A COOPERATIVE OBJECT TRANSPORT SYSTEM WITH BEHAVIOR-BASED ROBOTS

A Thesis

by

ARUN PRASSANTH RAMASWAMY BALASUBRAMANIAN

BE, Anna University - India, 2012

Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Texas A&M University-Corpus Christi
Corpus Christi, Texas

August 2018

August 2018

A COOPERATIVE OBJECT TRANSPORT SYSTEM WITH BEHAVIOR-BASED ROBOTS

A Thesis

by

ARUN PRASSANTH RAMASWAMY BALASUBRAMANIAN

This thesis meets the standards for scope and quality of
Texas A&M University-Corpus Christi and is hereby approved.

Scott King, PhD
Chair

Luis Garcia Carillo, PhD
Co-Chair

Ajay Katangur, PhD
Committee Member

August 2018

ABSTRACT

Cooperative object transport is an intriguing research area in swarm and multi-agent robotic systems. Global-view is a challenge in cooperative transport where different aspects such as providing the global picture, what information to share and how to share are still being explored. One simple way of addressing global view is by using a situated agent which has an elevated view of the environment and capable of communicating it. Various works that employ this strategy often rely on a centralized or global control where the agent with the global view makes the decisions. We propose a strategy modeled after Behavior-Based Robotics principles which enables the robots to react to the environment and thereby achieve cooperation to accomplish the task. Instead of relying on a sophisticated controller or a centralized leader, the robots simply react to the stimuli in different ways by executing simple behaviors from their own repertoire. The 'Observer', a situated agent with the global view has no decision-making responsibilities and simply serves as a means of stimulus. Also, the Observer employs simple techniques to extract and share very limited information with other agents. Different experiments were performed with real robots and various metrics were collected to demonstrate and evaluate the strategy.

ACKNOWLEDGMENTS

I would like to thank my chair Dr.Scott King and my co-chair Dr.Luis Garcia for guiding me and for being generous throughout the course of the research which I greatly appreciate. I would like to thank my committee, Dr.Ajay Katangur for the efforts he expended. I am deeply indebted to Texas A&M University-Corpus Christi for this great learning opportunity, for the people I met and for the fond memories.

A special thanks to my work supervisor, Joshua Green for being considerate and flexible. Thank you Daniella, for spending meticulous hours reviewing and fixing the grammar. Thanks to all of the IT Learning-Space staff for the motivation they extended. All of my beloved friends who have been kind and supportive in various ways: Charan, Vinay Datta, Karthik, Yeshwant, Chandu, President Ravi, Neela, Kusa, Harika, Ankit, Preethi, Sameera, Babloo, Mayuri, Sahana, Kuladeep, Sadis and, Evan: Thank you.

My brother Sree Ram, my parents Balasubramanian and Maheswari: thank you for the profound love and support, I owe you all more than ever. Finally, I thank you for being there for me when needed. Your perpetual love made this possible, holding everything in place. Thank you Anitha, my other half.

TABLE OF CONTENTS

CONTENTS PAGE

LIST OF TABLES

TABLES PAGE

LIST OF FIGURES

FIGURES PAGE

CHAPTER 1

INTRODUCTION

Box-pushing is an interesting research area where the basic goal is to move a box-like object to a predefined location using a robot. If the box is too heavy or too difficult to handle for a single robot, having multiple robots would be advantageous. Reduced time-consumption, increased resilience and maneuverability are some of the other advantages of employing a multi-agent system for this task. Thus, the box-pushing problem evolves into the Cooperative or Collective object transport problem and is relevant in various real-life applications such as warehouse stocking, mining, construction, agriculture and disaster management. The following sections of this chapter summarizes the motivation behind the research, describes the problem and lists contributions.

1.1 MOTIVATION

Using multiple robots for the object transport task brings various advantages. However, employing multiple specialized robots for the task undermines these advantages since using multiple sophisticated robots is not cost-effective. Thus, less sophisticated robots are often used which brings in other challenges: simple robots often have limited sensing and computing. These limitations inflict other side effects such as low computational power, lack of localization techniques and diminishes the overall capabilities of the robots. Also, in multi-agent systems, there are various other associated complexities such as coordination and consensus mechanisms, communication amongst the agents, sharing the global view. Global information is crucial if the robots are expected to be autonomous. Various aspects of sharing global information

such as 'What needs to be shared?', 'How to convey and how often?' are still being explored. This research aims at addressing these open challenges in a multi-agent collective transport system consisting of less-sophisticated robots by taking advantage of the Behavior-Based Robotics approach. The system explores a strategy where the robots 'sense' the global information as a 'stimulus' and react to it demonstrating the possibility of increased autonomy with relatively less computational cost.

1.2 PROBLEM DESCRIPTION

The basic objective of the cooperative transport problem is to move an object to a predefined location with a team of robots working together. The object is an elongated rectangular box which is large relative to the size of the robots and the robots move it from an initial location to a designated goal location by pushing it. Considering that the robots have located the object which is between them and the goal, now they have to work cooperatively to push the object towards the goal. Since the box is bigger than the robots, it occludes the robots' vision and they would not be able to perceive the goal. There is a situated agent i.e. the Observer, which has an elevated view of the environment and is capable of detecting both the object and the goal and thus can assist the robots in the task. Though the Observer can communicate with the agents and assist them, it has no control over the robots or their decision making. The robots will have to work cooperatively and move the box to the goal with the information form the observer. The problem is discussed in detail in Chapter 4: System Design.

1.3 CONTRIBUTION

The following are the contributions of this thesis:

- A Behavior-Based Robotics' system for the cooperative object transport task where simple behaviors are employed in place of sophisticated controllers to realize the objective. The strategy provides a mechanism for the robots to perceive the global information as a stimulus and react to it thereby increasing their autonomous capabilities.

- Capturing and sharing global information involves low computation as the Observer does not do any planning or decision making. The Observer simply assists the robots by generating the necessary stimulus rather than serving as the centralized controller.

1.4 STRUCTURE OF THE THESIS

The organization of the thesis is as follows. Chapter 2 provides background information and summarizes variously related literature, discusses the common challenges, and concludes with a discussion on Behavior-Based Robotics. Chapter 3 describes the hardware and software elements where Chapter 4 discusses the System Design and Implementation in detail. The experimental setup is discussed in Chapter 5 and Chapter 6 presents and analyzes the experiments and their results.

CHAPTER 2

RELATED LITERATURE

This section begins with background information on Swarm/Multi-agent based cooperative transport systems before moving to collective transport related literature and Behavior-Based robotics.

2.1 BACKGROUND

The problem of moving an object with multiple robots has been referred throughout the literature with various names. Cooperative/coordinated transport, collective transport, box pushing, object placement are some of the names used. Due to the diverse nature, categorizing the approaches is tricky since there is no concrete way of classifying the approaches. Tuci et.al. [1] classify the various approaches based on the method employed by the robots for handling the object. They classified the literature as Push-only, Grasping and Caging based cooperative transport systems. If classified based on this approach, the proposed system would be a Push-only cooperative object transport approach. Though this type of coarse grain classification is simple and straightforward, the classification becomes fragmented and confusing because of the diversity and overlap in strategies such as decision making and communication.

Another way of classifying the various collective transport approaches was shown by Rubenstein et.al. [2]. The approaches were vaguely categorized into two different genres based on the underlying control/ decision-making mechanism. The first genre comprises of approaches involving centralized decision-making approaches which mainly focus on developing strategies for planning, controlling and coordinating the robots. Centralized planners, centralized leader-based models, and ap-

proaches employing centralized decision-making mechanism would fall under this category. The second genre is the Swarm/collective robotics approach which aims at achieving the goal through highly decentralized coordination strategies similar to that of insect swarms. Typically, these systems do not have centralized decision making and rely on simple robots capable of demonstrating emergence. If classified based on these genres, the proposed system would be a decentralized strategy where there is no centralized controller.

Rubenstein et.al. [2] describe that Swarm robotics research has taken a different approach to the box-pushing problem. Swarm robotics is inspired by various social insects such as ants and termites. Ants show remarkable abilities to transport bigger and heavier objects. An ant can easily carry small prey whereas a large prey requires a coordinated effort. Various species of ants show a variety of impressive cooperative transportation abilities. In collective transport by ants, the physical factors of the object like the shape, orientation, and size do not matter to ants. They even achieve the objective without any direct communication amongst themselves. These phenomenal abilities of ants are extremely appealing to the box-pushing problem and thus inspired researchers to borrow various strategies from ants. Notably, the outlook of swarm robotics and social insects like ants show remarkable similarities in various aspects [3] and thus borrowing ideas from ants would be logically sound. The Swarm/collective robotics approach to the box-pushing problem aims at achieving the goal through highly decentralized coordination strategies similar to that of insect swarms. Another intriguing aspect of swarm robotics is that each robot is insignificant as an individual. They lack sophistication in most hardware aspects like computing power, sensors, effectors, communication, and battery. In fact, miniaturization and simplicity of individual members are of high emphasis in swarm robotics.

Also, each of these robots has relatively simple programs that usually consists of some basic behaviors which would be invoked based on certain sensor readings. However, these simple behaviors could result in a large set of complex swarm behaviors which is denoted as emergent behavior [4, 5, 6]. Emergent behavior in swarm robotics is analogous to those behaviors observed in nature such as birds flocking and fish schools avoiding a predator. In other words, swarm-intelligence strives to achieve meaningful behavior at swarm-level and the behavior at an individual level might be trivial. Another advantage that numbers give swarm is high fault tolerance [3, 7]. Swarm robots are homogeneous in most scenarios and thus any agent can replace or take the role of another as there are no specific pre-assigned roles [3]. This also makes dynamic and decentralized activities possible. Adaptability and faster transport are other advantages.

2.2 COOPERATIVE TRANSPORT

Before moving to different multi-agent-based concepts and corresponding approaches, this section begins listing some of the earliest box-pushing research work. The collective transport problem has been addressed by a multitude of approaches and this section attempts to cover the breadth of it and summarize them. Though the literature discussed in this section is classified as decentralized or swarm-robotics oriented, it is relatively easy to notice that one approach could differ from another drastically.

Most of the earlier approaches considered using either a centralized approach to coordinate the robots or considered using sophisticated robots with various capabilities. Also, due to the cost involved most of them were verified based on computer simulation [2]. One of the earliest physical demonstrations of collective transport by simple autonomous robots was done by Mataric et.al. [7]. They used the hexapod

robots named Genghis which took turns to push an elongated rectangular box. A notable point here is that just one of the robots was capable of generating enough force to move the box and yet the authors used two robots. The objective here was to evaluate whether using two robots for the same task would be faster than using one. There are various notable works that were inspired by social ants [2, 8, 9]. C. Kube and H. Zhang made detailed studies on collective prey retrieval in ants and published multiple works in the 90s addressing various aspects of collective transport by robots [8, 10, 11, 12, 13]. The literature titled "Cooperative transport by ants and robots" by C. Kube et al. [8] is one such work which was very closely modeled after prey retrieval in ants. This work established that the dynamics of ants and swarm robotic systems are very similar and thus a collective behavior resembling ants can be achieved by implementing ant-like behaviors on individual robots. They also demonstrated that coordinated effort among robots could be achieved even without any direct communication. Since there was no communication, all the robots had to sense the object and the goal. Even after finding the object, a robot had to move away from the box periodically to detect the direction of the goal to continue pushing the box successfully. Another interesting consequence was that all the robots tried to push the box at the same time even though only one of them could have pushed the box.

Ants show an amazing ability of collectively transporting complex, irregularly shaped objects without any prior knowledge of the object. Rubenstein et al. [2] demonstrated that this can be successfully achieved in collective robot systems. They investigated a simple, decentralized strategy for collective transport where all the agents act independently without any explicit coordination. They proved their strategy successful through a physics-based theoretical model. With this model, the

authors were able to predict various important outcomes such as speed of transport and the minimum number of agents required to move the object. They were also able to verify the object trajectory and establish a relation between the scalability of performance and the number of robots. They demonstrated their work successfully using the tiny Kilobot [14]. The Kilobot robots were designed to study flocking and emergent behavior in swarm robotics. They use some unconventional methods such as reflecting infrared light off the surface for communication and vibrating motors for locomotion. Kilobots use phototaxis (determine heading by sensing ambient light) to determine the goal, thus there was no need for any explicit communication.

One of the basic criteria for a robotic swarm is to solve the problem and achieve their goals through the use of numerous simple robots capable of some sensing and communication amongst themselves [3]. Communication is a critical area in swarm robotics which is often overlooked. A little communication amongst the agents would greatly increase performance, while too much communication would consume resources and undermine performance [6, 15]. Ants communicate in interesting ways, like stigmergy and touch. The concept of stigmergy is intriguing: it is the means of indirect communication through the environment using pheromone trails, or through the object which is being carried by the ants. One of the earliest approaches to adopt stigmergy for foraging was done by Beckers et.al. [16]. Stigmergic communication in a swarm of robots would be beneficial for reducing communication overheads and simplifying implementation etc. Many attempts have been made to emulate the indirect communication of ants in robots through chemical traces [17], peer-to-peer messaging virtual pheromones [18], and phosphorescent light trails.

