ENABLING HUMAN SUPPORT OF ROBOT SWARMS

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computer Science

by

Vaidehi Hitesh Patel

May 2018

ABSTRACT

Autonomous robot swarms may offer efficient and parallel task execution compared to single robot systems. They are also more robust than a single robot, but not free of failures. It is difficult for a human to monitor the swarm, because the state space grows exponentially with the number of robots presenting a very high cognitive load. Moreover, errors in robot swarms propagate non-intuitively and it becomes harder to monitor and debug the swarm. Previous works have looked into enabling control of robot swarms by a human user. We instead present a framework that allows a non-expert user to support the robot swarm in the presence of errors by use of a GUI. It enables monitoring, verification and viewing analysis on both robot and swarm behavior. We believe that this framework is a step forward to moving swarms from lab settings to real-world scenarios.

BIOGRAPHICAL SKETCH

Vaidehi Hitesh Patel, a native of Surat, Gujarat, India, obtained her education till 12th grade in India. Since childhood, she was intrigued by math and science, especially physics. This interest led to interest in computers and technology, and she decided to pursue a career in the area. During 11th and 12th grade, she went to obtain higher level math, physics and chemistry education at Bansal Classes and Vibrant Academy in Kota, Rajasthan, India, which are renowned for their IIT-JEE coaching. She came to the United States to pursue BS in Computer Engineering and BS in Business Administration (Dual Major, Graduated: May 2016) at Georgia Institute of Technology, Atlanta, Georgia (Georgia Tech). At Georgia Tech, she also obtained leadership experience through teaching assistantship in Math Department and through serving as a team leader for various course projects. To further her knowledge in Computer Science, and due to an interest in AI, she chose to pursue MS in Computer Science at Cornell University, Ithaca, NY (Graduating: May 2018). While at Cornell University, she took courses focusing on ML, AI, NLP, pursued research spanning swarm robotics, HSI, data analysis and cognitive science. She also worked as a graduate TA for the CS department. After graduation, she will be moving to the Bay Area to work as a Software Engineer and Data Scientist and plans to obtain an MBA after a few years of work experience, to have a career in executive side of tech business.

This thesis is dedicated to my father, mother and grandmother, who have supported me in pursuing all my dreams. Without their teachings, guidance, and support, I would not have been the person I'm today. I thank them for believing in me and for making me their priority. I also dedicate this thesis to my advisors: Kirstin Hagelskjær Petersen and Bart Selman, who have been very supportive throughout the journey of graduate research. Without their guidance, I'd not have been able to complete my research. In addition, I'd also like to thank all the family members and friends who have been there for me.

ACKNOWLEDGMENTS

My committee chair, Kirstin Hagelskjær Petersen has been the ideal advisor for me. Her guidance, feedback, patience and encouragement tremendously helped me complete this thesis research. My minor advisor for Cognitive Science, Bart Selman has also been very supportive throughout and I value his time in serving as an advisor for me. Ryan O'Hern, a Ph.D in ECE candidate, and a member of the Collective Embodied Intelligence (CEI) lab, has helped me a lot throughout the research, and has completed parts of the proposed framework. Without his contribution, it would not have been possible to have the framework running. I'd also like to thank all the members of the CEI lab for their support and feedback.

Table of Contents

1. INTRODUCTION

Robot swarms (Fig. 1) are a group of simple robots that collaboratively work to achieve a user-specified goal. They achieve the collective behavior by local interactions with neighbors and interactions with the environment. They may achieve goals such as exploration, caging, and construction more efficiently than an individual robot because many can work on the same task in parallel. The emergence of complex global behavior from simple local interactions is called swarm intelligence. Such robot swarms are more robust than a single robot since there is no single point of failure that would totally break the system, in contrast to failure of a part in a single robot. However, the robots used in swarm robotics are often cheaper and therefore robustness of individual robots can become an issue. Robot swarms are scalable with the number of robots. They are flexible in terms of adaptability to changing conditions. The main characteristics of an autonomous robot swarm are as follows [1]:

