when the robots were powered on, the box-light was off and the goal-light was on. System initialization consists of taking ambient light readings used to set the box-detection threshold. The robots began executing **FIND-BOX** and quickly dispersed themselves in the area. Shortly thereafter, the box-light was turned on and those robots that were facing the box and sufficiently close would move towards and make contact with a boxside using the **MOVE-TO-BOX** controller. Depending on an individual robot's position, with respect to the box when box-detection occurred, the distribution of robots around the box would vary and mark the beginning of the second phase.

In the second phase, some of the robots incorrectly positioned for pushing, as determined by the **PUSH-TO-GOAL** controller, began moving counterclockwise around the box perimeter searching for an open spot on a correct side. This behaviour is the result of several cycles through the transport Q-machine consisting of in turn **FIND-BOX**, **MOVE-TO-BOX** and **PUSH-TO-GOAL** subtask controllers and can be explained as follows. Once contact is made with a boxside the ?SEE-GOAL perceptual cue determines that the robot is on the wrong side for pushing. The **PUSH-TO-GOAL** controller then executes the REPOSITION behaviour which moves the robot away from the box in a counterclockwise direction. If the box is within view, determined by the ?BOX-DETECT cue, **MOVE-TO-BOX** brings the robot into contact with a new position on the box providing it is unoccupied. The obstacle avoidance behaviours keep a robot away from occupied positions on a boxside. If the box is not within view then **FIND-BOX** executes and searches for the box. The **PUSH-TO-GOAL** controller evaluates the new position and the cycle repeats.

The third and final phase is characterized by the box moving towards the goal position. Once a net force sufficient to move the box occurs the box begins to translate and possibly rotate. During the box movement phase a robot continuously determines if it remains on the correct side for pushing using the ?SEE-GOAL cue. A robot located at the edge of the pushing swarm may suddenly lose site of the goal and begin repositioning. The resulting drop in pushing force may be sufficient to halt the box movement until another robot joins the group effort. The dynamics of both the box and robots is such that the path taken by the box towards the goal is seldom straight. Rather, box movement can be said to converge towards the goal since its trajectory is the net result of several force vectors applied by individual robots. A typical box path might begin at position P_0 proceed towards P_6 and then move to P_7 as illustrated in Figure 6.1.

Figure 6.7 is taken from a 45 second video segment in which six robots starting from position P_4 moved the box a total of three meters ending on the goal line just behind

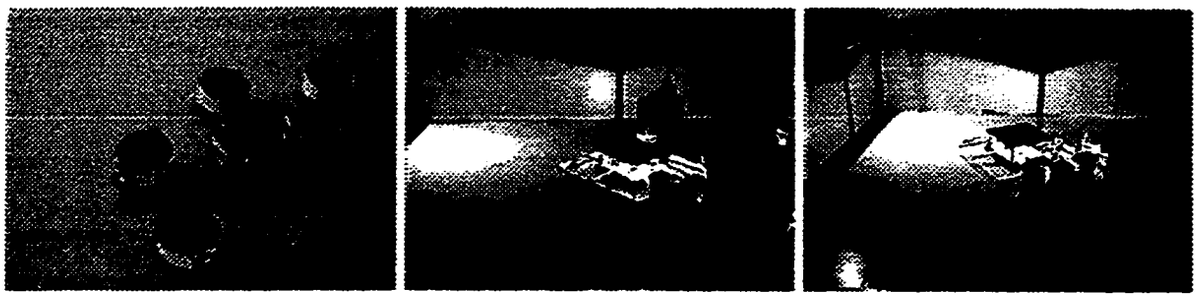


Figure 6.7: Shown are six robots pushing an large box from its initial position three meters towards a final goal. The mpeg video from which this sequence was taken is available at http://www.cs.ualberta.ca/~kube/

position P_7. On the whole, the time taken would depend on the size of the box and the number of robots as explained in section 6.3.2, but the success of the approach was not sensitive to the number of robots providing that the minimum of two robots were used.

Insensitivity to Some Types of Convex Object Geometry

To evaluate the controller's sensitivity to object geometry, 38 successful trials were performed using six robots and four different box types. The initial box, BOX A, tested was 42 centimeters square and large enough for two 18 centimeter robots on a side. A second 84 centimeter square box, BOX C, was built by extending the initial box with a second frame. This increased the box dimensions, but used the same base on which the box slid along the floor. A third 84 centimeter box, BOX B, was built on a new base which increased the number of points in contact with the floor and therefore its sliding friction. The fourth box, BOX D, was round with a diameter of 84 centimeters and the results of the 39 trials can be summarized as follows:

- BOX A. A total of 10 trials were successful in pushing BOX A from the initial position to the goal positions in quadrant I (see Figure 6.10). The robots started from positions P_{1-5}. In general as the number of robots increased the task took longer to complete as the robot interference was high since the limited box side space created competition among the robots.

- BOX B. A total of eight trials were successful in transporting BOX B from its initial position using 6 robots starting from position P_4 and ending at positions P_{5-7}.

- BOX C. A total of seven successful trials were recorded in which BOX C was moved to the goal area by six robots starting from positions P_{2-4}. This box had the highest

failure rate among the four boxes used and was due to a robot getting caught on the frame.

- BOX D. A total of 14 trials using a round box, BOX D, and four to six robots were successful in moving the box between two goal positions. The round box was the last box built and experienced the most success of the four types tested. The lack of corners provided the robots with a uniform contact surface to push against unlike the square boxes which had sharp points at its corners.

Insensitivity to Changes in Goal Position

The initial success of the directed box-pushing task led to the following extension which increased the task difficulty. Pictured in Figure 6.8 are two goal positions labelled P_A and P_B. The robots begin from position P_4 and goal-light at position P_A is illuminated causing the robots to push the box towards P_A. Once reached the goal-light at P_A is turned off and the goal-light at P_B is switched on. The robots reposition around the box and begin pushing towards the goal at P_B. Figure 6.9 is a sequence of three images taken from the a video segment in which two goals were used. A total of eight successful trials using three different goal positions were recorded using a single box.

6.3.2 Secondary Results

In the following discussion some interesting secondary results are presented which compare execution times as a function of system size in the first experiment and as a function of object geometry in the second experiment along with the following caveat. In experiments involving physical mobile robots, holding the many system variables invariant is near impossible making comparisons based on execution runtimes tenuous at best. In this experimentalist approach to robotics "things change" is axiomatic. Coefficients of friction change because the floor gets dusty, force is reduced because batteries run down, motors wear reducing repeatability, wheels slip in response to changes in load, and the list goes on. However, in general there still seems to be a trend in the data making it worth presenting.

System Size

The mean execution time for moving the smaller 42 centimeter square box from its initial position to the goal positions were compared for two to six robots as shown in Figure 6.10. Starting positions for the robots were varied and included $P_{1,3-5}$ with the final end position of the box recorded for timing to be $P_{5,7}$. Indicated in each plot are the number of trials

