Collective Foraging of Over-sized Payloads Using Robots in Swarms

Mehtab Iqbal[1*], Lei Chen[1]

[1] College of Engineering and Computing, Georgia Southern University, Statesboro, GA, USA
mehtab_iqbal@georgiasouthern.edu

***Abstract* – Swarm robotics is a very robust mechanism that can accomplish many complex tasks that require instant scalability or high fault tolerance. In this paper we want to propose how simple sensor based communication between robots can enable collaboration without sophisticated hardware or software and extract payloads of varying mass that require more than one robot to move.**

***Keywords* – Swarm Robotics, MarXbot, ARGoS, Simulation.**

I. INTRODUCTION

Robotics and artificial intelligence have experience tremendous advancement in the last couple of years, especially single agent systems [1]. Swarm robotics is not new but gaining a lot of traction in the recent years, and have provided very interesting use cases in military [2], construction [1], and navigation[3]. Swarm robotics inspired by nature's collective behavior towards achieving complex goals [4].

Due to the success of single agent systems, many have tried, and with some success too, towards scaling single agent models towards distributed multi agent systems [1]. However, what sets swarm robotics apart from a multi-agent robot environment is the fact that in a swarm, each individual robot perceive, communicate, and act based on its individual situation, and not a global information system or broadcast mechanism [4]. Swarm robotics provides prospects for cheap, redundant, fault tolerant system of robots which are highly scalable [5].

In leveraging the prospects of swarm robotics we would like to simulate a swarm robot system that is able to navigate in an environment with object payloads of varying mass. The goal is for the robots to collaboratively forage objects which require the combined strength of one or more robots to move.

II. LITERATURE REVIEW

A. Swarm Robotics

Swarm robotics is a fault tolerant multi robot system, that consists of simple robots in large numbers [5]. The idea is inspired by the collaborative nature of insects, and how very small generic entities in a swarm can perform very complex tasks [4], such as bees building a bee hive. According to Sahin [6] swarm robotics is useful due to its robustness, flexibility, and scalability. Swarm's robustness is derived from the multitude of redundant cheap robots which ensures that the swarm continues to function even if one or more entities fail [6].

Similarly, since the swarm is constituent of un-specialized entities, scaling a swarm should be as simple as introducing new entities into the swarm, without any specialized behavior or adaptability being a concern [5].

B. Simulation and Robot Development Environments

Over the years, robot development environments (RDEs) have become monumental in the field of robotics [7]. The taxonomy of available RDEs is indeed a gargantuan task and is not readily available. Kramer and Scheutz [7] in 2007 compiled a survey paper on the available RDEs and focused primarily on open source software. Some tools mentioned in their [7] survey have matured over the years and include primarily middle ware systems [8]. While many of these middle ware systems provide API which can be used for simulation testing, most focus on the code sharing between the development platform and the robot memory [8].

In the middle of all these developments, born out of the EU funded Swarmanoid project is ARGoS (Autonomous Robots Go Swarming) [9]. ARGoS was built with the goal of swarm robots simulations. ARGoS is extensible, has distributed physics engine, and has off the shelf simulated models of the MarXbot [10], Hand-bot [11], e-puck [12], and the Quadrotor [13].

III. METHODOLOGY

A. Simulation Environment and Robot Setup

For our research we used an open source multi-robot simulator called ARGoS [9]. ARGoS is only available on Linux or Mac OS X, and in our experimental environments we used Linux (Ubuntu 16.04) on a modest hardware set. ARGoS is extensible through C++ that allow writing controller programs for robot behavior, and loop functions for simulation behavior. Additionally, ARGoS uses Extensible Markup Language (XML) files with the extension ".argos" for experimental control points and configuration.

In this research, we extended the controller class to equip our simulated robots with simple behavior patters, that leverage their simulated actuators and sensors. We relied on the "footbot", which is a simulation of the MarXbot [10]. We utilized the sensors available to a MarXbot as shown in table 1, and consequently the actuators as shown in table 2.