Figure 1. A swarm of Kilobots from Harvard. Picture taken from: Bourque, Brad. "Harvard Researchers Have Created a Swarm of Learning Robots." Digital Trends. August 16, 2014. Accessed April 29, 2018. https://www.digitaltrends.com/cool-tech/harvard-kilobots-robot-swarm/.

i) The robots cooperate to work on a user-specified task.
ii) The robots are situated in an environment, which they can act upon.
iii) The robots only sense and communicate locally.
iv) The robots don't need any centralized control or global knowledge.

While robot swarms offer parallelism and better robustness, there are challenges involved with this approach. The system becomes harder to design and verify due to the lack of an intuitive mapping from local interactions to globally emergent behavior.

While robot swarms are more robust than a single robot, they are not free of failures. A classic argument for autonomous robot swarms is that small deviations in electronics and mechanics averages out [2]. But the robots are prone to failures related to mechanics, electronics, lost messages and noisy sensors and the failure of a few robots may affect the performance of the swarm and, in many realistic settings, be a liability. Adding human support for autonomous robot swarms could prove helpful in achieving the swarm task as the human can alleviate rare but inevitable errors. This, however, is challenging due to several reasons. There is a very high cognitive load associated with observing a robot swarm. Moreover, the state information available to the human might not be accurate or the robots might have incorrect information, due to faulty sensors, for example. It also takes expertise in dealing with the robots and understanding the intent of the swarm. Moreover, errors might propagate in non-intuitive or sometimes counter-intuitive ways and become amplified, clouding the root cause. These challenges hinder the process of moving robot swarms from lab to real-world applications.

We take a step forward in overcoming some of these challenges by proposing a decision support framework, OpenMind (Fig. 2), that uses a combination of statistical methods for analysis of data from the swarm. We present this analysis in an intuitive manner on a user-friendly interface that can be used by a non-expert user to evaluate and support the swarm. The name of the framework is chosen such to highlight its ability to reveal internal state of individual robots to the user. OpenMind framework contains:

i) a server to which the robots continuously communicate their internal state.

ii) an observer supported by an overlooking camera, that estimates the robot positions and orientations using computer vision.

iii) analysis tools, which models the robots and detects anomalies in the swarm.

iv) a graphical user interface (GUI) targeted at a non-expert user, displaying information about robots and the swarm in an intuitive manner.

We demonstrate the usefulness of the framework using a swarm of robots comprised of Elisa-3 robots from the K-team, that perform obstacle avoidance and flocking.

As previously mentioned, identifying errors in a swarm of robots is a highly complex cognitive task [3, 4]. The size of the state space to be debugged increases with the number of robots in the swarm. It involves simultaneous monitoring of multiple robots, which can be spread out in a wide area. Moreover, humans have a limited working/short-term memory and can only remember seven plus or minus two items at

