2018-04-30

Goal Based Human Swarm Interaction for Collaborative Transport

Yicong Xu
Worcester Polytechnic Institute

Goal-Based Human Swarm Interaction for Collaborative Transport

by

Yicong Xu

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Robotics Engineering

by

May 2018

APPROVED:

Dr. Carlo Pinciroli, Major Thesis Advisor

Dr. Michael A. Gennert, Professor

Dr. Zhi Li, Assistant Professor

Abstract

Effective human-swarm interaction is essential for the introduction of swarm-based solutions into real-world application scenarios. One of the main impediments obstructing productive human-swarm interaction is the difficulty in creating suitable interfaces for humans to convey their desired intent to multiple robots simultaneously. As the size of a swarm increases, the complexity of formulating and delivering explicit commands for each individual robot becomes increasingly intractable. In fact, it can be challenging for developers or operators struggle to direct robot swarms to finish even the most basic tasks.

In our work, we consider a different approach wherein humans specify only the desired goal rather than the individual commands necessary to achieve this goal. We explore this approach in a collaborative transport scenario, where the human user chooses the target position for an object, and a group of robots subsequently moves it by adapting themselves to the environment.

The main outcome of this thesis is the integration of a collaborative transport behavior of swarm robots and an augmented reality (AR) human interface. We implemented an AR application in which a virtual object is displayed overlaid on a detected target object. Users can manipulate the virtual object to generate the target configuration for the object. The designed centralized controller translates the target position to the robots and synchronizes the state transitions. The whole system is tested on Khepera IV robots through the integration of the Vicon system and the ARGoS simulator.

Acknowledgements

I would like to express my deepest appreciation to my advisor, Dr. Carlo Pinciroli, who hosts the Novel Engineering for Swarm Technologies (NEST) Lab, and whose guidance and support has been invaluable to the success of this thesis. I would also like to thank Dr. Gennert and Dr. Li for being my committee member. Their input and advice have surely enriched the work that follows.

Thanks to all my friends working in the NEST Lab; my time here has truly made me happy, and I feel a sense of belonging with you all. I am particularly grateful to Jayam, who helped me with the thesis, presentation and all aspects of the research. I would also like to acknowledge Nathalie and Adhavan, who gave us guidance on the use of clusters and the Vicon system, and Chris, who created the Vicon-ARGoS package and helped with debugging. Special thanks to Alicia for introducing Ivan to me. Ivan's great assistance on the writing brought the thesis a second life.

Finally, special thanks to my parents, who have encouraged me from the other side of the earth; and WPI that gave me the opportunity, knowledge, and discipline to succeed in my research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

All social species (especially insects) collaborate to some extent to co-exist in nature while facing the basic challenges of survival. For example, self-organizing behaviors can be seen in ants when they form bridges with their bodies [11] and in birds when they flock together to avoid predators [32]. Although each species displays a diverse range of behaviors, the feature common to all such species is that they display emergent properties when interacting collaboratively. The basic features which differentiate swarm behavior from other forms of intra-species collaboration are decentralized control, lack of synchronicity, and simple and identical members [3]. An emergent property of intelligence appears due to interactions among simple entities achieving a more higher-level global goal. The resulting system is more robust, scalable and flexible.

Swarm robotics is an approach to collective robotics, that takes inspiration from the self-organized behaviors of social animals [6]. Because swarm robotics can leverage the emergent properties of self-organizing systems, robots designed for swarm applications need not be highly sophisticated and typically have low cost and small size. Communication range limitations, noise in sensing and actuation architectures

and algorithms are necessary to make a large group of robots work coherently.

Several mature control algorithms have been designed and developed that can drive a robot swarm robot to behave "intelligently" [6]. Robot swarms can rendezvous [10] in arbitrary dimensions, self-assemble, and self-organizing [15]. Swarms can also realize forging and chain formation, which forms a certain pattern or keep communication topology. In these implementations, autonomy, emergence, and distributed functioning replace control, pre-programming, and centralization. Collective Transport, sometimes called group prey-retrieval, is one type of such intelligent behavior that requires explicit cooperation. It is inspired by the process by which ants transport a large piece of food or humans move a huge object cooperatively. The object to be transported is usually too heavy for a single robot to move, which naturally leads to multi-agent cooperation. Determining the goal position and planning the path can be time-consuming for a large number of tiny robots, especially when they have limited sensing and communication abilities. Most relevant research is still at the stage-of-proof [22]. Our proposed method employs remote interaction with a pre-set infrastructure to exchange information between the user and the robot swarm. Thus, the user is able to choose an object and a target position for it, and this choice will be relayed to the robot swarm.

Robots are expected to operate autonomously most of the time. However, the involvement of a human operator can be beneficial to the robot's execution or even necessary [25]. Humans often have more contextual information with regard to a given task. This can help to recognize and mitigate the shortcomings inherent to the autonomous swarm's limited contextual awareness and semantic abilities. It is necessary that changes in intent can be conveyed effectively from humans to robot swarms. Research on how and what humans should use to effectively interact with robots is ongoing. Traditional interfaces include PCs equipped with keyboards, mice,

screens, and joysticks. Advanced devices range from simple tablets and wearable devices to sophisticated brain-machine interfaces and electromyographic sensors. But the exponentially demanding computations resulting from the complex interactions between individual robot entities within robot swarms mean these existing interfaces cannot offer an effective means of robot swarm control. The limitations of these systems manifest in the challenges faced by human operators, who may share a working environment with swarm system and find themselves overwhelmed. How the current methods should be modified or even replaced to interact with swarms remains in doubt. However, augmented reality (AR) has many useful qualities which might be exploited to resolve these difficulties. First, it can be used to visualize robot information directly on the swarm member by adopting mature data visualization techniques. Second, the input method comes with the development AR, VR has an integrated solution to translate the human input into recognizable data. Third and finally, the touchscreen or motion-sensing technology is not only easy for humans to operate, but can also be used to designate new tasks precisely.

Our primary motivation is to expedite the advancement of effective real-world human-swarm interactions. From factory assembly to surgery, classical robots have replaced or assisted human laborers in many roles and helped people to better perform many complex processes. While the tasks performed by such classical robots are often complex and can demand extensive human ingenuity to algorithmatize, they are typically predefined and conducted in insulated, unchanging environments. For a robot swarm, as each component of it is an independent robot, it is easier to be deployed into an unknown environment because no explicit constraints need to be fulfilled for the relations between robots. This makes the swarm system has fewer positional or kinetic constraints compared with other kinds of robot systems. But this also makes it harder for a human operator to translate high-level tasks

into meaningful instructions for each swarm member. In our research, we attempt to execute high-level goal-based tasks through human-swarm interaction. By incorporating AR technology, we have enabled the operator to issue orders and get direct feedback in a natural, unobtrusive way. A basic state machine which can be arranged and extended easily are well implemented, concealing cumbersome inter-swarm relations from the user. As the user interface needs to communicate with all the robots, a special module is necessary to translate and convey the intent. After the commands from AR terminal are received, robots will deploy around the object, and conduct the transport behaviors automatically. We aim to find a balance between manpower and efficiency of finishing the task and establish the failure check and a remediation mechanism. To simplify the study and enhance our results, a global localization system was used for every robot and for the object, but the information every robot used for their task could also be acquired by local estimators if equipped with proper sensors and communication modules.

1.1 Problem Statement

This thesis aims at realizing a viable system for human-swarm interaction on the task of collaborative transport. This task requires a target object to be transported to an assigned location at an assigned angle. The system developed requires the task to be given from the user through a friendly interface. The users should assign the task in a natural way, meaning they do not need to have any professional knowledge, choose algorithms for each robot or input their intent using predefined commands through a keyboard. The execution should be planned and conducted by the swarm. The system should have robustness which means the robot swarm should either finish the task, or correct errors and indicate irresolvable failure.

On the hardware side, it is required that the robots can apply forces to the object and have at least 2 degrees of freedom (along the horizontal plane). Furthermore, the robots possess the inter-robot communications hardware, to enable collaboration, and proximity sensors to enable obstacle avoidance. For assessment, we also need the ability to acquire the configuration information about the robot, the object, and the target, to calculate the relative position of the object to the target, and robots to the object. This means the robot should have basic range sensors and communicate with a centralized controller.

The configuration consists the dimension and poses of every individual entity in the arena. Given the position of an object in the environment where obstacles exist, the device with a camera should be able to locate the object the user selects and send the localization information to swarm. The detected position of the object is $p_o = (x_o, y_o)$, which is a predefined anchor attached to a fixed place on the object. The user selects the goal position $p_g = (x_g, y_g)$. The robots, initialized in random locations, will gather and move the object to this target position. The points forming the path of transportation are assigned by the user in order. The obstacles can be detected by swarm or camera. The transport process can be intervened by dragging a path on the screen to assist the transportation process.

1.2 Thesis Structure

This thesis is organized in the following manner. The second chapter details related work, including others' work on swarms, human-swarm interaction, collaborative transport, trajectory control, etc. The third chapter, Design and Implementation, describes our specific implementation in detail and lays the groundwork for our testing on the implementation. The fourth chapter details the experiment design

to test the performance of the whole system, and list the corresponding result. Conclusion and future work are discussed in the last chapter.

Chapter 2

Related Work

We give a broad review on the topic related to each part of the system, covering the human-swarm interaction, augmented reality and collaborative transport.

2.1 Human Swarm Interaction

Emergent properties related to human-swarm interaction (HSI) needs to be clarified. The survey by Colling et al. of [25] identifies the core concepts needed to design a human-swarm system. In the work, HSI is introduced from the perspective of a human operator by discussing the cognitive complexity. The interface is introduced between swarm and operators. The challenges and solutions relating to human-swarm communication, such as state estimation, visualization and human control of swarms, are introduced. For the latter, they develop a taxonomy of control methods that enable operators to control swarms effectively. At the lowest level, the user can assign the motion primitives for the robot. At a higher level, different kinds of algorithms, consisting of a bundle of flocking, foraging, and engagement, can be switched and executed by swarm robots [24]. Other types of control from the operator include changing parameters of a swarm control algorithm, indirect influ-

ence through the environment or selecting and controlling through leaders [25]. The majority of interaction happens remotely (i.e., the human is outside the swarm and sending commands through a computer terminal). Since swarms are often intended to be deployed in inaccessible or dangerous areas, remote interaction is safe and more comfortable for a human user. Different from fully decentralized controllers which only need local information or communication with neighbor robots, human needs to exchange information with swarm robots during the interaction, which means a large scale of the communication network is required to be maintained. Another type is the proximal interaction where operators and swarms working in a shared environment [25]. The robot can sense the operator, who can be seen as a special member of the swarm.

Nine categories of human-swarm interaction metrics derived from the biological and robotic swarm literature are presented in [7]. It is judged that human-robot and human-computer interaction metrics may be inappropriate to describe the behaviors emerging in human-swarm interaction, and biological swarm metrics are carefully investigated. Based on a broad range of literature review, they identify a total of fifty-four metrics applicable to HSI. Categories include human attributes, task performance, timing, status, leadership, decisions, communications, micro-level movements and macro-level movements.