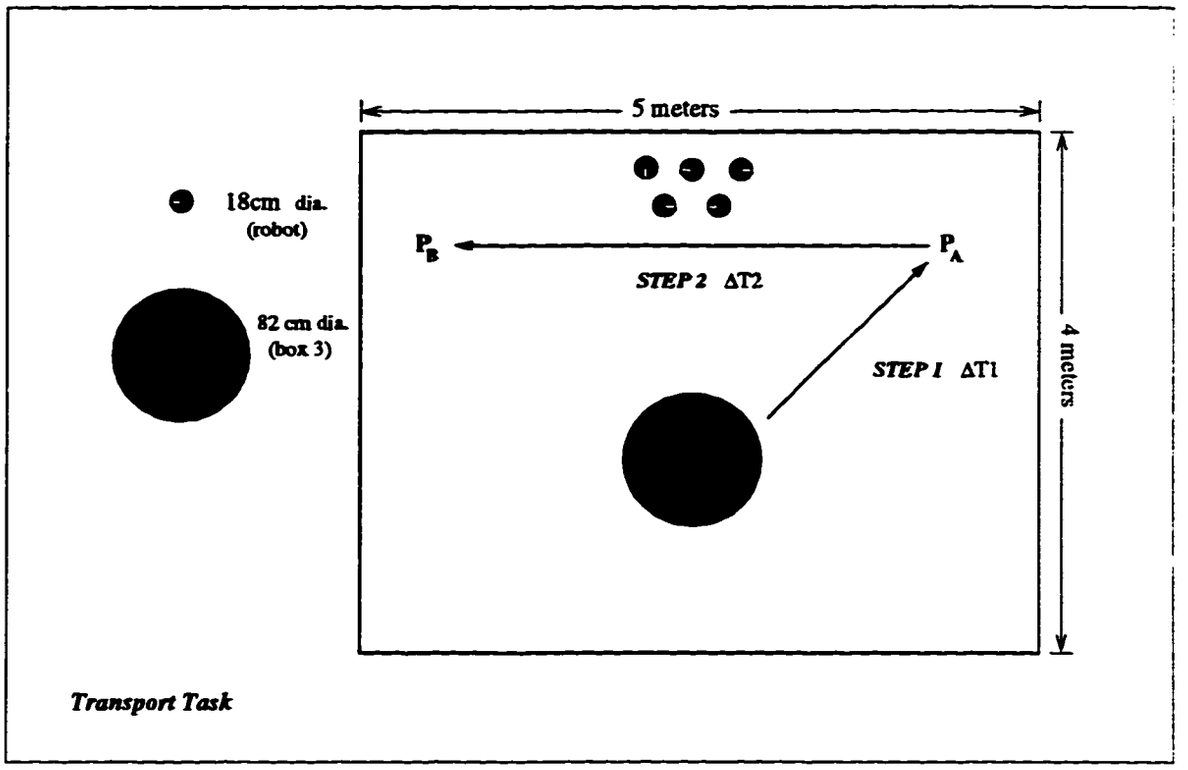


Figure 6.8: A schematic of the lab environment used to test the transport of a round box between two goal positions. Shown are the initial positions of the five robots and the box. The first step is to move the box from its initial position to the goal located at P_A. The second step moves the box from P_A to position P_B. The goal positions are indicated with a bright spotlight positioned at a height of 2.5 meters. To sequence the task steps the spotlight at position P_A is turned off and the light at P_B is turn on when the box reaches P_A.

used to compute the mean. The large variance in runtimes was due to robot start positions $P_{1,5}$ which could result in long repositioning phases [3]. In general, execution times increased as a function of the number of robots due to the increase in robot interference competing for the limited box space. A much larger number of trials is needed for any statistical conclusions.

Convex Object Geometry

Our previous simulation study had shown that in a box-pushing task performance, as measured by completion time or success rate, [4] could be improved if stagnation recovery behaviours were added to the controller to avoid deadlock from occurring when the robots applied an equal distribution of forces to the box [35]. What was also noted was the sudden

[3] Both the maximums indicated in the case of three and five robots occurred from P_5.

[4] Success was defined to be the movement of the box by 200 units in under 2000 simulation timesteps.

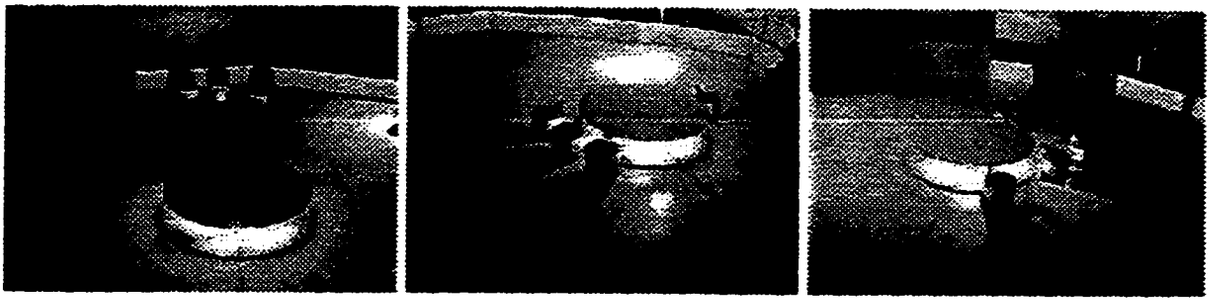


Figure 6.9: Shown are five robots pushing a round box from its initial position first towards a goal-light in the right of the picture and then towards a goal-light on the left of the picture. The mpeg video from which this sequence was taken is available at http://www.cs.ualberta.ca/~kube/

drop in performance as the size of the system grew for controllers without stagnation recovery. This was conjectured to be due to the number of robots able to fit on a box side. To test this hypothesis, simulations were run for the same behaviour controller and the robot diameter (RD) was tested for $RD = 10$ and compared with the results using $RD = 20$. The results are shown in Figures 6.11 and 6.12. If the diameter of the robots were reduced, for a fixed box side, the performance increases, which leads to the conjecture that for a given task, performance is dependent on some yet to be determined *task density* function.

In Figure 6.13 the mean execution times were compared for the four box types and six robots starting from the same initial position. In general, it appears that as the available contact space increases more robots are able to participate in pushing at the same time reducing the time taken to complete the task. However, due to the sparseness of the data additional experiments would allow statistical conclusions.

6.4 Summary

After reviewing the videotaped experiments one is reminded of the antagonistic forces present in ant group transport[48], yet the end result is invariably transport of the item back to the nest. In our multi-robot box-pushing experiments the path taken to the goal is neither optimal nor continuous, and it is not the same as one would get in a centralized controller, but rather it is a feasible solution to the problem given the limited abilities of the individual robots. There are even temporary setbacks as the box is moved incorrectly. At times the robots can lose contact with the box, be blocked by other robots, or be forced to relocate as the box rotates.

In the results presented, it can be said that in all cases the robots move the box towards

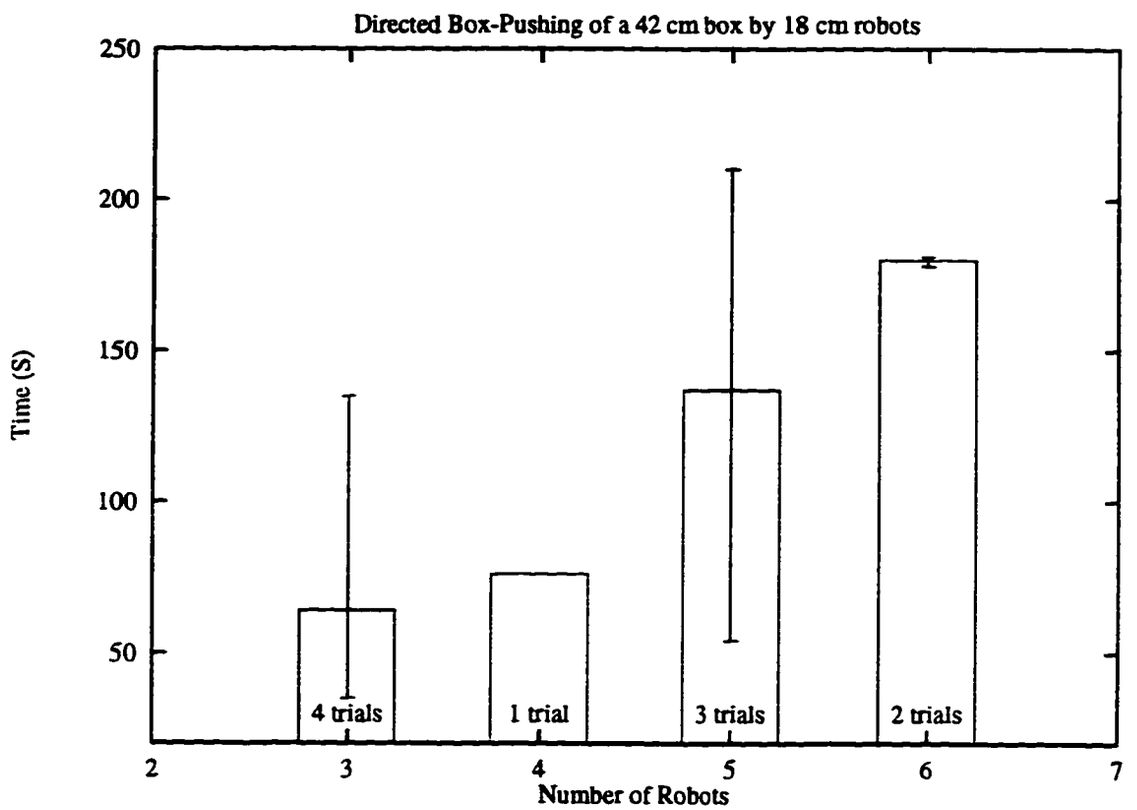


Figure 6.10: The mean execution time of moving a 42^2 centimeter box 2.5 meters towards a goal position (P5, P6, P7) as a function of the number of robots. For each plot the number of trials as well as the minimum and maximum run times are indicated. A boxside is approximately twice the robot's diameter and increasing the number of robots increases the robot interference as they compete for the limited space available.