Another swarm intelligence concept which is often studied in swarm robotics is flocking. Flocking can candidly be described as the behavior where an agent

aligns its heading by observing the heading of its neighbors. If all the agents change their heading in such a way, over iterations, the whole swarm would have an aligned heading. This type of consensus mechanism is useful in certain situations of the box pushing problem where not all agents can sense the object or goal. When only a few of the agents can observe the goal, the overall heading can be estimated relatively if the agents can communicate among themselves. Campo et al. demonstrated this in [19]. The same was demonstrated by Rubenstein et al. [2] to validate their theoretical model. The r-one robots [20] were used for this experiment. Another unconventional implementation of cooperative transport was demonstrated by J. Chen et al. [21]. In this approach, an agent continuously attempts to occlude itself from the goal (emulated as a light source) by maintaining the object between itself and the goal. In other words, the robots re-locate dynamically around the object such that they are always behind the objects. This strategy does not use any communication among the agents and is suited for bigger objects which could occlude the agent's view. Certain species of ants, like the common weaver ants, form pulling chains to retrieve the prey. This typically happens when more force must be applied to move the prey and the area is smaller. The ants which cannot access the prey thus grasp the other ants which have already attached themselves to the prey, thereby forming a pulling chain [8]. This strategy is highly useful in certain scenarios and has inspired various attempts at building modular or self-assembling robots. One such collective transport strategy with self-assembling robots was successfully demonstrated by M. Dorigo et al. [9]. This approach considered robots that could both push and pull. However, this was realized using the highly-sophisticated s-bots [18] which have a huge array of sensors including 15 IR proximity sensors, 8 light sensors, 8 multi-color LED, omni-directional camera, torque sensors, and touch sensors.

2.2.1 COMMON LIMITATIONS

This section begins by explaining occlusion and discusses some of the general challenges in multi-agent robotics and collective transport. Since there are numerous challenges, explanations are saved for those that are more related to this research while the other challenges are listed.

2.2.1.1 OCCLUSION

Occlusion which was introduced in 1.1 is a common challenge in cooperative transport with multiple agents where some or all of the agents working on the object might not be able to observe the goal. This could happen when the object is bigger than the robot or when the object is very close to the agent and thus occludes it's view. Only a handful of literature considers the issue of occlusion and address them while many others ignore occlusion and concentrate only on the cooperative task.

An interesting approach of tackling the occlusion problem was demonstrated by Gerkey et.al. [22] where out of the three robots, one would take the role of the 'watcher' and the other two would be the 'pushers'. Their roles were selected by an auction method called MURDOCH [23]. The pushers are occluded and cannot sense the goal, as such they must rely on the watcher. The watcher is equipped with a scanning laser range-finder that can precisely locate the goal and the object and, mathematically calculate the position and orientation of the box with respect to its own position. The observer then gives specific commands such as who has to push (left or right), what should be the velocity, and the pushers execute the commands. This approach is tightly-coupled since the objective of the authors is to implement a fault-tolerant cooperative system.

The system proposed by Wang et.al. [24] is a heterogeneous system where

three different types of agents are used. Figure 1 shows the system overview where other than the two physical robots, there are four software agents in the system. There is a vision agent that has a global view of the environment and is capable of generating positions and orientation coordinates of all robots, the object, and the obstacles. There is an evolutionary learning agent responsible for generating cooperation plans based on Genetic algorithm and Reinforcement learning approaches. The two physical robots execute the plan generated by the learning agents. The system is tightly coupled, and all the software agents run on sophisticated computers and are connected via Ethernet. Other than software simulations, one type of physical experiment is presented in the article where one of the robotic agents is a stationary robotic arm.

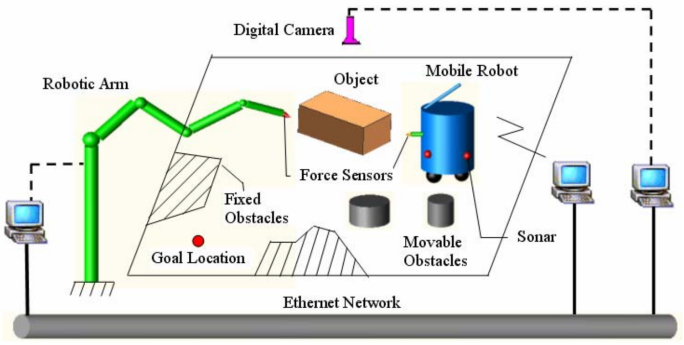


Figure 1. System demonstrated by Wang et.al.[24].

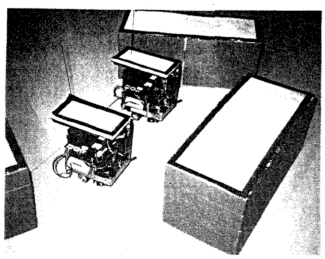


Figure 2. Object placement problem by Sugie et. al.[25].

Sugie et.al.[25] addresses a variation of the problem which is referred to as the 'object placement problem'. It is a special case of cooperative transport where the robots work together to move multiple objects to one of the goal locations i.e. any object can be placed at any goal location. The camera placed in the arena is capable of tracking all the objects, robots and goal locations as shown in Figure 2. The task planner generates plans for the robots. Though the robots do not communicate

explicitly, they can infer each other's intention. This is achieved by applying constraints such as the robots having synchronized watches. Also, all the robots use the same decision-making rules. If there is no uncertainty in the environment, and all the robots decide behavior at the same time, there would be no contradiction amongst the robots. Even if conflicts arise eventually, the robots execute one of another set of rules to resolve the conflict.

Hichri et.al. [26] has a centralized control algorithm in which an external server globally communicates with the robots. The study was done in a simulation where a set of homogeneous group of robots were equipped with manipulators for grasping and lifting an object in order to place the object on top of their bodies. This strategy requires a prior knowledge of the number of robots in the group, object's shape, mass and center of gravity. The server communicates position information to the robots to approach the object, to lift it, and to carry it to a destination. A similar approach where an external server coordinates the robots' actions is described by Wang et.al. [27] and Yamashita et.al. [28].

Rubenstein et.al. [2] work around the issue of occlusion through communication since their approach is swarm robotics oriented and an agent capable of communicating with its neighbors. In the work my Mataric et.al. [7], there is no occlusion since the pyroelectric sensors on the Genghis II robots are taller than the object and are capable of sensing the goal, a pair of tungsten lamps, throughout the course of the experiment. In the work by Kube et.al. [8], the object was a brightly lit box and the goal was a spotlight facing the floor. The robots were equipped with a narrow field-of-view sensor to detect the light. This light sensor was capable of sweeping up and down in an upward pointing arc with a help of a motor. If a signal peak occurs while facing up, it was caused by the spotlight. Even with this flexible design, the

robots would often lose sight of the goal and switch to a different behavior where it would move away and reposition so that the goal could be sensed again. The work by Machado et.al. [29] has a leader equipped with an omni-directional camera to generate trajectories whereas Pereira et.al. [30] implemented a design where all robots were equipped with omni-directional cameras and they directly shared information to complement each other's observation and create a global view of the object and its orientation. Even Tuci et.al.[9] employ omni-directional cameras, however, there is no problem of occlusion since the object is of the same height as the grippers on the robots.

2.2.1.2 OTHER LIMITATIONS

Though swarm robotics is inspired by simple insects, achieving performance levels closer to these insects is still a great challenge. One of the major challenges is a limitation of the resources such as computation, sensing, power, etc. Most of the swarm robots employ simple micro-controllers with a few bytes of memory. Kube et. al. [8] used a robot with a Motorola 68HC11 microcontroller with 8KB of RAM, whereas the Kilobots employ an Atmega328 microprocessor clocked at 8 Mhz with a mere 32K of memory hosting both the bootloader and the program [14]. This limitation in capabilities is sometimes intentional since swarm robotics desires miniaturization and strives to achieve meaningful behavior at the swarm level using simple robots. Localization is another challenge as most of the swarm robots are designed for indoor environments and they do not equip devices such as GPS. Additionally, the alternatives like WiFi localization or localization through environmental and perceptual cues are computationally expensive and less accurate. Though numbers in swarm robotics are one of its key advantages, it brings in its own share of complications.

Individual agents need to be reprogrammed and recharged every time which is demanding. Though there are novel strategies such as programming a group of robots using infrared light [14], these techniques are available only on sophisticated robots which are expensive.

2.3 BEHAVIOR-BASED ROBOTICS

Behavior-based robotics differs from traditional artificial intelligence by using biological systems as an inspiration. Typically, classical AI uses a set of steps to solve problems, where internal representations of events and world models play a role. Contrastingly, the behavior-based approach relies on adaptability rather than using preset calculations to handle a situation [4, 5]. Figure 4 compares the flow of information in Traditional AI and behavior-based approach. Traditional AI approaches typically follow the SENSE, MODEL, PLAN, and ACT approach. Whereas in a behavior-based approach, the robots simply react to their environment rather than build world models and plan.

Behavior-based robotics consider robots as an organism with some embedded cognition that can react to certain stimuli. Stimulus in a biological sense is defined as an event or something that evokes a specific functional reaction in an organ or tissue or in the organism itself. The stimulus may evoke a response in an organism but evoking a response is not the reason why the stimulus exists.

Much of the work in the school of behavior-based robots was done in the 1980s by Rodney Brooks and colleagues at the Artificial Intelligence Laboratory in Massachusetts Institute of Technology [31]. As shown in Figure 3, a series of wheeled and legged robots such as 'Shakey' and 'Genghis' were built based on Brook's subsumption architecture [32]. Because of the AI winter of the 70s and 80s, and due to

the processor impoverishment, they developed relatively low cost and fully functional robots with the principles of behavior-based robotics. Another important milestone in behavior-based robotics is the book named "Vehicles – Experiments in Synthetic Psychology" by Valentino Braitenberg [33]. Braitenberg, through a series of thought experiments, described how intelligent artificial creatures could be constructed incrementally starting from very simple ones those that can result in complex-appearing behaviors such as fear and love.

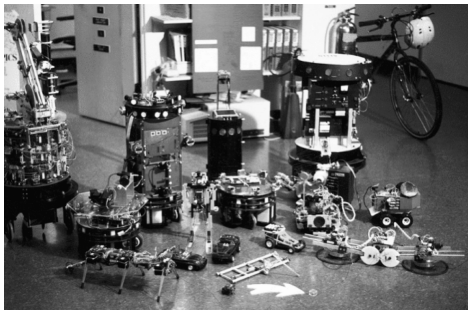


Figure 3. Behavior-based robots, Artificial Intelligence Lab, MIT.

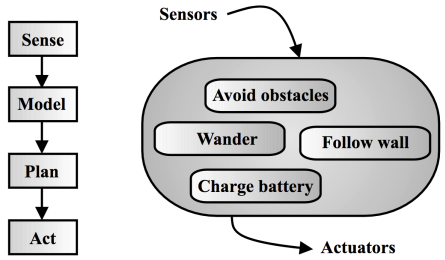


Figure 4. Classical AI and Behavior-Based robotics approach[5].

The methodology involved in developing a behavior for a behavior-based system as described by Arkin RC [4] is shown in Figure 6. This approach to designing behaviors is attractive in the perspective of software engineering principles due to the modular way of implantation or "programming by behavior" approach [6]. This also inherently supports good software engineering practices, especially low coupling, and high cohesion. While earlier behavior-based/reactive robots often relied only on SENSE-ACT primitives, they are not applicable universally. For example, avoiding an obstacle by reading a proximity sensor on a robot can be implemented in a reactive way, however making that robot reach a goal location on a terrain cannot be done completely reactive since navigation is a deliberative task. Thus, most of the modern

behavior-based robots employ a SENSE-PLAN-ACT strategy where they tend to have a certain hybrid architecture where the deliberation and reaction coexist.

Roomba, the vacuuming robot [34] by iRobot is one of the examples that demonstrates the principles and applications of behavior-based robotics. Roomba does not employ any complex navigation or path planning strategies [35]. It employs simple behaviors such as 'straight' and 'spiral', where the Roomba robot goes straight or in incremental spirals until it hits an obstacle. When it bumps into an obstacle, the 'bump' behavior makes it turn and go straight to a different direction. There is also a 'follow wall' behavior and 'backoff' behavior to avoid falling down stairs and so on. The Roomba executes these behaviors in any order and eventually gets the room cleaned.