Application closer to HSI has been done [28]. A novel end-to-end solution is presented for distributed multi-robot coordination that translates multitouch gestures into low-level control inputs for teams of robots. An iOS application developed by the authors in which the user is presented with a team of robots and a prism (a bounding box covering all the robots). The translating and scaling operation to the prism in a virtual environment will change the coordinates of the prism which are transferred to the server and act as the input of the onboard robot controllers. The

collision-free convergence to a goal is guaranteed between robots. This approach allows the human user to solve the cognitive tasks such as path planning while leaving precise motion to the robots. A photo of their iOS application is shown in figure 2.1.

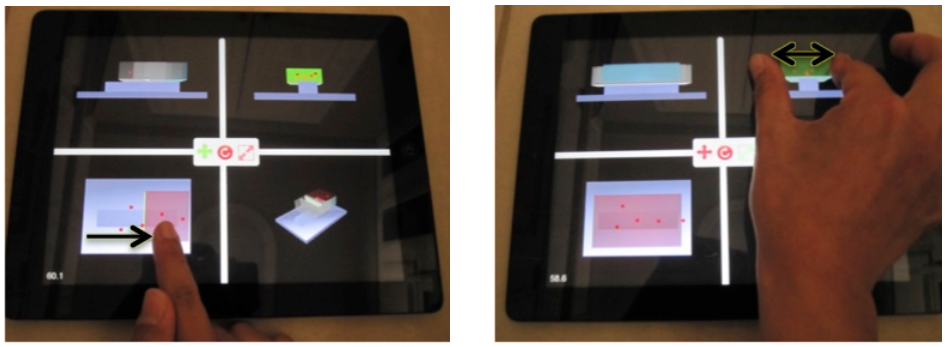


Figure 2.1: The iPad App interface of robot team control

2.2 Collaborative Transport

A hybrid force/position control architecture for object transportation is introduced in [27]. The architecture uses an object-level controller to compute the forces and torques needed to push an object along the desired path. A multi-robot actuation subsystem, which has a force/position controller, uses these desired forces/torques to generate commands, and applies proper net force and torque to the object, while maintaining relative the positions of individual robots with respect to the object. But they do not consider rotation and the force/torque calculation is an open-loop control.

Based on a specially-designed physics model, [26] investigates a simple decentralized strategy for robots to transport collectively without explicit coordination. What an agent knows is only the target direction; it does not know the object shape,

object weight, its own position, or the position of other robot entities in the system. Their paper mainly focuses on the transport phase after the robots have been deployed. The object can move exactly how the swarm moves. However, the success of their paper is limited to certain selections of robot platforms by which the robot can apply force in any directions no matter where it is attached to the object.

A fully decentralized pushing method is depicted in [22]. Object color is used as a guide by generating the occluded area behind the object where a direct view to the target is blocked. A robot swarm moves randomly to find the occluded area. Occluded robots move against the object to apply a lateral force. The mathematical derivation of this method is also given. If the changing occluded outline of the object is acted on by uniformly distributed line forces, the object will move and converge to the target position. As the object position changing, the outline acted by robots will keep changing with the occluded area. The translating direction does so. The object will converge to the target position in the end.

The cooperative control framework described in [14] has the ability to switch between decentralized controllers, which allows for changes information. A single type of omnidirectional camera is utilized by proposed estimators at different levels. The single-camera can be used as an obstacle detector, a collision detector, or a state observer. Like [19], which estimates the average position by constructing a tree, the estimator included in the controller address the relative localization problem with local information.

2.3 Augmented Reality

As detailed in [35], augmented reality (AR) combines real-world and computer-generated data (virtual reality). AR systems integrate computer graphics, computer

vision, and image processing to create a kind of visualization technique that is useful in a variety of domains including robotics. AR has been used for robotic navigation, mobile applications, computer-assisted surgery, education, gaming, military devices, manufacturing, product assembly, and repair, etc.

The most popular application of AR in robotics is in medical robotics. [1] includes information about the use of additional force feedback to a tele-operated robot-assisted surgical system. Instead of supplying the force or tactile feedback to the user's hand, the system displays the force level overlaid on top of the moving instrument tips. [31] describes the utilization of augmented reality in a laparoscopic surgical robot, da Vinci, and an endoscopic robot system containing an AR function that can show the location and direction of the system tip.

AR could be used in robotics software debugging, allowing the developer to see the real world as the robot does. [9] contributes a systematic analysis of the challenges in robotic software that make an AR debugging space an attractive option. Based on these, they describe an open-source implementation of an intelligent debugging system and present an evaluation of its efficacy. [13] details the transformation of a robot swarm into a distributed interface embedded within the environment. Each robot acts like a pixel within a much larger visual display space and only needs to display a small amount of data to human users. Information from a large number of small-scale robots can enable situation awareness, monitoring, and control.

A novel system designed for Kilobots is called augmented reality for Kilobots (ARK) [30]. It is able to communicate personalized location-and-state-based information to each robot and receive information about each robot's state. The Kilobots can sense additional information from a virtual environment in real-time. ARK is implemented in flexible base control software which allows users to define varying virtual environments within a single experiment using integrated overhead tracking

and control.

Mixed reality has also been used in robotics [21]. Mixed reality can enable robots to interact with physical and virtual objects in any number of physical or virtual environments. This enables the prototyping of algorithms on a combination of physical and virtual objects, including robots, sensors, and humans. Robots can be enhanced with additional virtual capabilities or can interact with humans without sharing physical space.

Chapter 3

Design and Implementation

3.1 System Workflow

A sketch for the implementation is shown in figure 3.1. The workflow of the

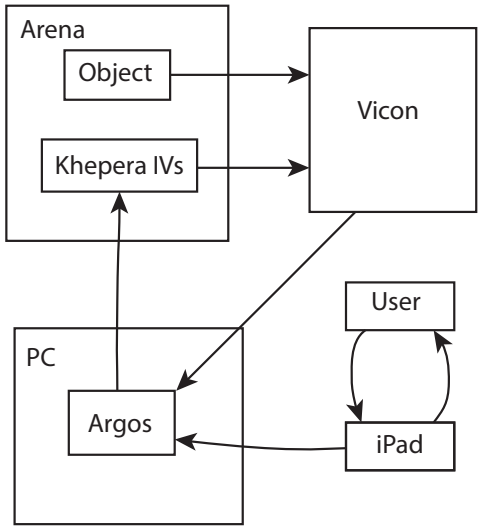


Figure 3.1: The implementation

whole system consists of three basic parts: a human-swarm interface, a centralized controller and multiple actuators. Augmented reality technology is used to provide

an intuitive interface for human interaction. No prior rigorous training is required for the user to understand and use the interface. The required pose of the object can be manipulated in the AR application using a virtual object to determine the goal position and orientation. This visualization of the goal will give the user a direct reference as to how the final position will ultimately look. This can be extended to the manipulation of multiple objects and forming complex construction using this framework. The detection capability of the AR resolves the problems of determining the world frame and target object detection. The platform of Vuforia is chosen from a set of tools. The AR human interface is designed as an iOS application running on an iPad due to its powerful processing ability, and an easy-to-use system, and versatile touchscreen.

The hybrid controller is a combination of one centralized controller governing the data exchange between the system and individual robot controllers. This controller is designed using a state machine control policy implemented in ARGoS, a multi-robot simulator. Compared with traditional simulators such as MATLAB, ARGoS is more efficient due to its unique multi-thread architecture specifically designed for swarm robot. The centralized controller is responsible for global position data and information exchange between all the system components. The robot actuators are chosen to adopt differential-drive carrying a decentralized controller to execute specific kinematic motions. We mainly focus on the software design and implementation and assemble existing hardware to utilize their functions. In order to achieve encapsulation and make all functions independent, different levels of controllers are divided into separate modules and called in assigned sequences. The centralized controller is contained only in ARGoS and runs prior to the decentralized controllers which are loaded in each robot entity and operate independently.

The communication between the above two modules is established using User

Datagram Protocol (UDP) for socket based networking on the iPad and the laptop. UDP does not need to establish the connection in advance and does not send check message from the server to client. This will decrease the workload for the tablet, which will improve the efficiency a lot.

3.2 Centralized Controller

As shown in figure 3.2, the centralized controller has the following 4 functional parts: state machine transfer, update information for the Khepera IV robots, communications with Vicon, check for failure and success.

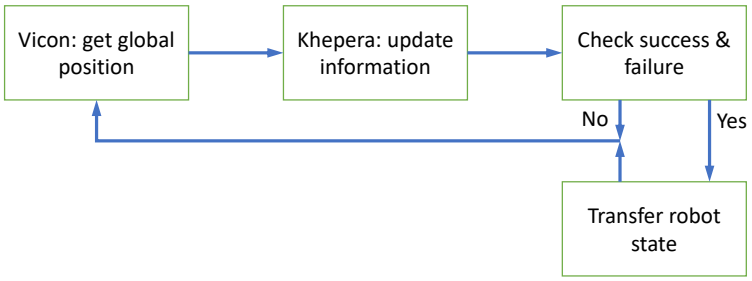


Figure 3.2: Four basic functions of the centralized controller.

After detecting the predefined object with iPad's onboard camera, the augmented reality module will send the object position and orientation. The centralized controller receiving the message will do the high-level task planning by generating deployment position, storing middle points of the trajectory and popping out them in sequence.

The primary responsibility of the centralized controller is to handle the synchronization of certain state transitions for all the robots. It controls the state machine in the robots by continuously monitoring the condition of the state transitions as it is able to acquire to global information such as the object and each individual robot positions. We use a group of a miniature assign-and-check state machine for the

whole procedure. In the control step of each task phase, the centralized controller first assigns the task by passing commands and information to each robot once, and checks the completion of the current phase repetitively. It will bring all robots to a state based on its current state and the special relative configuration the robot is in.

When the condition of state transition is fulfilled, the centralized controller will do the corresponding calculation that requires global information for the robots or order the robots to do the calculation that only needs local information. It should communicate between all entities including the localization system, iPad, and robots. In our framework, the centralized controller will run on a laptop. Each decentralized controlled robot has their own chip, runs program independently, and comes equipped with the ability to communicate directly with the laptop.

The workflow is shown in figure 3.3.

3.3 Augmented Reality

Augmented Reality (AR) is implemented as an interactive tablet application. Unlike conventional collaborative transport in which the goal position is well defined by using indicators, our swarm receives the goal position directly from the tablet. For augmented reality workflow, three parts are concatenated to achieve the following functions.

- Object detection.
- User touch operation.
- Message transfer and communication.

The algorithm in pseudo-code is shown in algorithm 1.

