CHALMERS

Collective transportation of objects by a swarm of robots

Master's thesis in Complex Adaptive Systems

SINA TORABI

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Adaptive Systems Group
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2015
Master's thesis 2015:47

MASTER'S THESIS IN COMPLEX ADAPTIVE SYSTEMS

Collective transportation of objects by a swarm of robots

SINA TORABI

Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Adaptive Systems Group
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2015

Collective transportation of objects by a swarm of robots
SINA TORABI

Master's thesis 2015:47
ISSN 1652-8557
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Adaptive Systems Group
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Chalmers Reproservice
Göteborg, Sweden 2015

Collective transportation of objects by a swarm of robots
Master's thesis in Complex Adaptive Systems
SINA TORABI
Department of Applied Mechanics
Division of Vehicle Engineering and Autonomous Systems
Adaptive Systems Group
Chalmers University of Technology

Abstract

A collective transport strategy, inspired by the food retrieval procedure of ant colonies, has been implemented on a swarm of robots that are smaller than the object. A simple odometry-based team coordination strategy in combination with an omni-directional camera has been implemented, resulting in a well-coordinated effort by the robots without using any communication. The strategy is fully decentralized. Moreover, a simple recruitment process has been introduced but it did not improve the transportation efficiency. The transportation strategy consists of four stages, namely prey discovery, team coordination, recruitment, and transportation. A simulation environment capable of handling robot swarms and their physical interaction is developed for this project. Using robots weighing 3 kg, a 3 kg object was successfully transported in 48 out of 50 trials, whereas a 4.5 kg object was successfully transportedin 44 out of 50 trials.

Keywords: Collective transport, swarm robotics

Acknowledgements

I would like to thank my supervisor, Mattias Wahde, and my friends and family without whom this thesis would not be possible.

Contents

Abstract i

Acknowledgements i

Contents iii

1 Introduction 1
1.1 Literature study . . . 1
1.1.1 Collective transport in nature . . . 1
1.1.2 Collective transport in robotics . . . 2

2 Simulation environment 4
2.1 Available robotic simulators . . . 4
2.2 General issues in simulation . . . 5
2.2.1 Timing of events . . . 5
2.2.2 Noise . . . 5
2.2.3 Sensors . . . 6
2.2.4 Actuators . . . 7
2.2.5 Physics engine . . . 9
2.2.6 Collision detection . . . 10
2.2.7 Robot brain . . . 11
2.3 Introduction to the simulator . . . 11

3 Collective transport algorithm 13
3.1 Collective transport algorithms in ants . . . 13
3.2 Collective transport in robotics . . . 13
3.3 Collective transport strategy . . . 14

4 Simulation results 18
4.1 Simulation setup . . . 18
4.2 Transport efficiency definition . . . 19
4.3 Results . . . 19
4.3.1 Number of successful trials . . . 19
4.3.2 Path efficiency . . . 19
4.3.3 Transportation time . . . 20
4.3.4 Transportation path . . . 20
4.3.5 Speed of the prey during the transportation . . . 21

5 Discussion and conclusions 23
5.1 Discussion . . . 23
5.2 Future work . . . 23
5.3 Conclusions . . . 24

Bibliography 25

1 Introduction

Recently, there has been a growing interest in swarm robotics as it provides an interesting alternative to more classical approaches such as classical artificial intelligence. Swarm robotics can be defined as an alternative approach, as opposed to more centralized approaches, to coordination of large numbers of robots and the study of how large numbers of relatively simple agents can be designed in such a way that a desired collective behavior emerges from the local interactions among the agents and between the agents and the environment [4, 21, 22, 1].

Swarm robotic systems consist of autonomous robots with local sensing and communication capabilities, but without any access to centralized control or global knowledge of the environment. A more important feature of a swarm robotic systems is the collective behavior shown by its agents. Swarm robotic systems are often used to solve tasks that might be inherently impossible or too complex for a single robot to tackle, such as, collective transport, self-assembly, task allocation, chain formation collective exploration, etc [4].

The main source of inspiration for swarm roboticists comes from observing the behavior of social animals and, in particular, social insects such as bees, ants and termites. Studies [6] have revealed that there exists no centralized coordination mechanisms behind the synchronized operations in social insects, and yet their system's performance is often robust, scalable and flexible. These properties are desirable for multi-robot systems, and can be regarded as motivations for the swarm robotic approach [26].

Robustness requires the swarm robotic system to be able to continue to operate, although at a lower rate, despite individual failures. Robustness can be characterized by several factors. Redundancy in the system, that is, any loss of an individual can be compensated by another individual. Decentralized coordination, which means the system can operate without a leader, and, finally, the simplicity of the individuals.

Scalability means that the system is able to carry out a task under a wide range of group sizes. This implies that the underlying coordination mechanisms that ensures the system work is undisturbed by changes in group sizes.

Flexibility of the system is a feature of the system that enables it to generate modularized solutions to different tasks. Just like ant colonies in which individuals take part in different tasks such as foraging, prey retrieval and chain formation, swarm robotic should be able to respond to different tasks in different environments, utilizing different coordination strategies.

Collective transport, also known as group prey retrieval, is a collective behavior which can be combined with other behaviors such as exploration, pattern formation, self-assembly and task allocation for solving a complex task, for instance, a search and rescue operation in a dangerous situation such as after an earthquake. In collective transport, a group of robots need to cooperate in order to transport an object which is heavy for a single robot to move. This task needs a coordinated movement of robots with collective decision making when necessary, such as in stagnation situations. The task of collectively transporting an object can be carried out by a swarm robotic systems, taking ants as a source of inspiration, since ant colonies are able to successfully and efficiently transport an arbitrary object without *a priori* knowledge of the object's shape, mass or its location. Collective transport has a lot of applications such as agriculture, warehouses, mining, etc.

1.1 Literature study

There has been a large amount of research on collective transport in social insects [3, 2, 23], as well as in the field of swarm robotics [20, 13, 27, 28]. In this section, a review of previous work in robotics as well as in biology will be given.

1.1.1 Collective transport in nature

Group food retrieval has evolved several times in ants, but to a strongly varying degree between ant species [2]. It should be mentioned that there are some ants that show high social skills, but, at the same time, show little or no skill at collective transport [16]. In those species, food is mostly retrieved by single ants. Larger items will be divided into smaller pieces so that they can be transported by single ants. There are only a few species efficient in group transport, a behavior that has evolved for different reasons. Among the impressive examples are the group raiding species known as **army ants** or **marauder ants** [12]. Their massive colonies can only be fed by a large amount of fresh food. Therefore, it is very important for the colony to capture and retrieve the prey quickly and efficiently. Collective transport of the food will serve both goals, since it reduces the need

for in-place food dissection and will use fewer ants for delivering the prey and thus make more labors available for hunting.

In [12], Franks reported that teams exist in species of army ants. He also discovered that the workers in the colony are able to assess their performance and their contribution to a group effort. Therefore, army ants seem to avoid having too large or too small groups of workers while retrieving a prey. For instance, workers will join a group only until they have brought the item to the standard retrieval speed. Moreover, he reported that groups in army ants have a definite sociological composition which will help them to carry an item with maximum efficiency.

Another ant species that shows high skills of cooperation in food retrieval is the desert ant **Aphaenogaster cockerelli** [17]. However, the underlying reason is different than for the army ants. These desert of ants have small colonies that contain about only ten thousand workers, compared to millions in army ants. They also lack the aggressive behavior shown by army ants. Therefore, A. cockerelli had to find a better solution to overcome these disadvantageous and their solution is rapid collective retrieval of large items before other competitors can monopolize them. In [17], Hölldobler *et al.* reported that three to five A. cockerelli can carry an item of 750 mg, jointly, in about 5 minutes over a distance of 7 m. An individual forager needs about 1.5 minutes to transport an item of 3 mg, over the same distance. Moreover, they showed that A. cockerelli ants uses a sophisticated approach for recruiting a team. This species of ants, use both short- and long-range recruitment. In short-range recruitment, when a forager finds an item, she starts moving around the prey and deposits pheromone which can be detected by other ants at distances up to 2 m. If the short-range recruitment does not work, then the finder will go back directly to nest and bring more ants. This procedure can bring a band of 5-10 foragers.

Collective transport in ants consists of four phases in ant species which are more efficient at transportation [23]. These four phases are (1) locating a large item of food; (2) recruiting more ants for the transportation; (3) coordinating and organizing the transportation in which the direction of the transportation. The coordination could emerge from the behavior of the ants, or, it could be organized by ants themselves; (4) and, finally, accumulated workers move the item toward the nest. A detailed discussion of these four phases and how they are performed by ants can be found in [23]. Moreover, Berman *et al.* in [2] studied the behaviors of A. cockerelli ants, in an experimental setup and extracted the rules that govern the ant transport behavior. They also developed a behavioral model which was verified by reproducing the observed behaviors of ants during experiments.

1.1.2 Collective transport in robotics

Collective transport has been studied for years in multi-robotic systems. In general, there has been three major different strategies for transporting an object by robots, namely pushing, pulling and caging.

One of the earliest works that adopted a pushing strategy was the study by Kube and Zhang [20]. They utilized a bottom-up approach for design of a robot's controller, and developed a behavior-based model for collective transport of an object using five simple behaviors that were validated by experiment. There are some problems with the pushing approach to collective transport such as stagnation, coordination of motion, and the effect of the shape of the object being transported. In [19], Kube and Bonabeau addressed the issue of stagnation and proposed a recovery mechanism including realigning the pushing angle of repositioning the pushing force. They also pointed out that directed box-pushing, in which there is a fixed goal, requires no communication and it is insensitive to box size and geometry. However, it should be mentioned that in their experiments, all robots could sense the direction of the goal and therefore move toward it. The pushing strategy have been also used in [25, 8], although with different behaviors for controlling the robots.

The pulling strategy involves making a number of robots connect themselves to the object using physical mechanisms. In nature, in ants for example, transporting an object using pulling strategy is the most common way of transportation since ants are equipped with appropriate physical mechanisms to carry out the task [28]. However, implementing the pulling strategy on actual robots is still a difficult task due to the complicated physical mechanisms required for pulling an object. This strategy has been implemented in [13, 15, 10], using so called **s-bot** [10]. S-bots are fairly simple robots with a number of sensors and motors, basic communication devices and limited computational capabilities, suitable for multi-robot tasks involving self-assembly. These robots are equipped with physical mechanisms that allow them to form physical structures. Moreover, the ability of these robots to carry out collective task were validated in different experiments involving self-assembly, cooperative transport, exploration and navigation.

The caging strategy can be regarded as a special case of the pushing strategy. In this method, a several robots organize themselves around the object in such a way that the object is caged (trapped) inside the robot

formation [27, 32, 24, 11]. Depending on the object's shape, caging can be a complex problem since it requires a certain number of robots to be available and considerable amount of information about the object. The caging strategy can also be implemented using robots capable of self-assembly, such as in [13, 15].