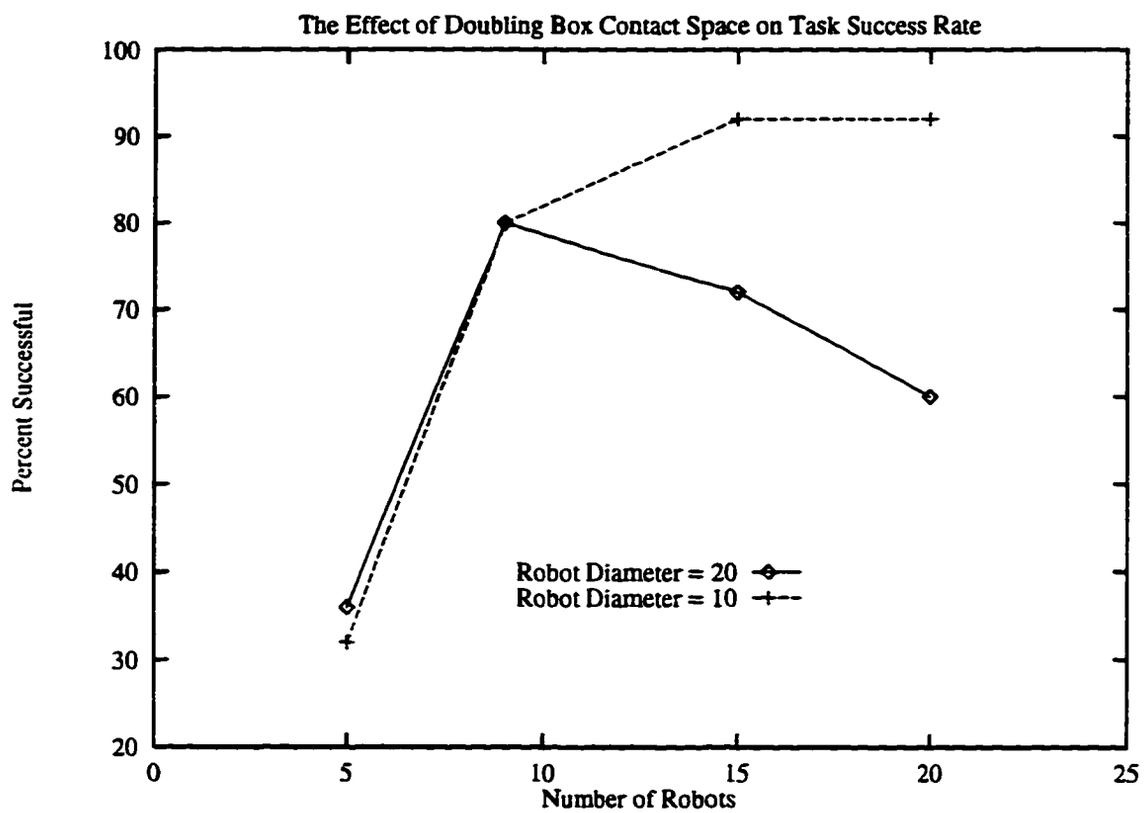


Figure 6.11: The effects of doubling box contact space on the task success rate. The results from two simulation experiments in which the only parameter changed was the robot's diameter, with the size of the box side fixed at 90 units. Robot diameters of 20 and 10 were compared for a task in which a box was moved 200 units from its initial position. Each data point is the average of 25 simulation runs each with a different random initial configuration.

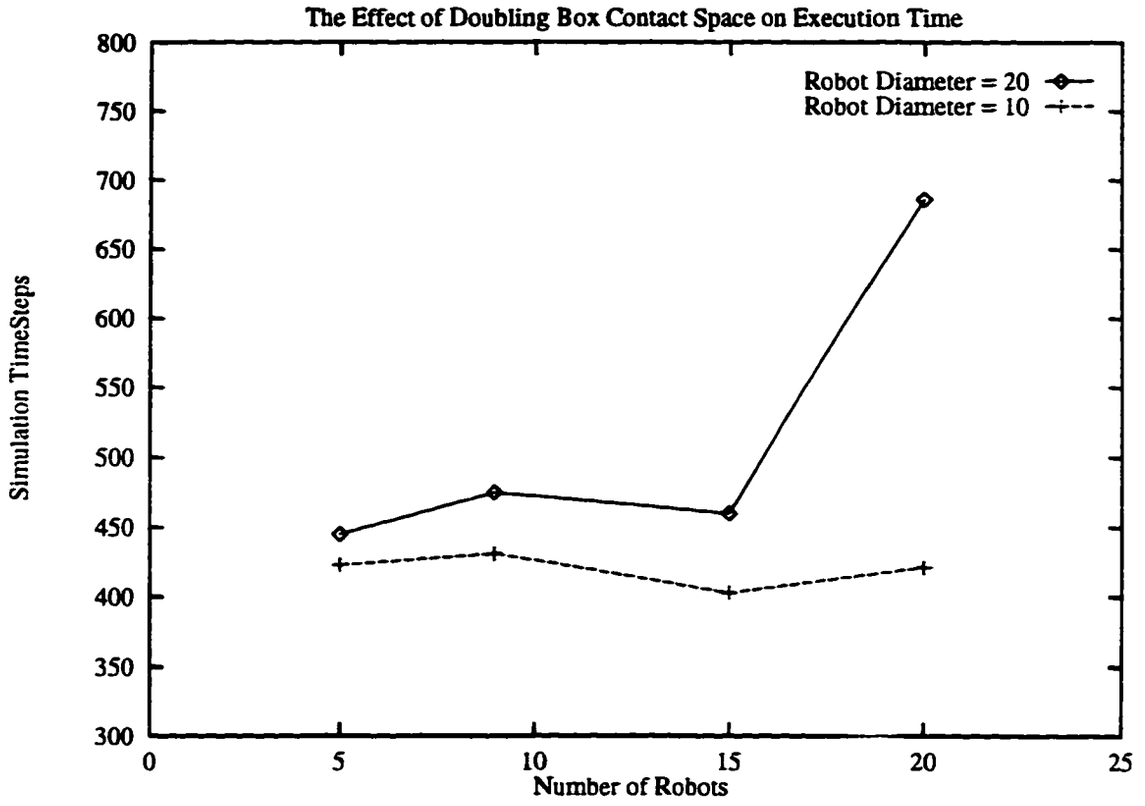


Figure 6.12: The effects of doubling box contact space on execution time. The results from two simulation experiments showing execution time versus system size. The only parameter varied was the size of the robot; the size of the box side was held constant at 90 units.

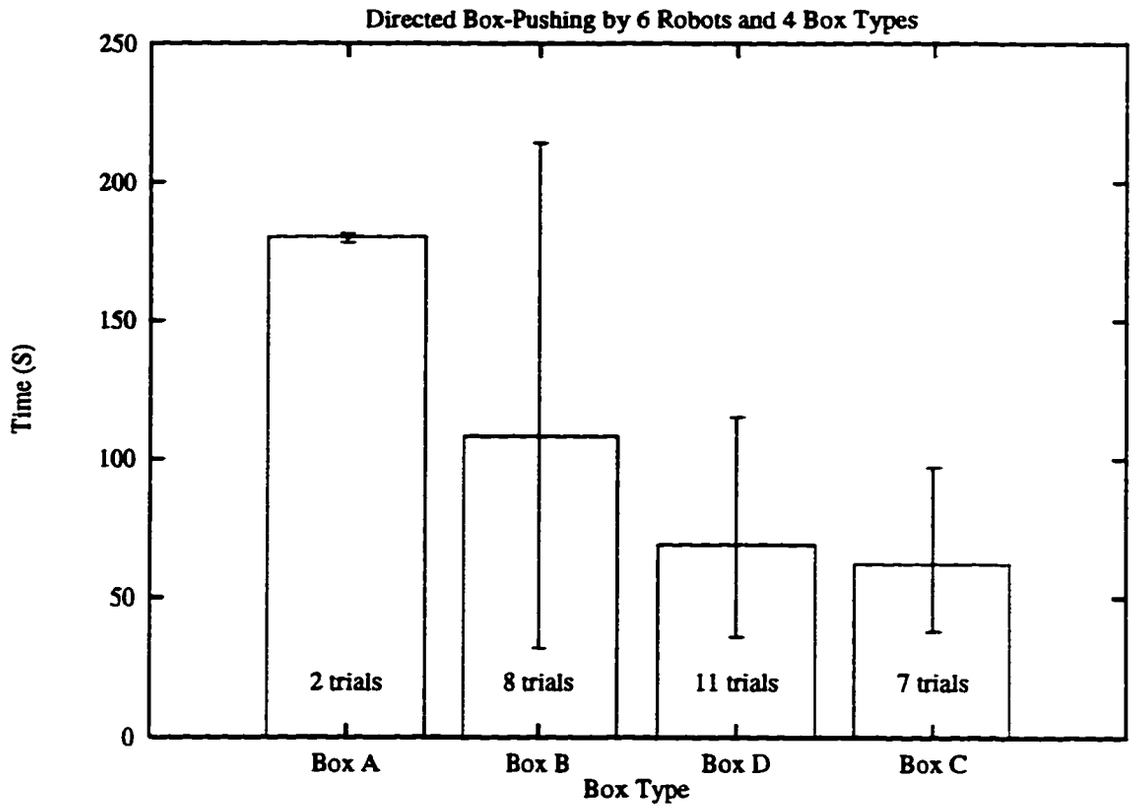


Figure 6.13: The mean execution time of moving a box towards the goal as a function of box type. Box A is a 42 centimeter square box, Box B and C are 84 centimeter square boxes with B having a higher sliding friction than Box C, Box D is an 84 centimeter diameter round box. All box types are approximately the same weight and can be pushed by at least two robots. For each plot the number of trials as well as the minimum and maximum run times are indicated. All trials used six robots. Robot interference is minimized by increasing the available contact space around the box.