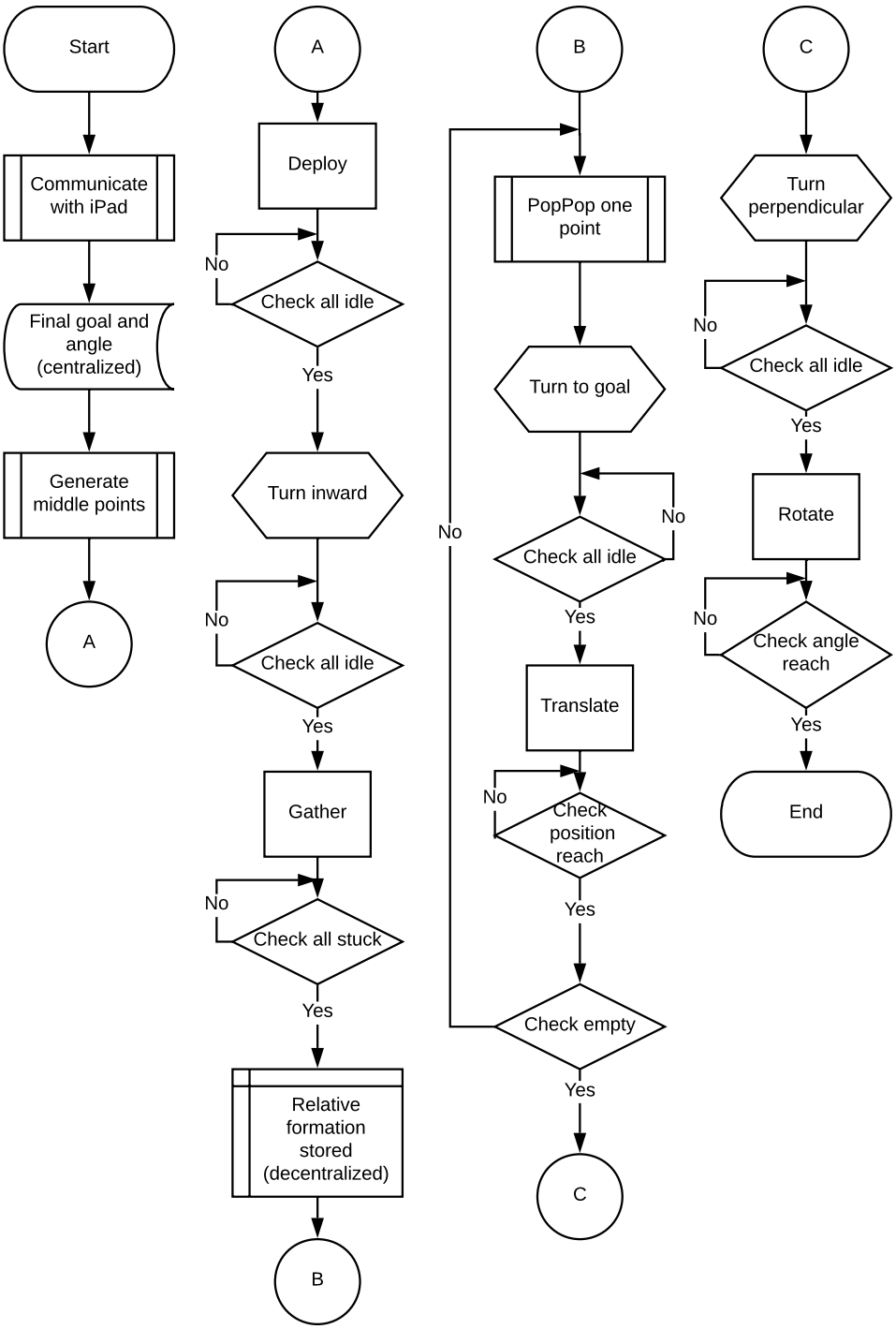


Figure 3.3: A specific workflow of the state machine and communication.

```
Function: Object Manipulation
Input: video_stream
Send : pose : (x, y, θ)
Init(database);
Detect(config);
while obj is detected do
    if #touch > 0 then
        p ← touch[0];
        switch p.phase do
            case Begin do obj ← RayCast(p, config);
            case Move do Translate(obj, p);
            case Release do Send(obj.pose);
        end
        if #touch==2 then
            q ← touch[1];
            switch q.phase do
                case Begin do α0 ← AngleBetween(p, q);
                case Move do
                    α ← AngleBetween(p, q);
                    Rotate(obj, α − α0);
                end
            end
        end
    end
end
```

Algorithm 1: Object manipulation method through touch screen. The rotation and translation are distinguished by the number of touches on the screen. The phase of every occuring touch has 3 types: **Begin**: the finger start to touch the screen; **Move**: the finger is skimming over the screen; **Release**: the finger leaves the screen.

3.3.1 Detection

Detection is responsible for determining the world origin and the real object. The configuration of the objects to be transported is first acquired by the AR application using a camera. The object to be transported is predefined in a database, which means its size and surface pattern is known to the application in advance. This information is used by the AR toolkit to realize the detection automatically.

The world origin is defined as a special predefined stationary target F_0 placed in the real arena. The planar positions and orientations of all the objects are represented by the pose of a frame, $F(x, y, \theta)$ w.r.t. F_0 attached to a fixed point on the project, which is the geometry center in our condition. Max of N objects to be transported could be identified and tracked simultaneously. N is based on the capability of the detection module used.

After the success of detection, the position and orientation of the object will be stored as $V(x_V, y_V, \theta_V$. A virtual object with the same pose information, $V = F$, is shown overlaid in real time on the target object once the target is detected. As the object is predefined, the virtual object's shape and scale can be exactly the same as the real object. Thus, the overlaid virtual object is not only a reminder to let the user know which objects can be manipulated, but also the goal poses in the next steps. The selected object ID will be sent to the centralized controller as well. Translation and rotation operations are enabled for the user to manipulate by hand.

3.3.2 Translation

A touch point $P_0 = (p_{x0}, p_{y0})$ in pixel coordinate will determine a ray starting from the camera's focus point through P_0. If this ray intersects with the virtual object V whose initial position equals to the real object, the intersecting point is

denoted $H(x_H, y_H, z_H)$, which can be considered the touch point in the real world coordinate. A reference virtual plane parallel to the ground (x-y plane of F_0) is created. $D = H - V$ is the offset between the touch point and the virtual object center. The motion of the finger, the new single touch point is denoted as P_0' at each time step, H' is the ray-cast point between the ray and the virtual plane. The new position of the virtual object is updated by $V' = D + H'$.

3.3.3 Rotation

The rotation operation is accomplished through the coordination of two fingers. The second finger is denoted as $P_1 = (p_{x1}, p_{y1})$ and form a vector with P_0 under pixel coordinate

$$u = P_1 - P_0$$

With the change of P_0 and P_1 at next time step, new vector u' is calculated in the same way.

$$\delta = angle(u') - angle(u)$$

The orientation of the virtual object around z axis θ is added by δ, and u is updated to u'. After all fingers are released, the updated V is saved and transmitted.

3.3.4 Implementation

Vuforia is one of the most widely used software platforms for augmented reality applications on hand-held and head-worn devices. For developers, it delivers a cross-platform solution for attaching digital content to physical objects and environments. Developers can add AR cameras and targets easily in Unity editor, and make use of the integrated physics engine, rendering function and event handler in unity. Vuforia is still under development, and we continuously update our system to make

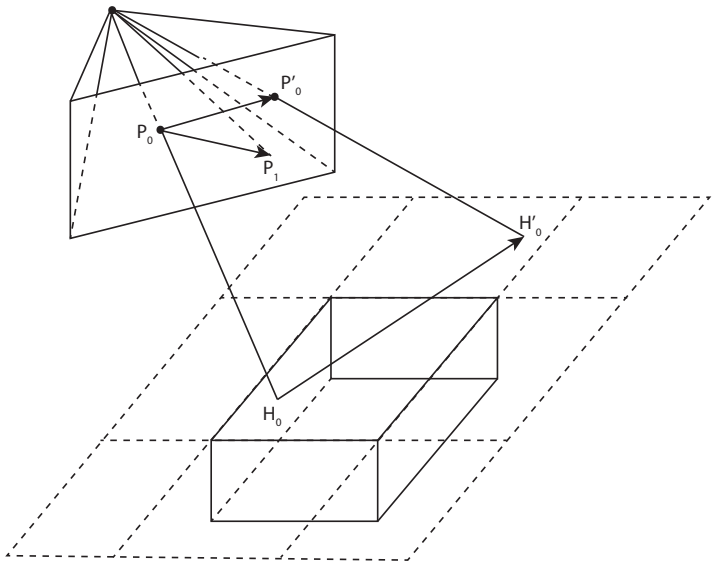


Figure 3.4: The ray cast and vectors

use of the newest features. The whole toolkit works as a plugin of Unity platform. This integration provides a number of benefits including a streamlined development workflow and a more seamless software update process. The Unity platform has a unified development process for Android and iOS platform.

A normal Unity project consists of scenes, assets, scripts and plugins. Assets contain the 3D models, images and textures and other resources. Scenes are where all the elements from assets are placed, combined and arranged in the unit of the game object. Scripts are C#/Javascript source file that can be bundled with a certain game object, which can be linked to other game objects as an object to call their member functions or implement own algorithms. Plugins are where code other than C#/Javascript is stored. Programs written in C can be maintained here and called if a certain interface routine is followed. This Unity project can be built and generate an Xcode project that will be built again and uploaded to the designated device. The AR application can run on iPad Air 2 as the augmented reality input device of our system.

(a) The Unity interface (b) The AR application

Figure 3.5: AR application related

3.4 Collaborative Transportation

3.4.1 Control Policy

As mentioned in chapter 2, three major ways of transportation are pushing, pulling and caging. We mainly focused on caging methods inspired by pushing proposed in [22]. In their work, the robots keep pushing only along the section of the object's perimeter that occludes their views of the goal, as shown in figure 3.6. Pushing

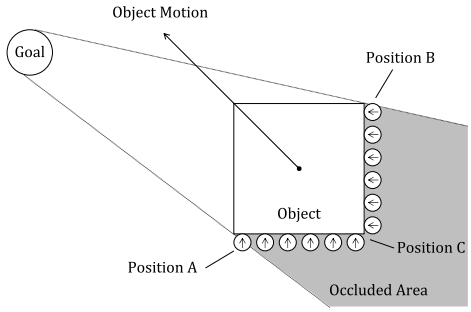


Figure 3.6: Illustration of how a swarm of robots can push a large object in a 2-D planar environment from [22]. We extend their method by deploying the robots around the object. In our case, only ones in the occluded area are effective, which does not require a continuous re-deployment.

needs narrower working space than caging, as the robots are only deployed behind the object with respect to the goal position. On the other hand, due to the limited

angle of affecting force the robots can apply to the object, it lacks a means to steer or rotate the object. Continual re-deployment makes the translation slow and full of unintentional rotation. The instability brought by the adjustment is not promising. There are also other transport methods that design special connectors attached to the object guaranteeing the robot's capability of applying force in any direction to the object, which, in an intuitive way, lacks the sense of generality.

Caging needs more workspace, by which means we need to dilate the object if obstacle avoidance is applied. Considering that an individual of a swarm or a multi-robot group is small, this dilation is acceptable. The caging method requires all the robots to form a closure around the object to avoid it escaping from the capture in every motion step. If the robots can be arranged closely to stick the outline, the object will move exactly as the robots. A strict motion control is required to keep the closure while conducting translation and rotation. The robot is differential drive non-holonomic robot. A corresponding control method is adopted based on what kind of tasks a single robot is executing.

3.4.2 Deployment

Position Generation

Playing the role of the information exchange hub, the centralized controller knows the position of the object and is suitable to manage the robot positions near the object. Currently, a centralized deployment generation method is used. Based on the number of the robots, a deploying arrangement with an even angular gap between the robots is generated. The side where robots occupy needs to form a closure ensuring that the object won't escape from the robots gap.