There are various ways to carry out these transport strategies. Most of the work dealing with the transport strategies belong to behavior-based robotics, since it offers feasible solutions. In [20], Kube and Zhang defined five simple behaviors for the collective transport task. They compared two behavior selection mechanisms, subsumption network and adaptive logic network. Subsumption networks use a fixed priority assignment between behaviors and simply picks the behavior with highest priority. Adaptive logic networks are neural networks formed in binary tree configurations. They showed that adaptive logic networks were much simpler to design compared to subsumption network, however, subsumption network was more efficient in accomplishing the task. Same behavior selection method was adopted in [8, 25]. However, in [13, 15, 10], artificial neural networks, synthesised by evolutionary algorithms, for action selection were implemented. Artificial neural networks, in general, require a lot of training time which makes them difficult to implement and test on actual robots.

One of the issues that should be addressed during a cooperative transport process is the coordination of motions and the forces of robots. For a successful transport, robots must move and apply their forces in the right way to ensure that the object will move, at first, and that it also moves in the right direction. For solving this problem, several methods have been used. In earlier work like in [20, 19], it was assumed that the target location to where the object is supposed to be transported, was visible by all the robots. Therefore, robots only needed to push the object in the perceived direction. In recent works, more sophisticated approaches were used to overcome this limitation. In [25, 7], consensus based coordination was proposed in which it was assumed that not all the robots can detect the goal. In this case, those robots that can see the goal will move toward the goal, while other robots try to minimize the difference between their direction and their neighbors' direction (flocking consensus behavior). Although consensus based algorithms work well in coordinating the movement of the robots, robots must be equipped with communication devices which can be problematic. Another method that has been used by researchers, is based on occlusion. In this method, robots are capable of detecting the goal whenever it is not occluded, which can happen if there are other robots in their way or the goal is occluded by the object [8, 14].

It would appear that current research in the collective transport field is focused on fining a way for coordination of the robots. However, in the author's opinion, an interesting topic, to explore is the effect of recruitment on transport efficiency and time, just as in ant species. In this thesis, an algorithm for collective transport has been implemented featuring searching for the object, coordinating the robots movement and recruitment, if needed. Moreover, in order to solve the coordination problem in robots, instead of enabling all, or some, of robots to see the goal, they are equipped with wheel encoder which will allow them to use odometry for estimating their motion. The method will be explained in detail later in section **??**. The algorithm has been implemented in behavior-based robotic fashion and a subsumption network has been chosen for behavior selection. In order to do so, a simulation environment has been written and the algorithm has been tested.

2 Simulation environment

Robotic simulation is an important part of the research in the field of robotics, since it enables one to rapidly test algorithms, design robots, and perform different tests using realistic situations and scenarios. Moreover, creating virtual robots and simulating their components can lead to simplification of the construction process. In addition, most of the applications used in the simulation can be transfered to an actual robot without any changes or at least major changes. In a simulation environment, interactions and behaviors of a robot can be simulated with high accuracy compared to the actual real life model. For example a mobile robot can be simulated as it moves around in an environment with lots of obstacles, in order to analyze the responses from its sensors and the efficiency of its algorithm without any risk of damaging the robot and spending a lot of times on preparing the robot and the environment.

Robotic simulators have other advantages such as reducing the cost of producing a robot from scratch, testing different programming code based on the specifications, modification of the design without further costs, testing different sensors and parts of a robot and to determine whether or not that the robot will meet the specifications. It is worth mentioning that using a simulator can reduce the time of the design of a robot's behaviors. For instance, it is common to use optimization algorithms, such as evolutionary algorithms, when designing the interaction rules for a robot, which is very time-consuming if it has to be carried out on a real robot given its limited computation power. On the other hand, simulation environment enables researchers to use the optimization algorithms in the simulation stages and then transfer the final results to the actual robots. However, it has to be mentioned that it is, in most of the time, necessary to iterate between the simulation environment and the actual robot several times to have the desired outcome.

On the other hand, there are two disadvantages of using simulations in the field of robotics. First, the simulation can only simulate what it is programmed to simulate. Second, there are some unpredicted scenarios in real life that are very hard to simulate in a virtual environment such as the presence of humans in the actual environment.

In order for the simulations to be applicable on real systems, there are several issues that need to be addressed during the simulations such as implementation of sensors, kinetics and kinematics of a robot, etc. These issues and the implementation of various sensors and actuators will be discussed in further detail in the following sections. For this project, a simple robotic simulator has been written using the C# programming language in Microsoft Visual Studio Express 2010.

2.1 Available robotic simulators

In the field of swarm robotics, several simulation environments have been developed. A survey and comparison of the robot development platforms, including a limited discussion of the simulators is given in [29]. Here, a few of the influential systems will be introduced briefly.

- **TeamBots**[1] : TeamBot is a simple 2D simulator and was popular around the year 2000, due to its ease of use and free distribution, and the ability to run the same code in simulation and the real robots.
- **Gazebo**[2]: Gazebo simulates multiple robots in a 3D environment, with extensive dynamic interaction between objects. It enables one to use different sensors including laser range finder, 2D/3D cameras, Kinect style sensors, contact sensors, force-torque sensor, and more. Gazebo has access to different physics engine including the open dynamics engine, Bullet, Simbody, DART.
- **Webots**[3]: Webots is a commercial simulator focuses on accurate dynamical models of popular robots. It is fast [29], easy to use and user friendly interface. This simulator uses the open dynamics engine to simulated dynamic worlds. It has a Fast2DPlugin extension that is optimized for simple, fast simulations, based on the Enkie engine and comes with detailed models of popular robots in swarm robotics such as EPFL Alice, Khepera, and E-Puck robots.
- **Microsoft robotics studio**[4]: Microsoft robotics studio released in early 2007, and it is functionally

[1] http://www.cs.cmu.edu/~trb/TeamBots/
[2] http://gazebosim.org/
[3] http://www.cyberbotics.com/
[4] http://www.microsoft.com/en-us/download/details.aspx?id=29081

similar to Gazebo. Like Gazebo and Webots, Robotics studio is based on a high-fidelity dynamics engine. Currently, there is no project with multiple robots that uses Robotic studio.

- **Swarmbot3D (S-bot simulator)**[5]: The SWARM-BOTS project and s-bot robot system has been developed by Mondada, Dorigo and other researchers. It is a highly successful and influential swarm robotics project. The project has its own simulator which includes dynamic interactions of the robot and its environment and it is based on the Vortex^{TM} commercial physics engine.

- **Swarmanoid simulator**[6]: Swarmanoid project is the successor to the SWARM-BOTS project and examines the heterogeneous swarms if robots. This simulator has modular design, whereby controllers, sensors and actuators, physics engine and visualization are implemented as plug-ins.

In this thesis, a simulator based on the ARSim simulator [31], has been developed. The core of the simulator is similar to the one used in [31] with some modification necessary for the task at hand, such as adding extra sensors, adding the ability of handling collisions, and having multiple robots.

The robot that has been used in this simulator is a differentially steered two-wheel robot equipped with two IR sensors, five bumper sensors, and an omni-directional camera. Implementation details are discussed in further sections.

2.2 General issues in simulation

In a simulation, after the initialization of the robots and the environment, several events occur in a stepwise fashion. At each step, simulator loops through all the robots to obtain their sensory readings and sending it to their robotic brain, in which motor signals will be determined. Next, based on the calculated motor signals, robots' accelerations are calculated which will lead to new velocities for each robot. After obtaining new velocities, physics engine of the simulator will resolve any contact and collisions between robots and objects and it will alter the velocities. Then based on the given velocities by physics engine, robots and objects will move. And finally, termination criteria are checked. If the criteria are not met, simulator will repeat the steps. The flow of the simulator is given in figure 2.1.

2.2.1 Timing of events

Simulation results must be tested and verified on an actual robot. However, in order to test the simulation on a real robot, some issues need to be taken care of, such as obtaining the sensory inputs, processing of the information and the computation of the motor signals. These steps need to be taken in an actual robot during less or equal amount of time that it takes in simulation. Here, there are two types of events, namely, those events that take a long time to complete in simulation, but would take a very short time in a real robot, and those events that are carried out rapidly in simulation, but would take a long time to complete in a real robot [31].

2.2.2 Noise

Another important aspect of the simulation is to implement noise. Real sensors are noisy on several levels. Moreover, even noise free sensors somehow show different values in practice. In addition, the reading frequency can introduce another source of noise. Therefore, it is very important to model noise and add it to the process of obtaining the sensory informations.

Noise can be added in several ways, one of the common ways is to take the original reading of the sensor S and add noise in order to form the actual reading S'.

$$S' = SN(1, \sigma) \tag{2.1}$$

here $N(1, \sigma)$ is the normal distribution with the mean 1 and the standard deviation σ. Another method for implementing the noise is to take some measurements of the real sensor and store the readings in a lookup table. Then, this lookup can be used in simulation by the virtual robot. However, the lookup table method is limited to very simple sensors.

[5] http://www.swarm-bots.org/index.php?main=3&sub=33
[6] http://www.swarmanoid.org/swarmanoid_simulation.php

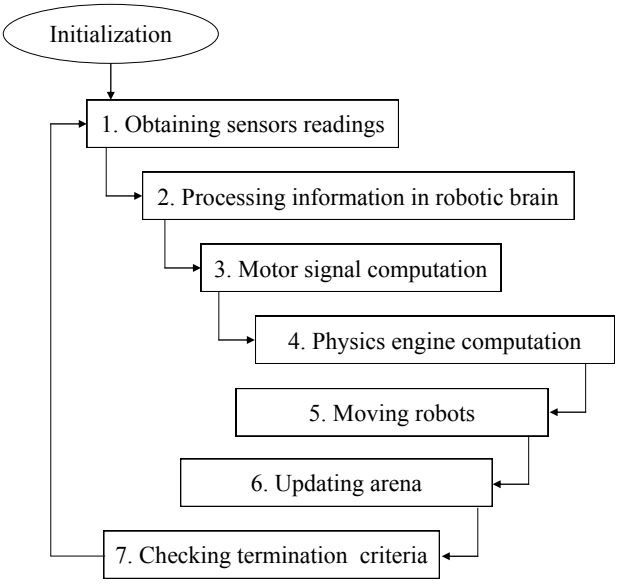


Figure 2.1: *Simulation flow for the simulator used here. Steps 1 through 3 are carried out first for all robots. Then, simulator will go to step 4.*

2.2.3 Sensors

The robots used for the simulation is equipped with different sensors, such as infrared (IR) sensors, touch sensors (bumpers) and a camera with a 360 degrees view. These sensors have been implemented in the simulator in the presence of noise, as mentioned before. The detail description of the implementation has been given in following.