the goal; however, there is ample room for improved performance once the robot parameters are fine tuned. For the box-pushing transport task the decentralized approach to control is insensitive to system size, some types of convex object geometry and changes in the goal positions. Statistical conclusions can be better supported with more experiments.

Chapter 7

Discussion: From Social Insects to Collective Robots

Visualize a room in which a group of robots sit in one corner and a large box sits approximately center with a spotlight placed in another corner. The robots begin moving and soon disperse into the room. Soon after the box-light comes on the robots begin moving towards it and eventually come into contact with a side. Then imagine some of the robots beginning to reposition themselves by moving around the box in a counterclockwise fashion, while others which are correctly positioned begin to push the box towards the spotlight. The box begins to move in the direction of the lit corner of the room, but the path is not quite straight and veers to the right and the box movement stops. Again some of the robots begin their counterclockwise repositioning and assume a new position more correctly oriented for pushing. Finally, the box begins to move in a new direction towards the goal-light.

Once the box reaches the goal position the spotlight turns off and a second goal light on the opposite corner of the room is illuminated. Now all the robots begin repositioning, eventually making it to the opposite side and begin to once again push the box towards the new goal destination. Robots leave the task, seemingly at random, and wander off only to return and join the group effort in transporting the box towards its goal. The experiments are repeated, this time with boxes of different shapes and sizes and the number of robots in a group are varied. Our video recordings shows, and those that have seen them agree, that the robots make a coordinated effort in pushing the box in a direction that converges towards the indicated goal position.

7.1 Coherent Behaviour without Explicit Cooperation

The results show in the many successful trials of directed box-pushing that *a coordinated group effort is possible without use of direct communication or robot differentiation.* Rather a form of indirect communication takes place through the environment by way of the object being manipulated. For directed box-pushing, the control strategy was shown to be insensitive to system size, some convex object geometries and changing goal positions. The results of experiments with physical robots presented here, adds support to Arkin's simulation studies which showed that cooperation in some tasks are possible without direct communication [4].

In Arkin's study, the task of retrieving objects in the environment was decomposed into three subtasks analogous to the transport task presented here. Although in Arkin's study robot differentiation was used, the results presented here are complimentary in their support of the hypothesis that cooperation in some tasks is possible without communication. The present study expands on the previous simulation work by indirectly considering the dynamics involved in box manipulation.

The data presented in this study also agrees in certain aspects with other studies in which stigmergy is used as the task coordinating mechanism. Stigmergy as proposed by Grassé is a model used to explain the regulation of building behaviour in termites [26]. Stigmergy theory holds that transitions between a sequence of construction steps is regulated by the effect of previous steps. In more general terms, the theory has been used to explain and describe the process by which task activity can be regulated using only local perception and indirect communication through the environment as applied to algorithms for coordinating distributed building behaviour [70] and foraging tasks by multi-robot systems [11]. In the box-pushing task the results support the use of indirect communication through the environment as proposed by stigmergy theory. However, Downing and Jeanne found that stigmergy theory does not explain the use of additional cues, not dependent on previous steps, in regulating task execution in nest construction by paper wasps [19]. For collective robotics this means that perceptual cues can also be formed from stimuli other than that which are immediately available from the task itself. For example, in directed box-pushing the box-detection cues are adaptive to the ambient light level of the environment by specifying box-detection as a multiple of the ambient light level.

Stigmergy theory also does not account for the multiple cues proposed in this study for creating transition cues based on the use of orthogonal sensing and described in Chap-

ter 4. The use of multiple cues for controlling transitions between the subtask controllers is supported by studies on wasp nest construction in which a processing hierarchy reduces the number of cues that need to be evaluated at the same time [20]. In a similar manner perceptual cues, used to specify transitions between subtask controllers is a hierarchical method of evaluating an action control decision. Hence, stigmergy theory would have to be expanded to include both additional and multiple cues which may adapt to the environment as proposed in [19].

Coherent behaviour from a collective system of robots must also account for task resource management. Coordination improves by minimizing antagonistic actions that can result from conflicts over limited resources. In box-pushing antagonistic forces are mitigated by increasing the available boxside space while enforcing a noninterference behaviour. The data on transporting small boxes versus large boxes by the same number of robots confirms the observations made during task execution. For box-pushing, this result implies that group size is important for a fixed resource size in a given task and agrees with the result obtained by Beckers *et al.* [11] for a foraging task in which one to five robots were used to gather 81 objects randomly distributed in their environment then placing them into one large pile. Their study showed that group size was a critical factor in determining task efficiency and that increasing the number of robots used without increasing the available task resources increased task execution time due to the increase in inter-robot interference. In general, increasing task resources minimizes inter-robot interference. Thus, reducing robot interference increases group coordination and consequently leads to a more efficient coherence as demonstrated by the decreasing execution times.

The coherent behaviour displayed for the transport task can also be attributed to the common goal shared by the individual robots along with an identical set of interaction rules. This is the same effect noted by Seeley while considering the collective decision making in honey bees [61]. As an explanation for how a swarm of honey bees could reach the same decision on the profitability of several food sources, Seeley hypothesized that each bee's nervous system was calibrated in a similar manner. Since all members of the colony share the same rules for adjusting response thresholds, the bees can operate independently yet generate a collective response to various nectar sources. Thus common goals and common rules of interaction allow a decentralized decision making process to produce a coherent global response.

7.2 Research Contribution

The central thesis in this research was that coherent behaviour in some tasks, namely collective box-pushing, does not require explicit mechanisms of cooperation. Rather, a decentralized system of asynchronous machines could perform a multi-step task in a coordinated fashion, given a common set of operational rules (Q-machines) and a decision process (perceptual cues) that depends on local information only. This synergistic approach was demonstrated by way of a system of multiple mobile robots and a common collective task. The contribution is the evidence in support of the hypothesis that some coherent behaviour does not require explicit mechanisms of cooperation. During the course of investigation a simple task-programming architecture, called *Q-machines*, was developed which included a novel framework, called *perceptual cues*, that offers a new approach to environment-specific task modelling in collective robotics.

The methodology of modelling tasks as Q-machines, is similar to the well established theory of sequential machines, and in multi-robot systems allows for tasks to be described as a sequence of steps and control behaviours designed to accomplish each step. Given that the environment in which the system functions can be controlled, the methodology results in a deterministic system behaviour. The modelling of the task to be accomplished as a hierarchy of state machines allows for alternate control mechanisms to be employed at each level. This modularity allows for nonreactive control techniques to be used along with reactive ones. Most existing systems integrate task-level knowledge with tool-level knowledge in a way that does not permit this separation during implementation.

The design approach advocated here is analogous in some ways with task specific solutions by social insects. In both the robot and insect cases, a solution must consist of two parts: the environment and the agent with its environmentally tuned sensing systems. The solution is therefore environment-specific as well as being task-specific. This is congruent with the idea that task-specific robotics will prevail over the more general multiple task do-everything-for-you robotics typically presented by the media and held by some roboticists.

The second contribution lies in the use of the kinesthetically-driven stagnation recovery behaviours. Task progression depends on the ability of the system to automatically solve problems relating to system deadlock or stagnation. In Q-machines, stagnation recovery behaviours make use of kinesthetic orientation to solve the local minimum problem of deadlock in reactive control. The method specifies the stagnating condition in terms of a locally sensed stimulus. The resulting action used to break the deadlock condition makes use of

either fixed action or random motion patterns like those found by Sudd and used by ants during prey transport [66, 67].