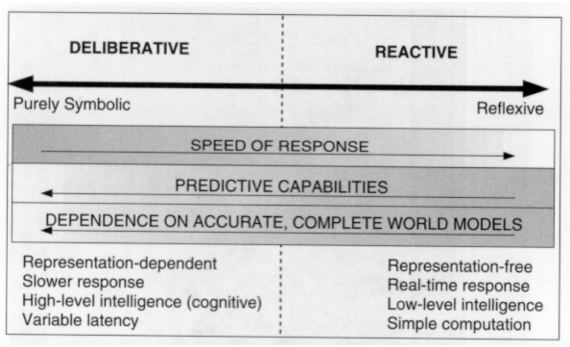


Figure 5. Deliberative and Reactive approaches presented as a sliding scale [4]The more reactive a system tends to be, lesser the representation and computation.

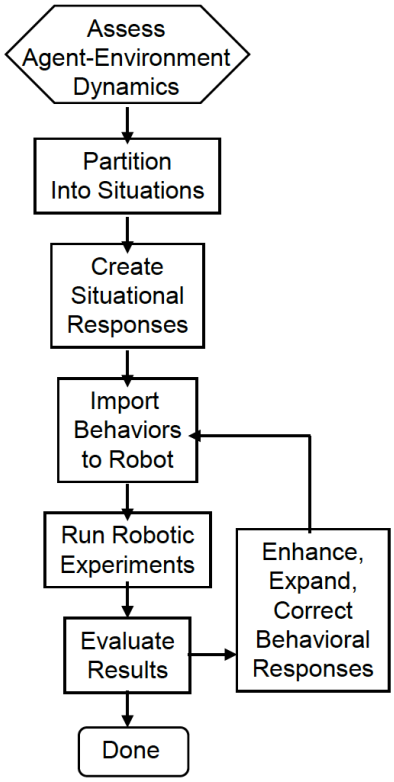


Figure 6. Steps for developing behaviors for a behavior-based robot, Arkin RC, 1998[4].

CHAPTER 3

HARDWARE AND SOFTWARE

Figure 7 summarizes the hardware platform and software while they are discussed in detail in this chapter.

Robot Platform	Khepera IV
Language	C++
Libraries	Libkhepera 2.1, OpenCV 3.3
IDE	Eclipse C++ (Mars 2)
Toolchain	poky-glibc-i686-khepera4-image-cortexa8hf-vfp-neon-toolchain-1.8
Development Platform	Alienware Area 51 PC with Intel core i7 processor and 32 GB RAM, Ubuntu 14.4 OS
Observer	Logitech C920 web-cam connected to the Development PC via USB

Figure 7. Hardware and Software.

3.1 KHEPERA IV

For the experiments, Khepera IV robots [36] were used. Khepera IV shown in Figure 8 is a robot for indoor use (table, lab floor) with various state of the art technologies. It is equipped with an 800MHz ARM Cortex-A8 Processor and 512 megabytes of RAM and an RGB color camera with a maximum resolution of 752X480 pixels. The robot includes an array of 8 Infrared Sensors for obstacle detection and 4 more for fall avoidance or line following. There are 5 Ultrasonic Sensors for long-range object detection they are also equipped with a gyroscope and an accelerometer. Wi-Fi and Bluetooth are available for communication and Khepera IV is differential-wheel drive based and the internal battery provides a running time of about 4-5 hours. Khepera IV is running a full, standard embedded Linux Operating System i.e. Yocto Linux which provides several benefits. Also, a library 'libkhepera' is available and provides

an interface with all the robot peripherals. It should be noted that though Khepera IV has a wide range of sensors, only the RGB camera, and 3 Infra-Red proximity sensors are used in the current implementation. The RGB camera is used at a resolution of 192x144 to detect the object and the 3 IR proximity sensors (Front, Front-Left, and Front-Right) are used for aligning the robots for the pushing task.

Figure 8. Khepera IV [36].

3.2 LANGUAGE AND LIBRARIES

3.2.1 DEVELOPMENT ENVIRONMENT

The implementation was done in C++ with the Eclipse Mars IDE. C++ was chosen Since the 'libkhepera' library is available only in C/C++. Also, the image processing library OpenCV 3.3 is available in C++ and thus the implementation was done in C++. The development was done on a generic desktop computer running Ubuntu 14.4. The Eclipse IDE was set up for the cross-compilation such that the resultant executable would be compatible with the ARM processor on the Khepera IV. There was no need to cross compile the Observer program since it runs on the desktop PC

as a standalone C++ application and it uses the OpenCV 3.3 library for its image processing functions.

3.2.2 LIBRARIES

The 'libkhepera' library is currently in version 2.1 and it is the latest. It is capable of granting control over the robot hardware at a fairly low level in most of the instances. It allows the user to configure devices, read sensor data and send commands. Some functions in the library return primitive data types, while others such as the camera and IR data retrieval functions still output data in unstructured buffers.

As mentioned above, the OpenCV 3.3 library was used for the image processing on both the Khepera IV and the Observer. Though 'libkhepera' provides some image processing functions, it is primitive and limited. Hence, the decision to use the OpenCV library was made to save development time.

3.3 CHALLENGES AND LIMITATIONS

Though Khepera IV is a capable robot in a small package, it is not without any limitations. One of the major challenges came in the form of a limitation in the physical design of Khepera IV. Khepera is an indoor robot with a ground-to-top height of 5.77 cm (wheels included). Khepera was designed to have a minimum ground clearance of 4 mm to increase the stability and is advised to use only on hard and flat surfaces. However, considering the other design factors such as the castor wheels, their solution results in 0.5-1 mm of ground clearance, making the robot very stable but preventing its use on any surface that is not effectively flat and smooth [37]. This restricted ground clearance has a great negative impact, making the robot unfit for anything but flat surfaces as claimed in [37]. This proved to be true our test

environment: the lab floor was not even throughout and had various bumps and tiny pits which would often thwart the free movement of the Khepera IV. Thus, most of the times the Khepera is stuck when it encounters such a bump on the floor. The wheels with the thin O-rings that act as tires does not help much in such a situation as they do not provide enough traction and just slip.

From the implementation point of view, there were other minor limitations resulting from the 'libkhepera' library. Though the 'libkhepera' library is capable of granting fine-grained control over the robot hardware, the lack of higher-level constructs often forces the users to write repetitive code and re-implement frequently used functionality [37] and thus consumes significant time. Also, the image processing functions provided by 'libkhepera' were primitive, and hence the decision was made to use the OpenCV library. However, installing OpenCV on the Khepera 4 robot was not feasible since the robots did not have sufficient storage. Thus, it was decided to cross-compile the OpenCV library with the recommended toolchain $poky - glibc - i686 - khepera4 - image - cortexa8hf - vfp - neon - toolchain - 1.8$ and load only the required shared object files (.so)to the Khepera IV.

CHAPTER 4

SYSTEM DESIGN

This section gives the overview of the system design and then breaks down the system components into its basic behaviors in the subsequent sections.

4.1 PROBLEM OVERVIEW

The task addressed here is cooperatively moving a rectangular box which is large relative to the size of the robots from an initial location to a designated goal location by pushing it. Considering that the robots have located the object which is between them and the goal, now they have to work cooperatively to push the object towards the goal. Since the box is bigger, it occludes the robots' vision and they would not be able to perceive the goal. There is a situated agent i.e. the Observer, which is having elevated view of the environment and capable of detecting both the object and the goal can assist the robots in the task, however, he has no control over the robots or their decision making. Figure 9 shows an overview of the system.

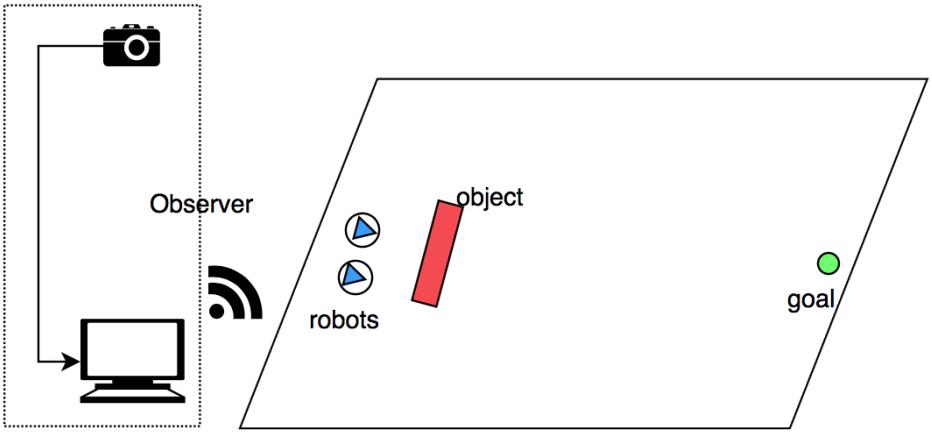


Figure 9. System Overview.

4.2 PROPOSED STRATEGY

Since the Observer has a global view, a straightforward solution would be to take advantage of it to accomplish the task. The previous works that employ such strategies where the agents make use of the global information available to another agent are discussed in Section 2.1.1 in detail. However, there is a recurrent pattern in all of these approaches to be noted: the agent (also referred as host or server in the literature) which is capable of capturing the global view takes control of the cooperative task. The agent with the global view either does all the processing and gives explicit commands to the robots [22, 24, 26, 27, 28] or it has to keep track of all the objects, robots and the goal position(s) [24, 25]. This requires computationally expensive tracking, planning and prediction methods and a global localization strategy which might suffer when scaled.

We propose a strategy where the Observer does not have to do extensive computing or tracking or control the other agents but can still assist them by supplying vital information. The system takes inspiration from Behavior-Based Robotics where during the design process the robots are considered as a reactive organism which responds to various external stimuli [4, 5].

Approaching the collective transport problem from a Behavior-Based robotics perspective, one could realize that the problem of occlusion can be explained as a lack of stimulus. The stimulus that denotes the goal cannot be directly perceived in the environment since the sensory organ i.e. the camera of the robot(s) is occluded by the object. Thus, if a different stimulus that can mimic the deprived stimuli is perceived by the agents, they can react to it. This is what the Observer is designed to do: it provides an external stimulus to the agents who cannot perceive the goal. As discussed in Section 2.3, it should be noted that the input being provided should

be simple to be considered as a stimulus.

The proposed strategy is summarized below while the individual behaviors are discussed in the following subsection. The Observer captures the scene and employs basic color image processing technique such as color blob detection to identify the objects and the goal. The Observer extracts the centers (2D Cartesian coordinate) of the object and the goal after identifying them. These points include two coordinate points representing the extremities of the object and one coordinate point representing the goal. These points are essentially just pixels on the image that the Observer captured and does not mean anything in the real world considering the fact that the images are warped due to the position of the Observer. The Observer can be located anywhere in the environment in a corner or an edge (so that it could cover a larger area) which warps the image. No translation or additional processing is done on the three 2D Cartesian points and are simply stored in a shared memory. The shared memory is overwritten each time an image frame is captured, and the points are extracted.

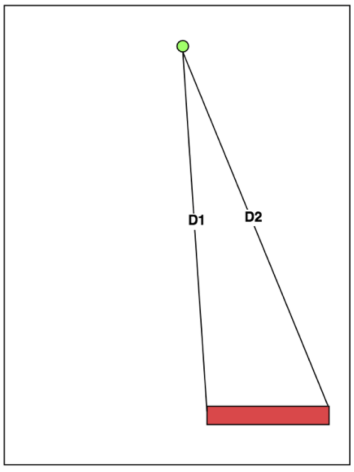


Figure 10. Sample environment where the green circle is the goal and the red box is the object. D1 and D2 denotes the distances between the goal and the object.

The robots rely on their own sensors (the camera) and the global information from the Observer to locate and approach the object. Once a robot is sufficiently close to the object, it uses three of its IR proximity sensors (Front, Front-Left, and Front-Right) to reposition itself to push the box. Now the robot is in a situation where it has no information on how to proceed since the box occludes its camera. At this stage, the robot reads the coordinates from the shared memory associated with the Observer. However, the coordinates are in the reference frame of the observer and the robot do not know how it would be in their own reference frame. The robot interprets the points by calculating Euclidean distance between the goal and the two points representing the object. Figure 10 shows a sample environment where the green circle is the goal and the red box is the object. The distances between the goal and the extremities of the box are denoted as $D1$ and $D2$. Any following decision made by the robot is based on the absolute difference between the two distances. This absolute difference between the distances would be denoted as Δ . If the absolute difference between the distances is less than a threshold $\Delta\tau$, it means that the object's current alignment is good enough and the robots can work together to push the box towards the goal. If Δ is greater than $\Delta\tau$, it means that the object is out of alignment and needs to be oriented. Figure 11 demonstrates these two scenarios.