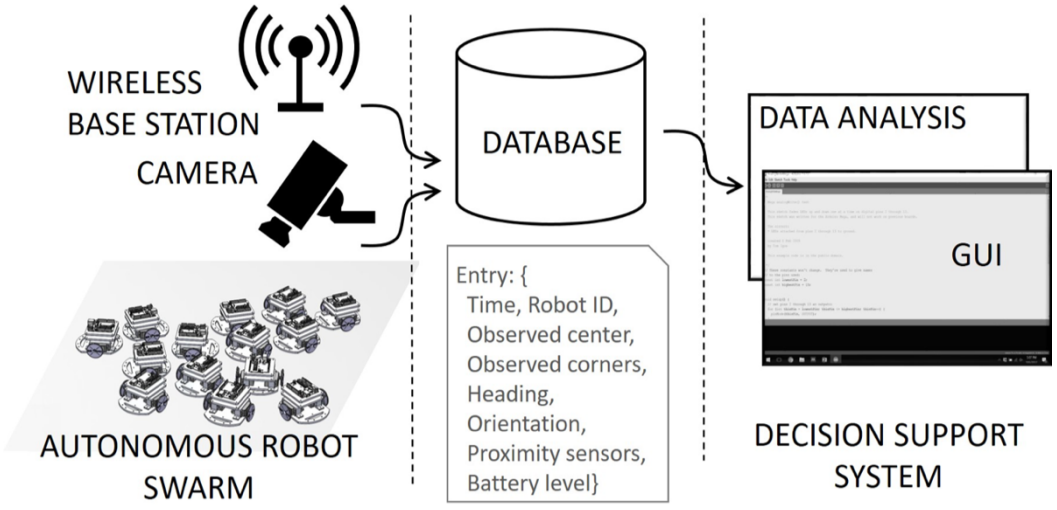

Figure 2. OpenMind Framework.

a time [5]. The environment involving a robot swarm is cognitively much more complex. Some properties like the history of the robots are hard to remember,

especially when all the robots look identical and when there are many robots. Interacting with robots also needs expertise in the domain. Thus, general operation and maintenance of autonomous robot swarms remain difficult, creating the need for integrated tools to augment the cognitive ability of humans to interact with collectives [6]. The work presented in this thesis contributes to solving this problem, by proposing a framework: OpenMind, to support robot swarms. It assumes little prior knowledge on the model of the robots. The framework enables the user to see discrepancies within the swarm and of the swarm over time, for example a robot out of battery or a misbehaving subset of robots. The user interface is designed keeping in mind a non-expert user and it is made highly intuitive and simple to use.

2. TAXONOMY OF ROBOT SWARMS AND RELATED WORK

The strength of robot swarms has been utilized in a number of research works. Robot swarms have different characteristics that determine their application and method of support. One possible classification of robot swarms is given in [7], attempting to divide swarms by properties like:

i) **size** *in terms of number of robots*. In tasks involving synchronization, speedups can be obtained using multiple robots, motivating a framework to support large number of robots.

ii) **communication** — *range* limited, *hierarchy* limited, *bandwidth* limited. Communication is a function of robot distribution, thus making it important to monitor the spatial locations of the robots.

iii) **reconfigurability** — speed with which swarm can reorganize, also related to the communication range. It can be looked at as the rate at which robots move with respect to other members of the swarm.

iv) **unit processing ability** — computation model used by the robots in the swarm.

v) **composition** — swarms may be composed of robots with different behavior and/or physical structure or of same type of robots.

Knowing these characteristics of robot swarm is useful in designing a framework for the support of the swarm. Research on real implementations of robot swarms is increasing, although most work is restricted to simulation [1]. Demonstrations include assembly of furniture [8], collective construction [9], and self-assembly where robots become the structure [10]. Other applications include assistance, also mentioned later, of firefighters [3] and in battlefield [4], which are more challenging environments. Works focusing on swarm flocking [11], formation [12], exploration [13], consensus

[14], coverage [15], transport [16] and construction [9] are some real applications, utilizing the aforementioned robustness, scalability, parallelism and flexibility of swarm of robots.

The field of swarm robotics also motivates the field of Human Swarm Interaction (HSI), focusing on how humans can interact with the robot swarms. Ability to integrate a human operator with a robot swarm becomes important for real-world applications of the swarms, especially for complex missions [17]. This is because the swarm can benefit by human operator even if the swarm is autonomous because the robot swarms are sensitive to failures. A faulty proximity sensor can confuse a robot, affecting the performance of the swarm. Similarly, a robot on a low battery might need to be replaced as soon as possible, to ensure efficiency. The human operator can [18]:

i) identify and alleviate shortcomings.
ii) make out-of-range information available, which can then be utilized to increase the performance.
iii) convey any change of task information, for example the goal. The robots otherwise lack this higher-level knowledge.