- **IR-sensors:** IR sensors belong to the class of ray-based sensors, which use a simple form of ray tracing in order to form their readings. Other examples of the ray-based sensors are sonar sensors and laser range finders. In order to simulate the IR sensors, one needs ray tracing which will be explained later.

  In ray-based sensors, basically, a number of rays, 3 to 5 normally, are sent out from the sensor in different directions (depending on the opening angle of the sensor), and then the distance to the nearest object is determined. If there is no object in the range of the sensor, then there will be no reading (0 in this case). The range of the IR-sensor implemented here is 0.8 meters. Since the objects, including the robots and walls and other objects are represented by circles and rectangles, then the readings of the sensor are obtained using line-line intersection in the simulator. At each time-step, the IR-sensors readings are updated by checking whether sensor's ray intersects with another object. Full procedure of how to implement the IR-sensors are given in [31].

- **Bumper-sensors:** Bumper sensors are used for detecting physical contacts with an obstacle by sending a 0 or 1, i.e. true or false, signal to the robot. A bumper signals 1 when it is in contact with another object, and signals 0 when it is not. In this simulator, bumpers are considered like small rectangles, Figure 2.2. Therefore, a bumper is activated when the bumper's rectangle collides with an object; and since all the objects are represented by lines or circles, the reading of a bumper can be simulated by implementing a line-line intersection detection algorithm.

- **Omni-directional camera:** The omni-directional camera that has been used in this project is similar to the camera used in *S-bots*, [10]. This camera can detect an object with its color (usually red, green, blue, and sometimes a combination of three) in the range of 70-110 centimeters, the range is different depending on the color. Moreover, the camera can estimate the distance as well as it is angular position with respect to the robot's position. Since it is a bit unclear how a camera can be implemented in a simulation environment, a simple approach has been used here.At each iteration, the distance to different detectable objects are calculated and then after adding noise to the distance, if it falls in the camera's

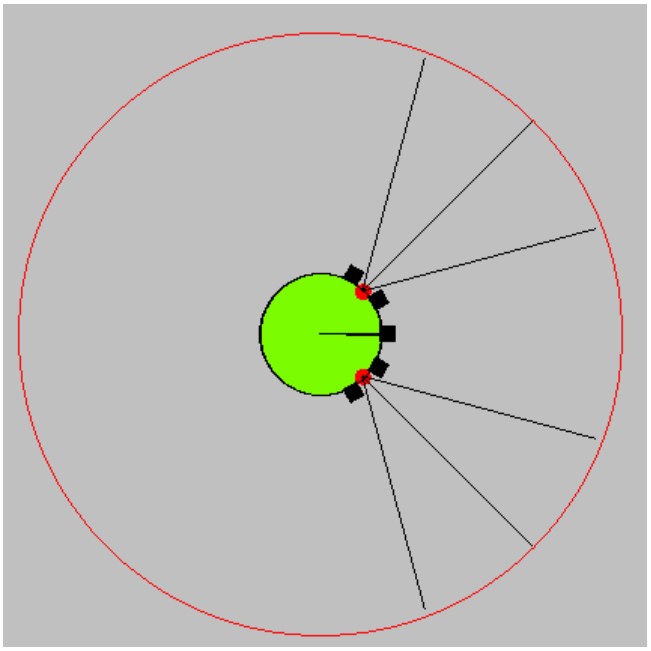


Figure 2.2: *Snapshot of the simulated robot. The IR-sensors are represented by small red circles in front of the robot, and the IR-rays are modeled by three black lines (for each sensor). Each robot has also equipped with five bumper sensors, modeled by small black rectangles in front. Omni-directional camera is represented by a red circle around the robot.*

range, then it will be added to the detected objects by the robot. The detectable objects are the prey, the nest, and the robots that have their LEDs light on.

An alternative method is to model the camera as a ray-based sensor with an opening angle of 360 degree and find the distance to all the detectable objects within the range using the line-line intersection method described before.

- **Wheel encoders:** One of the simplest way of estimating the position of a robot is to use wheel encoders. Wheel encoders determine the position of the robot based on the distance traveled by each wheel. Wheel encoders provide information that in combination with robot's kinematic allows one to estimate the position and heading of a robot. This process is an example of **odometry**. In order to implement a wheel encoder, following steps should be taken in each iteration:

  1. Computing left and right wheel's speed.
  2. Computing the change in the distance that each wheel has taken from previous iteration.
  3. Computing the change in the heading from previous iteration using wheels' speeds.

  There are several problems with wheel encoders in actual robots which can lead to inaccurate estimation of position and heading of a robot, such as when the wheels are slipping but the robot is not moving. In order to include the wheels slipping in the simulation, noise should be added to the estimated position and heading after their calculations.

2.2.4 Actuators

In this project, robots are only using DC motors as actuators. In this simulator, a standard DC motor has been implemented. The motors take the applied voltage as the input signals. In simulation, mechanical and electrical dynamics of the motors are neglected, and therefore, the output torque is given by following equations,

$$\tau_g = \frac{c_t}{R} V - \frac{c_e c_t}{R} \omega \tag{2.2a}$$

$$\tau = \tau_g - c_C \operatorname{sign}(\omega) - c_v \omega \tag{2.2b}$$

$$\tau_{out} = G\tau \tag{2.2c}$$

where τ_g is the generated torque, τ is the load torque, τ_{out} is the output torque, V is the voltage, R is the resistance, ω is the angular velocity, G is the gear ratio, and c_t, c_e, c_C and c_v are torque constant, electrical constant, coulomb friction constant and viscous friction constant respectively.

Kinematics and Dynamics of the robot

- **Kinematics:** Kinematics is the process of determining the movement of a robot with the various constraints on the motion of the robot, without taking into account the acting forces on it.The kinematics of a robot depends on its structure such as the number of wheels and their types. In this project, a differentially steered two-wheeled robot is considered. The robot used in this project is similar to the one used in [31], therefore only the governing equations are described here.

$$V = \frac{v_L + v_R}{2} \tag{2.3a}$$

$$\omega = -\frac{v_L - v_R}{2R} \tag{2.3b}$$

where V is the speed of the robot and v_L and v_R are the left and right wheel's speed. ω is the angular speed and R is the robot's radius. Therefore, the position of a robot at time t_1 is given by

$$X(t_1) - X(t_0) = \int_{t_0}^{t_1} V_x(t)dt \tag{2.4a}$$

$$Y(t_1) - Y(t_0) = \int_{t_0}^{t_1} V_y(t)dt \tag{2.4b}$$

$$\phi(t_1) - \phi(t_0) = \int_{t_0}^{t_1} \omega(t)dt \tag{2.4c}$$

Where $V - x = V\cos(\phi(t))$ and $V_y = V\sin(\phi(t))$, $(X_(t_0), Y(t_0))$ is the previous position of the robot and $\phi(t_0)$ is the previous heading.

- **Dynamics:** Kinematics only consider the motion of an object and say nothing about how to achieve a particular motion. Dynamics, on the other hand, considers the motion of an object while taking into account the forces acting on it. In the case of a two-wheeled robot, for deriving the equations of motion, one need to take into account the torques generated by the motors as well as the friction and any other forces acting on the robot. Like previous section, the full derivation of the equations are not considered and only the final equations of motion are described here. The full derivation of these equations are presented in [31]. The equations of motion are

$$M\dot{V} + \alpha V = A(\tau_L + \tau_R) \tag{2.5a}$$

$$\overline{I}\ddot{\phi} + \beta\dot{\phi} = B(-\tau_L + \tau_R) \tag{2.5b}$$

where M is the mass of the robot, V is the velocity, $\overline{I}$ is the moment of inertia, ϕ is the angular position, τ_L, τ_R are the torques produced by left and right motors, A and B are constant coefficient depending on the physical feature of the robot and α and β are constants.

- **Robot motion:** Once the applying torque on each wheel is determined, the motion of the robot is implemented using numerical integration of the robot's kinematic equations (equations (2.5a) and (2.5b)). The integration is carried out using simple first order Euler integration. At each time step $\dot{V}$ and $\ddot{\phi}$ are

computed and then the new values for the linear velocity, the angular velocity, new position and heading are computed using following equations.

$$V' = V + \dot{V}\Delta t \tag{2.6a}$$

$$\dot{\phi}' = \dot{\phi} + \ddot{\phi}\Delta t \tag{2.6b}$$

$$\phi' = \phi + \dot{\phi}'\Delta t \tag{2.6c}$$

$$V'_x = V' \cos(\phi) \tag{2.6d}$$

$$V'_y = V' \sin(\phi) \tag{2.6e}$$

$$X' = X + V'_x\Delta t \tag{2.6f}$$

$$Y' = Y + V'_Y\Delta t \tag{2.6g}$$

2.2.5 Physics engine

In order to simulate certain physical systems, such as rigid body dynamics (including collisions), one needs to use a **physics engine**. Physics engine is a computer software that provides an approximation of physical systems. In this project, a physics engine is needed for simulating contacts and collisions between robots and other objects in arena, e.g. the prey, in order to get a realistic simulation.There are numerous physics engines available that one can use, for example unreal engine [7] which is highly detailed and realistic that has been used extensively in video games, or simpler engines such as **Bulltet Physics Library** [8] that has been used in [8], or **Box2D**[9] physics engine.

In this simulator, the scenario is very simple and there is no need to use a library for simulating rigid body dynamics. The physics engine written for this simulator takes the following steps at each iteration. The procedure is a modification of the work in [5]:

1. Advance velocities using equations (2.5a,2.5b).
2. Detect collisions based on candidate positions in next time step.
3. Resolve collisions and update velocities.
4. Apply the friction force between the prey and the surface.
5. Advance positions.