For pushing, the robots are deployed behind the object. The robots will de-

ploy with a gap just a few millimeters wider than the robot's radius. This can approximate an occluded area generated by a light source from an infinite distance.

All robots are deployed in a circle, as shown in figure 3.7, with a radius larger than the dimension of the object. The gap between robots should be equivalent as well as close enough to ensure a form of closure avoiding the object escaping from the grasp. A circular formation whose radius is larger than the object's dimension is generated.

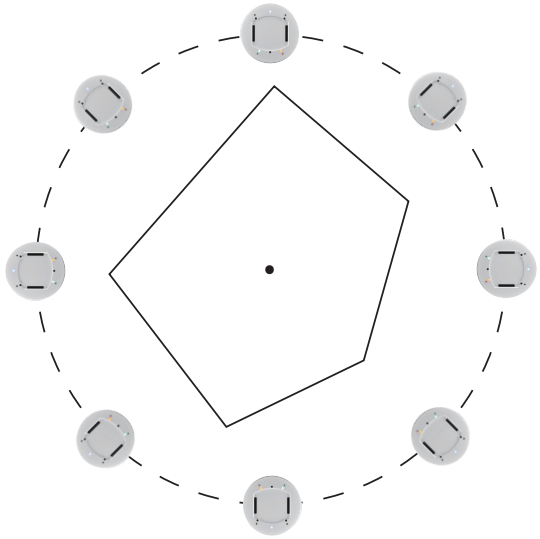


Figure 3.7: Robots deployed around the object

Position Assignment

As shown in algorithm 2 and figure 3.8, we will limit the maximum value of the shortest distances from each robot to all deployment positions. This is intended to minimize the waiting time between motions, as gathering requires a synchronization of all robots. To achieve this, we create a table listing all pairs of robot-to-position distance pairs of robots and positions. Then a select-and-delete operation will be executed from the largest minimum distance values of these pairs. The chosen positions for each robot will be sent to according robot through WiFi. One assignment

	P0	P1	P2	P3
R0	D00	D01	D02	D03
R1	D10	D11	D12	D13
R2	D20	D21	D22	D23
R3	D30	D31	D32	D33

Figure 3.8: The table of distance from each robot to each generated deploy position. First find the minimum value of each row ($D01, D13, D22, D30$), then find the maximum value among these minimum values $D22$. The index of this searched element is (i, j). Delete row i and column j, and apply the algorithm to the remained table until the table is empty.

Function: Positon Allocation
Input : The robot-position distance table D_k
Output: plan
$Init(min\_list)$;
while D_k is not empty **do**
  foreach row d_i in D_k **do**
    for d_{ij} in d_i **do if** $d_{ij} < min\_list(i)$ **then** $min\_list(i) \leftarrow d_{ij}$;
    ;
  end
  for $min\_list(i) \in min\_list$ **do if** $maxmin > min\_list(i)$ **then**
   $maxmin \leftarrow min\_list(i)$;
  ;
  $(i, j) \leftarrow FindRowColumnOf(D_k, maxmin)$ $delete(D(i :))$;
  $delete(D(: j))$;
  $plan.add(i, j)$;
end
return plan;

Algorithm 2: Deployment position allocation

is made by the pair:

$$(i, j) = \max_i(\min_j(|p_i - q_j|)), i, j \in 1, \ldots, N$$

means robot i will take position j. After the assignment, the ith row and jth column of the distances table will be removed. After the deployment position is generated, they will be assigned to each robot based on their pairwise relation.

Robot Kinematic

During the execution of deployment, each individual robot needs to go to the deployment position guided by their own controller. This controller needs to take the goal position, obstacles, and their own pose information into consideration. The ARGoS-Khepera package supply a velocity control interface, which will allow setting the linear velocity for the left and right wheel of differential drive.

Approaching The notion of the basic approaching motion is straightforward. If there is no obstacle, the robot will go to the goal directly by spinning around to face to the goal first and move forward with the speed controller. We adopt proportional control to realize this motion. The control input is the linear velocity v and angular velocity ω.

Assume the robot is at position p_r with an orientation θ_r. Denote the linear velocity of the center of the robot to be v, angular velocity to be ω. We assume the robot will only go in the forward direction due to the obstacle avoidance policy adopted. The assigned deployment position for robot p_r is p_d. The approach vector is

$$q = p_d - p_{rbt}$$

. Its angle is θ_q. The angle deviation is

$$\theta_g = \theta_q - \theta_{rbt}$$

and the distance deviation is $|q|$.

When implementing on hardware, there is a maximum speed for the differential drive. We also don't want the robot to move at a certain lower speed which cannot drive the robot when the battery is low or the ground condition is bad. As the sigmoid function has a constant value when the independent variable is high, and linear when it is low as shown in figure 3.9, we modify it a little to map the deviations to the control variables:

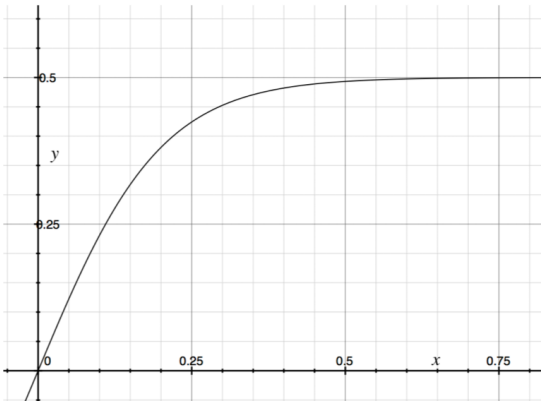


Figure 3.9: The curve of sigmoid function. The function is ninear when close to the origin, and constant when far.

$$v = 0.5\frac{v_{max}}{1 + e^{-k_1|q|}}$$

$$w = (\frac{1}{1 + e^{-k_2\theta_g}} - 0.5)\omega_{max}$$

where $v_{max} = r \times \omega_{max}$

Gain k is set to accelerate the converging speed into the converging area. Since the deployment position may be compacted and very close to the object, after it

reaches a neighbor area of the goal position, it will move directly without obstacle avoidance to get the convergence. The speed will overflow since the max condition for the linear velocity and angular velocity may happen simultaneously. We divide it by 2, which can be regarded that the max velocity is limited by half.

$$v_l = \frac{v - \omega r}{2}$$

$$v_r = \frac{v + \omega r}{2}$$

Obstacle Avoidance The obstacle avoidance method adopts proximity sensors, which in our case, are Khepera's ultrasonic sensors, that can return a value from 0 to 1. The larger the return value is, the closer an obstacle is. Therefore a vector from one sensor can represent the obstacle distance and orientation.

Compared with potential field or using continuous feedback to realize obstacle avoidance, our method is suitable for a Khepera to run in a decentralized way without global information. We use the front 5 sensors for obstacle avoidance. If a symmetric obstacle-avoidance policy is used (a turn without translation, or exactly the same awareness areas for the front 5 and back 5 sensors that can trigger symmetric motions), the robot may bounce between two nearby obstacles, shake its head forever, or move back and forth ended up getting stuck. As shown in figure 3.10, the middle three sensors have a lower threshold to trigger a hard turn, because obstacle in this area will definitely block the heading direction. Sensors on two sides are used to make a soft turn with a higher threshold, as the obstacle on these two sides.

The obstacle avoidance policy is depicted in algorithm 3.

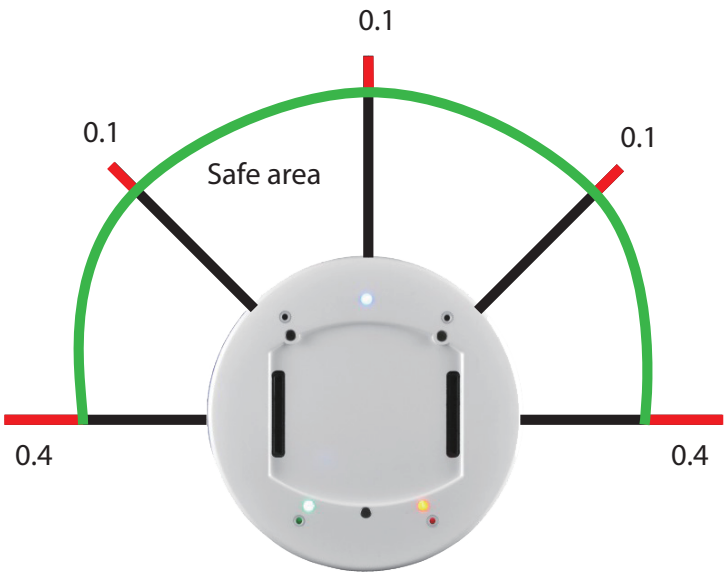


Figure 3.10: The sensor detecting the obstacle

Function: Obstacle Avoidance
Input : $SensorReading$
Output: Reach Goal
while not reach goal **do**
  if obstacle detected **then**
    approach goal
  if obstacle in front **then** $HardTurnParam(v_l, v_r)$;
  else if obstacle on side **then** $SoftTurnParam(v_l, v_r)$;
  else $ApproachParam(v_l, v_r)$;
  $SetVelocity(v_l, v_r)$;
end
return true;

Algorithm 3: Obstacle avoidance policy

Gathering After correctly deployed, the robots will conduct a "gathering" motion in order to grab the object. As we want to grab the object tightly, we are not using Proportional control where the feedback strength will decrease when the goal is about to be achieved.

Function: Gather(a, θ)
Input : Approach vector a, robot angle θ
Output: stuck flag
Initialize $counter = 0$;
while not all_stuck **do**
  if $(a.angle - \theta > turning\_threshold)$ **then**
    $(v_l, v_r) = (0, 0.3max\_speed)$;
  else if $(a.angle - \theta < -turning\_threshold)$ **then**
    $(v_l, v_r) = (0.3max\_speed, 0)$;
  else $(v_l, v_r) = (0.3max\_speed, 0.3max\_speed)$;
  if $(robot\_actual\_speed < 0.1max\_speed)$ **then** $counter+ = 1$;
  else $(counter = 0)$;
  if $(counter \geq 30)$ **then** $stuck = true$;
end
return true;

Algorithm 4: Gather motion

After all robots getting stuck, the relative position of each robot will be recorded as a parameter for the formation keep behavior in the pre-rotate calculation. This includes:

- The relative distance R_{ij} from the robot j to object i.
- The relative angle β_{ij}.
- Average relative position of all the robots G_i.

3.4.3 Translation

The translation here is the process when the center of the object is moving toward the goal position. For transportation, we use caging method. All robots form a

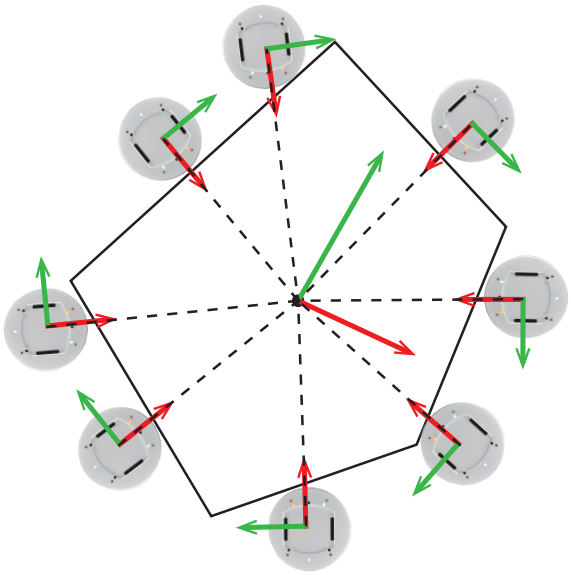


Figure 3.11: The robots grab the object. The black dash line is used to calculate the desirable relative position w.r.t. F_i frame

close enclosure strictly and move in the same direction in order to keep the object within the shape.