Table 1: Sensors Utilized in this Project

Sensor	***Purpose***
Light Sensor	Detect light intensity
Proximity Sensor	Detect object and provide readings on their relative distance
Motor Ground Sensor	Detect ground intensity using Infra Red
Colored Blob Omnidirectional Camera Sensor	Detect light color from nearby sources

Table 2: Actuators Used in this Project

Actuators	***Purpose***
LED Actuator	Emit colored light using LED
Gripper Actuator	Grip or release object at the front of the robot
Differential Steering Actuator	Independent motor controlled bidirectional wheels

Each MarXbot is equipped with a light sensor which is able to detect light sources, and the intensity of the light. We rely on this light sensor to differentiate between the direction of exploration environment and our nest area. The proximity sensors uses diffusion vectors to calculate the relative distance between the robot in question and any objects in the environment, which may include other robots. The motor ground sensors are infrared sensors located at the bottom of the robot, situation in the front, back, left, and right side of the robots. These ground sensors can tell us the light intensity of the ground the robot is on. We rely on the ground sensors to tell us whether we are in the exploration environment (white or full intensity) or the nest area (gray or lowered intensity). Finally, The omnidirectional camera is able to detect light readings and determine the color of the light.

The actuators are mechanical components of the robot which allow it to interact with the (simulated) environment. Our experiments require three such actuators — LED, gripper, and differential steering actuators. Each robot uses the LED actuator to emit a green light when it is exploring the environment, or pulling a payload, and changes to red when it is stopped or requires assistance. The gripper is used to hold on to an object in front of the robot to be able to pull it to a different location. Finally, differential steering actuators are basically two wheels controlled by independent motors.

When both motors rotate at the same velocity, the robot moves forward or backward. However, when one wheel is moving, the robot turns in the direction at an angular velocity equal to the summation of the tangential speed at each wheel towards the direction of the tangents. In other words, when the left wheel turns backwards and the right wheel turns forward, the robot turns left, the converse is true for reversed rotation of the wheel. This allows us to control both the direction, speed, and the sharpness of the turns the robot can make.

In ARGoS loop functions is a class that enables customization of the experimental run. We used our custom loop function to create our arena which could provide ground feedback as shown in figure 1, the gray area is the nest and the white area is our exploration environment. We needed the loop function to do this, so that the ground material could be used to collect positional information. Additionally, since it is methods that can intercept before and after each execution tick in an experiment, it provided us with a lot of useful information, both for debugging and actual development of our bot controllers.

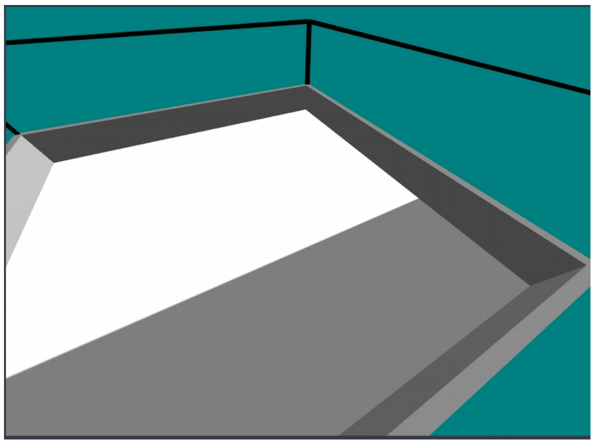

Figure 1: Arena Setup

Finally, each experiment setup in ARGoS is maintained by an XML file, or the Argos file. These files contain experiment simulation parameters to either pass to our controllers, and/or loop functions as configurations, and also the configurations that is used by ARGoS to run each experiment. The next section will discuss our experimental environment setup in more details.

B. Experiment Setup

Table 3 shows the components that needs to be set up in order for Argos to run an experiment simulation.

Table 3: Configuration Options

Configuration	***Purpose***
General	Simulation based properties, such as number of ticks, ticks per second, etc.
Controllers	Specify which controllers to use and nest parameters to pass to controller during initialization.
Loop Functions	Specify which loop functions to use and nest parameters to pass to loop functions during initialization.
Arena	Specify the arena size, object and robot distribution, light, etc.
Physics Engines	Specify which physics engine to use and provide custom parameters if needed.
Media	Media components that should be available to the simulation, these include LED, RADAR, LIDAR, etc.
Visualization	Rending mode to use for the arena along with camera setup for the view-port.

We ran several experiments each with their own varying configurations. For visualization, we used the default camera setup, and the QT rendering libraries. All our experiments had robots that depended on LED's in the environment and hence the LED type was also used as the default provided. Finally, we used the 2D physics engine provided by the ARGoS simulation system. All our experiments required that the robots move about and interact on the xy-plane and as a result there were no physics simulation required along the z-axis.

Our experimental setups can be grouped into 4 broad categories, each demonstrating a specific behavior pattern.