Determining the robots' velocities is a straight forward process, however, it is a difficult problem for the prey since the prey is in contact with surface, and therefore, experiencing a friction all the time. For modeling the friction and therefore the prey's velocity, a similar approach to the work in [5] has been taken. The velocity of the prey at each time step is computed by

$$\mathbf{V}^{n+1} = \max\left(1 - \mu\frac{\Delta\mathbf{V}_N}{|\mathbf{V}^n|}, 0\right)\mathbf{V}^n \tag{2.7}$$

where $\mathbf{V}^{n+1}$ is the prey's velocity at time $n+1$, μ is the static friction coefficient between the prey and the surface, $\Delta\mathbf{V}_N$ is the change in velocity due to gravity at each time step, and $|\mathbf{V}^n|$ is the speed of the prey at

[7] `https://www.unrealengine.com/what-is-unreal-engine-4`
[8] `http://bulletphysics.org/`
[9] `http://box2d.org/`

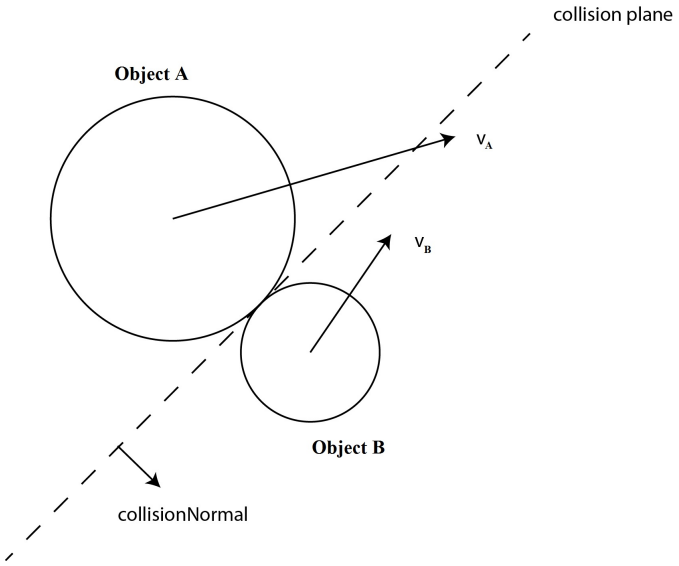


Figure 2.3: *Collision response of two objects.*

time n. In practice, this models the frictional contact with micro-impulses between the prey and the surface in the direction of normal to the surface.

When the new velocities of the robots and the prey is available, the engine will check if any collision occur. This process is done by virtually moving all the objects to their predicted positions, based on the new velocities, and then running a collision check. If any collision occurs, they will be resolved using impulse-based dynamics (explained below). Since collisions change velocities, new collisions might occur based on the new values. For resolving this issue, steps 2 and 3 will be repeated for few iterations, five here, and then the engine will go to the next step.

Collision response: When two objects collide with each other, their velocities change due to their impact. Collision response is modeled using impulse-based dynamics. Suppose that object A is moving with velocity $\mathbf{V}_A$ towards object B which has a velocity $\mathbf{V}_B$, Figure 2.3. Their velocities after collision can be computed using following equations,

$$j = \frac{-(1+e)(\mathbf{V}_B - \mathbf{V}_A)\cdot \hat{n}}{\frac{1}{m_A} + \frac{1}{m_B}} \tag{2.8a}$$

$$\mathbf{V}'_A = \mathbf{V}_A - \frac{j \times \hat{n}}{m_A} \tag{2.8b}$$

$$\mathbf{V}'_B = \mathbf{V}_B + \frac{j \times \hat{n}}{m_B} \tag{2.8c}$$

where j is the impulse, e is the minimum of the two objects' coefficient of restitution, $\hat{n}$ is the collision normal vector from object A to object B, and m_1 and m_2 are objects' masses. This model is applied to two objects, if their relative velocity in collision normal direction is negative, i.e. $(\mathbf{V}_B - \mathbf{V}_A)\cdot \hat{n} < 0$, since this means they will collide in next step. However, when the relative speed is positive in the collision normal direction, then the two objects will not collide.

2.2.6 Collision detection

In general, it is desirable to prevent collisions between robots and other objects in the environment. However, in some situations, one can let the robots collide with other objects, for example when the robots are pushing an object. In any case, collisions need to be detected, and necessary actions be taken, such is in the pushing scenario. In this project, collision checking is implemented using line-line and line-circle intersections since all of the objects in an arena are represented by circles or rectangles. In this project, collision is inevitable since robots are transporting an object which involves collisions. Moreover, during transportation phase, it is likely for robots to collide with other robots. Therefore, the simulation will continue after collisions, however, necessary changes are made using the physics engine.

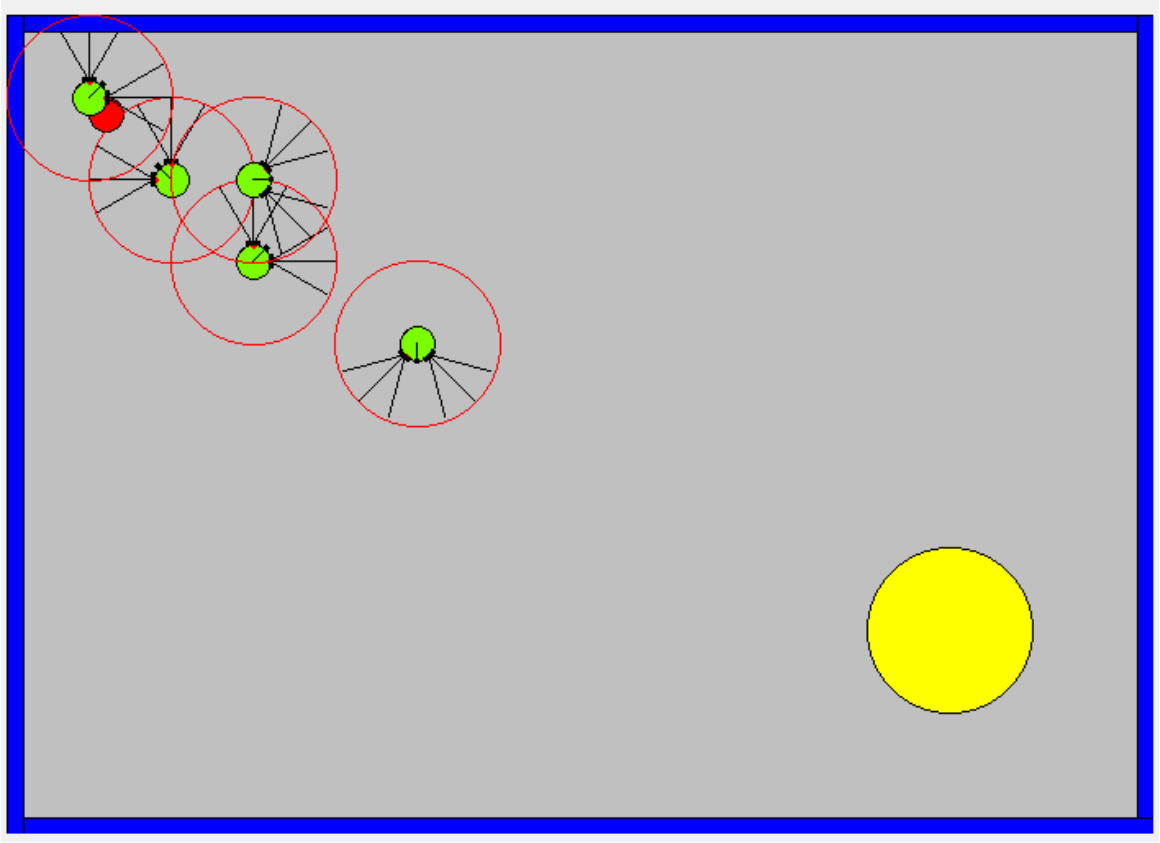


Figure 2.4: *A typical screeshot from the written simulator for this project. The red small circles are the IR sensors and the black lines from each IR sensor are the rays used for modeling the sensor reading. The red circle indicates the range of omni-directional camera and the five black squares in front of each robot is the bumper sensors.*

2.2.7 Robot brain

While the physical components of a robot, such as its sensors and motors, often remain unchanged between simulations, the robotic brain must be adapted to the current task. Robotic brains can be implemented in many different ways.

In behavior-based robotic (BBR), the brain of a robot is often implemented from a set of behaviors designed specifically to meet the requirements of the current task. Moreover, a decision-making procedure needs to be implemented so that the robot can select the appropriate behavior from the set.

2.3 Introduction to the simulator

A screenshot of the simulator is given in Figure 2.4. The robots appear in a quadratic arena with four walls and a prey which is represented by a yellow circle. Moreover, the nest is represented in a red circle. The IR sensors (of which there are two in each robot) are shown by small red circles. The rays used for determining the sensor readings are shown as black lines. Moreover, the omni-directional camera is depicted by a red circle around the robot. Simulator executes the steps until the prey is transported to the nest, or until simulator has executed 50,000 steps of 0.01 s.

The flow of the simulation basically follows the structure in Figure 2.1. First, the arena and the prey and the nest are put in their specified locations. Next, the robots are created with their sensors. Since more than one robot is needed for transporting the prey, several robots are created in vicinity of the nest and the simulator checks that they do not overlap at the initialization step.

After the initialization step, the simulator begins a loop in which several steps are executed. Each time step, the sensors are read and then their informations are passed to the robot's brain. These readings are

obtained in the order of; first, IR-sensors; second, odometer; third, the bumpers. After obtaining the sensor readings, the robot brain processes the information and takes the necessary action by producing the appropriate motor signals. When all the motor signals are obtained from robots, the physics engine will first update all the objects' (including the prey and robots) speed, and then, predict their position given their updated velocities. At next step, physics engine will check for the collisions and follow the structure given in section 2.2.5. When the physics engine complete the movement of the objects, the termination criteria are checked, in this case the prey transportation, and if they are met, simulation ends.

3 Collective transport algorithm

3.1 Collective transport algorithms in ants

According to [23], there are four phases in the cooperative transport in ant colonies. These four phases are (1) decision phase; (2) recruitment phase; (3) organization phase; (4) and transport phase. These four phases are described in further details in the following part.

1. **Decision phase:**

   In some ant species, the decision to initiate a cooperative transport is adaptive, and based on the likelihood that the transport would succeed. Sometimes making a decision is not necessary since cooperative transport could emerge as workers accumulate at the prey's location. However, in ants that actively recruit helpers, the worker that finds the food must decide to initiate the cooperative transportation. This decision is affected by several factors, such as, the prey's resistance to movements, the type and size of the prey, and the likelihood of the prey being found by other ants, i.e. if the probability of the other ants finding the food is low,

2. **Recruitment phase:**

   Recruitment mechanisms vary greatly among different ant species. Recruitment mechanisms usually involve a short-range and long-range procedure. In short-range recruitment, the finders releases a volatile chemical that attracts nearby ants which can attract ants at distances up to 2 m. In long-range recruitment, the finder goes back directly to the nest, and on her way back to the nest, she deposits a pheromone trail and brings back three to five ants to the prey. There are some ant species in which the recruitment process involves both mechanisms. Moreover, in army ants, the finder does not recruit other ants by laying a pheromone trail or going back to the nest. Instead, she sends signals to the nearby ants. Some of these ants immediately join the finder and try to move the prey, while other ants recruit more ants from the nest.

3. **Organization phase:**

   Organization is an important phase that differentiates the efficient transportations from inefficient ones. In some ant species, it seems that there is no organization and the coordination is a self-organized procedure. In this species, there is a transient state throughout which ants frequently change the angle at which they are applying force. At some point this uncoordinated effort yields sufficient force in the correct direction to initiate the movement. In some other ant species, distinct roles are established for the duration of the effort. In these species, the ant that originally found the prey is more important to the success of the transportation. Moreover, the group size is determined during the organization phase, since when there are more ants than needed, their effort is wasted. For instance, in army ants, an ant would join a group if it can help them to carry the prey at a certain speed. Therefore, an ant would not join a group that is transporting an item at a desired speed.