Caging is actually an omnidirectional pushing of sorts. The robots-formed shape will grab the object tightly similarly to a gripper, so the object will move with the robot shape. But since the communication latency and that the kinematic of every robot cannot be sensed accurately, an ideal firm shape cannot be achieved. When controlling the shape, we also need to find a balance between efficiency and shape-keeping. We cannot set the trajectory as depending on time because the object is what we want to keep shape around. All robots tend to move slowly to keep the shape, but they need to move forward.

In [36] a stable trajectory control method is proposed. The method is easy to implement and very effective. It only requires the error of orientation and planar position, which is ideal for us since our reference is the object, not a time-based

trajectory. We want to map the relative position between the robot and the object, and that between the object and the target position.

For a differential drive robot, assume the robot's pose is represented as a triple:

$$\mathbf{p}(t) = \begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \end{pmatrix}$$

The $(x(t), y(t))$ is the planar position of the robot's center in the world coordinate frame F_0 at time t. $\theta(t)$ is the robot's front orientation from the x-axis in the range of $(-\pi, +\pi]$. The front of the robot is the direction the robot goes toward when both wheels are set to equivalent positive speed, in other words, is the direction of linear velocity v of the robot center.

$$\theta(t) = \tan^{(-1)}(\frac{\dot{y}(t)}{\dot{x}(t)})$$

Another kinematic of the robot center is its angular velocity ω. The robot's Jacobian matrix can be written as follow:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \dot{p} = Jq = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix}$$

In our scenario, the desired trajectory for the robot p_d is not determined by t. Instead, it is determined by the object's current position p_o. In the gathering motion, we calculate the relative distance r and angle β under the object's coordinate frame F.

Assume the direction of the object is θ_o.

$$\begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\beta & \sin\beta & x_o \\ -\sin\beta & \cos\beta & y_o \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R\cos\beta \\ R\sin\beta \\ 1 \end{pmatrix}$$

and θ_r equals to the angle of $p_g - p_o$ where p_g is the position of the goal in F_0.

The position error from the robot's position is calculated by:

$$p_e = \begin{pmatrix} x_e \\ y_e \\ \theta_e \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} (p_d - p)$$

When the robot is at the desirable position p_d, it should move in planned velocity

$$q_d = \begin{pmatrix} v_d \\ \omega_d \end{pmatrix}$$

the so-called approaching vector is $\mathbf{u}(t) = (\mathbf{p_t} - \mathbf{p_o}(t))$. The transfer matrix from the world coordinate to the object

The initial relative position of robot w.r.t. the object is $\vec{v} = T^(-1)(\vec{x_o} - \vec{x_r}, y_o - y_r)$.

Our control method is based on this error and the desirable velocities.

$$q = \begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} v(p_e, q_d) \\ \omega(p_e, q_d) \end{pmatrix}$$

In [36], a stable control method is proposed:

$$q = \begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} v(p_e, q_d) \\ \omega(p_e, q_d) \end{pmatrix} = \begin{pmatrix} v_d \cos(\theta_e) + K_x x_e \\ \omega_d + v_d(K_y y_e + K_\theta \sin\theta_e) \end{pmatrix}$$

Under a critical damping scenario, $K_y = 6.4 \times 10^{-3}$/cm, $K_x = 0.7$/sec and $K_\theta = 0.16$/cm.

After get the control variable (v, ω), the wheel speed an be set

$$\begin{pmatrix} v_l \\ v_r \end{pmatrix} = \begin{pmatrix} \frac{v-\omega r}{R} \\ \frac{v+\omega r}{R} \end{pmatrix}$$

3.4.4 Rotation

A rotation process will be conducted after the object reaches the final target position following a set of re-deploy and gathering. An ideal rotation without additional translation for the object should be executed. The stable control method for the differential drive robots proposed in [36] also works for circular path control.

Pre-rotation phase The control policy we adopts requires a small initial error in both orientation, θ and position, (x, y). Same as translation, the rotation requires the robots to face the direction tangent to the perimeter of the circle. After the robots are deployed and gather to grab the object tightly, a pre-rotation calculation occurs to find the radius of the circular motion.

Different from translation, rotation may fail to conduct when all the robots can't touch the object instantly when start moving figure 3.12. During the control

calculation, we set the relative angle

$$\beta_j = \arctan(\frac{y_j - y_o}{x_j - x_o}) \pm 90^\circ$$

The sign is positive if the rotation counterclockwise negative if clockwise.

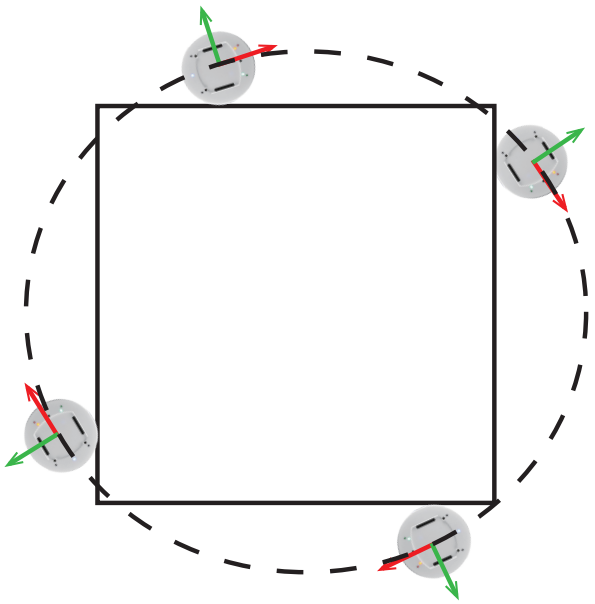


Figure 3.12: All the robots are not able to rotate the object since their coming segment of planned trajectory cannot guarantee sustained contact with the object. If the current position is desirable, due to the control strength, all robots remain static.

3.4.5 Software and Hardware

ARGoS

ARGoS is a specialized multi-physics robot simulator. It is written in C++ and can be utilized in both C++ and Buzz. It can simulate large-scale swarms of robots of any kind efficiently and be customized easily by adding new plug-ins. Before ARGoS, a multi-robot system can only be simulated in MATLAB and the user needs

to design controllers and model the physical environment from the very beginning. In ARGoS, controllers can be implemented and deployed to all robots identically. The localization and pose information of each entity can be accessed easily by using the pointers and handles. The ARGoS plays as a bridge between Vicon, iPad and Kheperas.

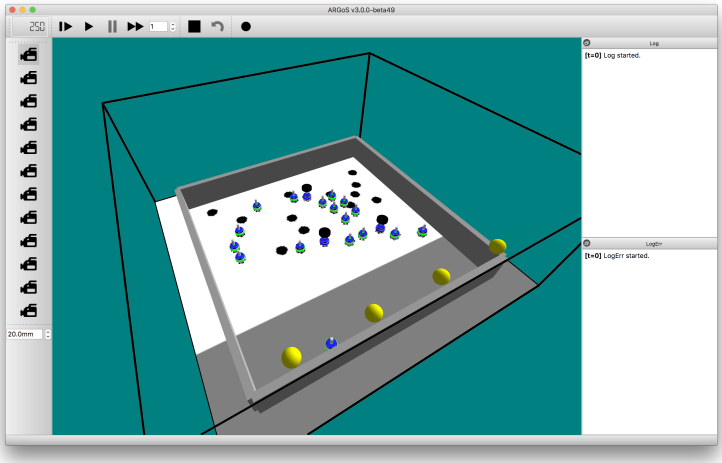


Figure 3.13: The window of ARGoS simulator

An ARGoS project, basically a CMake project, consists of three folders: experiments, controllers, and loop_functions. The conceptual diagram is shown in figure 3.14. The experiments folder contains XML files with extension **.ARGoS** that configure the information needed for an experiment. Controllers are C++ program compiled in one or more libraries to be called by the ARGoS simulator when running. The virtual function "ControlStep()" will be executed in every time step, manipulates the calculation, sensor readings and drivers on the robot. Loop functions or other C++ code manipulating the work (excluding robots), such as visualization, manipulating abstract class in the physical world, or communication happens in pre-step or post-step functions. The UDP receiver is implemented in loop functions.

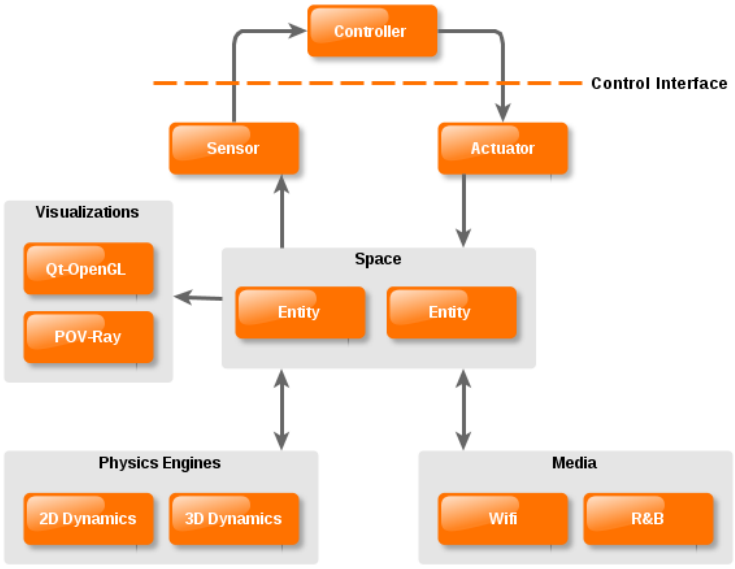


Figure 3.14: The structure of ARGoS simulator. *loop_fucntion* folder contains the program of the control interface between controller and sensor//actuator and the visualizations block. Controller folder implements Controller block. Experiments folder set parameters for each trial. ARGoS core manipulates other background process.

Khepera

The Khepera IV (figure 3.15) is a compact robot designed for any indoor lab application (table, lab floor) such as navigation, swarm, artificial intelligence, computation, demonstration, etc. It requires a little space to operate, even in a swarm. The use of a KB-250 bus allows users to stack many different extensions on the top of the robot in mere seconds, providing an unbeatable modularity.

Khepera IV has 12 infrared sensors and 5 ultrasonic sensors. The range of the IR sensor is 2 to 250 mm and that of the ultrasonic sensors is 2 to 200cm. For obstacle avoidance during the deployment phase, 5 infrared sensors arranged evenly in the front of the robot are used as proximity sensors to detect obstacles. Wireless communication is used to exchange information with the centralized controller. Due to the lack of local communication with other Khepera or detect certain signal, the

(a) The front view of Khepera IV.

(b) The bottom of Khepera IV.

Figure 3.15: Pictures of Khepera IV

Khepera IV is not able to perform in an isolated environment as a pure centralized swarm.