This dissertation also complements the existing research studies by examining another of the three typical multi-robot tasks. Foraging, a task in which objects are retrieved, can be accomplished using a single robot. Multi-robot studies have shown that this task can also be accomplished without centralized control or explicit communication, but that some limited forms of direct communication can improve execution times [42, 3]. Formation marching, a task in which robots move in a fixed pattern, has only been studied in simulation except for very simple two robot cases. Box-pushing, the task studied here, has previously only been considered in the two robot case with robots assuming unique left/right functional roles in task execution. Both direct communication [58, 43] and indirect communication [18] using two robots has been studied. Simulation studies for the transport task which utilized decentralized control have also been presented in the literature [65]. For the box-pushing task presented here, data has been collected for over 100 trials with a physical system of robots making this study the largest exploration to date of this task domain.

A novel framework is used to specify transitions in sequential subtask controllers. Transitions are specified using locally sensed information. By using both state information and perceptual cues, tasks that require sequential execution are possible. The use of action-oriented perception, also referred to as selective perception, further demonstrates how a specific stimulus in the environment can be used in the action selection process as also argued by Horswill in [29]. The approach presented in this dissertation for transition control is closely analogous to "trigger events" used in the temporal coordination of perceptual algorithms, called finite state acceptors, proposed by Arkin and MacKenzie [6]. The difference lies in how the transition cues are specified. In the proposed framework, cues are specified using orthogonal local sensing only, whereas in finite state acceptors transitions occur from a variety of conditions including elapsed time, algorithm completion, algorithm failure, or termination of a motor activity. The taxis-based model used for action is somewhat analogous with Agah and Bekey's *Tropism System Cognitive Architecture* in which robot actions are based on likes and dislikes [1]. Their system allows simulated colonies of robots to learn relationships between task performance and perception, success and failure.

Finally, also gathered was empirical evidence that supports the notion of a task density function which relates the number of robots to the available task resources. Like group transport behaviour where ants can carry prey in excess of the sum of individual pieces by distributing their efforts, the mass effect of many robots on the box-pushing task coupled

with simple obstacle avoidance rules allow resources to be distributed. The resulting distribution of robots along the box perimeter coordinates the individual pushing actions towards the common goal position. The results obtained in this investigation will contribute to the body of knowledge in the relatively young field of collective robotics.

7.3 Further Study

Three areas for further exploration are: system reliability, learning and perceptual expansion. First, the approach to controller design although procedural is not automatic and of interest for further study is a technique for comparing different control designs for the same task. Reliability theory is one possible tool for such a comparison. Second, in order for systems to be truly autonomous they must learn to adapt. Two possible avenues for exploration are the adaptation of the perceptual cues and the primitive actuation behaviours. Finally, the current system made use of the minimal number of sensors thought needed for the task; however, still to be explored is the relationship between system performance and the amount of information provided by the perception system.

7.3.1 System Reliability

The experimental objectives are not to demonstrate optimal, but rather feasible solutions to collective tasks. In doing so, the aim is to provide a means of making relative comparisons among competing solutions highlighting the variables that effect system reliability. When a task is decomposed into an aggregation of behavioural actions performed by many redundant simple robots, how can comparisons between alternative decompositions in terms of their reliability be made? Herbers [27] has shown, using reliability theory, how a large system of redundant behaviour sequences in ant colonies can increase system reliability in the foraging task. Five foraging strategies used by ants to find food were compared by modelling each strategy's system structure as the probability that an individual behavioural act is performed correctly. Thus the probability of food being returned to the ant's nest can be calculated and compared for each strategy. In doing so, Herbers was able to show that the foraging strategy of group hunting was more likely to be successful in retrieving food when compared to the strategy of foraging alone. This was due, in spite of the fact that the probability of performing an individual behavioural act correctly was low, to the highly redundant series-parallel foraging strategy employed in group hunting. In a similar manner, can this approach be mapped to a redundant set of robots and used to compare alternate box-pushing task decompositions?

7.3.2 Learning to Adapt

Also of interest, for future study, is the manner in which a perceptual cue or primitive motion can change over time or be adaptive to changes in the environment and task requirements. Entomological studies of nonlinear sequences of behaviours suggest that cues used to regulate the latter stages of nest construction vary in response to a changing environment [20]. In a manner similar to the way the box detection cue was varied to adapt to the ambient light found in the robot's environment, the thresholds used in a perceptual cue could be adjusted based on a reinforcement signal, as could the currently fixed motion primitives. For example, both obstacle avoidance cues and pushing behaviours could be adapted over time as follows:

- *Learning to See.* Currently two fixed threshold values are used in obstacle avoidance. If a robot could vary the threshold based on the number of collisions over a given time, then the avoidance behaviour would learn to adjust the threshold to an optimal value for a given environment. And if the environment changed over time then the threshold values would adjust. To test this idea, the robot would need an array of contact sensors to provide the negative feedback for each motor action. An initial threshold value for obstacle avoidance would allow the robots to detect obstacles within a fixed distance of the robot. If a movement resulted in a collision, as detected by the contact sensors, then the threshold would be adjust downward. This would have the effect of varying the detection field around the robot until an optimal collision free value is found.

- *Learning to Act.* Pushing behaviour currently makes use of a fixed motion primitive the result of which is a simple decision on whether to push or not to push. However, a feedback stimulus is available which would allow a robot to assess the effect of each discrete pushing action. That stimulus is the intensity of the goal indicator light. A successful push from the robot's point of view, is one that brings the box closer to the goal as indicated by an increase in signal intensity from the goal direction sensor. If the angle of pushing (currently perpendicular to the box) was adjustable and made a function of the position of the signal peak within the field of view of ?SEE-GOAL, then it might be possible to learn optimal pushing angles.

7.3.3 Perceptual Expansion

While viewing the video taped experiments, one immediately notices that robots often leave the box and seemingly wander off. Since box detection is limited to two forward pointing

light sensors, repositioning behaviour often moves the robot so that its view of the box is lost. If the number of sensors were increased so that box detection became omnidirectional then one would assume that the robots would spend less time searching for the box. Consequently, the time taken to transport the box to the goal should decrease thereby increasing the system performance. This argument could also be made for the obstacle avoidance sensors, which currently are also two forward pointing sensors. Although Shannon's information theory states that ambiguity decreases as the amount of information increases it is not clear how an increase in the amount of sensing information would translate to an increase in system performance resulting in "better" control decisions.

7.4 Epilogue

By way of the social insects, nature is showing us how to build decentralized and distributed systems that are autonomous and capable of accomplishing tasks through the interaction of many simple and highly redundant agents. From their local perception to the mass effect that results in a global action these biological systems serve to elucidate the mechanisms thought to be at the heart of self-organizing behaviour. With their rich source of inspiring examples, social insects serve as nature's proof that solutions to complex tasks in collective robotics may in fact be found underfoot.

Bibliography

[1] Arvin Agah and George Bekey. Learning by trying in an autonomous multi-robot system. In *submitted to IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995.

[2] Karl Altenburg. Adaptive resource allocation for a multiple mobile robot system using communication. CSOR TR-9404, North Dakota State University, 1994.

[3] Karl R. Altenburg. Cocobots: Robots, ants, and randomness. Master's thesis, North Dakota State University, 1995.

[4] Ronald C. Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.

[5] Ronald C. Arkin. Survivable robotic systems: Reactive and homeostatic control. In M. Jamshida and P. Eicker, editors, *Robotics and remote systems for hazardous environments*, volume 1, chapter 7, pages 135–154. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[6] Ronald C. Arkin and Douglas MacKenzie. Temporal coordination of perceptual algorithms for a mobile robot navigation. *IEEE Transactions on Robotics and Automation*, 10(3):276–286, June 1994.

[7] Hajime Asama, Maki K. Habib, Isao Endo, Koichi Ozaki, Akihiro Matsumoto, and Yoshiki Ishida. Functional distribution among multiple mobile robots in an autonomous and decentralized robot system. In *1991 IEEE International Conference on Robotics and Automation*, pages 1921–1926, 1991.

[8] Shuji Asami. Robots in Japan: Present and future. *IEEE Robotics and Automation Magazine*, pages 22–26, June 1994.

[9] G.P. Baerends and J.P. Kruijt. Stimulus selection. In R.A. Hinde and J. Stevenson-Hinde, editors, *Constraints on Learning: Limitations and Predispositions*. Academic Press, 1973.

[10] Tucker Balch and Ronald C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1:1–25, 1994.

[11] R. Beckers, O.E. Holland, and J.L. Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. In *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems Artificial Life IV*, pages 181–189, 1994.