4. **Transport phase:**

   In ant species with forward-facing transport, workers do not grasp the food all at once. Instead, a large worker begins moving the prey and other workers join the transportation. Franks in [12] described a simple rule that could lead to this joining behavior. In some other species, ants drag the item while walking backward. This transporting behavior increases the stability by keeping the food's center of gravity low. Dragging the food and keeping it low can also help the coordination of the transportation since all ants can line up along the food and faces the same direction.

3.2 Collective transport in robotics

Tasks that requires massive parallelism, high level of redundancy, and adaption to, possibly hazardous, environments can potentially be performed by a swarm robotic system. Such systems consist of hundreds of autonomous robots with limited communication, sensing and computation capabilities, identical in hardware and controlling algorithms. By observing natural swarms, such a system would enable the parallel execution of tasks, robustness to individual failure, and also, they scale well with the size of the swarm.

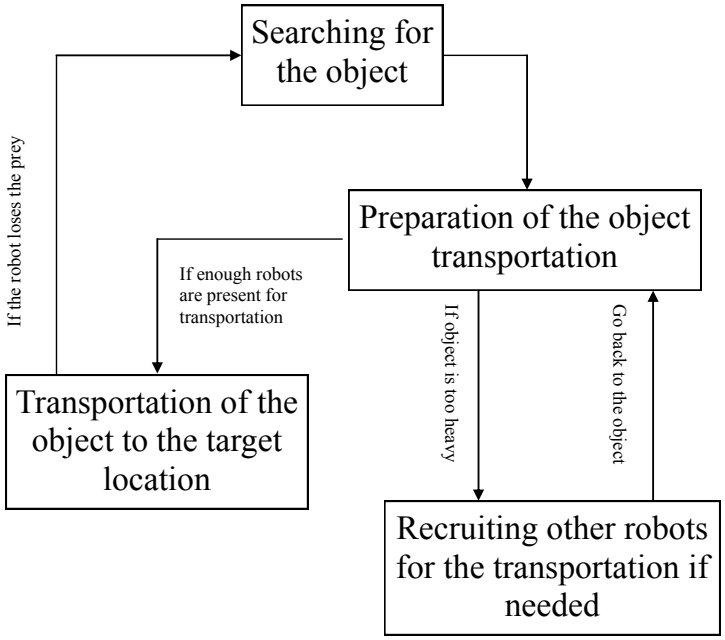


Figure 3.1: *General structure of a robot's major behavior while equipped with the recruitment behavior in its transport strategy.*

The size of a swarm makes it impractical to use centralized approaches, which can provide globally optimal solutions. On the other hand, decentralized approaches, although resulting in suboptimal solution, are more practical and scalable with the size of the swarm. Moreover, decentralized approaches are easier to implement since they require local information/communication without global knowledge of the system. One of the common approaches in decentralized robotic control is **behavior-based robotics**.

Behavior-based robotics offers a bottom-up approach, and in most cases, inspired by biological systems. Behavior-based robotics consists of a collection of behaviors for achieving a goal, e.g. formation control. A *behavior* in this context is defined as a set of actions, mostly for the motors, performed by the robot in order to accomplish some goals.

Since the robots are designed with different behaviors, there must be a system for selecting the appropriate behavior, usually called the arbitration systems. There are several methods for use as an arbitration system, such as subsumption architecture [20], machine learning approaches [15, 13] and using the concept of utility function and rational decision making [30].

In this project, a subsumption architecture has been chosen for arbitration of the behaviors due to its simplicity and ease of implementation. The subsumption network uses a fixed priority assignment between behaviors. The selected behavior is simply the one with highest priority. This architecture usually requires the designer to consider all the behaviors in the control system and decide on how to assign priorities.

3.3 Collective transport strategy

In order for a robot to operate in a real environment, some behaviors need to be implemented such as obstacle avoidance. Moreover, for enabling a swarm of robots to collectively transport an object, special behaviors need to be defined such as locating the object , searching for help to transport the object, and direction negotiation, etc. In this section, the behaviors used for the collective transport will be described. The finite-state machine which represents the behaviors and their relations is given in Figure 3.1.

In this structure, each behavior itself consists of several simpler behaviors and motor actions. In other words, The robot will first decide on which major behavior to perform, and then, it will follow a set of simpler behaviors in order to perform the selected major behavior. For instance, when a robot is searching for the prey, it carries out a random walk as well as obstacle avoidance until it finds the prey.

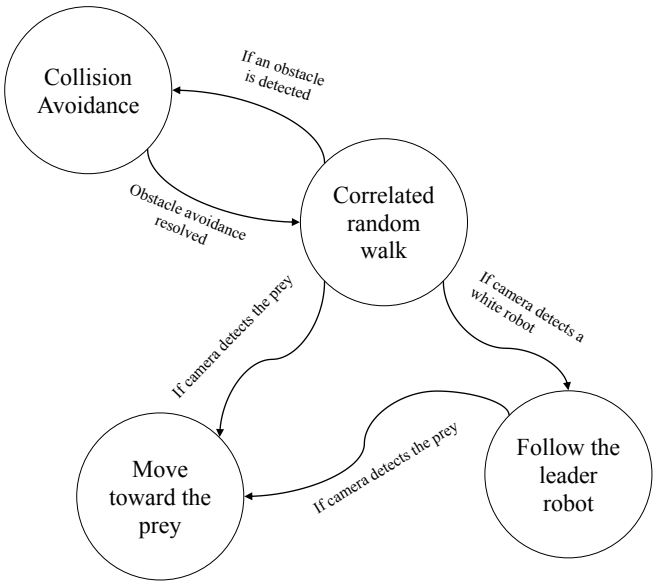


Figure 3.2: *Structure of the search for the prey behavior, carried out by each robot.*

Robot's major behavior implementation

In this section, the description of the behaviors necessary for the transportation of an object is described in details. In this work, each behavior has been implemented as a series of actions which are selected based on the sensory input. In most of the cases, each major behavior is consist of simpler behaviors implemented using sets of *if-else* rules.

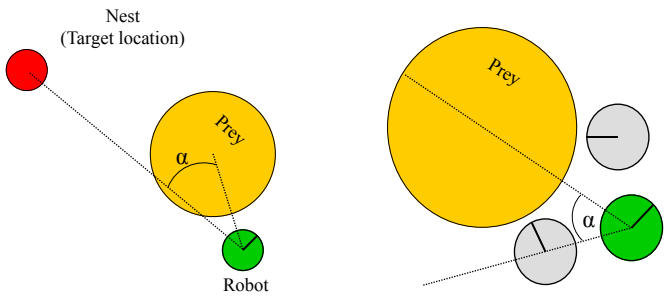
1. **Search for the prey:**

   For finding the prey, a correlated random walk in combination with other simple behaviors is implemented for each robot. More complicated approaches can be taken for exploration as well. However, since it is assumed that the arena in which the robots are operating is bounded, a correlated random walk in combination with obstacle avoidance behavior is sufficient for covering the whole arena. Moreover, correlated random walk has been observed in insects movements [9] when performing a task, such as foraging, which can serve as an additional inspiration.

   Correlated random walks involve a correlation between successive step orientations, known as **persistence** [9]. Persistence produces a local directional bias that each step tends to point in the same direction as the previous one. Nevertheless, the influence of the initial direction of the motion progressively decreases over time and step orientations are uniformly distributed in the long term. Correlated random walks have been frequently used to model animal movements in various contexts [18]. When a robot is performing the correlated random walk, it has to avoid collisions with other objects and robots. Moreover, at each time-step, while performing the correlated random walk, the robot uses its omni-directional camera to find the prey or to locate other robots which have found the prey. If the robot has found the prey, it will change its color to white, then it will move towards it and then changes its behavior to *prepare for transporting the prey*. However, if a robot, which has found the prey, is detected by camera, then the robot will follow it. While the robot is in following mode, it has a purple color so other robots can detect it. The leader-follower behavior will eventually lead the robot to the prey that was found by the leader robot. The structure of this behavior is given in Figure 3.2. The transition between each part is activated based on a prespecified sensory input. For instance, if the robot is performing the correlated random walk and its IR-sensors detect an obstacle, the robot will switch to obstacle avoidance behavior and switch back to the correlated random walk when the collision avoidance is resolved.

2. **Preparing for transporting the prey:**

   This behavior is designed for robots, so that, they can find a suitable position for pushing the prey considering their own positions, and other robots positions. The idea is to find a position in which when the robot starts pushing the prey, it roughly moves toward the nest. This is achieved by combining the



(a) (b)

Figure 3.3: *Left panel shows how a robot will determine if it is approximately behind the prey in the direction of the nest and right panel shows how a robot will determine if there is enough space between robots.*

odometry information with the assumption that all robots know the target location. Therefore, at each time step, robots calculate the nest direction from their estimated position (provided by odometry) and use this information later.

In order to do so, first, the robot walks around the prey (a circular object) simply by performing a wall following behavior. Then, at each time step, the robot will compute the relative angle α in Figure 3.3a.

If its absolute value is less than a threshold, then the robot is behind the prey. When the robot is behind the prey, the robot's behavior changes to *transporting the prey to the target location* as soon as the robot finds an empty spot behind the prey. This is achieved by using the omni-directional camera by checking the robot's distance and relative angle to other robots in vicinity of it, Figure 3.3b. If these values are in a predefined range, then the robot has found an empty place and can start the transporting process.

Another behavior that can be activated from *preparing for transporting the prey*, is the recruitment process. This process is activated automatically and does not require any assessment of the prey's weight and robot's forces. Instead, it is assumed that all robots know for transporting the prey, at least three robots is required. Therefore, when a robot finds the prey, it first goes behind the prey, by following the behavior described above. Then if there is no robot there, the robot will wait for another robot to come. When there are at least two robots ready to transport the object, one of them, by flagging itself, starts the recruitment process. The other robot stays behind the prey in order to inform others, by changing its color, that the recruitment task has been assigned to a robot and others can start the transportation.

3. **Recruiting other robots for the transportation if needed:**

   The process of recruitment is one of the sophisticated mechanisms not only in collective robotics, but also, in ant colonies. In this thesis, a modification of the long-recruitment approach often used by ants has been implemented. In this approach, the recruiter goes back to the nest slowly and look for another robot. Once a new robot is found, the recruiter turns back to the prey. It should be mentioned that the recruiter goes back to the nest and return to the prey just once. The recruiter has a specific color which helps other robots to identify the recruiter and follow it to the nest.