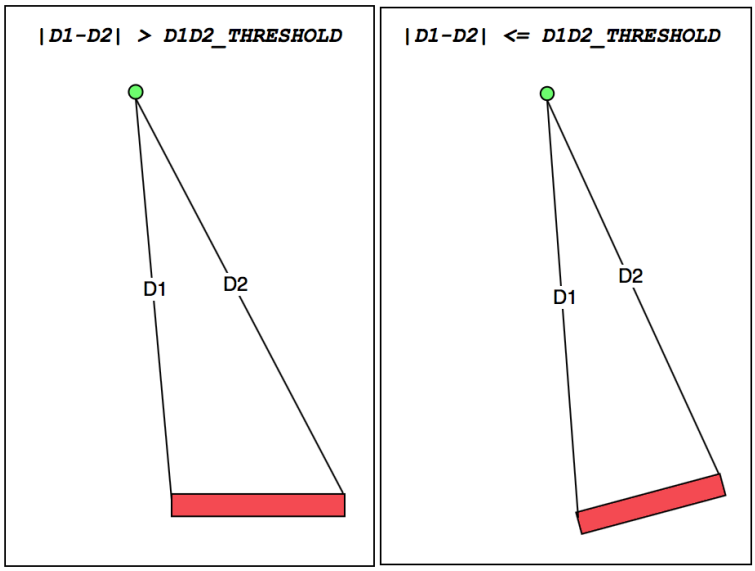


Figure 11. Left image shows a scenario where the object is out of alignment while the right image shows a case where the object is aligned towards the goal. Δ is denoted as $|D1 - D2|$ while $\Delta\tau$ is $D1D2\_THRESHOLD$.

Consider that there are two robots 1 and 2 working on the transport task and robot 1 has located the box and aligned. Robot 1 now reads the coordinates from the Observer's shared memory and calculates D1, D2, and Δ . Finding that the Δ is lesser than the threshold $\Delta\tau$, robot 1 decides that the box is ready for cooperative action and update its *STATUS_FLAG* to '*STATUS_WAITING*'. Robot 1 then reads its peer's i.e. robot 2's *STATUS_FLAG* to learn its status. If robot 2 is not ready, robot 1 waits for a fixed number of seconds, constantly polling and to check the robot 2's status at regular intervals. When the timer runs out, robot 1 resets its status flag to '*STATUS_INACTIVE*' and continues the loop. When robot 2 is also ready for the cooperative task, they robots work together and push the box for a certain amount of time called T . T is not a fixed value, it is dynamically calculated based on the distance between object and the goal (average of D1 and D2). T is directly proportional to the distance between the object and the goal: when the distance is

more, T is longer and vice versa. Also, there is a defined range of duration within which T lies. This is defined by the limits $Tmin$ and $Tmax$ respectively.

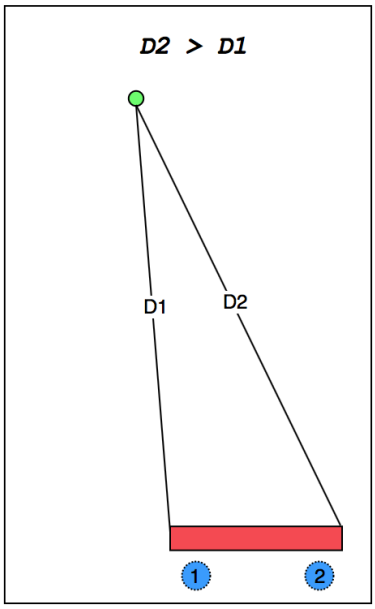


Figure 12. D2 is greater than D1, while 1 and 2 represents robot 1 and robot 2.

To explore the other possible sequence of events, consider that robot 1 has found the object and decided that the object is out of alignment and needs to be oriented ($\Delta > \Delta\tau$). However, there is no global localization strategy, and this causes various challenges: the robots do not have a way of deciding who is on which side or who has to push at a given instant so that the box can be aligned in a way suitable for cooperative pushing. Thus, the robots start pushing by trial and error. A robot checks the status of the system before and after it has pushed. Since the objective of aligning the box is to reduce the difference between D1 and D2, the robot checks for the distances before and after pushing. For example, if D2 was larger than D1 before pushing, it means that the side that belongs to D2 is far away from the goal and thus the box is facing outside. This is shown in Figure 12.

To align the box, D1 must be reduced which improves the position of the box relative to the direction of the goal. However, if the robot was on the wrong side, rather than helping, this would worsen the condition by turning the box away. In such a case, the robot would preempt and let its peer push. Thus, analyzing the status of the system before and after pushing allows the robot to decide whether it has made a positive change that could continue until $\Delta < \Delta\tau$. However, the robot would preempt in case where it had made any negative change on the system. Any robot which is ready to push the box for aligning sets its status to '*STATUS_ACTIVE*' in order to let the other robot know that it is actively pushing the box. Figure 13 shows the flowchart of the approach while the individual behaviors along with the pseudo-codes are presented in the next subsection.

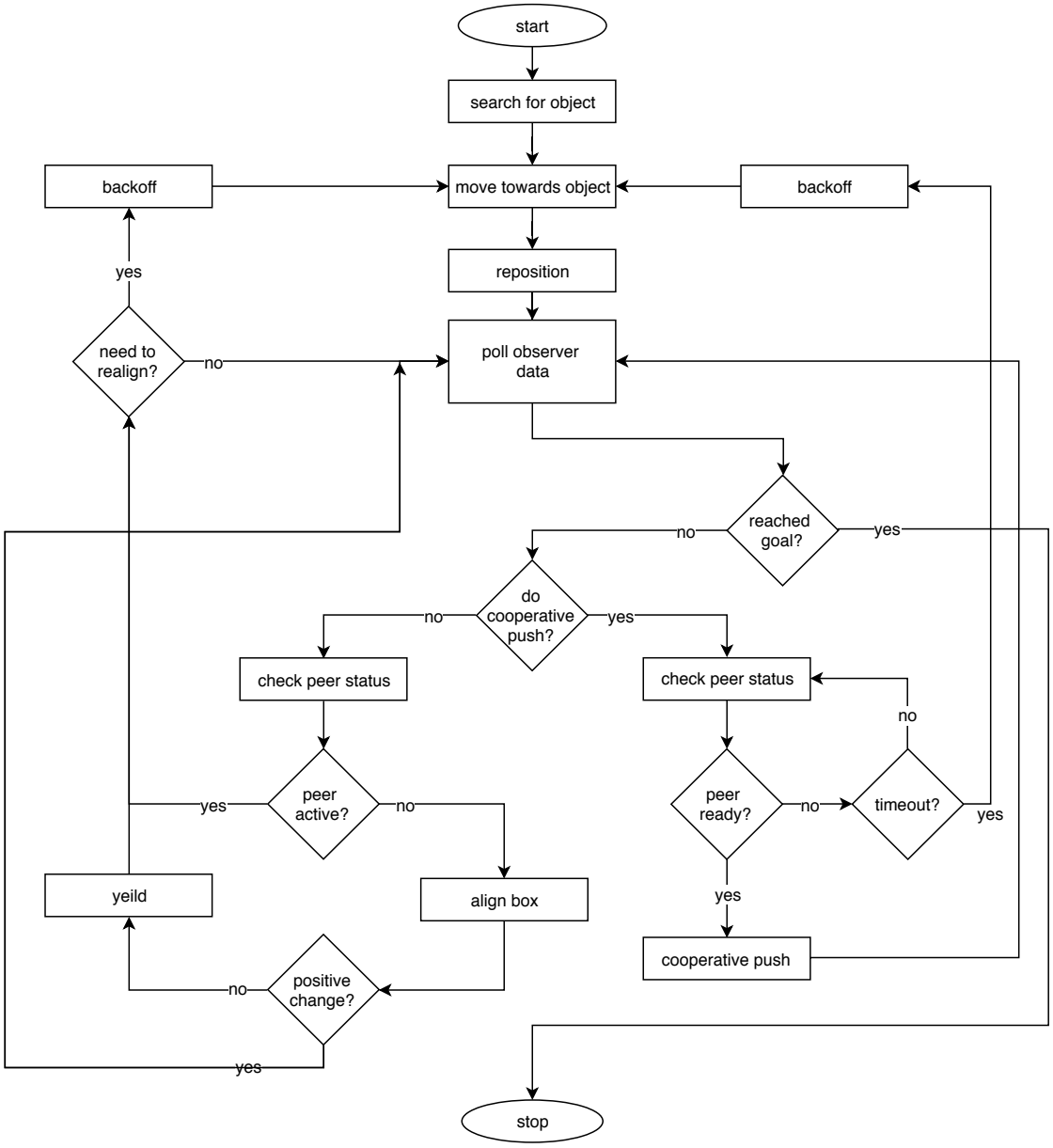


Figure 13. Algorithm flowchart.

4.3 BEHAVIORS

This section explains the behaviors presented in the state machine (Figure 14) in detail along with the transition and accompanying conditions. Other minor behaviors

such as *BACKOFF* and *JERK* are not presented as separate states in the state machine. Though the proposed strategy does not emphasize on the search behavior, it has been included in the state-machine for delivering a well-rounded picture of the system.

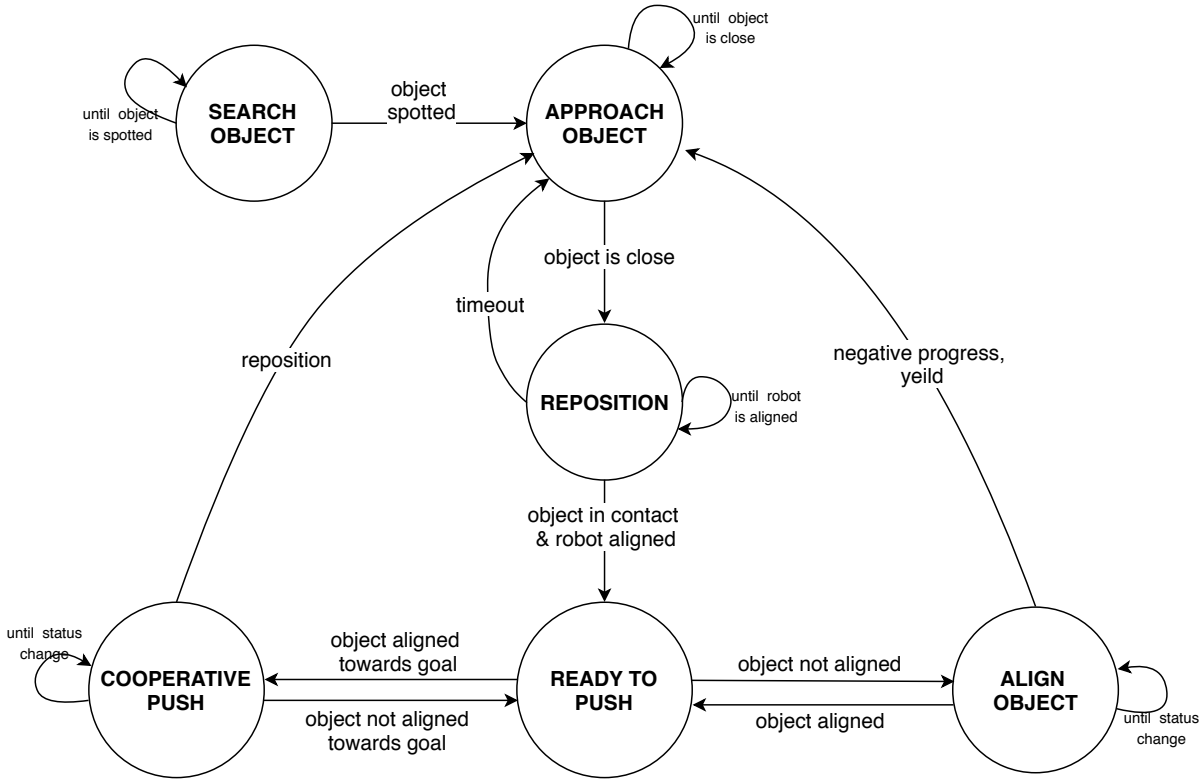


Figure 14. State machine representing the behaviors and transitions.

4.3.1 SEARCH OBJECT

The robot searches for the object. If the object is not visible, the robot uses the global information from the Observer and determines its heading direction by moving in a random direction for a second. It then reads the current location of the object as seen by the Observer and estimates the turn angle with respect to the current heading direction and proceeds to move towards it. The robot periodically

checks its own sensors and moves towards the object. When the robot has found the object, it verifies whether it is on the correct side of the object. If the robot is on the correct side, it executes the next behavior, *APPROACH OBJECT*. If the robot is on the wrong side (between the object and the goal), it executes a minor behavior called *GO-AROUND-BOX* where it goes around the box to reach the other side of the box.

```
1   If(no red blob)
2      determine heading direction
3      read coordinates from observer
4      calculate angle and turn
5      read coordinates & verify direction
6      go forward for four seconds
7      If(object found)
8         If(on wrong side of the object)
9            go around the box
10        Else
11          SEARCH behavior
12     Else
13       continue
```

Figure 15. Pseudo-code for *SEARCH* behavior.