[12] Alan H. Bond and Les Gasser. An analysis of problems and research in distributed artificial intelligence. In *Readings in Distributed Artificial Intelligence*, pages 3–35. 1988.

[13] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology.* MIT Press, 1984.

[14] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, March 1986.

[15] Philippe Caloud, Wonyun Choi, Jean-Claude Latombe, Claude Le Pape, and Mark Yim. Indoor automation with many mobile robots. In *1990 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 67–72, 1990.

[16] Y. Uny Cao, Alex S. Fukunaga, Andrew B. Kahng, and Frank Meng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997. to appear.

[17] Russell J. Clark, Ronald C. Arkin, and Ashwin Ram. Learning momentum: On-line performance enhancement for reactive systems. In *1992 IEEE International Conference on Robotics and Automation*, pages 111–116, 1992.

[18] Bruce Donald, James Jennings, and Daniela Rus. Analyzing teams of cooperating mobile robots. In *1994 IEEE International Conference on Robotics and Automation*, pages 1896–1903, 1994.

[19] H. A. Downing and R. L. Jeanne. Nest construction by the paper wasp, Polistes: a test of stigmergy theory. *Animal Behaviour*, 36:1729–1739, 1988.

[20] H. A. Downing and R. L. Jeanne. The regulation of complex building behaviour in the paper wasp, *polistes fuscatus* (insecta, hymenoptera, vespidae). *Animal Behaviour*, 39:105–124, 1990.

[21] Joseph F. Engelberger. *Robotics in Service.* Kogan Page Ltd., 1989.

[22] H. R. Everett. *Sensors for Mobile Robots: Theory and Application.* A. K. Peters, Ltd., 1995.

[23] N. R. Franks, A. Wilby, B. W. Silverman, and C. Tofts. Self-organizing nest contruction in ants: sophisticated building by blind bulldozing. *Animal Behaviour*, 44:357–375, 1992.

[24] Nigel R. Franks. Teams in social insects: Group retrieval of prey by army ants. *Behavioral Ecology and Sociobiology*, 18:425–429, 1986.

[25] Norman E. Gary. Pheromones of the honey bee, apis mellifera l. In David Wood, Robert Silverstein, and Minoro Nakajima, editors, *Control of Insect Behavior by Natural Products*, pages 29–53. Academic Press, 1970.

[26] P. Grassé. La reconstruction du nid et les coordinations interindividuelles chez *bellicositermes natalensis et cubitermes* sp. la théorie de la stigmergie: essai d'interpré. *Insectes Sociaux*, 6:41–81, 1959.

[27] J. M. Herbers. Reliability theory and foraging by ants. *Journal of Theoretical Biology*, 89(1):175–189, 1981.

[28] Bert Hölldobler and Edward O. Wilson. *Journey to the Ants: A Story of Scientific Exploration*. Belknap Press of Hardvard University Press, 1994.

[29] Ian D. Horswill. *Specialization of Perceptual Processes*. PhD thesis, Massachusetts Institute of Technology, 1993.

[30] Rudolf Jander. Insect orientation. *Annual Review of Entomology*, 8:95–114, 1963.

[31] C. Ronald Kube. Collective robotic intelligence: A control theory for robot populations. Master's thesis, University of Alberta, 1992.

[32] C. Ronald Kube. A minimal infrared obstacle detection scheme. *The Robotics Practitioner*, 2(2):15–20, 1996.

[33] C. Ronald Kube and Hong Zhang. Collective robotic intelligence. In *Second International Conference on Simulation of Adaptive Behavior*, pages 460–468, December 7-11 1992.

[34] C. Ronald Kube and Hong Zhang. Collective robotics: From social insects to robots. *Adaptive Behavior*, 2(2):189–219, 1993.

[35] C. Ronald Kube and Hong Zhang. Stagnation recovery behaviours for collective robotics. In *1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 3, pages 1883–1890, 1994.

[36] C. Ronald Kube and Hong Zhang. The use of perceptual cues in multi-robot box-pushing. *Robotica special issue on Interacting Robots*, 1995. submitted October.

[37] C. Ronald Kube and Hong Zhang. The use of perceptual cues in multi-robot box-pushing. In *1996 IEEE International Conference on Robotics and Automation*, pages 2085–2090, 1996.

[38] C. Ronald Kube and Hong Zhang. Task modelling in collective robotics. *Autonomous Robots*, 4(1):53–72, 1997.

[39] C. Ronald Kube, Hong Zhang, and Xiaohuan Wang. Controlling collective tasks with an ALN. In *1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 289–293, 1993.

[40] H. Martin. Zur Nahorientierung der iene im Duftfeld Zugleich ein Nachweis für die Osmotropotaxis bei Insekten. *Zeitschrift für Vergleichende Physiologie*, 48(5):481–533, 1964.

[41] Maja J. Matarić. Designing emergent behaviors: From local interactions to collective intelligence. In J. A. Meyer, H. Roitblat, and S. Wilson, editors, *Second International Conference on Simulation of Adaptive Behavior*, pages 432–441. MIT Press, 1992.

[42] Maja J. Matarić. *Interaction and Intelligent Behavior*. PhD thesis, Massachusetts institute of technology, 1994.

[43] Maja J. Matarić, Martin Nilsson, and Kristian T. Simsarian. Cooperative multi-robot box-pushing. In *1995 IEEE International Conference on Intelligent Robots and Systems*, volume 3, pages 556–561, 1995.

[44] David McFarland and Thomas Bosser. *Intelligent Behavior in Animals and Robots.* MIT Press, 1993.

[45] Merriam-Webster Inc. *Webster's Ninth New Collegiate Dictionary*, 1985.

[46] Hans A. J. Metz. State space models for animal behaviour. *Annals of Systems Research*, 6:65–109, 1977.

[47] Micromachine Center (MMC). *Micromachine*, 3 edition, July 1993.

[48] M. W. Moffett. Cooperative food transport by an asiatic ant. *National Geographic Research*, 4(3):386–394, 1988.

[49] Hans Moravec. *Robot Rover Visual Navigation.* UMI Research Press, 1981.

[50] J. C. Moser, R. C. Brownlee, and R. Silverstein. Alarm pheromones of the ant atta texana. *Journal of Insect Physiology*, 14:529–535, 1968.

[51] J.C. Moser. Pheromones of social insects. In D. Wood, R. Silverstein, and M. Nakajima, editors, *Control of Insect Behavior by Natural Products*, pages 161–178. Academic Press, 1970.

[52] N. Nilsson. Shakey the robot. A.I. Center Technical Note 323, SRI International, 1984.

[53] Fabrice R. Noreils. An architecture for cooperative and autonomous mobile robots. In *1992 IEEE International Conference on Robotics and Automation*, pages 2703–2710, 1992.

[54] Fabrice R. Noreils. Multi-robot coordination for battlefield strategies. In *1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1777–1784, 1992.

[55] Jun Ota and Tamio Arai. Motion planning of multiple mobile robots using dynamic groups. In *1993 IEEE International Conference on Robotics and Automation*, pages 28–33, 1993. Vol. 2.

[56] Koichi Ozaki, Hajime Asama, Yoshiki Ishida, Akihiro Matsumoto, Kazutaka Yokota, Hayato Kaetsu, and Isao Endo. Synchronized motion by multiple mobile robots using communication. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1164–1169, 1993.

[57] Lynne E. Parker. Designing control laws for cooperative agent teams. In *1993 IEEE International Conference on Robotics and Automation*, pages 582–587, 1993. Vol. 3.

[58] Lynne E. Parker. ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In *1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, pages 776–783, September 1994.

[59] Lynne E. Parker. *Heterogeneous Multi-Robot Cooperation.* PhD thesis, Massachusetts Institute of Technology, 1994.

[60] B. Schricker. Die Orientierung der Honigbiene in der Dammerung zugleich ein Beitrag zur Frage der Ocellenfunktion bei Bienen. *Zeitschrift fur Vergleichende Physiologie.* 49:420–458, 1965.

[61] Thomas D. Seeley, Scott Camazine, and James Sneyd. Collective decision-making in honey bees: how colonies choose among nectar sources. *Behavioral Ecology and Sociobiology*, 28:277–290, 1991.

[62] A. Seitz. Die Paarbildung bei einigen Cichliden I. *Zeitschrift für Tierpsychologie*, 4:40–84, 1940.

[63] Takanori Shibata and Toshio Fukuda. Coordinative behavior by genetic algorithm and fuzzy in evolutionary multi-agent system. In *1993 IEEE International Conference on Robotics and Automation*, pages 760–765, 1993. Vol. 1.