1) Foraging Experiment: for this we took one of the examples provided with the ARGoS source code and modified the environment to fit our experimental needs. The arena is a four sided area with nest area depicted in gray and exploration area in white as shown in figure 1. There are randomly scattered “food” objects in the exploration area, and 20 robots distributed in the nest area as initialization. The food objects are drawn in the environment using a loop function as a black patch on the floor. This helps with a simple demonstration because we can use the ground sensor reading to determine if the robot is on a food object instead of performing sophisticated collision detection.

The goal of the experiment is for each robot to avoid collision and explore the environment until it finds a food object and then return it to the nest are. A capture frame of the experiment is shown in figure 2.

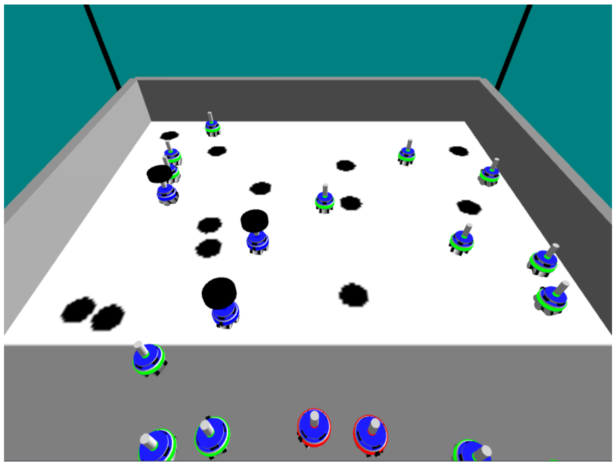

Figure 2: Foraging Experiment

2) Flocking Experiment: this is another experiment where we relied on the environment setup provided by the example source code in ARGoS source code. This experiment simulates the behavior of clustered exploration where the robots move towards a point in the environment by staying within proximity of one another based on the Lennard-Jones potential [14]. The interaction can be seen in figure 3.

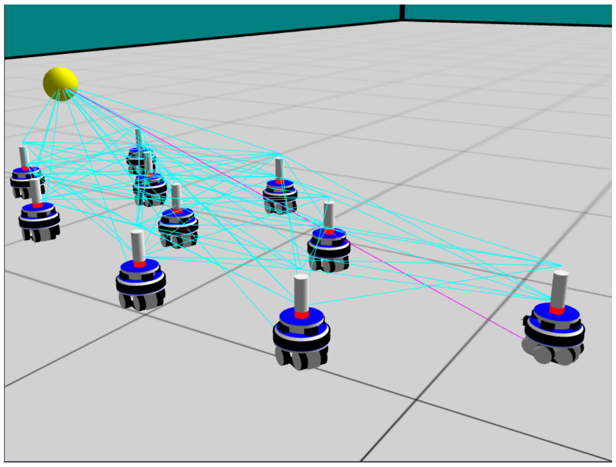

Figure 3: Flocking Experiment

3) Gripping Experiment: this is a simple experiment to test the use of the gripper actuator. The environment setup was a simple open spaced environment with one footbot and a cylinder of mass 5kg. The footbot moved forward in a straight line until it collides with the cylinder and then pull the cylinder back to its initial position. This experiment was repeated with varying masses assigned to the cylinder.

4) Putting it all together: finally we set up and experimental environment with objects (cubes and cylinders) scattered around the exploration area. There are two sizes and masses of both cubes and cylinders. Fifty percent all the cubes are of mass 5kg and other remaining are 50kg, similarly cylinders were also distributed based on the former ratio. Details on how the values were chosen would discuss in greater details in the result section of this paper.

The area setup is similar to the one shown in figure 1. Twenty footbots were distributed in the nest area to explore the environment with the goal of pulling the objects back to the nest. Figure 4 shows the experimental setup. Additionally, right behind the nest there are 4 light beacons that are setup to provide the footbots as a basis for the direction to follow out of the nest, or back to the nest.