4.3.2 APPROACH OBJECT

After locating the object, the robot approaches it. Since the object is identified by the red-markers attached on it the robot essentially moves towards the red. The behavior Inspired by Braitenberg's vehicles [33] and the color red serves as the stimulus at this stage. If the robot sees red on the left, biases its right wheel with higher speed to turn towards the left and so on. When the front IR proximity sensors read a value greater than a minimum threshold limit *Imin*, the robot transitions to the reposition state.

```
1   If(red blob)
2      While(IR_FRONT < MIN_IR_THRESHOLD)
3          determine whether object is on LHS or RHS
4          calculate speed bias
5          add speed bias to opposite wheel
6   Else
7      SEARCH behavior
```

Figure 16. Pseudo-code for *APPROACH OBJECT* behavior.

4.3.3 REPOSITION

The robot tries to get as close as possible to the object and it tries to reposition itself such that the absolute difference of Front-Left and Front-Right Infra-Red proximity sensors is within a threshold $I\delta$. Since the Khepera IV robot has a circular form factor, there is a possibility that the robot could slide-off while pushing the object depending on the angle of approach. This behavior is implemented to align the robot somewhat perpendicular to the side of the box so that the possibility of the robot sliding-off the box reduced. Figure 18 depicts how the IR sensors are used for aligning the robot where L, R, and F stand for the front-left, front-right and the front IR sensors. This stage is somewhat analogous to docking since the robot simply tries to align itself and gets ready to push the object.

```
1   If(IR_FRONT >= MIN_IR_THRESHOLD && red blob)
2      While(IR_FRONT <= MAX_IR_THRESHOLD)
3          move towards object
4      While(IR_DIFF > IR_DIFF_THRESHOLD)
5          determine which side to reposition
6          JERK left or right
7          If(timeout)
8              BACKOFF behavior
```

Figure 17. Pseudo-code for *REPOSITION* behavior.

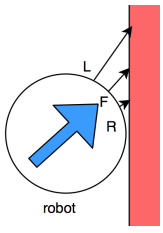


Figure 18. Reposition using IR.

4.3.4 READY TO PUSH

When the robot has repositioned itself close to the box, it enters this state where it checks for the external stimulus i.e. the coordinate points form the Observer. The robot calculates the D1 and D2 distances and if the absolute difference i.e. Δ is lesser than the threshold $\Delta\tau$, the transition leads to the execution of *COOPERATIVE PUSH* behavior, else it leads to the *ALIGN BOX* behavior.

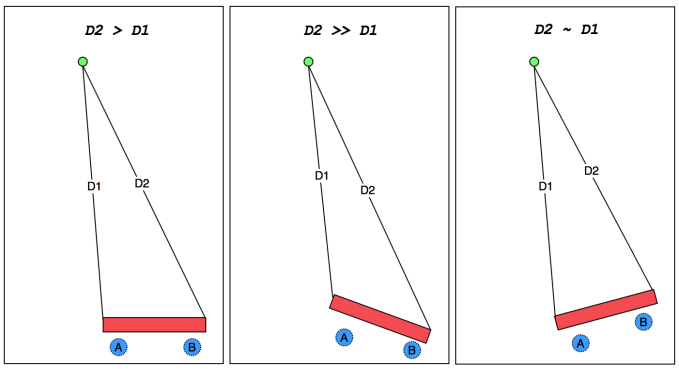
```
1    If(IR_FRONT >= MAX_IR_THRESHOLD &&
2                   IR_DIFF <= IR_DIFF_THRESHOLD)
3      read coordinates form observer
4      calculate D1D2_DIFF
5      If(ABS_D1D2_DIFF < D1D2_THRESHOLD)
6          COOPERATIVE-PUSH behavior
7      Else
8          ALIGN-OBJECT behavior
```

Figure 19. Pseudo-code for *READY TO PUSH* behavior.

4.3.5 ALIGN BOX

A robot enters this state when the box is not sufficiently aligned towards the goal and thus needs to be aligned. The robot reads the peer status and if is the peer is not actively working on the box, the robot saves the current status of the system and pushes the box for a fixed amount of time. The robot then reads the current coordinates form the Observer calculates the status of the system after it has pushed. If the robot has made a had a positive change on the position of the box and box needs to be aligned further, the robot continues in this behavior. To continue this behavior, Δ should be greater than $\Delta\tau$ even after the robot has pushed. If the impact was negative, say when the robot is on the wrong side and makes the situation worse

by pushing it, the robot relinquishes the control to its peer. Figure 20 portrays an example of such positive and negative changes that can result from the robots pushing. The robot could move to the *REPOSITION* behavior if the IR proximity sensor readings change suggesting that the box has moved.



20(a) 20(b) 20)(c)

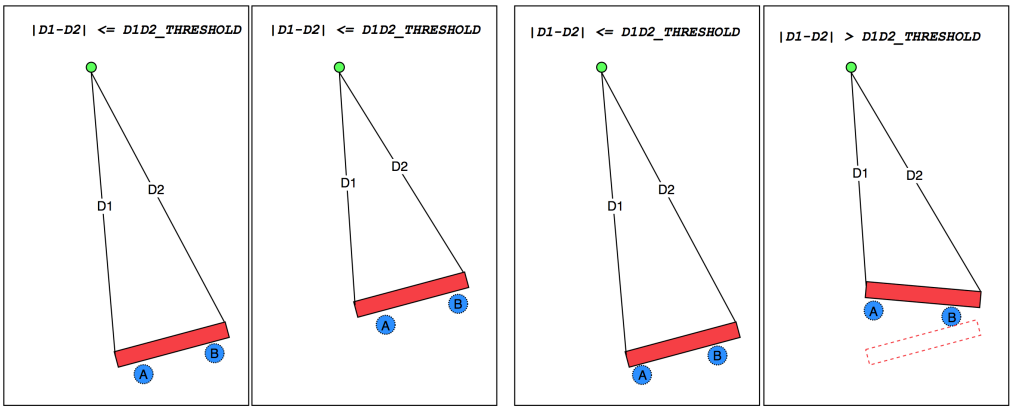
Figure 20. (a) Initial state where D1 is greater than D2; (b) robot A has pushed the box and made a negative impact; (c) robot B has pushed and made a positive change.

```
If(ABS_D1D2_DIFF > D1D2_THRESHOLD)
   IF (peer is not active)
       update status as active
       save current D1 & D2
       push box
       read coordinates from observer
       calculate new D1 & D2
       If(positive change &&
           ABS_D1D2_DIFF > D1D2_THRESHOLD)
           continue ALIGN OBJECT
       Else
           reset status
           yield
   Else If(peer active)
       continue
```

Figure 21. Pseudo-code for *ALIGN* behavior.

4.3.6 COOPERATIVE PUSH

The robot executes this behavior when the absolute difference between D1 and D2 i.e. $\Delta < \Delta\tau$. This means that the object is sufficiently aligned towards the goal and the robots can cooperatively push. This stage is usually a significant stage since two robots pushing the box together is much quicker than two robots taking a turn pushing the box. The transition from *READY TO PUSH* or *ALIGN BOX* could lead to this stage. A robot which is in this stage reads its peer's status and if the peer is ready, the robots do a cooperative push for T . This is a dynamic variable which derived from the distance between the goal and the object. If the distance is more, the duration would be longer and vice versa. *Tmin* and *Tmax* are the boundaries within which the push duration would fall. These boundaries were selected after various experiments and they facilitate striking a balance between and speed and recoverability. If the peer is not ready, the robot checks the peer status periodically until a timer goes off after which the robot executes the *BACKOFF* behavior to reposition towards the object. A cooperative push may lead to the same behavior continuing or to the *ALIGN* behavior as illustrated in Figure 22. This is possible despite the fact that the robots push with the same speed and for the same amount of time as there are various other non-deterministic factors such as the condition of the floor, non-uniform friction, position of the robots and so on.



22(a) 22(b) 22(c) 22(d)

Figure 22. (a) Cooperative push was successful and results in another cooperative push shown in (b). (c) Cooperative push that was successful, but results in Align behavior(d). In this case, robot A is comparatively closer to the edge and thus the box moves towards the right.

```
1  If(ABS_D1D2_DIFF <= D1D2_THRESHOLD)
2      IF (peer is not ready)
3          While(until timeout)
4              periodically check if peer is ready
5                      If(box moved)
6                  BACKOFF behavior
7      Else If(peer is ready)
8          save current D1 & D2
9          calculate PUSH_DURATION
10         push for PUSH_DURATION
11         read coordinates from observer
12         calculate new D1 & D2
13         If(positive change &&
14                  D1D2_difference <= D1D2_threshold)
15             continue COOPERATIVE-PUSH
16         Else
17             READY-TO-PUSH
```

Figure 23. Pseudo-code for *COOPERATIVE PUSH* behavior.

4.3.7 BACKOFF

This behavior makes the robot back off a few centimeters from the object. This is a vital behavior which helps the robot recover when it is stuck or realigns when the box has been moved by another robot. The behavior also helps the robot to maintain the object in view since the robot's camera is blacked out when is in contact with the box and thus cannot receive any useful feedback. This behavior has multiple entry points or in other words, various events could trigger this behavior. A common trigger is a change in the IR proximity sensors readings since it denotes the box has moved (because of some activity by the peer). Another trigger is timeout: while trying to *REPOSITION* or *APPROACH* the object the movement of the robots could be hindered because of the uneven surface. When *BACKOFF* is triggered three times continuously denoting that the robot is stuck in a position, the robot invokes a variation of the behavior where it jerks left and right alternatively to overcome the inertia and then back off.

4.3.8 GO AROUND BOX:

This is a minor behavior executed when the robot realizes that it is on the wrong side of the object. The robot follows steps similar to that observed in wall-following where a robot moves along a wall maintaining a certain distance. Once the robot reaches the other side, the regular behaviors such as *APPROACH OBJECT* or *REPOSITION* are executed.

4.3.9 JERK LEFT/RIGHT:

This another minor behavior which helps the robot reposition or recovers from a position where it is unable to move from. This behavior sends short pulses the wheels

in succession thus helping it move when stuck or turn in place while docking with the box.

```
1  //Main Loop
2  While (goal not reached)
3      capture image and look for red blobs
4      If(red blob)
5          APPROACH OBJECT
6          //object is close
7          If (IR_FRONT >= MIN_IR_THRESHOLD)
8              REPOSITION
9             //aligned with object
10            If(IR_FRONT >= MAX_IR_THRESHOLD &&
11                           IR_DIFF <= IR_DIFF_THRESHOLD)
12                //READY TO PUSH
13                IF (ABS_D1D2_DIFF <= D1D2_THRESHOLD)
14                    COOPERATIVE PUSH
15                Else If (ABS_D1D2_DIFF > D1D2_THRESHOLD)
16                    ALIGN BOX
17     Else If (no red blob)
18         SEARCH
```

Figure 24. Pseudo-code representing the complete algorithm as a loop. Each condition which leads to the execution of particular behavior is analogous to a stimulus. The behaviors are highlighted with bold-face fonts.

CHAPTER 5

EXPERIMENTS AND EVALUATION

The effectiveness of the approach was tested with different sets of physical experiments in the laboratory. Various metrics were collected to evaluate and analyze the approach and are presented in this chapter.

5.1 EXPERIMENTAL SETUP

The test environment consists of two different setups in our Pixel Island lab located in room 209 of the Engineering building. Arena 1 as is smaller measuring about 12 x 11 feet. The Observer camera was placed about 9.5 feet from the floor and 4 inches from the ceiling. Arena 2 is about twice the size, measuring 22 feet long and 8 feet wide. Figure 26 and 27 shows arena 1 and arena 2 as seen by the observer. The black markers on the floor denote the boundaries of the arena for the viewers' reference.

Figure 25. From left: Goal, Khepera IV robots, object(back) and red markers.

As discussed earlier in Section 3.3, not the entire floor in the arena was flat due to the worn-out condition. Certain parts of the floor had small bumps which made

it harder for the robots to move. Thus the experiments were conducted in the area where the surface was with less hindrance and workable for the robots. The object is an elongated rectangular cardboard box measuring 33 inches (2.75 feet) in length and 7.2 inches tall and 4.4 inches wide. The Khepera IV robots are 58 mm (2.2 inches) tall while their camera is 2.5 cm above the ground. The objects were marked with red square patches of 4.5inches while the goal was a bright green box measuring 7.1 x 6.8 inches.