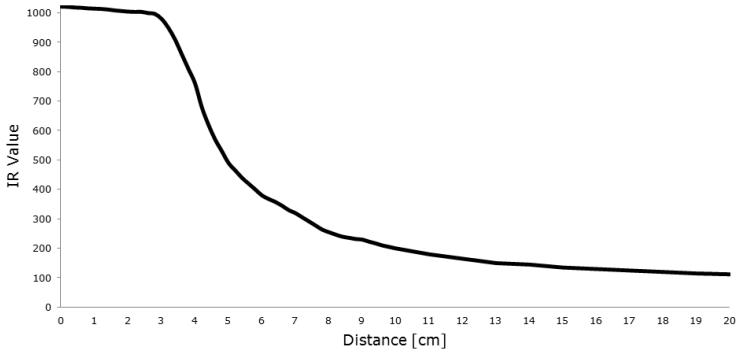
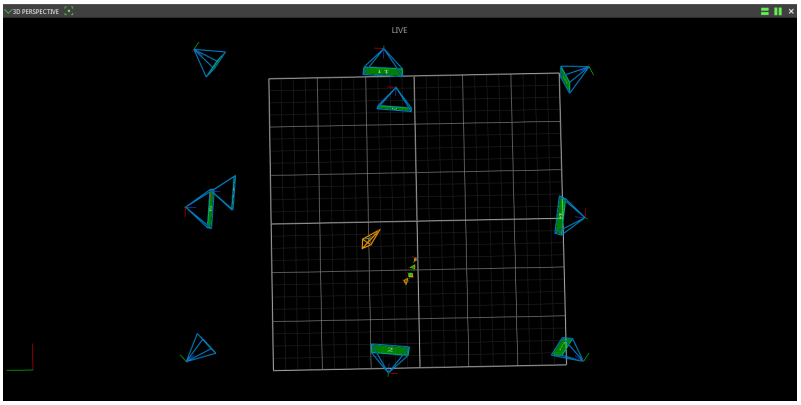


Figure 3.16: The reading value vs. sense range

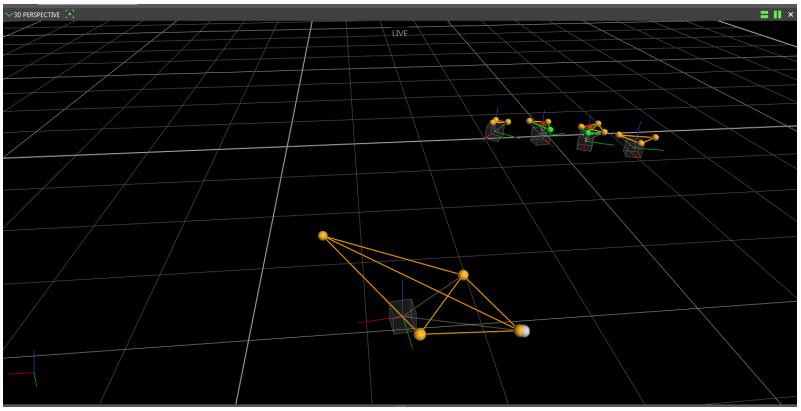
3.5 Other Components

3.5.1 Global Tracking System

Vicon system is an advanced motion capture system using optical cameras. We adopt Vicon Vero with on-board sensors that monitor camera position and temperature to ensure optimal performance. Aside from the main camera, the Vicon



(a) A broad view of the Vicon system



(b) A closes view to Vicon

Figure 3.17: Vicon Screenshot

camera has a group of small infrared transmitters. In our experiment, the precise position and pose of the Khepera IVs and the object to be manipulated are localized by the Vicon system. At least 4 markers are affixed to each distinguishable entity to guarantee reliable localization.

Thanks to the work of the ARGoS-Vicon package, the configuration data from the Vicon system can be received and visualized by ARGoS in real time. All dimensions and pose information of robots and objects are either defined or detected in Vicon environment rather than in the .argos file. Other parts of the code are independent of these pre-configurations.

3.5.2 Communication

Once all finger contacts are released, the indicated planar position and orientation (x, y, θ) of the object will be packed into a package and sent through the socket. It should be noted that the pose should be transformed from the coordinate system of the object to that of the robot controller. To achieve the maximum sending efficiency, we only send a given position packet once. This requires the receiver to listen to for messages continuously.

We use UDP protocol instead of TCP/IP to send data from the iPad to the centralized controller. TCP is Transmission Control Protocol, which is more reliable as the sender requires the receiver to give feedback and error check to make sure that the message is properly delivered. UDP protocol, the User Datagram Protocol, is just a datagram sent to the target IP address. There is no guarantee that the message is delivered, but the communication is more efficient.

Because there is no guarantee of packet delivery, the socket using UDP must already be active and working before the very first message is sent. All received messages are put in the buffer without being flushed by the operating system kernel. As such, it is unpractical to know which is the newest message. Consequently, if the packets are continuously sent without time stamps until the buffer overflowed, the buffer would simply drop the incoming messages of unknown sequence. In order to mitigate this risk, we only send the position from the iPad when the user conducts a set of touch motions to ensure the receiver buffer is clear.

As mentioned before, the UDP listener will be activated before the first message is sent and need to continue listening thereafter. The recvfrom() function will be stuck and wait if the socket buffer is empty, which will block the main single thread. We use multi-thread programming to deal with this problem. The UDP server thread starts when the whole system is initialized, and remain in the loop of receiving

the message. When the buffer is empty, this message is just stuck in one round of the loop. When the message is received, the message is recorded and read by the centralized controller. Then the child thread will go to the next step of the loop and listen to the message again. Since the user won't manipulate the object faster than the process speed of a single loop containing the recvfrom() function, the buffer will never overflow and all incoming messages will be received under normal conditions. The child thread will jump out the loop and destructed when the ARGoS simulator is closed.

3.5.3 Failure Check

As the target object may have various physical properties, an improperly configured robot swarm (with a deployment shape for example), might fail to keep formation during the transportation process. Similarly, the swarm may be compromised by the sudden unexpected rotation of the object or slipping on one side of the object. As the deployment is dependent on the object, one or more re-deployments can potentially restart the whole process and fix the issue under certain scenarios.

Loose shape during translation Due to the loosening of robot swarm shape, the object may rotate unexpectedly, apply a large force to the robots, and escape from the enclosure during the translation process. The "loose" check will be based on a large value of x_e and y_e which will cause a robot to fully halt.

Deviation during rotation After the object has been grabbed, the average position of all the robots will be calculated. If this average position deviates too much from the original one, we assume that the formation has been broken.

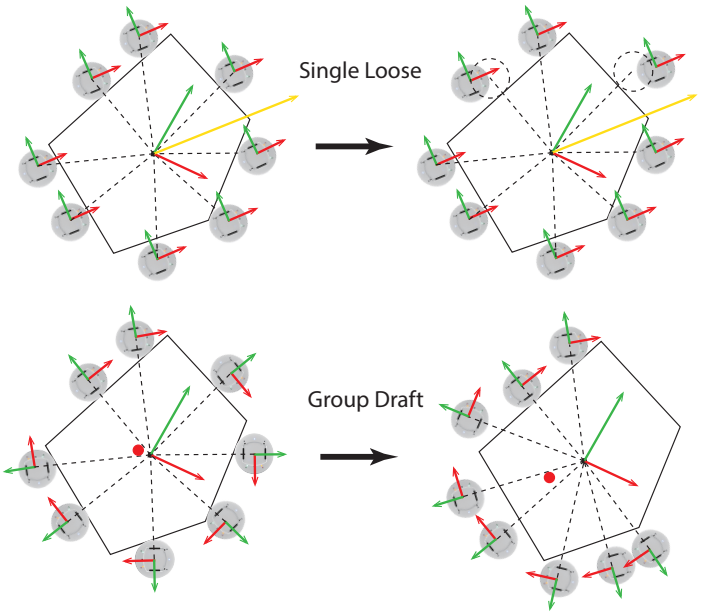


Figure 3.18: The robots grab the object. The black dash line is used to calculate the desirable relative position w.r.t. F frame

Chapter 4

Experimental Evaluation

4.1 Experimental Design

One part of our experiment included trials using real robots. The target is dictated by the user through the AR application or by the program, and robots are running in the real scenario supervised by Vicon. In these trials, the localization of the Vicon system is treated as the ground truth. The other part of the experiment is running purely on the laptop by adding noise to the sensor and the localization information, actuator speed and sensor reading.

The choice of key performance metrics is flexible for human-swarm interaction. Different considerations emphasize different focuses on metrics ranging from the object placement precision to the efficiency of transport. Hereafter we detail some general specifications relevant to the whole system.

4.1.1 AR application performance

In the very beginning of our experiment, the localization error of the real object is measured. Each time the world center (Vuforia Marker) and the object are placed,

the origin of Vicon will be set by placing an infrared stick on the origin marker and taking 10 distinct measurements.

Experiments on dragging were conducted in the following manner. The virtual object was initialized at the origin. The real object was placed at a random position and its Vicon position was recorded. The user was asked to drag the virtual object to a specified configuration, and the AR localization was recorded and compared to the above Vicon readings.

Figure 4.1: The object we design and use.

4.1.2 Transport metric

The transportation performance is our main metric of interest. Once the precise position of the real and virtual objects are recorded accurately, the efficiency of the task depends on how accurately and quickly the object is transported. The Vicon system acts as the true data source of our measurement, which passes data to ARGoS simulator where everything can be processed.

The following terms, covering the speed, efficiency, and precision of the transport, were measured. Trials using different objects and different numbers of robots

(swarms of four or more) were conducted.

- **Precision**: The error of the final position and angle of the object.
- **Translation efficiency**: The total actual distance traveled by the object ratio to the planned path length.
- **Rotation efficiency**: The total actual rotating angle of the object ratio to the planned rotating angle of the object.
- **Rotation speed**: The average rotating speed during the rotation phase ratio to the desired angular speed.
- **Translation speed**: The average translating speed during the translation phase ratio to the desired linear speed.

The error of the final position is calculated by $|p_g - p_f|$, where p_g is the goal position (x_g, y_g) and p_f is the final position of the object. The angular error is calculated as tthe absolute value $|\theta_g - \theta_f|$.

The path the robot needs to drive the object over is the planned path, which is a feasible and continuous path. Even the smallest extra additional distance is regarded as waste. So the translation efficiency will be measured as the ratio of ideal path length to the actual path length.

In a normal translation, the object angle may change due to the translational kinematics involved. But in our application, the translation method we used does not include any angle adjustment. Thus, any angular change is undesirable, though sometimes it may contribute to the rotation. Therefore, the rotation efficiency will be calculated by the following equation:

$$\frac{1}{2} \cdot \frac{|\theta_p| + |\theta_c|}{\int \delta\theta}$$

As the planned angle is the effective part as in the travel distance part. The rotational part also compensates for the unwanted angular change in the translation phase, which cannot be regarded as waste. The extra useless rotation should be minimized by decreasing the total rotational change instead of decreasing the rotation angle.

Consider a scenario where the object reaches the goal angle after the translation by accident. The result is what we wanted, but the process is uncontrollable, so the efficiency should not be considered superior.