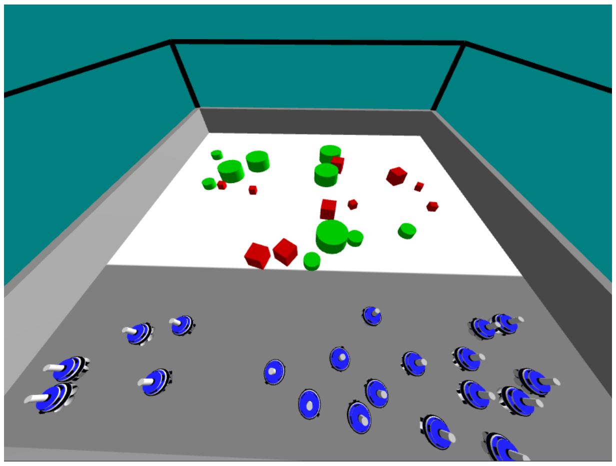

Figure 4: Area Setup for Varying Payload Sizes

C. Robot Controller and Behavior

Our goal was to keep the robot controller as simple as possible. Instead of building complex robots, our goal was to find the simplest possible cognitive algorithm that would allow our robots to perform our task at hand leveraging the strengths of swarm behavior. To that end for each of our experimental setup, we used incrementally more complex robots. For the first foraging experiment the behavior was very straightforwards, also since we designed a shortcut for our food objects to just be black patches on the ground, we can rely on our ground sensors to gives us a zero reading or a very close to zero reading to determine if we are on a food object or not. The following algorithm shows the simple decision-making process of the footbot in order forage food.

The algorithm already has provisions for more complexity by adding additional sensors for energy level of the robots when they are already in exploring mode. Similarly, in the flocking experiment the footbots light up their LED red. Then using the omnidirectional camera to detect each other, the Lennard-Jones force between each one of them is calculated. This causes them to flock together as they move towards a common goal without wandering away from the flock.

The behavior introduced in the gripper experiment is also straightforward, where a footbot moves towards an object (a cylinder), as soon as collision is detected using the proximity sensor, the gripping actuator is activated, causing the footbot to grip the object. Once the object is gripped, it pulls it back to its starting location.

The final experiment required more sophisticated robot behavior based on sensor stimuli. Each robot starts off at the nest and is set to explore mode, where they move away from the light at the back of the nest area. Additionally, each robot is equipped with a green LED which is initialized to be on at the beginning of the experimental simulation. When in explore mode, each robot move in a straight line until their proximity sensor reading provides readings for the relative distance of a possible nearby collision.

When a probable collision is detected, the robot uses its omnidirectional camera to detect for other robot's LED

```
if groundSensors = GRAY
  state ← explore
elif groundSensors = WHITE
  state ← explore
elif groundSensors = BLACK
  collectFood ← True
  state ← returnToNest
```

Algorithm 1: Simple Foraging Behavior

color. If the there is a possible collision with another robot whose LED is green, the robot changes its direction to avoid collision. However, if the LED is either red, or there is no led reading, the robot is within a proximity of either a movable object, or that of another robot who is awaiting assistance. The robot then moves towards this object (robot or obstacle) and grips the target object upon collision. If the robot is able to move the object, it moves it back to the nest.

If for some ticks the robot records no displacement in its sensors the robot changes its LED color to red and wait. This causes any nearby explorer footbot to move towards the other beacon footbot asking for assistance. Upon reaching the stalled footbot, the new footbot grips the stalled footbot to provide combined strength in the direction of the nest. The detailed algorithms is shown in algorithm 2.

```
if groundSensor = GRAY
   if gripper = True
      gripper ← False
   state ← explore
   wheelSpeed ← away from light
if groundSensor = WHITE
   if proximitySensor < threshold
      if omnicamera = GREEN
         changeDirection ← True
      else
         forward ← True
         gripper ← True
         state ← nest
         wheelSpeed ← towards light
         if displacement = 0
            led ← RED
         else
            if led = RED
               led ← GREEN
```

Algorithm 2: Collaborative Payload Extraction Behavior

In order to avoid milling in one position, the initial state starts of with swarm moving in a flock using the Lennard-Jones potential and thus maintaining a balance of both attraction and repulsion from one and another as shown in figure 5.