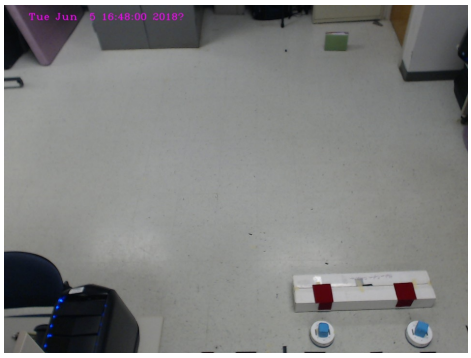


Figure 26. Arena 1: 12x11 feet.

Figure 27. Arena 2: 22x8 feet.

5.2 METRICS

To analyze and evaluate the approach, various metrics were collected for each of the trials and such metrics are introduced here while the experiments and the corresponding results are presented in the following sub-sections. The metrics are presented below:

1. ***Path efficiency(PE):*** Path efficiency of a trial is defined as the ratio of the shortest distance between the centroid of the object and the goal to the actual distance traveled by the object's centroid in that trial. This metric gives an idea about the distance traveled by the object versus the shortest possible distance.

In formula 5.1, $Distance_{optimal}$ is the shortest distance between the object and the goal and $Distance_{actual}$ denotes the actual distance traveled by the object before entering the goal region or making contact with the goal. The distances are Euclidean and are derived directly from the image frames captured by the Observer and thus all the distances are in pixels. Since hundreds of frames are generated for a given trial, the frames are sampled at regular intervals to calculate the cumulative distance traveled by the object.

$$PathEfficiency = \frac{Distance_{optimal}}{Distance_{actual}} \tag{5.1}$$

2. ***Completion Time:*** The completion time of a trial is defined as the time elapsed from the start of a trial until the end of the trial which is when the centroid enters the goal region. The completion time is measured in seconds and essentially denotes how much time a trial took.

3. ***Angular Displacement:*** Angular displacement is defined as the difference between the relative difference in the orientations of the object at the beginning and the end of a trial. Let $p(t_0)$ and $q(t_0)$ be the centroids of the two tracking markers on the object at the beginning of a trial. The orientation vector $a(t_0)$ of the object at the beginning shown in 5.2.

$$a(t_0) = p(t_0) - q(t_0) \tag{5.2}$$

Similarly, considering that the centroids are $p(t_1)$ and $q(t_1)$ at the end of the trial, the angular displacement $D(t_0, t_1)$ is calculated as shown in Figure 28.

$$D(t_0, t_1) = \left| \arccos \frac{\mathbf{a}(t_0) \cdot \mathbf{a}(t_1)}{\|\mathbf{a}(t_0)\| \, \|\mathbf{a}(t_1)\|} \right|$$

Figure 28. Angular displacement

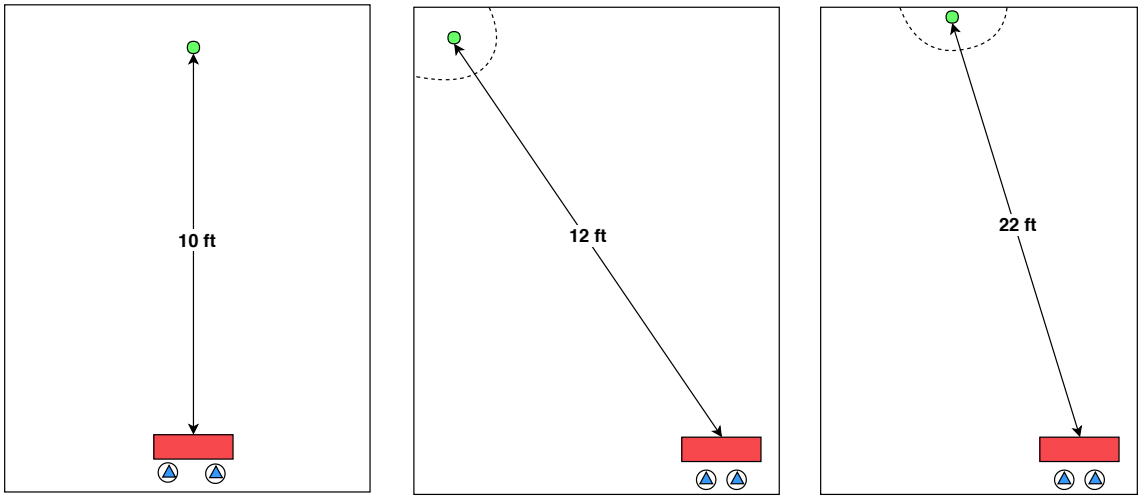
4. ***Cumulative Angular Displacement:*** Cumulative angular displacement is calculated as the sum of the angular displacements of the object at each step of the trial. This metric gives an idea of how much the orientation of the object is changing throughout a trial or in simple words gives an estimate of how much the object 'wiggles' while being transported. The metric *Angular displacement* described above simply shows the relative orientation of the object and does not represent anything significant. Thus cumulative angular displacement is calculated as another metric that gives the relative change in the orientation of the object throughout a trial.

5. ***Final Distance from Goal:*** Final distance from goal is calculated as the distance between the centroid of the object and the goal when a trial ends. The distance is Euclidean and gives an idea about how much closer the object is when a trial ends.

5.3 EXPERIMENTS AND RESULTS

Various experiments were performed to demonstrate the approach and are categorized as Experiment A, B, C, D and E. Each experiment with the corresponding procedure and results are presented below. Figure 29 shows a plan view of the experiments A, B, and C respectively. Experiments A, B, D and E were conducted on Arena 1 (Figure 26) while experiment C was done in Arena 2 (Figure 27). Apart from demonstrating that the approach works, each experiment had other underlying

objectives. Experiment A and B were to examine the stability of the approach while Experiment C was conducted to demonstrate and examine the factors that affect performance in a bigger arena. Experiment D was conducted to evaluate how how well the approach can work in situations where the object and goal locations are different within the test environment and how the orientation of the object affects the performance. The purpose of Experiment E is to test the *SEARCH* behavior. To convert the image coordinates to real world coordinates, the observer cameras were calibrated with a checkerboard pattern to derive the intrinsic and extrinsic parameters. The *cameraCalibrator* application in Matlab was used for this task. The experimental procedure and analysis are presented in the following sections.



29(a) 29(b) 29)(c)

Figure 29. (a)(b)(c) Experimental setup for Experiment A, Experiment B, and C respectively(not drawn to scale). The red box is the object and the blue triangles within circles represent the robots. The green dot denotes the goal while the dashed line around the goal represents the goal region.

5.3.1 EXPERIMENT A

The objective of this experiment is to evaluate the performance in a situation where the robots cooperatively work to move an object which is already aligned towards the goal. The robots have to move the object to the goal which is located about 10 feet ahead. The experimental procedure is described below.

The initial conditions are the same as stated in the problem description where the robots have already located the object and positioned themselves appropriately. Now, the robots have to cooperatively work and push the object towards the goal which is about 10 feet ahead of the object. As shown in Figure 29 (a), the plan view of the experiment, the object is placed close to the bottom of Arena 1 while the object is placed at the top. The box is sufficiently aligned towards the goal location and thus the robots can start cooperatively pushing the box quickly instead of aligning it. The stop condition is when the object makes contact with the goal. 20 trials were performed with the same initial conditions in Arena 1. The mean and standard deviation of the various metrics for the 20 trials performed are presented below in Table I while the histogram representing the final distance between the goal and the object is presented in Figure 31.

	Completion Time(s)	Angular Displacement(deg)	Cumulative Angular Displacement(deg)	Distance from Goal (mm)	PE
MEAN	167.14	15.74	105.69	154.14	0.9595
SD	67.41	12.35	101.19	157.84	0.0554

Table I. Mean and standard-deviation for Experiment A(20 trials) where PE stands for path-efficiency.

The average completion time is about 160s while the median is about 154s. Though the task is simple where the robots need to push the object forward, due

to various factors such as non-uniform friction, position of the robots and the floor condition, the completion times vary drastically. The completion times recorded ranges between 86 and 273s. This can be explained based on the sequence of behaviors that were executed in each trial. As mentioned earlier in chapter 4, the *ALIGN OBJECT* behavior is much slower than the *COOPERATIVE PUSH* behavior. Though the initial condition implies that the robots can start by executing the quicker *COOPERATIVE PUSH* behavior, the outcome is non-deterministic. The initial cooperative push might lead to either the align object state or the cooperative push state as shown in 22. If the robots enter the align state frequently, they would be spending more time aligning the box which is slower. Comparing that to a situation where the robots enter the cooperative push stage frequently, it could be understood that how the behaviors influence completion time. Thus, if a trial involves frequent align operations, it tends to be slower. Also, other parameters such as the $\Delta\tau$ and the T influence the performance and this is discussed in section 5.4.

Figure 30. Paths taken by the object's centroid for various trials of Experiment A.

Figure 30 shows the paths of the centroids of the object traced from the logs for all the trials of Experiment A.Though the experiment is straight forward where the object has to be simply pushed to the goal, from Figure 30, it is easy to spot how non-deterministic the paths taken by the object is. This demonstrates how experiments being conducted in a real-world environment are subject to various influential factors such as non-uniform friction, the position of the robots and floor condition.

5.3.2 EXPERIMENT B

The objective of this experiment is to evaluate the performance in a situation where the robots cooperatively work to move an object which is not aligned towards the goal. This means that the robots need to align the object first and then cooperatively work on the task. The experimental procedure is described below.

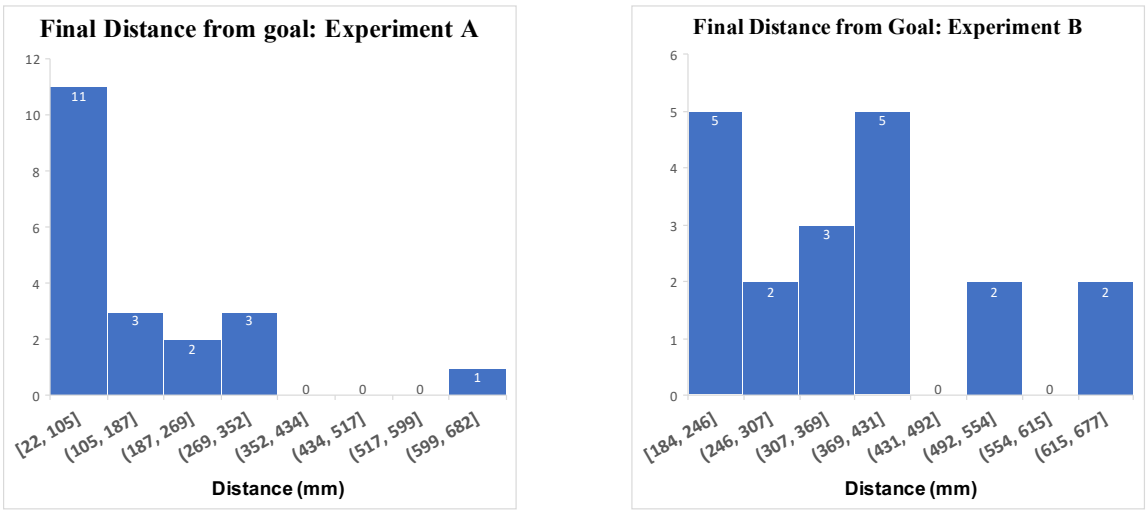
The initial conditions for Experiment B is same as the problem description: the robots have already located the object and repositioned themselves as needed. However, unlike Experiment A, the box is not aligned towards the goal and thus the robots will have to align the box before beginning the cooperative push. Also, the goal is about 12 feet form the object and placed such that the path taken by the object would be diagonal. The object itself is located at the bottom right corner of Arena 1 while the goal is placed diagonally on the top left as shown in Figure 29(b). The stop condition is either the object's centroid is within 160 pixels from the goal location or the object colliding with the goal. 20 trials were performed in Arena 1 and the results are presented in II.

	Completion Time(s)	Angular Displacement(deg)	Cumulative Angular Displacement(deg)	Distance from Goal (mm)	PE
MEAN	141.57	32.68	174.47	369.96	0.9646
SD	48.00	7.58	43.72	151.65	0.0216

Table II. Mean and standard-deviation for Experiment B(20 trials) where PE stands for path-efficiency.

The mean and standard deviation of the various metrics for the 20 trials performed are presented above in Table II where PE denotes the path efficiency. The average time taken for completion is about 141s which is comparatively lesser than that of Experiment A where the task was much simpler and the goal was com-

paratively closer. This is because Experiment B and Experiment A used different parameters. Since Experiment A and B are different, comparing their results to analyze them is not appropriate and thus the impact of the parameters are discussed in section 5.4 with different sets of experiments. While path efficiency is similar to that of Experiment A, the final distance from the goal in is comparatively higher in B. This is again attributed to the influence of the different parameters.