[64] Richard Silby and David McFarland. A state-space approach to motivation. In D. J. McFarland, editor, *Motivational Control Systems Analysis*, pages 213–250. Academic Press, 1974.

[65] Daniel J. Stilwell and John S. Bay. Toward the development of a material transport system using swarms of ant-like robots. In *1993 IEEE International Conference on Robotics and Automation*, pages 766–771, 1993. Vol. 1.

[66] J. H. Sudd. The transport of prey by an ant, *pheidole crassindoa. Behaviour*, 16(3-4):295–308, 1960.

[67] J. H. Sudd. The transport of prey by ants. *Behaviour*, 25(3-4):234–271, 1965.

[68] John Sudd. How insects work in groups. *Discovery*, pages 15–19, June 1963.

[69] Texas Instruments Inc. *The Optoelectronics Data Book for Design Engineers*, fifth edition, 1978.

[70] Guy Theraulaz and Eric Bonabeau. Coordination in distributed building. *Science*, 269(4):686–688, 1995.

[71] Marc R. Tremblay and Mark R. Cutkosky. Using sensor fusion and contextual information to perform event dete ction during a phase-based manipulation task. In *1995 IEEE International Conference on Intelligent Robots and Systems*, pages 262–267, 1995.

[72] D. M. Vowles. The orientation of ants I, the substitution of stimuli. *Journal of Experimental Biology*, 3(31):341–355, 1954.

[73] P. K. C. Wang. Navigation strategies for multiple autonomous mobile robots moving in formation. *Journal of Robotic Systems*, 8(2):177–195, 1991.

[74] R. Wehner and M.V. Srinivasan. The world as the insect sees it. In T. Lewis, editor, *Insect Communications*, pages 29–47. Academic Press, 1984.

[75] Rudiger Wehner. Matched filters - neural models of the external world. *Journal of Comparative Physiology A*, 161:511–531, 1987.

[76] E.O. Wilson. *The Insect Societies.* The Belkap Press of Harvard University Press, 1971.

[77] E.O. Wilson, N.I. Durlach, and L.M. Roth. Chemical releasers of necrophoric behavior in ants. *Psyche*, 65(4):108–114, 1958.

[78] E.O. Wilson and B. Hölldobler. *The Ants*. The Belkap Press of Harvard University Press, 1990.

[79] Stewart W. Wilson. The animat path to AI. In *First International Conference on Simulation of Adaptive Behavior*, pages 15–21. MIT Press, 1990.

[80] J. Zeil, G. Nalbach, and H.O. Nalbach. Eyes, eye stalks and the visual world of semi-terrestial crabs. *Journal of Comparative Physiology A*, 159:801–811, 1986.

Appendix A

Collective Robotics Hardware

A unique modular system for building mobile robots was developed during the course of the described research. The system was also used to support an undergraduate course in Mobile Robotics taught within the Department of Computing Science and has been successfully licensed by the University of Alberta to a commercial partner. Shown in Figure A.1 is the system block diagram of a box-pushing robot.

Electronic Control Modules

In order to simplify the robot's electronics, control functions have been packaged into modules and fabricated on printed circuit boards. Each robot control system is built from an assortment of modules from the following list:

- **68HC11 Microcontroller.** This single board computer from New Micros Inc. is based on the popular Motorola 68HC11 with the addition of an embedded Forth operating system/language/compiler. The board is complete with RS232 communications, expandable RAM and the UA-ROM BIOS/Utilities/Tools extension.
- **DC Power Regulator.** Converts battery voltage to regulated +5 VDC and provides several +5 VDC connections to other modules.
- **DC Power Interconnect.** A DC power expansion module with power indicator LED, 11 +5 VDC and 3 +Battery connections, and expansion connector. The module provides a convenient source of regulated voltage.
- **Motor Control.** Provides a digital interface to 2 DC motors (50V, 3A max). Each motor has a direction and enable control line allowing for Pulse Width Modulation (PWM) speed control. Each motor has forward/reverse direction indicator LEDs.

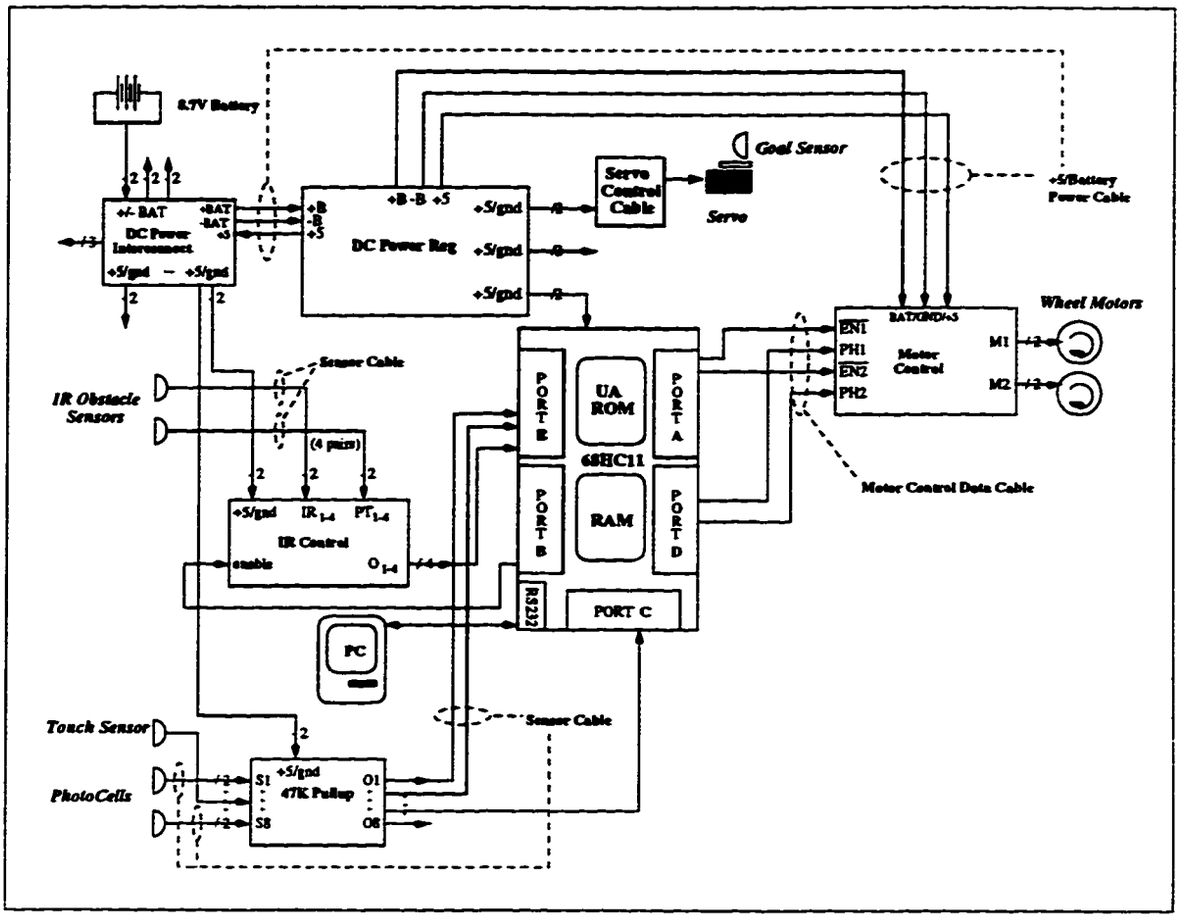


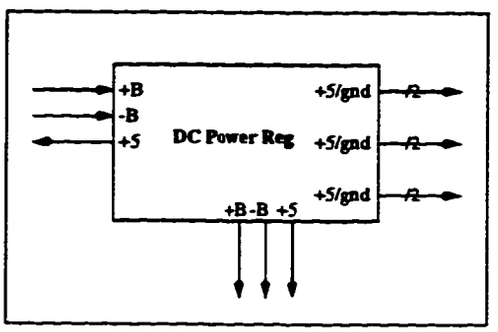
Figure A.1: A system block diagram of a box-pushing robot illustrating the use of the electronic modules.

- **Infrared Control.** Provides on/off control of 4 infrared emitter/detector pairs used in obstacle detection.
- **Sensor Pull-Up Resistors.** A 47K-Ohm pullup resistor module allows for 8, 2-wire, sensor connections.