   For the recruitment to start, two robots should be behind the prey. One of them will pick up the recruiting role and the other robot stays behind the prey to inform other robots that the recruitment process has already started and there is no need for recruitment. In this way, no more robots go back to nest and they can start the transportation.

4. **Transport the prey to the target location:**

   When this behavior is activated, the robots face the prey and start pushing it by moving forward. When the prey is moving, and at the same time, other robots are pushing the prey from different angles, it is possible for a robot to be pushed away from a suitable position. Therefore, whenever the relative angle between the object and the robot, computed using odometry and omni-directional camera, goes above a certain threshold, the robot look for a suitable position again (described before). If a robot loses the prey, i.e. it is not detectable by its camera, its behavior changes to *search for the prey*, and since the robot is close to the prey, it will find the prey soon again.

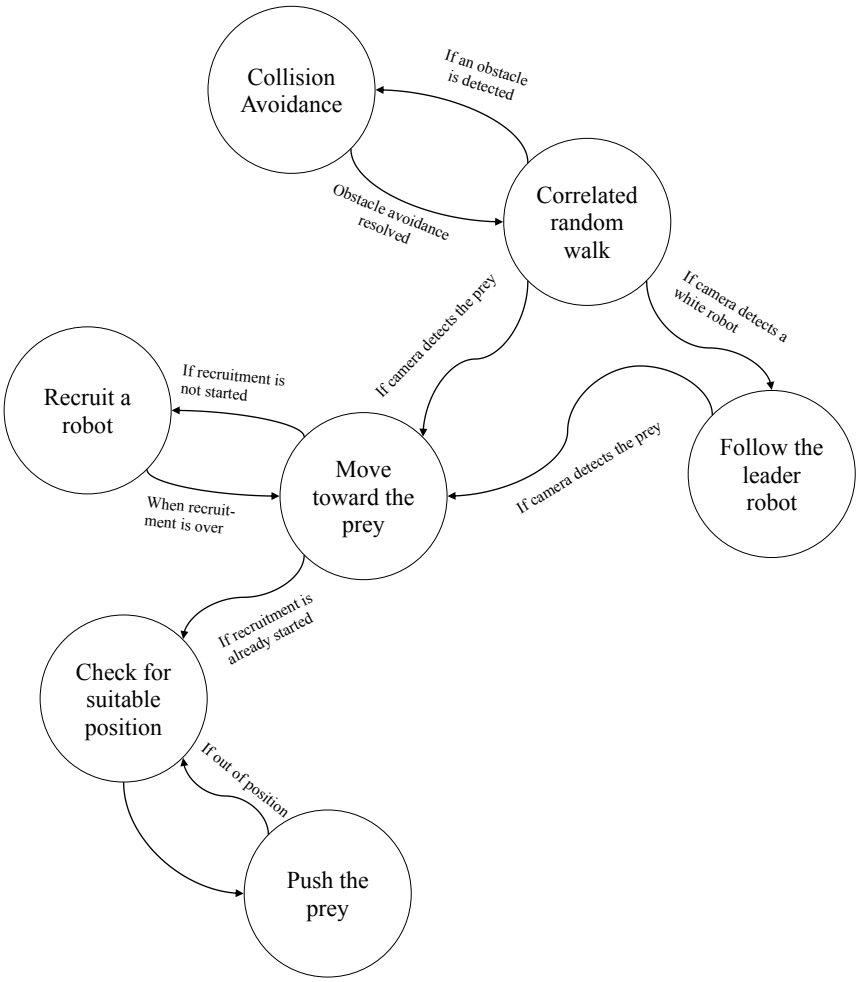


Figure 3.4: *Complete structure and arbitration of the behaviors in a robot's brain. The transitions are based on sensory inputs. At each stage, except when looking for the food, if the robot loses the prey, its behavior changes to the search for the prey.*

Full structure of the algorithm is given in Figure 3.4. It should be mention that after that the robot found the prey, if it loses it, its behavior changes to the *search for the prey*. However, when a robot finds the prey again, it will not go through the recruitment process again. This mechanism has been implemented to avoid any waste of effort when the other robots are pushing the prey. For instance, once the transportation of the prey has started by some of the robot, if a new robot finds the prey and automatically starts the recruitment process, it will check for a flagged robot. If the robot can detect a flagged robot, then it will finish the recruitment process and start the transportation process.

4 Simulation results

In order to evaluate the collective transport strategy in a 2D planar environment, a decentralized algorithm just like the described algorithm in the section 3.2 is implemented in the simulation environment, on small mobile robots platform.

4.1 Simulation setup

The arena in which the robots carry out the transportation task is rectangle. In the arena, there is a prey and the nest and several robots. The robots are created with two IR-sensors, an omni-directional camera, and five bumper sensors. The specification of the objects, such as the objects' position and sensor details, are given in Tables 4.1 ,4.2. All the lengths are represented in meter and the angles in radians.

Table 4.1: Arena setup and objects specifications

Arena	Shape	Rectangle
	Width	10
	Length	14
Nest	Shape	Circle
	Position	$\{1.2, 1.2\}$
	Radius	0.2
	Color	Red
Prey	Shape	Circle
	Mass	$\in \{3, 4.5\}$ kg
	Position	$\{11.5, 7.5\}$
	Radius	1
	Color	Yellow
	Friction coefficient	1.2
	Coefficient of restitution	0.5
Robot	Shape	Circle
	Mass	3 kg
	Number of robots	5
	Radius:	0.2
	Initial position	$x \in [1, 7], y \in [1, 5]$
	Initial heading	$\in [-3\pi/4, 3\pi/4]$
	Initial velocity	0
	Initial angular velocity	0
	Coefficient of restitution	0.5

Table 4.2: Robot's hardwares details

IR-sensors	Number of rays:	3
	Relative angle to robot's center:	$\in \{-\pi/4, \pi/4\}$
	Opening angle:	$\pi/3$
	Range:	0.8
Omni-directional camera	Position	Center of the robot
	Range	1
Bumper sensors	Number of sensors	5
	Relative angle to robot's center	$\in [-\pi/3, \pi/3]$
	Width	0.05
	Length	0.05

4.2 Transport efficiency definition

After initializing the simulation, several parameters need to be defined for measuring the efficiency of the proposed algorithms. These parameters are, namely, number of successful trials, path efficiency, transportation time, and prey's path. These parameters, along with other parameters, has been used to measure the efficiency of the cooperative transportation both in robotics [8] and in biology [23].

1. **Number of successful trials:** For each strategy, several run has been made and the percentage of the successful tries are given. In a successful trial, the prey should be transported to the target location within a certain amount of time.

2. **Path efficiency:** In order to measure the efficiency of the robots pushing coordination, i.e. the way robots distribute themselves around the prey, the ratio of the path length to the distance between the initial position of the prey and the nest is computed. Therefore, the path efficiency can be computed using following equation

$$\text{path efficiency} = \frac{\text{distance traveled by the prey}}{\text{distance between prey's initial position and nest's position}} \tag{4.1}$$

3. **Transportation time:** One of the important factors for an efficient transportation is the time that robots need to transport an object. In order to compute the transportation time, the elapsed time from start of a trial until the center of the prey overlaps with the nest, i.e. the distance between the centers of the prey and the nest is less than a threshold, is computed and defined as transportation time.

4. **Transportation path:** The last parameter shows the path of the object which can be an indicator of any unnecessary effort that leads to an unwanted rotation.

In order to compare the effect of the recruitment on the collective transport strategy, the algorithm described in the section 3.3 is compared to a similar algorithm, however, without the recruitment process. Therefore, one can see how the recruitment process can affect the efficiency parameters.

4.3 Results

For both strategies, several runs with different prey masses are made and each of the efficiency parameters are computed. In this section, the outcome of the simulation for both strategies are given.

4.3.1 Number of successful trials

For measuring the number of successful trials, robots have to transport the prey within a certain time threshold. The threshold varies with the prey's mass in such a way that for transporting a light prey, robots have less time than when they are transporting a heavy prey. In this project, the time limit for a prey of mass 3 kg is 300 s and for a prey of mass 4.5 kg is 450 s (simulation seconds).Therefore, when the cooperative strategies are used, if a trial takes longer than the limits, it is considered as an unsuccessful.

Overall, 48 out of 50 trials with the 3 kg prey, using both strategies, and 44 out of 50 trials with 4.5 kg prey were successful. In simulation, it was allowed for robots to continue the transportation after time limit and it should be mentioned that most of the failed trials could be completed in about 50 more seconds.

4.3.2 Path efficiency

For all successful trials, the ratio of the prey's path length to the distance between its initial position and the nest's position has been computed. The distance between the prey's initial position and the nest is $11.27m$. One can see from the table 4.3 that in average, the prey's path length is around $13m$. Moreover, one can see from the table 4.3 that the robots efforts are in the right direction and they are not causing a lot of unnecessary movement of the prey. The difference between the path length and the distance between the nest and prey's initial position is around $1.5m$ on average and $7.63m$ at maximum. The path efficiency also reveals that the mass of the prey slightly decreases the efficiency, indicating that team coordination is more difficult when dealing with a heavier object, which could be resolved using more robots.

Table 4.3: Path efficiency measurement for transportation of the prey.

		Path length (m)		
Prey mass (kg)	Strategy	Average	Standard deviation	Path efficiency
m = 3	With recruitment	13.068	±0.67	1.159
	Without recruitment	13.161	±1.51	1.167
m = 4.5	With recruitment	13.37	±1.24	1.186
	Without recruitment	13.43	±0.87	1.191

4.3.3 Transportation time

As mentioned earlier in section 4.3.1, usually more than 90% of the transportations are carried out in time. In order to compare the effect of the recruitment process on transportation time, for each prey's mass, 50 trials were made, 25 trials with recruitment process and 25 without it. The results are given in Table 4.4.

Table 4.4: Transportation time of the prey.

		Transportation time (s)			
Prey mass (kg)	Strategy	Average	Standard deviation	Min	Max
m = 3	With recruitment	186.7 s	±25.6 s	133.3 s	232.6 s
	Without recruitment	169.6 s	±72.3 s	110.9 s	246.6 s
m = 4.5	With recruitment	331.8 s	±84.4 s	221.4 s	446.0 s
	Without recruitment	320.8 s	±62.7 s	228.2 s	402.2 s

At first, it might seem that the recruitment process increases the transportation time, around 12 simulation seconds, for the lighter prey. However, one should note that, although the recruitment process delays the start of the transportation, the standard deviation from the average transportation time is smaller. Therefore, one can expect the prey to always be transported in more specific period of time when the robots use the recruitment process. However, this is not the case with the heavier prey. This can be explained by the fact that for transporting the heavier prey, more robots need to push it, and by activating the recruitment process, one of the robot is excluded from pushing which can increase the overall transportation time.