31(a) 31(b)

Figure 31. (a)Final distance from goal presented as histogram for Experiment A and Experiment B(b). The number of bins is 8.

The histogram representing the final distance between the goal and the object for both Experiment A and Experiment B is presented in Figure 31. The histograms have eight bins each while the bin-width is about 83mm and 62mm respectively. Figure 32 shows the paths taken by the the centroids of the object traced from the logs for the trials of Experiment B.

Figure 32. Paths taken by the object's centroid for various trials of Experiment B.

5.3.3 EXPERIMENT C

Experiment C carried out in Arena 2 to demonstrate that the algorithm is capable of working in a larger area. Arena 2 is 22 feet long and the robots need to push the object across the room. The experimental procedure is described below.

The initial conditions are that the robots have already found the object and repositioned them sufficiently. Another initial condition is that the object is not aligned towards the goal and thus the robots have to align the object first. As shown

in Figure 29 (c), the object has to be moved from the bottom right corner to the goal location across the room. For some of the trials, the goal location and the object location was swapped. The stop condition is that the object's centroid is within 40 pixels from the goal. Due to the longer distance, the image was warped and the regular markers used to track the the object were tiny and undetectable by the Observer. Different markers were used: two red cubes of 4.5 inches were affixed on either end of the object to serve as markers. The mean and standard deviation of the various metrics for the 10 trials performed are presented below in Table III.

	Completion Time(s)	Angular Displacement(deg)	Cumulative Angular Displacement(deg)	Distance from Goal (mm)	PE
MEAN	335.9	6.35	182.48	814.44	0.9602
SD	92.59	6.40	69.74	382.61	0.0275

Table III. Mean and standard-deviation for Experiment C(10 trials) where PE stands for path-efficiency.

Since the distance traveled is significantly higher, the completion time is also relatively higher than the previous experiments. Path efficiency similar to the other experiments while the final distance from goal is significantly higher. This is due to the warping of the image: farther an object is from the camera, lesser the pixels representing the object in the captured image.

5.3.4 EXPERIMENT D

This experiment consists of various trials performed to demonstrate that the proposed approach is capable of working with the different initial position of the goal and the object. In other words, these experiments demonstrate that the strategy works for different object and goal location provided that the problem constraints are met. The experiments were done in Arena 1 and the initial location of the goal

and the object were decided at random. The experimental procedure is described below.

The object and the goal locations were chosen at random while the initial conditions were the same as the other experiments: The object has been located by the robots. The goal position was changed to different places and the box orientation was also different for the experiments. Figure 33 shows time-lapse images captured from various trials. 15 Trials were performed in Arena 1 and the results are presented in Table IV.

	Completion Time(s)	Angular Displacement(deg)	Cumulative Angular Displacement(deg)	Distance from Goal (mm)
MEAN	129.76	46.98	237.82	578.36
SD	72.94	43.78	150.54	365.68

Table IV. Mean and standard-deviation for Experiment D(15 trials).

From the results, it can be observed that the cumulative angular displacement is higher than all the previous experiments. This would be attributed to the varying initial orientations of the box. Since the initial angle between the goal and the object is higher for these trials, the box has been moved and aligned more often and thus the overall change in the orientation of the box is higher. Completion time does not relay anything significant as the object did not travel the same distance in all the trials. The current implementation of the algorithm works well in most cases. However, there is a chance that the approach would fail when the angle between the object and the goal is close to ninety degrees.

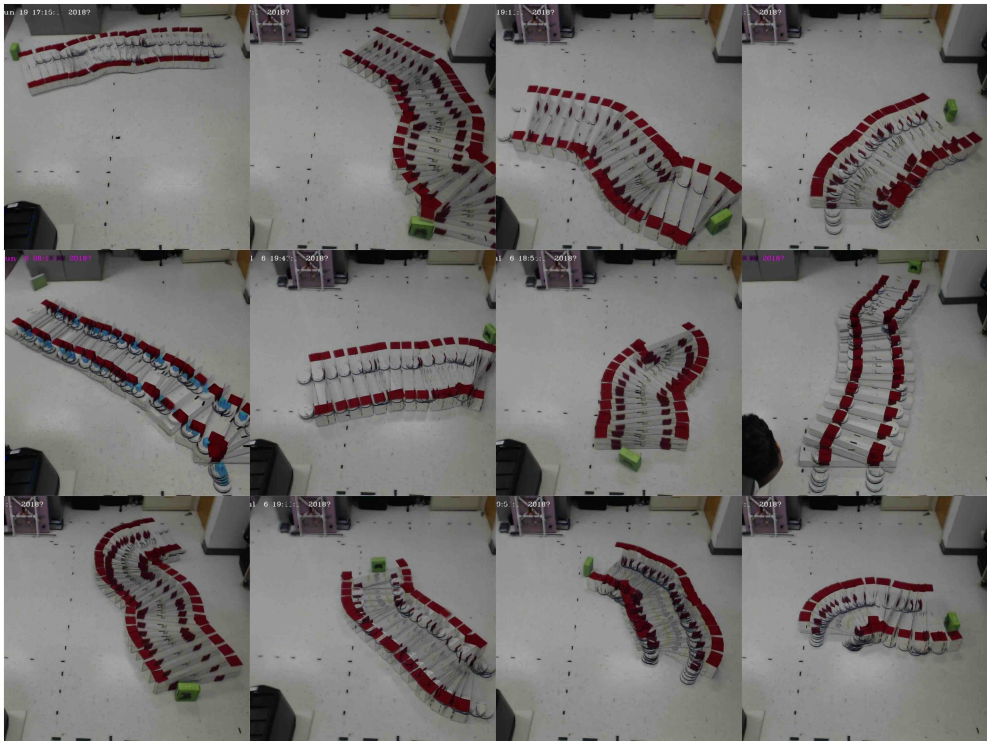


Figure 33. Time-lapse images of sample trials of Experiment D (12 trials are displayed).

5.3.5 EXPERIMENT E

The objective of Experiment E is to demonstrate the *SEARCH* behavior. Due to the limited vision capabilities of the robot, the object would spotted often and thus the robot has to rely on the search behavior to locate the object. For this experiment, the robot would be placed at random initial location with a random orientation and it has to search and locate the object. In some of the trials, the orientation and position of the object was also changed randomly. Figure 34 shows all the traced paths for all the 15 trials.

Figure 34. Paths taken by the robot during the search operation of Experiment E (15 trials). The dots denote the starting location while the arrows denotes the end point of the traced path.

The mean completion time for the 15 trials was 116.06 seconds while the standard deviation was 57.76. Figure 35 below shows some the traced paths for some of the sample trials under Experiment E. It could be noted from Figure 34 and 35 that the search algorithm is not very efficient. This is because the coordinates being shared by the observer is less meaning full for the observer since there is no direction component in it. For example, the robot can estimate the angle it has to turn however it would be hard to know which direction(turn left or right). Thus the robot calculates an angle and proceeds in that direction for one second and stops to verify

if it is heading in the right direction. If the direction is wrong, the robot corrects its heading direction and moves forward. Also, since the robot's camera is blurry and running at a low resolution, they often mistake colors of similar shade for red and head in a wrong direction. However, when they get sufficiently close, they realize that the color is different and execute the search behavior again. All these factors result in the robot taking erratic paths to find the object as shown in Figure 34

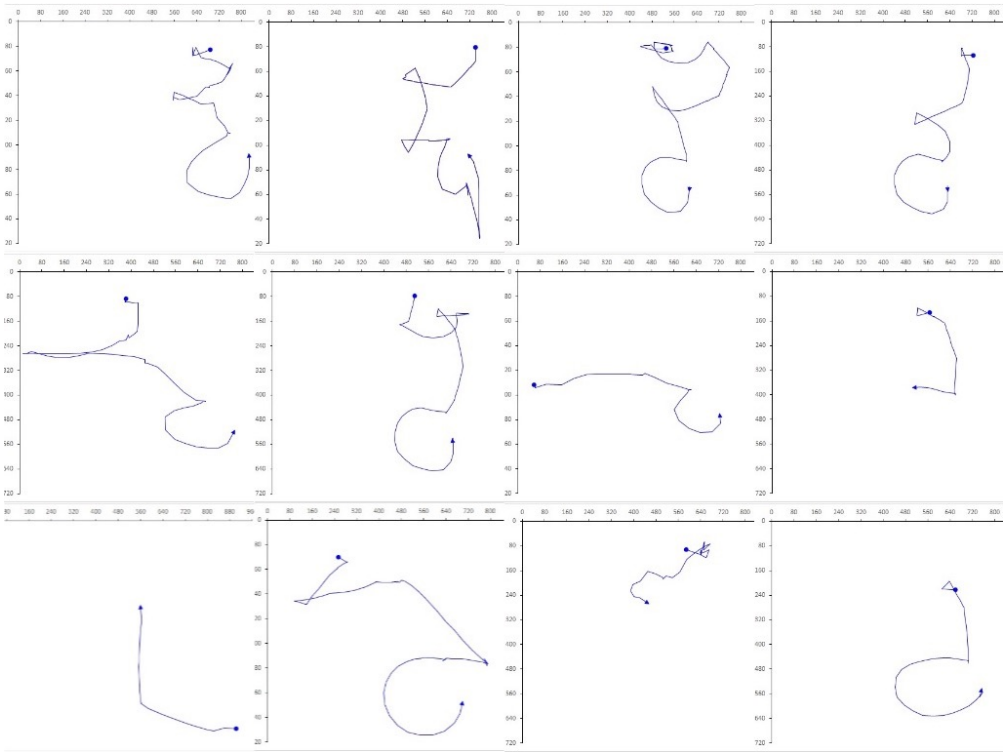


Figure 35. Experiment E: Traced paths for 12 sample trials. The dots denote the starting location while the arrows denote the end point of the traced path. The curves are resulting from the *FOLLOW-THE-BOX* behavior where the robot goes around the box to reach the other side.

5.4 ANALYSIS ON PARAMETERS

Comparing the metrics from Experiment A and B, a trend observed is that Experiment B had shorter completion times than Experiment A even though the distance traveled was more. Another pattern is that the angular displacement is comparatively lower in the trials belonging to Experiment A. These outcomes are due the reason that Experiment A and B used different parameters in their algorithm. There are two major parameters which could be changed to influence the performance and are described below:

1. $\Delta\tau$ *or* ***D1D2_THRESHOLD***: As described earlier in Chapter 4, $\Delta\tau$ is the threshold denoting when the robots execute the align object and cooperative push behaviors. This parameter value can influence the performance greatly as they dictate which behavior needs to be executed. If the threshold value is higher, the robots can execute the cooperative push behavior more often. Since the cooperative push behavior is quicker than the align object behavior, the completion time would be comparatively lesser in this case. However, with a higher threshold, the error values such as the cumulative angular displacement and final distance form goal tend be comparatively higher since the robots do not align the object frequently. The key aspects of the $\Delta\tau$ are summarized below.

    - Influences which behavior is executed.
    - Larger value: Robots enter *COOPERATIVE PUSH* behavior quicker and more frequently. Thus, completion time would be comparatively less. However, larger $\Delta\tau$ leads to more errors since robots align the box less frequently.

- Lesser value: Robots take more time to align the box. Since the *ALIGN* behavior is relatively slower, the completion time will also be higher.
- Appropriate limits: 20 - 45 pixels (determined through various experiments).

2. ***PUSH_DURATION***: T determines how long the cooperative push can happen. The push duration is dynamically calculated based on the distance between the object and the goal. However, the calculated push duration should be within an range. Through various trials, it was found that the appropriate limits of the push duration is 3 - 8 seconds. When the push duration is about 3 seconds, the robots do not seem to make as much progress. When the robots start pushing, a lot of effort is spent to overcome the inertia and to get the box moving. If the push duration is low, the robots would not make much progress in moving the box since they would have already stopped by the object started moving. Longer push duration can reduce time taken, however very large values increases the possibility of the box being pushed to a state where the robots cannot realign the box and recover. The key aspects are summarized below

   - Determines how long cooperative-push happen.
   - Two variables *Tmin* and *Tmax* are the boundaries within which the push duration falls.
   - Longer cooperative-push time reduces completion time.
   - Appropriate limits: 3 – 8 seconds (determined through various experiments).

To demonstrate the influence of the parameters, experiments were conducted

with different parameters in the same setup and the results are presented in Figure 36 and Table V.

Experiment	Completion Time (s)		Cumulative Angular Displacement (deg)		Final Distance from Goal (mm)	
	Mean	SD	Mean	SD	Mean	SD
D1D2_DIFF = 20	139.7	36.32	100.90	50.74	217.56	145.43
D1D2 _DIFF= 40	116.11	45.11	79.69	62.45	270.87	279.4

Figure 36. Mean and standard-deviation for varying Δ . A bigger value leads to quicker completion but the error values such as angular displacement, distance from goal tend to be higher as well.