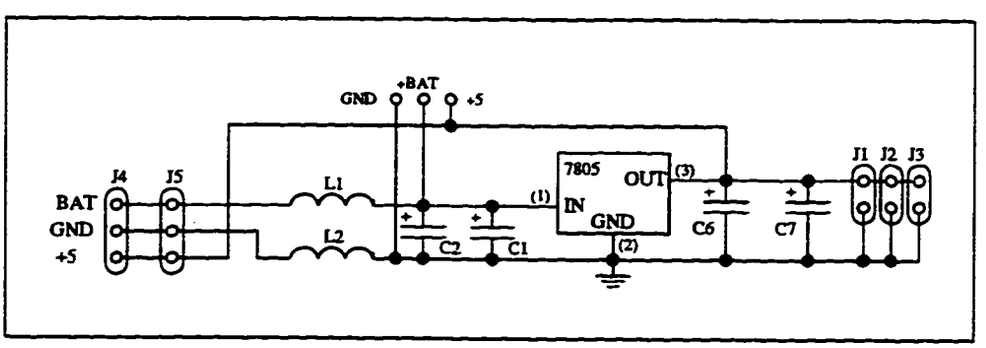
DC Power Regulator

Functional Description

The DC Power Regulator converts input battery voltage (+/-B) to regulated +5 VDC output. The battery voltage must be $\geq$ 7.5 VDC, with a one ampere output current available.



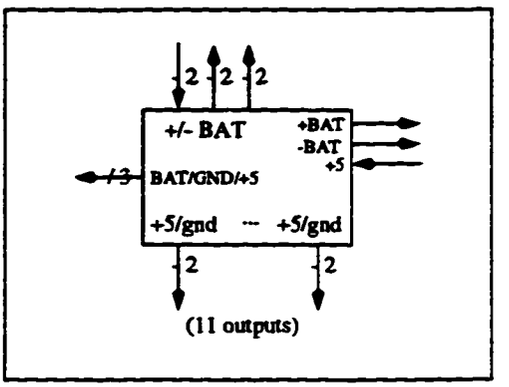
Electrical Description



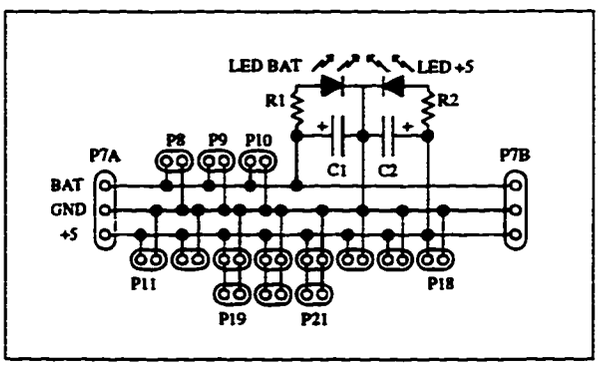
DC Power Interconnect

Functional Description

A DC power expansion module with power indicator LED, 11 +5 VDC and 3 +Battery connections, and expansion connector. The module provides a convenient source of regulated voltage.



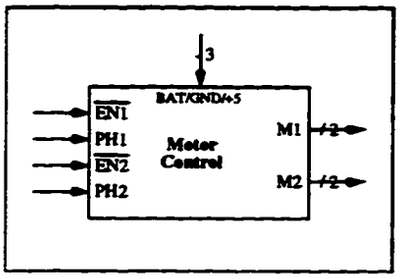
Electrical Description



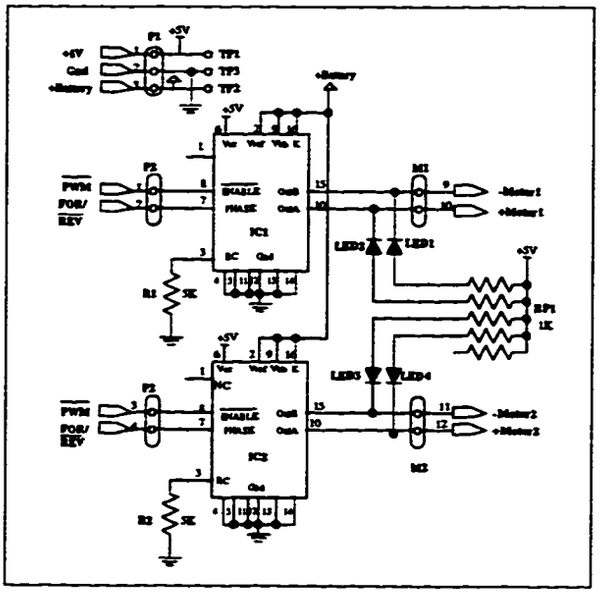
Motor Control

Functional Description

Provides a digital interface to 2 DC motors (50V, 3A max). Each motor has a direction and enable control line allowing for Pulse Width Modulation (PWM) speed control. Each motor has forward/reverse direction indicator LEDs.



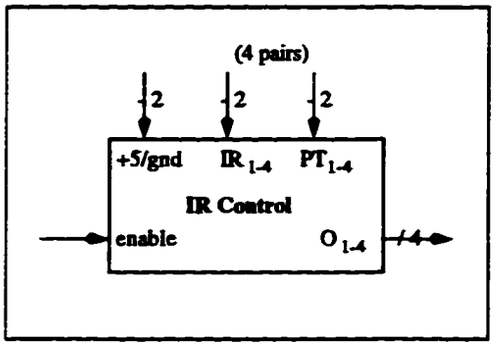
Electrical Description



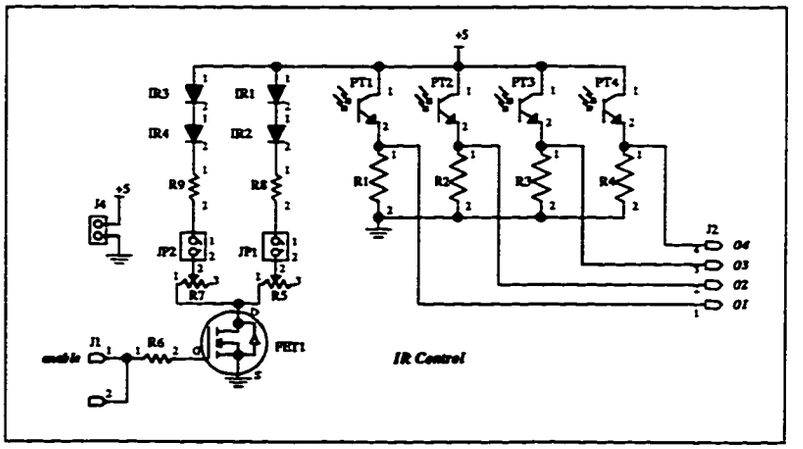
IR Control

Functional Description

Provides on/off control of 4 infrared emitter/detector pairs used in obstacle detection.



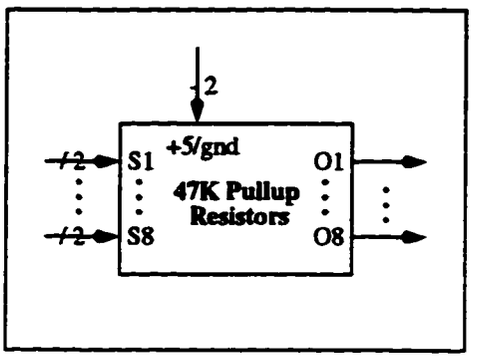
Electrical Description



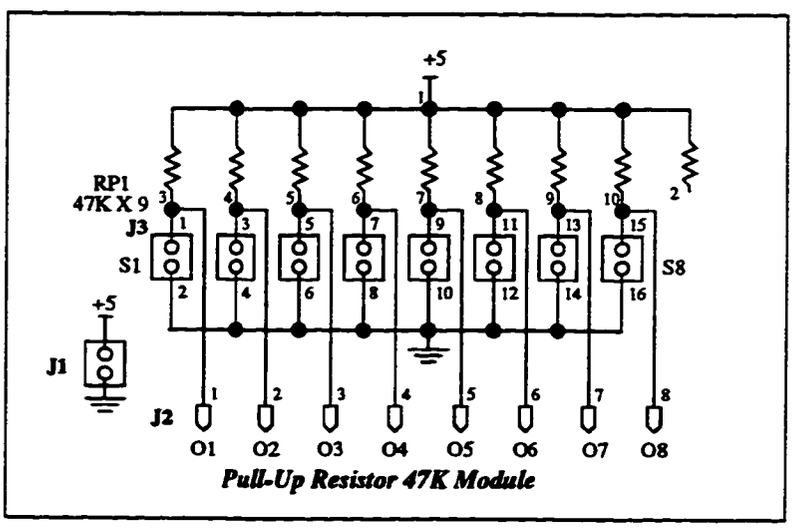
Sensor Pullup Resistors

Functional Description

A 47K-Ohm pullup resistor module allows for 8, 2-wire, sensor connections.



Electrical Description



Pull-Up Resistor 47K Module