4.3.4 Transportation path

One of the interesting and important part of the cooperative transportation algorithms is the transportation path. In ant colonies, especially in colonies efficient in cooperative transportation, the prey is usually transported in a straight way towards the nest. This can be achieved by well coordinated team effort. In this project, at each time step during the simulation, the position of the centroid of the prey is saved until the prey is in the nest. Some of the sample paths for different prey masses and strategies are shown in Figure 4.1.

One can see from the figure that the prey follows an almost straight path. Although it seems that, when the prey is heavier, the efforts of the robots are better coordinated than when the prey is lighter, a heavier prey requires more time to be transported and its path is longer. The reason is that it is easier for the robots to move the prey even when all of them are not pushing the prey in the same direction. Therefore, when the prey is heavy it will only move when a sufficient number of robots are pushing it in the same direction. Through simulation, it was observed that the robots faced sever trouble when the prey was close to the nest. In this situation the prey is also close to the arena's wall and, consequently, it is harder for the robots to push from same direction.

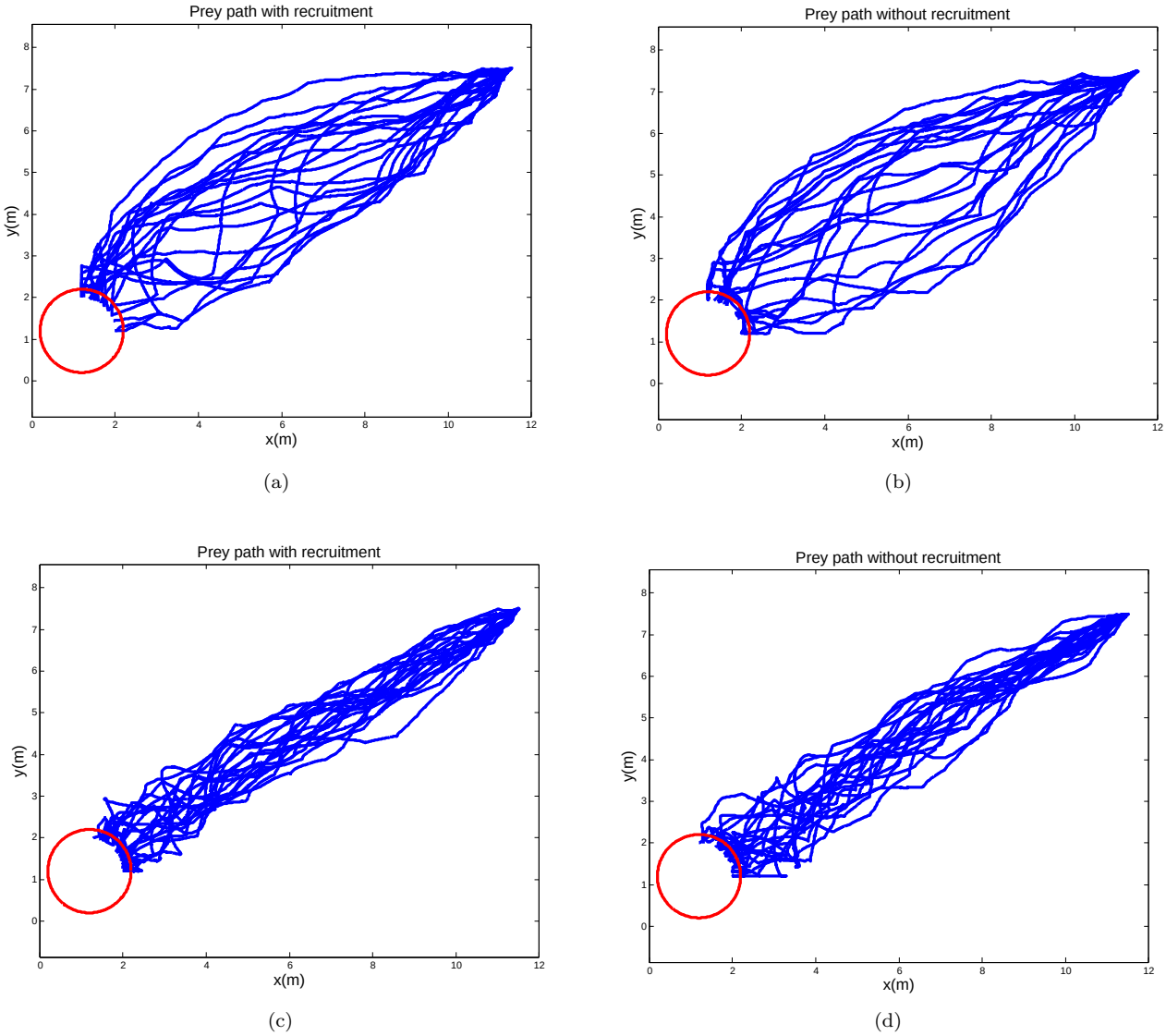


Figure 4.1: *Sample paths of the prey transported by robots from its initial position in the upper right corner in each panel, to the nest position shown as a red circle. (a) Sample paths of the centroid of the 3 kg prey, being transported by robots equipped with the recruitment process. (b) Sample paths of the centroid of the 3 kg prey, being transported by robots not equipped with the recruitment process. (c) Sample paths of the centroid of the 4.5 kg prey, being transported by robots equipped with the recruitment process. (d) Sample paths of the centroid of the 4.5 kg prey, being transported by robots not equipped with the recruitment process.*

4.3.5 Speed of the prey during the transportation

It is uncommon to include the speed of the prey, during the transportation, as an efficiency parameters. However, it can gives great insight about when the robots are highly cooperating in the transportation, i.e. pushing in the same direction. For example, in the study by Berman *et al.* in [2], when the ants passes the team organization phase, they can keep their team organized during the food retrieval process. However, from the simulation, it was observed that robots are not capable of maintaining their team coordination well due to several reasons. One of the important reasons that robots cannot keep the right formation is the physical interaction between robots and robots, and robots and the prey. Unlike the ants that do not push each other for the food transportation, robots need to do that which leads to robots being pushed out of their position. As it can be seen from the figure 4.2, the speed of the prey usually drops after a period of high cooperation, i.e. with high speed, to zero and robots need to relocate themselves so that they can move the prey again.

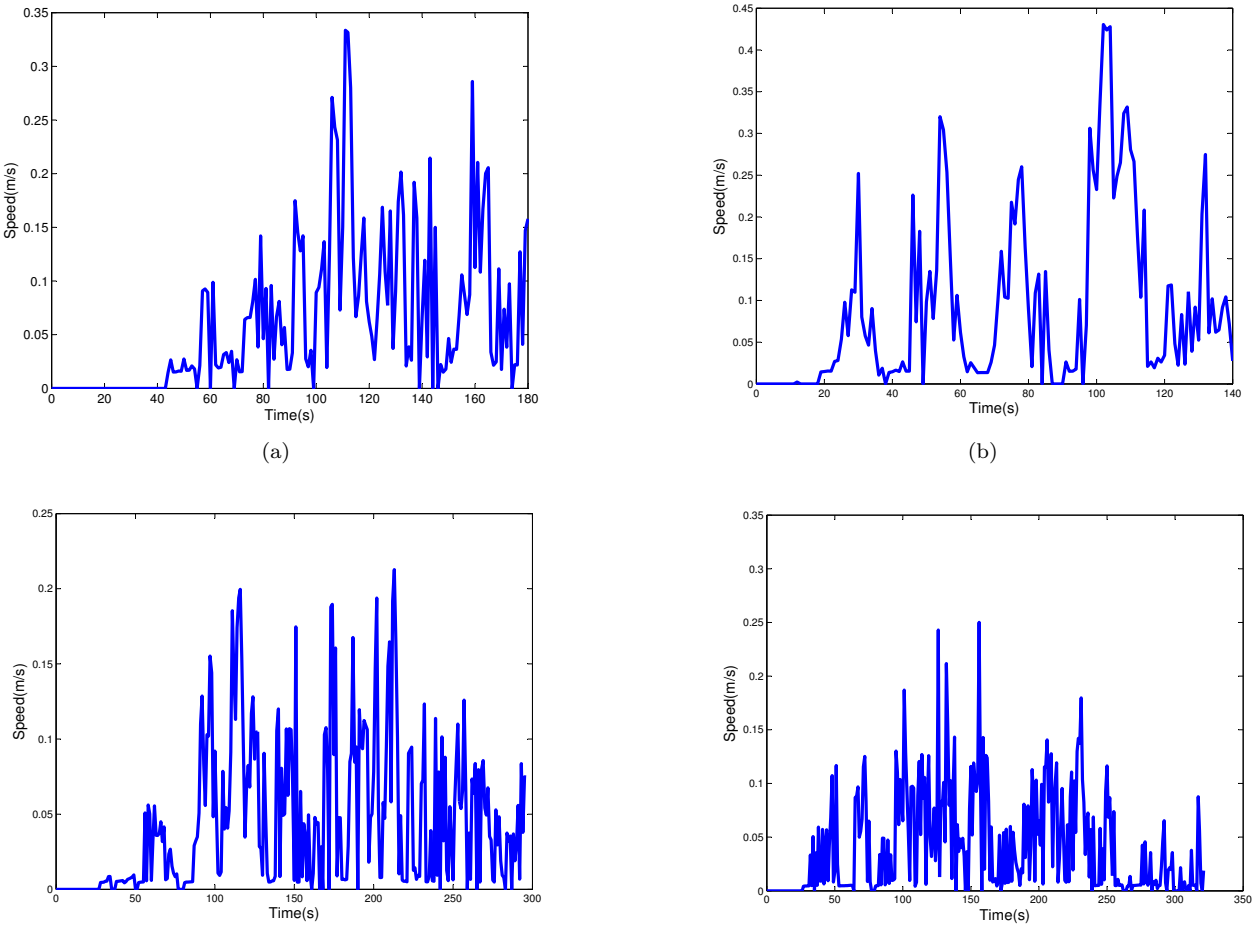


Figure 4.2: *Sample speed of the prey during transportation. As one can see, there are periods in which the speed of the prey is high which is an indication of a well-coordinated effort between robots. However, due to their physical interactions, robots are pushed out of their positions by other robots, something that forces them to find another position to push the prey. Therefore, the speed of the prey drops to a small value. In panels (c) and (d), the speed of a 4.5 kg prey is shown. Unlike the upper panels, the speed of the heavy prey fluctuates rapidly. This is due to the fact that for moving a heavy object, all robots need to apply their forces in the right direction. Therefore, if a robot is pushed out if its position, it can lead to a complete stop of the prey.*

5 Discussion and conclusions

5.1 Discussion

It is interesting to note that, even without communication or perfect sensors, organizing and coordinating the robots' efforts when they are transporting the object can be achieved using odometry and a short-range omni-directional camera. Unlike the procedure used in [25, 8, 15, 20], robots are not able to detect the nest (target location) all the time, and therefore they rely on their noisy knowledge of the nest's direction obtained by odometry. In addition, the implemented strategy is scalable since it does not use any communication devices.