During the translation and rotation phases, we designate a desired speed for the robots. As the robots should keep a relatively static position in an ideal scenario, the object should just move as the robot's speed. Any delay or slowing down is regarded as waste. So the speed will be normalized by the ideal translational and rotational speeds under different desirable configuration.

4.1.3 Robot metric

The performance of the robots themselves will also be considered. Prior to the transport, we consider the efficiency of the deployment phase and the balance of the robots. The following items were measured for each individual:

- Energy balance: the ratio of min and max total travel distance was recorded.
- Deployment efficiency: Average velocity during the displacement to the position.
- Robot use rate: The average robot travel distance during translation ratio to the object translation distance.

The first step was to measure the energy consumption balance between robots. Too much consumption of a certain robot is undesirable since imbalance can cause

accelerated electromechanical degradation of individual robots. The speed we calculate which is based on displacement is the speed of approach to the target, not the path length divided by time. The straighter the path is (which means the influence of the obstacles are diminished) it finds, the more efficiently it has operated.

During the translation process, the ideal condition is that every robot should move precisely with the object. So the average travel distance is calculated to compare to the travel distance for the robots.

4.2 Results

The performance of the system is given in this section. Separate tests were conducted for the AR application, hybrid controller, and the integration of these two parts. The result are shown in figure 4.2.

4.2.1 AR precision

The box used in this part is the small box with WPI logo.

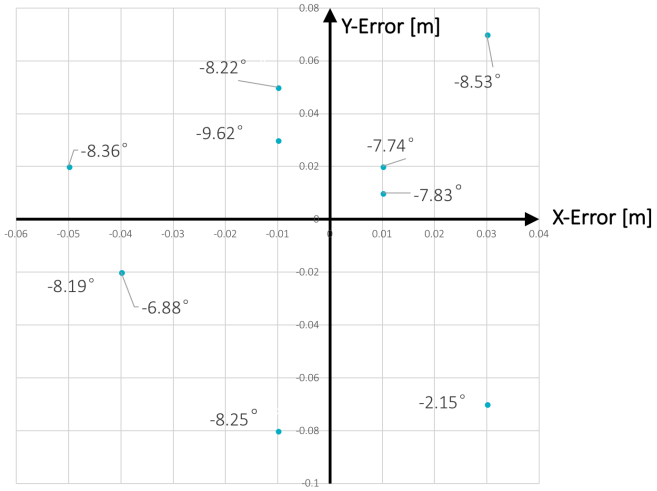


Figure 4.2: The precision of AR application. The degrees near each point is the respective degree error.

AR error					
Metric/Rbt#	4	5	6	7	8
2D Error(cm)	0.8	2.1	0.6	0.1	0.4
Angle error(°)	0.7	0.7	0.7	1.0	0.8
Distance Efficiency	97.5%	98.1%	99.4%	98.7%	99.2%
Degree Efficiency	17.1%	97.9%	97.6%	91.4%	71.4%
Translation speed	71.3%	99.8%	99.7%	99.8%	97.1%
Rotation speed	78.1%	96.5%	94.5%	94.6%	90.0%
Deploy speed(cm/s)	4.24	4.36	4.42	4.59	3.97
Energy balance	0.92	0.87	0.85	0.86	0.84
Robot Efficiency	64.9%	81.9%	81.9%	82.1%	78.6%

Table 4.1: Result from experiments using simulator, from $(0, 0, 0)$ to $(0, 0, \pi)$

Real Robot Performance					
Metric/Rbt#	4	5	6	7	8
2D Error(cm)	4.6	5.0	8.6	9.4	3.5
Angle error(°)	1.40	0.93	1.15	1.4	1.2
Distance Efficiency	91.6%	81.3%	70.5%	66.3%	78.5%
Degree Efficiency	85.7%	87.2%	83.1%	85.2%	82.0%
Translation speed	46.1%	51.0%	45.6%	30.9%	110.0%
Rotation speed	88.5%	89.7%	99.2%	93.6%	97.0%
Deploy speed(cm/s)	4.70	4.12	3.39	3.07	3.32
Energy balance	0.75	0.69	0.67	0.65	0.54
Robot Efficiency	51.4%	58.4%	53.1%	46.1%	58.5%

Table 4.2: Result from experiments using real robots

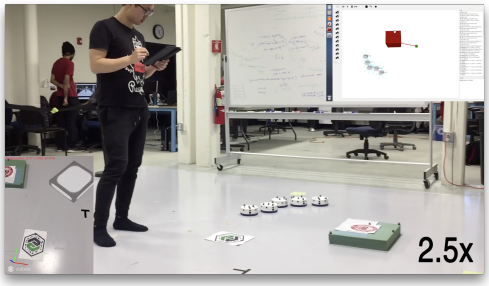
4.2.2 Trials in ARGoS

Several trials are run in ARGoS with noise added. The noise is uniformly distributed. For proximity sensors, the noise range is $(-0.1, 0.1)$. For the wheel velocity, the noise is $\pm$1cm/s. The result is shown in table 4.1.

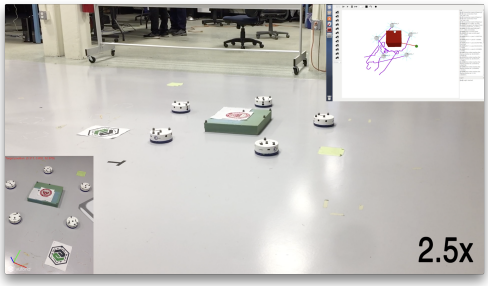
4.2.3 Trials with real robot

Trials using iPad as the user input are shown in table 4.2.

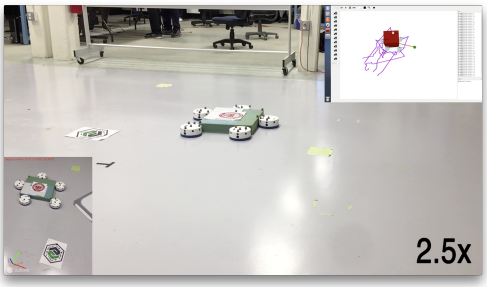
Experiments specially designed for the collaborative transport were conducted.



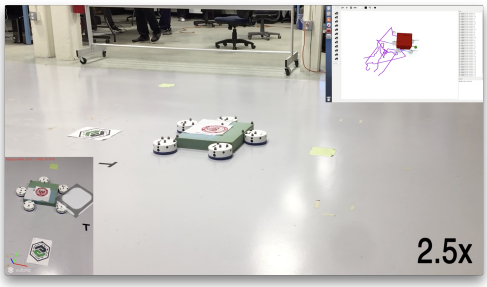
(a) $t = 0s$, initial configuration



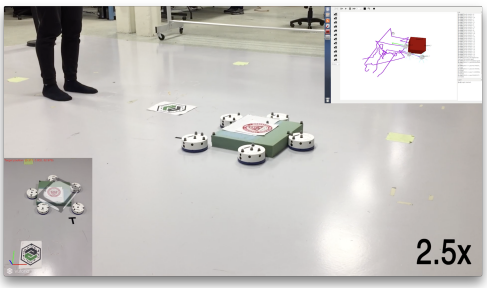
(b) $t = 53s$, robots are deployed.



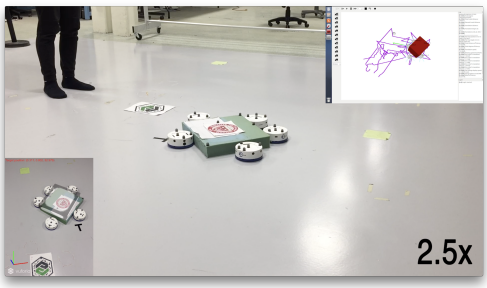
(c) $t = 67s$, the robots grab the object.



(d) $t = 73s$, translating.



(e) $t = 80s$, reaches the final goal.



(f) $t = 109s$, rotation finishes.

Figure 4.3: Human swarm interaction with 5 robots transporting the object.

W-shape performance		
Metric/Trial#	1	2
2D Error(cm)	10.0	7.1
Angle error(°)	1.28	1.43
Distance Efficiency	85.2%	79.6%
Degree Efficiency	54.0%	54.9%
Translation speed	55.7%	54.7%
Rotation speed	93.6%	98.4%
Deploy speed(cm/s)	4.84	4.82
Energy balance	0.91	0.93
Robot Efficiency	69.0%	67.7%

Table 4.3: Mesurement from 2 trials for conducting. Use 5 robots.

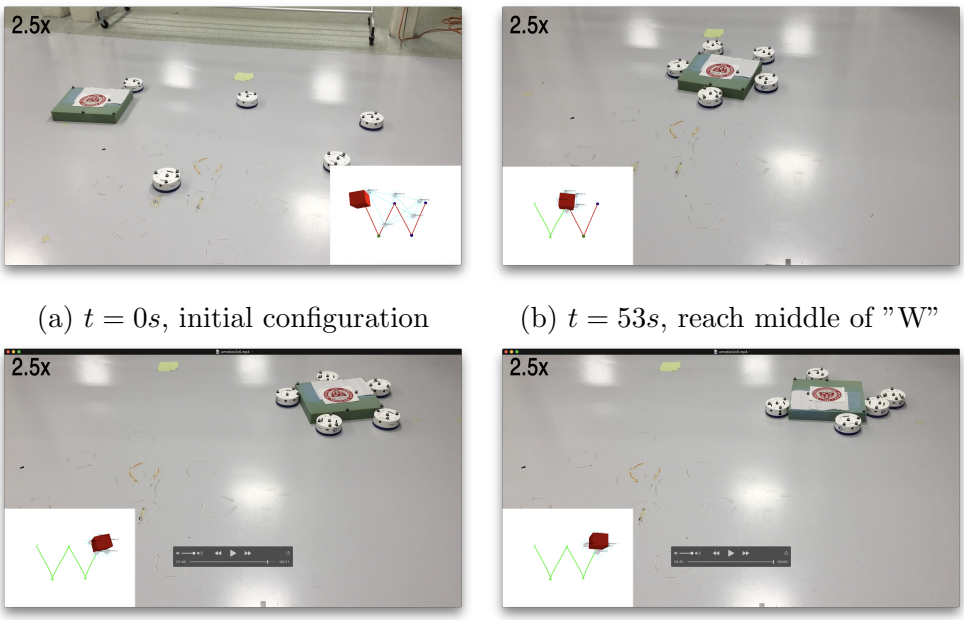
The object and final goal are far away from the origin marker, which is hard for the iPad to detect or display. The transport range exceeds the iPad view angle. This tests the capability of the transport system to endure tasks of long duration.

We chose to make the object move in a W-shape for "WPI".

4.3 Analysis

It can be calculated from figure 4.2 that the average error of x is around 1 cm, y is less than 1 cm. Both are acceptable. The average angular error was -7.58 degrees. These errors are identified as systematic errors and will be added to the original value before being sent to the ARGoS. These systematic error comes from how the image is attached to the object which has reflective markers on it, since the detection mechanism is based on image detection for the AR app, but reflective markers for Vicon. After the compensation, the AR part can correctly reflect the user's intention on the target position.