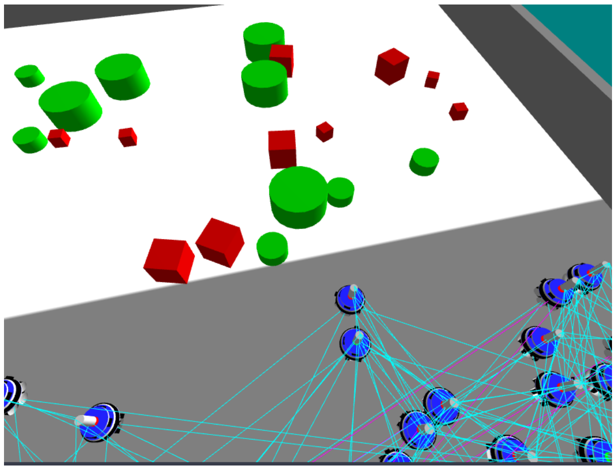

Figure 5: Flocking towards the Goal

IV. RESULTS

Since the first two experiments (foraging and flocking) were primarily focused as a test demo, we will not be discussing them in this section. However, the gripper experiment was tried with various mass to figure out at what mass does it necessitate a footbot to fail at simple foraging. Table 4 shows the 4 different masses we tried and the observed behavior from the robot.

Table 4: Gripper Experiment Results

Mass	***Observation***
1kg	Effortless movement
5kg	Effortless movement, the slowdown was almost negligible as far as velocity concerned.
10kg	Considerably slow movement nearing crawling velocity.
50kg	Absolutely no movement even after a long period.

Based on our findings from the gripper experiment, we decided to create our final environment with a mixture of 5kg (very easily moved), and 50kg (immovable). Our aim was to focus on the simple sensor based communication enabling each robot to switch from explorer to beacon mode by switching its LED color from green to red. This causes other footbots to flock towards the red beacon and provide assistance all the while not straying too far from the flock.

V. FUTURE RESEARCH

These experiments enable us to create a simulation model for swarm of footbots (MarXbot)s to work collaboratively towards foraging in an environment with object with varying mass. However, since the goal of this research was to see how simple communication can be achieved in these low memory robots, we did not consider the energy constraints involved. Each robot is battery powered and faces limitations on how long they can go about exploring. Additionally, activating the gripping actuator on heavier object drains more power than on a lighter object.

Taking the energy constraint into consideration so that the beacon state can be achieved well before a robot runs out energy will mitigate the chance of robots becoming obstacles themselves. Also, making the scenario more usable in a real life situation. In addition to the energy constraint, we also assumed that the introduction of early flocking behavior using the Lennard-Jones potential would provide enough distance between each individual robot, to reduce the probability of more than required robots accumulating at the beacon, giving rise to inefficiency.

REFERENCES

[1] G. Sartoretti, Y. Wu, W. Paivine, T. S. Kumar, S. Koenig, and H. Choset, “Distributed Reinforcement Learning for Multi-Robot Decentralized Collective Construction,” 2018.

[2] Y. Li, J. Klingner, and N. Correll, “Distributed Camouflage for Swarm Robotics and Smart Materials,” *ArXiv170907051 Cs*, Sep. 2017.

[3] K. McGuire, G. de Croon, and K. Tuyls, “A Comparative Study of Bug Algorithms for Robot Navigation,” *ArXiv180805050 Cs*, Aug. 2018.

[4] H. Hamann, *Swarm Robotics: A Formal Approach*. Springer, 2018.

[5] M. Salvaro, “Virtual sensing technology applied to a swarm of autonomous robots.”

[6] E. Şahin, “Swarm robotics: From sources of inspiration to domains of application,” in *International workshop on swarm robotics*, 2004, pp. 10–20.

[7] J. Kramer and M. Scheutz, “Development environments for autonomous mobile robots: A survey,” *Auton. Robots*, vol. 22, no. 2, pp. 101–132, 2007.

[8] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, “Rosbridge: Ros for non-ros users,” in *Robotics Research*, Springer, 2017, pp. 493–504.

[9] C. Pinciroli *et al.*, “ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotics,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 5027–5034.

[10] M. Bonani *et al.*, “The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 4187–4193.

[11] M. Bonani, S. Magnenat, P. Rétornaz, and F. Mondada, “The hand-bot, a robot design for simultaneous climbing and manipulation,” in *International Conference on Intelligent Robotics and Applications*, 2009, pp. 11–22.

[12] F. Mondada *et al.*, “The e-puck, a robot designed for education in engineering,” in *Proceedings of the 9th conference on autonomous robot systems and competitions*, 2009, vol. 1, pp. 59–65.

[13] J. F. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, “Quadrotor using minimal sensing for autonomous indoor flight,” in *European Micro Air Vehicle Conference and Flight Competition (EMAV2007)*, 2007.

[14] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, “Swarm robotics: a review from the swarm engineering perspective,” *Swarm Intell.*, vol. 7, no. 1, pp. 1–41, 2013.