As one can see from the simulation results, the recruitment process increases the transportation time. Although the standard deviation of the transportation time was smaller when the robots were transporting the lighter object, the recruitment process increased both transportation time and its standard deviation in the case of heavier objects. Several reasons can cause this drop in performance. For example, since there is no decision-making process involved in the recruitment process, there is always a robot performing the recruiting process without considering its necessity. However, there might be other possible recruitment processes by which one can improve the transportation efficiency.

There are other challenges that need to be addressed for an efficient transportation. For example, when robots are searching for the prey, they only use a correlated random walk that is guaranteed to result in prey localization in a reasonable amount of time only if the environment is bounded. On the other hand, in most real scenarios, robots need to operate in an unbounded or very large environment, something that requires more sophisticated approaches for the prey localization. Moreover, the recruitment process is inspired by the long-range recruitment used in ant colonies in which the ant moves back to nest directly and generally brings back three to five more ants. However, unlike in ant colonies, there is no guarantee that by going back to the nest, the recruiter would find more robots. Therefore, the recruitment process would often prevent a robot from participating in the transportation of the prey. Additionally, since the arena is relatively small, the chance of other robots to find the prey is higher by performing the correlated random walk than being recruited by a robot.

Another important factor to take into account is that the proposed algorithm should cope well with different arena sizes, varying number of robots, and different object shapes and sizes. Moreover, it is very important for the algorithm to be able to handle the obstacles, if any, in the arena while transporting the object. Robots should be equipped with behaviors enabling them to overcome any stagnations caused by pushing the prey against an obstacle. Furthermore, it would be interesting to enable the robots to push the prey in a specified path as a solution to overcome the stagnation problem and move the prey passed the obstacles.

5.2 Future work

There are many ways to improve the developed simulator. Several assumptions were made during writing of the simulator that should be addressed for a more general purpose simulator. First of all, it was assumed that all the objects are either rectangles or circles, and the physical interaction can occur only between circles and circles, or circles and rectangles. A more advanced simulator should enable one to include any convex objects in the arena.

As mentioned in the previous section, the odometry-based team coordination performed well. However, it can be improved by using local communication, such as IR-based communication between robots, combined with consensus-based algorithms. This could help robots to correct their noisy knowledge of the nest's direction and prevent them from wasting their effort in unnecessary directions.

The proposed recruitment process proved to be inefficient. In order to improve it, one can equip the robots with a decision-making behavior that can help them to decide if it is necessary to initiate a cooperative transport or not. However, one should note that including the decision-making process might require more computational power as well as complicated sensors. For instance, in ant colonies, the decision to initiate a cooperative transportation depends on the size and mass of the prey which can be easily estimated by ants. However, robots require sophisticated sensors for assessing these parameters. Another way to improve the efficiency of the recruitment process is to introduce different roles for robots, just like in ant colonies. For instance, there can be two groups of robots for transporting an object; one group of scouts and one group of transporters.

5.3 Conclusions

In this thesis, a collective algorithm was implemented in several mobile robots to transport an object. The algorithm enables the robots to detect the prey's location, coordinate their efforts, and move the prey in the right direction. The algorithm is fully decentralized, using only information available through the robot's sensors without any communication tools.

In this study, a different approach towards team coordination has been taken than in most previous work in which usually local communication based algorithms have been used. In this work, instead of using communication, which can limit the scalability of the system, robots rely on their odometry information and the assumption that they know the position of the nest. Using the coordination algorithm, the prey traveled about 1 m more than the euclidean distance between the nest and the prey's initial position (which is around 12 m). Moreover, it was shown that the recruitment process could not improve the transportation time.

Bibliography

[1] E. Şahin. "Swarm Robotics: From Sources of Inspiration to Domains of Application". English. In: *Swarm Robotics*. Ed. by E. Şahin and W. Spears. Vol. 3342. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 10–20. ISBN: 978-3-540-24296-3. DOI: 10.1007/978-3-540-30552-1_2. URL: http://dx.doi.org/10.1007/978-3-540-30552-1_2.

[2] S. Berman et al. "Experimental Study and Modeling of Group Retrieval in Ants as an Approach to Collective Transport in Swarm Robotic Systems". In: *Proceedings of the IEEE* 99.9 (2011), pp. 1470–1481. ISSN: 0018-9219. DOI: 10.1109/JPROC.2011.2111450.

[3] S. Berman et al. "Study of group food retrieval by ants as a model for multi-robot collective transport strategies." In: *Robotics: Science and Systems*. Citeseer. 2010.

[4] M. Brambilla et al. "Swarm robotics: a review from the swarm engineering perspective". English. In: *Swarm Intelligence* 7.1 (2013), pp. 1–41. ISSN: 1935-3812. DOI: 10.1007/s11721-012-0075-2. URL: http://dx.doi.org/10.1007/s11721-012-0075-2.

[5] R. Bridson, R. Fedkiw, and J. Anderson. "Robust treatment of collisions, contact and friction for cloth animation". In: *ACM Transactions on Graphics (ToG)*. Vol. 21. 3. ACM. 2002, pp. 594–603.

[6] S. Camazine. *Self-organization in biological systems*. Princeton University Press, 2003.

[7] A. Campo et al. "Negotiation of goal direction for cooperative transport". In: *Ant Colony Optimization and Swarm Intelligence*. Springer, 2006, pp. 191–202.

[8] J. Chen et al. "Occlusion-Based Cooperative Transport with a Swarm of Miniature Mobile Robots". In: *Robotics, IEEE Transactions on* 31.2 (2015), pp. 307–321. ISSN: 1552-3098. DOI: 10.1109/TRO.2015.2400731.

[9] E. A. Codling, M. J. Plank, and S. Benhamou. "Random walk models in biology". In: *Journal of the Royal Society Interface* 5.25 (2008), pp. 813–834.

[10] M. Dorigo. "SWARM-BOT: an experiment in swarm robotics". In: *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*. 2005, pp. 192–200. DOI: 10.1109/SIS.2005.1501622.

[11] J. Fink, M. A. Hsieh, and V. Kumar. "Multi-robot manipulation via caging in environments with obstacles". In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE. 2008, pp. 1471–1476.

[12] N. R. Franks. "Teams in social insects: group retrieval of prey by army ants (Eciton burchelli, Hymenoptera: Formicidae)". In: *Behavioral Ecology and Sociobiology* 18.6 (1986), pp. 425–429.

[13] R. Groß and M. Dorigo. "Evolution of solitary and group transport behaviors for autonomous robots capable of self-assembling". In: *Adaptive Behavior* 16.5 (2008), pp. 285–305.

[14] R. Groß and M. Dorigo. "Group transport of an object to a target that only some group members may sense". In: *Parallel Problem Solving from Nature-PPSN VIII*. Springer. 2004, pp. 852–861.

[15] R. Groß and M. Dorigo. "Towards group transport by swarms of robots". In: *International Journal of Bio-Inspired Computation* 1.1 (2009), pp. 1–13.

[16] B. Hölldobler and E. Wilson. *The Ants*. Belknap Press of Harvard University Press, 1990. ISBN: 9780674040755. URL: https://books.google.co.in/books?id=ljxV4h61vhUC.

[17] B. Hölldobler, R. C. Stanton, and H. Markl. "Recruitment and food-retrieving behavior in Novomessor (Formicidae, Hymenoptera)". In: *Behavioral Ecology and Sociobiology* 4.2 (1978), pp. 163–181.

[18] P. Kareiva and N Shigesada. "Analyzing insect movement as a correlated random walk". In: *Oecologia* 56.2-3 (1983), pp. 234–238.

[19] C. R. Kube and E. Bonabeau. "Cooperative transport by ants and robots". In: *Robotics and autonomous systems* 30.1 (2000), pp. 85–101.

[20] C. R. Kube and H. Zhang. "Collective robotics: From social insects to robots". In: *Adaptive behavior* 2.2 (1993), pp. 189–218.

[21] K. Lerman, A. Martinoli, and A. Galstyan. "A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems". English. In: *Swarm Robotics*. Ed. by E. Şahin and W. Spears. Vol. 3342. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 143–152. ISBN: 978-3-540-24296-3. DOI: 10.1007/978-3-540-30552-1_12. URL: http://dx.doi.org/10.1007/978-3-540-30552-1_12.

[22] A. Martinoli, K. Easton, and W. Agassounon. "Modeling swarm robotic systems: A case study in collaborative distributed manipulation". In: *The International Journal of Robotics Research* 23.4-5 (2004), pp. 415–436.

[23] H. McCreery and M. Breed. “Cooperative transport in ants: a review of proximate mechanisms”. In: *Insectes sociaux* 61.2 (2014), pp. 99–110.

[24] G. A. Pereira et al. “Cooperative Transport of Planar Objects by Multiple Mobile Robots Using Object Closure”. In: *Experimental Robotics VIII*. Springer, 2003, pp. 287–296.

[25] M. Rubenstein et al. “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*. AAMAS ’13. St. Paul, MN, USA: International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 47–54. ISBN: 978-1-4503-1993-5. URL: http://dl.acm.org/citation.cfm?id=2484920.2484932.

[26] E. Şahin. “Swarm robotics: From sources of inspiration to domains of application”. In: *Swarm robotics*. Springer, 2005, pp. 10–20.

[27] A. Sudsang and J. Ponce. “A new approach to motion planning for disc-shaped robots manipulating a polygonal object in the plane”. In: *Robotics and Automation, 2000. Proceedings. ICRA ’00. IEEE International Conference on*. Vol. 2. 2000, 1068–1075 vol.2. DOI: 10.1109/ROBOT.2000.844741.

[28] E. Tuci et al. “Cooperation through self-assembly in multi-robot systems”. In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 1.2 (2006), pp. 115–150.

[29] R. Vaughan. “Massively multi-robot simulation in stage”. English. In: *Swarm Intelligence* 2.2-4 (2008), pp. 189–208. ISSN: 1935-3812. DOI: 10.1007/s11721-008-0014-4. URL: http://dx.doi.org/10.1007/s11721-008-0014-4.

[30] M. Wahde. “A general-purpose method for decision-making in autonomous robots”. In: *Next-Generation Applied Intelligence*. Springer, 2009, pp. 1–10.

[31] M. Wahde. *Introduction to Autonomous Robots: Lecture notes in Autonomous Agents*. Chalmers University of Technology. Göteborg, Sweden, 2012.

[32] Z. Wang, Y. Hirata, and K. Kosuge. “Control multiple mobile robots for object caging and manipulation”. In: *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*. Vol. 2. 2003, 1751–1756 vol.2. DOI: 10.1109/IROS.2003.1248897.