Although the resulted transport precision is considered to be enough for the current stage of basic object placement, the transport range is limited due to the dramatic drop in precision of detection when the object is far away. This is con-



(a) $t = 0s$, initial configuration

(b) $t = 53s$, reach middle of "W"

(c) $t = 90s$, position reached

(d) $t = 111s$, angle reached

Figure 4.4: Collaborative transport through "W" shape

strained by the quality of the on-board camera and the iPad's internal odometer.

In table 4.1, it can be seen that the 2D error and angular error are very low and nearly ideal. The distance and degree efficiencies are also high at a level over 90% most of the time. The trial with 4 robots has low degree efficiency due to the spiral motion of all the robots during the translation phase. Such extensive useless rotation drug the rotating efficiency down significantly, and also negatively impacted the translational speeds.

When the above results are compared with table 4.2, the performance drops. For the 2D error, apart from the localization error from Vicon, the remaining error mainly comes from the gathering motion before the rotating motion, which disturbs the object due to the time interval between different robots contacting with the object. The decrease of the translational speed is caused by that more re-deployment motions and loose conditions were triggered. Finally, as the contact face between the

object and the floor was uneven, the real-world object movements were less stable than that of its simulated counterpart. It may rotate or get stuck during translation, affecting the robot's trajectory. This also resulted in greater re-deployment correction, further decreasing to the robot's efficiency.

From both screen shots of the video, figure 4.3 and figure 4.4, it can be seen that the object can be transported to the correct position and angle. Especially in figure 4.3, it is clear that the object can align with the virtual object observed through iPad's camera within tolerable error.

Chapter 5

Conclusion and Future Work

In our work, we successfully implemented a novel system for human-swarm interaction for the task of collaborative transport. We designed and implemented the augmented reality application on iPad for human users to select the predefined target object, and overlaid a virtual object on it for moving and rotating. The centralized controller used can receive the user-specified goal through UDP communication, and translate it to the correct control commands for the robots. It can then synchronize all robots by checking their states and exchange necessary information with the robots. Each robot can receive signals and execute the tasks designated to them. Deployment, pushing, caging, and rotation have been implemented.

Future work could include certain manipulation methods for multiple objects to be transported to form a particular architecture. Multiple objects can exist in the same environment, and robots should organize them in a desired configuration determined by the user. To achieve this kind of task, a robot swarm channel should be formed wherein precisely-angled pushing along the desired motion vector is the only possible movement. The future centralized controller should be implemented in the AR application or even be removed. This leads to a possible research topic that

the whole system to be made as decentralized as possible, or make the interaction terminal to be a special member (a leader) of the swarm. A decentralized estimator can be adopted required in this decentralized architecture. Local communications should be adopted without the need to use an existing network. The current angle of view of the tablet is also limited, which constrains the translating range of the object. A drone with a camera or some other devices with a broader view angle can be used to transmit the video back to the tablet for interaction. Simpler robots such us Kilobots instead the expensive and powerful Khepera can be used.

Bibliography

[1] Akinbiyi, T., Reiley, C., Saha, S., Burschka, D., Hasser, C., Yuh, D., and Okamura, A. Dynamic Augmented Reality for Sensory Substitution in Robot-Assisted Surgical Systems. In *Proceedings of the 28th IEEE* (New York City, 2006), p. 4.

[2] Alkilabi, M. H. M., Narayan, A., and Tuci, E. Cooperative object transport with a swarm of e-puck robots: robustness and scalability of evolved collective strategies. *Swarm Intelligence 11*, 3-4 (Dec. 2017), 185–209.

[3] Beni, G. From Swarm Intelligence to Swarm Robotics. In *Swarm Robotics* (July 2004), Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 1–9.

[4] Bonabeau, E., Dorigo, M., and Theraulaz, G. *Swarm intelligence: from natural to artificial isystems*. Oxford University Press, New York, 1999.

[5] Bowley, S. J., and Merrick, K. A Breadcrumbs Model for Controlling an Intrinsically Motivated Swarm Using a Handheld Device. In *AI 2017: Advances in Artificial Intelligence*, W. Peng, D. Alahakoon, and X. Li, Eds., vol. 10400. Springer International Publishing, Cham, 2017, pp. 157–168.

[6] Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence 7*, 1 (Mar. 2013), 1–41.

[7] Caroline E. Harriott, Adriane E. Seiffert, Sean T. Hayes, and Julie A. Adams. Biologically-Inspired Human-Swarm Interaction Metrics. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting 58*, 1 (Sept. 2014), 1471–1475.

[8] Collett, T. H. J., and MacDonald, B. A. An augmented reality debugging system for mobile robot software engineers. ACM Press, p. 49.

[9] Collett, T. H. J., and Macdonald, B. A. An augmented reality debugging system for mobile robot software engineers. *JOURNAL OF SOFTWARE ENGINEERING IN ROBOTICS 1*, 1 (2010), 18–32.

[10] Cortes, J., Martinez, S., and Bullo, F. Robust Rendezvous for Mobile Autonomous Agents via Proximity Graphs in Arbitrary Dimensions. *IEEE Transactions on Automatic Control 51*, 8 (Aug. 2006), 1289–1298.

[11] Czaczkes, T. J., Gruter, C., Jones, S. M., and Ratnieks, F. L. W. Synergy between social and private information increases foraging efficiency in ants. *Biology Letters 7*, 4 (Aug. 2011), 521–524.

[12] Dai, Y., Kim, Y.-G., Lee, D.-H., and Lee, S. Symmetric caging formation for convex polygon object transportation by multiple mobile robots. In *Advanced Intelligent Mechatronics (AIM), 2015 IEEE International Conference on* (2015), IEEE, pp. 595–600.

[13] Daily, M., Cho, Y., Martin, K., and Payton, D. World embedded interfaces for human-robot interaction. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on* (2003), IEEE, pp. 6–pp.

[14] Das, A., Fierro, R., Kumar, V., Ostrowski, J., Spletzer, J., and Taylor, C. A vision-based formation control framework. *IEEE Transactions on Robotics and Automation 18*, 5 (Oct. 2002), 813–825.

[15] Dorigo, M., Trianni, V., ahin, E., Gro\s s, R., Labella, T. H., Baldassarre, G., Nolfi, S., Deneubourg, J.-L., Mondada, F., Floreano, D., and others. Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots 17*, 2 (2004), 223–245.

[16] Fujii, M., Inamura, W., Murakami, H., Tanaka, K., and Kosuge, K. Cooperative control of multiple mobile robots transporting a single object with loose handling. In *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on* (2007), IEEE, pp. 816–822.

[17] Grob, R., Mondada, F., and Dorigo, M. Transport of an object by six pre-attached robots interacting via physical links. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on* (2006), IEEE, pp. 1317–1323.

[18] Gross, R., and Dorigo, M. Towards group transport by swarms of robots. *International Journal of Bio-Inspired Computation 1*, 1-2 (2009), 1–13.

[19] Habini, G. Distributed Centroid Estimation and Motion Controllers for Collective Transport by Multi-Robot Systems.

[20] Hassanien, A. E., and Emary, E. *Swarm intelligence: principles, advances, and applications*. CRC Press, 2016.

[21] Hoenig, W., Milanes, C., Scaria, L., Phan, T., Bolas, M., and Ayanian, N. Mixed reality for robotics. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on* (2015), IEEE, pp. 5382–5387.

[22] Jianing Chen, Melvin Gauci, Wei Li, Andreas Kolling, and Roderich Gro. Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots. *IEEE Transactions on Robotics 31*, 2 (Apr. 2015), 307–321.

[23] Jorge M. Soares, Iaki Navarro, and Alcherio Martinoli. The Khepera IV Mobile Robot: Performance Evaluation, Sensory Data and Software Toolbox. In *Robot 2015: Second Iberian Robotics Conference*, Lus Paulo Reis, Antnio Paulo Moreira, Pedro U. Lima, Luis Montano, and Victor Muoz-Martinez, Eds., vol. 417. Springer International Publishing, Cham, 2016, pp. 767–781.

[24] Kapellmann-Zafra, G., Salomons, N., Kolling, A., and Gro\s s, R. Human-Robot Swarm Interaction with Limited Situational Awareness. In *International Conference on Swarm Intelligence* (2016), Springer, pp. 125–136.

[25] Kolling, A., Walker, P., Chakraborty, N., Sycara, K., and Lewis, M. Human Interaction With Robot Swarms: A Survey. *IEEE Transactions on Human-Machine Systems 46*, 1 (Feb. 2016), 9–26.

[26] Michael Rubenstein, Adrian Cabrera, and Justin Werfel. 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* (St. Paul, MN, USA, May 2013), AAMAS

'13, International Foundation for Autonomous Agents and Multiagent Systems, pp. 47–54.

[27] Neumann, M. A., and Kitts, C. A. A Hybrid Multirobot Control Architecture for Object Transport. *IEEE/ASME Transactions on Mechatronics 21*, 6 (Dec. 2016), 2983–2988.

[28] Nora Ayanian, Andrew Spielberg, Matthew Arbesfeld, Jason Strauss, and Daniela Rus. Controlling a Team of Robots with a Single Input.

[29] Pereira, G. A., Kumar, V., Spletzer, J. R., Taylor, C. J., and Campos, M. F. Cooperative transport of planar objects by multiple mobile robots using object closure. In *Experimental Robotics VIII*. Springer, 2003, pp. 287–296.

[30] Reina, A., Cope, A. J., Nikolaidis, E., Marshall, J. A. R., and Sabo, C. ARK: Augmented Reality for Kilobots. *IEEE Robotics and Automation Letters 2*, 3 (July 2017), 1755–1761.

[31] Suzuki, N., Hattori, A., and Hashizume, M. Benefits of augmented reality function for laparoscopic and endoscopic surgical robot systems. 8.

[32] Thomas Caraco, Steven Martindale, and H. Ronald Pulliam. Avian flocking in the presence of a predator. 400–401.

[33] Wang, Z., Yang, G., Su, X., and Schwager, M. OuijaBots: Omnidirectional robots for cooperative object transport with rotation control using no communication. In *International Conference on Distributed Autonomous Robotics Systems (DARS), London, UK* (2016).

[34] Wilson, S., Pavlic, T. P., Kumar, G. P., Buffin, A., Pratt, S. C., and Berman, S. Design of ant-inspired stochastic control policies for collective transport by robotic swarms. *Swarm Intelligence 8*, 4 (Dec. 2014), 303–327.

[35] Yu, D., Jin, J. S., Luo, S., Lai, W., and Huang, Q. A Useful Visualization Technique: A Literature Review for Augmented Reality and its Application, limitation & future direction. In *Visual Information Communication*, M. L. Huang, Q. V. Nguyen, and K. Zhang, Eds. Springer US, Boston, MA, 2009, pp. 311–337.

[36] Yutaka Kanayama, Yoshihiko Kimura, Fumio Miyazaki, and Tetsuo Noguchi. A Stable Tracking Control Method for an Autonomous Mobile Robot. IEEE.

[37] ZhiDong Wang, Hirata, Y., and Kosuge, K. Control a rigid caging formation for cooperative object transportation by multiple mobile robots. IEEE, pp. 1580–1585 Vol.2.

[38] ZhiDong Wang, Hirata, Y., and Kosuge, K. Control a rigid caging formation for cooperative object transportation by multiple mobile robots. IEEE, pp. 1580–1585 Vol.2.