Completion Time	T = 4s-6s	T = 5s-8s
MEAN	373.20	298.6
SD	103.59	71.27

Table V. Completion time for varying T limits in Arena 2. A longer duration results in comparatively faster completion.

The impact of the parameter Δ is explained here by comparing the results of Experiment A and B from the previous section as examples. As described earlier, $\Delta\tau$ determines how much effort is made in aligning the box towards the goal before pushing. In Experiment A, the threshold value of the difference between the two extremities of the object is lower and thus the robots invest more time in orienting the box. This reduces the possibility of the box traveling in a less optimal direction. However, this also results in the side effect where the completion time is comparatively longer for Experiment A. Contrastingly, if the objective is to push the object quickly, the threshold value of $\Delta\tau$ would be higher since and thus the angular displacement would also be comparatively higher. Another interesting observation is that though the overall distance traveled in Experiment B is longer than Experiment A, the completion time is comparatively less.

Experiment C brought a different set of challenges. Though the initial experiments were successful, they took a long time to reach the goal. This was due to the fact that the image was warped. The farther the object is form the camera, lesser the pixels representing the object on the image. Thus, the image pixels does not represent the actual distance between the object and the goal. This resulted in situations where the Euclidean distance calculated from the image pixels were much lesser than the actual distance. Since the T is directly proportional to the calculated distance, the robots tend to cooperatively push only for shorter bursts and thus taking a long time eventually. To optimize the completion time for this case, the $Tmin$ was increased from 4s to 6s, resulting in much shorter completion times.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

A Behavior-Based Robotics strategy was developed for the cooperative transport problem demonstrating that the task can be achieved through a simple rule-based approach instead of relying on global decision making or sophisticated controllers. Contrasting to other approaches that use a centralized global view, the Observer has no decision making responsibilities in this system. Since the system is based on simple SENSE-REACT primitives, the information being shared by the Observer is also simple thereby reducing the computational load on the Observer. The Behavior-Based Robotics approach considers the robots as simple organisms with embodied cognition to a certain extent. This could be observed to be true in our approach as well since the robots have a way of distinguishing or sensing 'good' and 'bad' with respect to the given task. Also, The approach demonstrates that the robots could gain more autonomy by relying on their senses to make decisions instead of relying on external instructions.

Various experiments were also performed in a real environment to evaluate and analyze the approach. The completion time is greatly influenced by behaviors being executed. When the robots push the object cooperatively, they are able to move the object further and faster. Also, there are various other supporting behaviors that are little slow, but play important roles. As the robots use euclidean distance as a measure and as there is no directional component in the information being shared by the observer, various additional behaviors are required to align, reposition and push the object. Also, the orientation of the object influences the outcome. This is because rectangular objects are easily susceptible to rotation and thus it is more

challenging to transported them.

The completion time of the proposed system can be improved. For example, the *ALIGN* behavior is very slow compared to the *COOPERATIVE PUSH* behavior. This is because when both the robots push together, it is much easier to overcome the friction and thus the box could be moved further with the same effort form each of the robot. Though the *ALIGN* could not be made as fast as the cooperative push, optimizing the aligning process would shorten the completion time. The robots often make wrong decisions on who needs to push and when a wrong robot pushes, the box needs to realigned. If more wrong decisions are made, more time has to be spent correcting the orientation of the box and thus leading to longer completion times. Performance would improve if this type of wrong decisions are avoided. In the current implementation, though the robots have a sense of realizing good and bad, they do not remember or keep track of it for future decisions. The robots do not keep track of which side of the object they are pushing(left or right). If the robots are programmed with some mechanism of remembering their position, the possibility of wrong robot pushing the box can be avoided. This would improve performance by reducing the completion time.

Also, the robots spend more time repositioning themselves to dock with the box. Some of the behaviors such as Back-off and Reposition are important due to factors such as limited sensing and the shape of the robot. These behaviors are executed frequently and the robots spend a sizable chunk of their time in these behaviors. Since these behaviors are little slow, it gives an impression that the robots are slow overall. Improvement in performance could be observed if these behaviors could be optimized. For example, the Back-off behavior is executed very often: when the robot is very close to the object, its camera is rendered less useful. Thus, in order to

make sure that a robot is maintaining the object in view and not pushing another robot, it executes the back-off behavior. Then the robot captures a frame to make sure the object is ahead and approaches it. This behavior need not be eliminated, however it could be optimized. To optimize this scenario, the robot can either use the coordinates from the observer to make sure of its current position or it can rely on the infra-red proximity sensors to keep track of its orientation with respect to the box. This way the robot can execute these expensive behaviors less frequently thereby saving time.

Also, the current system was implemented considering only two agents where each agent can communicate with the other agent. This strategy of communication might not scale when the number of agents increases and thus should be approached differently. For example, if there are ten agents in the system, it might not be possible to communicate with all of the agents. In the various multi-agent system, polling and voting for consensus is a common strategy which could be integrated to this system. For example, if there are five agents (out of ten) that agree on a cooperative task, the robots can all execute the cooperative push. This kind of approach would make it possible for multiple agents to be involved in the task, however there are various other challenges such as prior knowledge on the number of agents and a broadcast type of communication.

By designing the system based on behavior-based principles, the need for the Observer to be the centralized controller can be eliminated. Though the observer is not the centralized controller, the system is still centralized as the Observer is the source of the stimulus. Thus, this system is prone to single-point of failure like any centralized system. If the observer fails because of any reason, the robots would be helpless. To address this such as scenario, a set of additional behaviors could be

added to the robots which would enhance their capabilities. If the robots are capable of realizing that the observer has failed and electing one of them as an observer, the system would continue to function. This would give the robots the ability to switch roles as the Observer or a pusher based on the situation. However, this would require the robots to have better sensing capabilities and reliable communication channels. In such as situation, other flavors of multi-agent systems especially various swarm-robotic concepts such as collective computation and distributed coverage would be helpful.

The current implementation is a synchronous closed-loop system. As presented earlier in the pseudo-code Figure 24, all the behaviors reside in a loop and the behaviors are executed based on the various stimuli present at a given instant of time. This type of implementation is analogous to a state-machine and thus is more intuitive and easy to implement. However, a common shortcoming in such a synchronous implementation is I/O wait. For example, when the the program is reading data form the camera or polling its IR sensors, the program is essentially halted as the inputs being gathered are required for further decision making. If these functions could be performed asynchronously, greater performance could be achieved.

REFERENCES

[1] Elio Tuci, Muhanad Hayder Mohammed Alkilabi, and Otar Akanyeti. Cooperative object transport in multi-robot systems: A review of the state-of-the-art. *Frontiers in Robotics and AI*, 5:59, 2018.

[2] Michael Rubenstein, Adrian Cabrera, Justin Werfel, Golnaz Habibi, James McLurkin, and Radhika Nagpal. Collective transport of complex objects by simple robots: theory and experiments. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 47–54. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[3] Erol Şahin and Alan Winfield. Special issue on swarm robotics. *Swarm Intelligence*, 2(2-4):69–72, 2008.

[4] Ronald C Arkin, Ronald C Arkin, et al. *Behavior-based robotics*. MIT press, 1998.

[5] Mattias Wahde. Evolutionary robotics. *The use of Artificial Evolution in Robotics, Tutorial presented at IROS*, 2004.

[6] Robin Murphy and Robin R Murphy. *Introduction to AI robotics*. MIT press, 2000.

[7] Maja J Mataric, Martin Nilsson, and Kristian T Simsarin. Cooperative multi-robot box-pushing. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 556–561. IEEE, 1995.

[8] C Ronald Kube and Eric Bonabeau. Cooperative transport by ants and robots. *Robotics and autonomous systems*, 30(1-2):85–101, 2000.

[9] Elio Tuci, Roderich Groß, Vito Trianni, Francesco Mondada, Michael Bonani, and Marco Dorigo. Cooperation through self-assembly in multi-robot systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(2):115–150, 2006.

[10] C Ronald Kube and Hong Zhang. Collective robotics: From social insects to robots. *Adaptive behavior*, 2(2):189–218, 1993.

[11] C Ronald Kube and Hong Zhang. The use of perceptual cues in multi-robot box-pushing. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 3, pages 2085–2090. IEEE, 1996.

[12] C Ronald Kube and Hong Zhang. Stagnation recovery behaviours for collective robotics. In *Intelligent Robots and Systems' 94.'Advanced Robotic Systems and the Real World', IROS'94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 3, pages 1883–1890. IEEE, 1994.

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

[14] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3293–3298. IEEE, 2012.

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

[16] Ralph Beckers, OE Holland, and Jean-Louis Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. 1994.

[17] Ryusuke Fujisawa, Hikaru Imamura, and Fumitoshi Matsuno. Cooperative transportation by swarm robots using pheromone communication. In *Distributed Autonomous Robotic Systems*, pages 559–570. Springer, 2013.

[18] David Payton, Regina Estkowski, and Mike Howard. Pheromone robotics and the logic of virtual pheromones. In *International Workshop on Swarm Robotics*, pages 45–57. Springer, 2004.

[19] Alexandre Campo, Shervin Nouyan, Mauro Birattari, Roderich Groß, and Marco Dorigo. Negotiation of goal direction for cooperative transport. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 191–202. Springer, 2006.

[20] James McLurkin, Andrew J Lynch, Scott Rixner, Thomas W Barr, Alvin Chou, Kathleen Foster, and Siegfried Bilstein. A low-cost multi-robot system for research, teaching, and outreach. In *Distributed Autonomous Robotic Systems*, pages 597–609. Springer, 2013.

[21] Jianing Chen, Melvin Gauci, Wei Li, Andreas Kolling, and Roderich Groß. Occlusion-based cooperative transport with a swarm of miniature mobile robots. *IEEE Transactions on Robotics*, 31(2):307–321, 2015.

[22] Brian P Gerkey and Maja J Mataric. Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 464–469. IEEE, 2002.

[23] Brian P Gerkey and Maja J Mataric. Sold!: Auction methods for multirobot coordination. *IEEE transactions on robotics and automation*, 18(5):758–768, 2002.

[24] Ying Wang and Clarence W de Silva. Cooperative transportation by multiple robots with machine learning. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 3050–3056. IEEE, 2006.

[25] Hiroshi Sugie, Yasuhiro Inagaki, Shuchir Ono, Hidyuki Aisu, and Tatsuo Unemi. Placing objects with multiple mobile robots-mutual help using intention inference. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 2, pages 2181–2186. IEEE, 1995.

[26] B Hichri, Lounis Adouane, J-C Fauroux, Youcef Mezouar, and Ioan Doroftei. Cooperative mobile robot control architecture for lifting and transportation of any shape payload. In *Distributed Autonomous Robotic Systems*, pages 177–191. Springer, 2016.

[27] Zhi-Dong Wang, Eiji Nakano, and Takuji Matsukawa. Cooperating multiple behavior-based robots for object manipulation. In *Distributed Autonomous Robotic Systems*, pages 371–382. Springer, 1994.

[28] Atsushi Yamashita, Tamio Arai, Jun Ota, and Hajime Asama. Motion planning of multiple mobile robots for cooperative manipulation and transportation. *IEEE Transactions on Robotics and Automation*, 19(2):223–237, 2003.

[29] Toni Machado, Tiago Malheiro, Sérgio Monteiro, Wolfram Erlhagen, and Estela Bicho. Multi-constrained joint transportation tasks by teams of autonomous

mobile robots using a dynamical systems approach. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3111–3117. IEEE, 2016.

[30] Guilherme AS Pereira, Mario FM Campos, and Vijay Kumar. Decentralized algorithms for multi-robot manipulation via caging. *The International Journal of Robotics Research*, 23(7-8):783–795, 2004.

[31] Rodney A Brooks. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159, 1991.

[32] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23, 1986.

[33] Valentino Braitenberg. *Vehicles: Experiments in synthetic psychology*. MIT press, 1986.

[34] iRobot. Roomba robot vaccum. `https://www.irobot.com/For-the-Home/Vacuuming/Roomba.aspx`. Accessed: 2018-07-2.

[35] Joseph L Jones and Philip R Mass. Method and system for multi-mode coverage for an autonomous robot, October 26 2004. US Patent 6,809,490.

[36] K-Team. Introducing khepera iv. `https://www.k-team.com/mobile-robotics-products/khepera-iv/introduction`. Accessed: 2018-06-29.

[37] Jorge M Soares, Inaki Navarro, and Alcherio Martinoli. The khepera iv mobile robot: performance evaluation, sensory data and software toolbox. In *Robot 2015: Second Iberian Robotics Conference*, pages 767–781. Springer, 2016.