There are several challenges involved in human control of a robot swarm [17]:

i) the cognitive load associated with observing the robot swarm scales poorly
ii) the state information of the robots may not be accurate, due to faulty sensors
iii) it's difficult for a human to interpret the robots' intent
iv) the robots need to understand the intent of the human operator accurately

Research work in controlling robot swarms fall into three categories:

i) centralized control, allowing user to alter the overall goal of the swarm [6]

ii) intermittent control, allowing user to take over one or more robot in the swarm and thus affecting robots nearby [19-23]

iii) environmental controls, allowing users to place beacons in the environment, which are acted upon by the robots [6, 21, 22, 24].

There has also been research related to uncertainty and latency of feedback from the swarm, as well as execution error, delay, and stabilization time after reception of new commands by the robots [3, 17, 19, 25, 26]. In [26], it's suggested that predictive display can alleviate effects of latency.

There have been several works on control of robot swarms. Some of these use gesture-based control [27, 28, 29], or hands-free visual, auditory, or tactile feedback [30, 31, 32, 33], while others base the feedback and control on a graphical user interface [23, 34]. Ability to control swarms using gestures or other sensory input, becomes necessary in hazardous missions such as firefighting [3, 31, 34], battles [4] and removal of dangerous materials [20]. This is due to the remote and safe nature of these methods for the user, which allows for interaction without physical contact with the robot. In other environments, user interfaces may well serve the purpose of providing missing information to the swarm. There are two ways to look at a swarm of robots, dictated by task or the environment in which they operate:

i) a simple and confined environment, which can be monitored by an overhead camera and a GUI, with all the robots in the view.
Example: a work on swarm robots involving simple robots uses virtual environment to overcome hardware limitations in providing information about the environment to the individual robots [35]. The information about location is communicated to robots via an overhead controller and a GUI is used to

provide users a means to configure the system and gain feedback on progress of experiments [35] (Fig. 3).

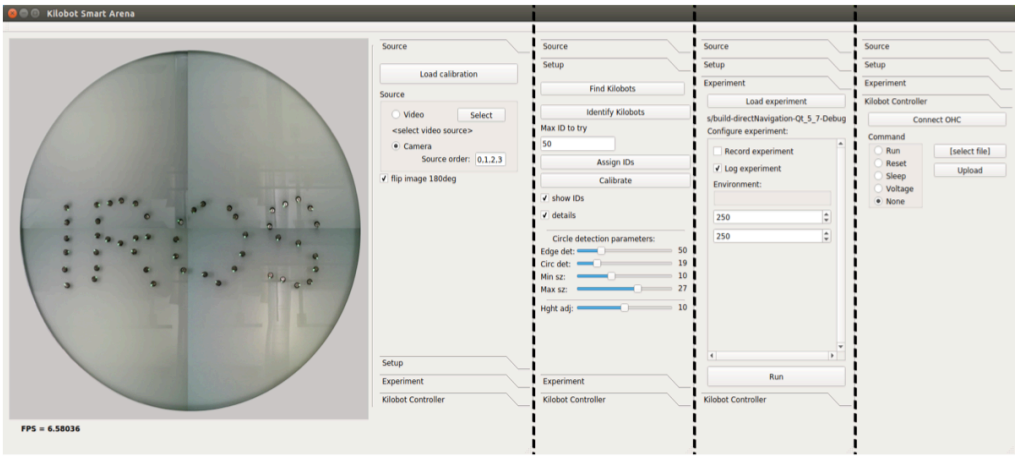

Figure 3. GUI used to control a robot swarm. Picture taken from [35].

ii) complex environments, in which only a part of the swarm is visible. Example: [27] takes a gesture-based approach to control the function of the swarm, focusing on applications to entertainment robotics (Fig. 4). They divide the control into two categories: a) controlling a subset of robots directly to complete a goal or trajectory, b) controlling the shape of the swarm. The user can control the position, movement and color of the robots using an interactive display. The applications of this approach also extend to search and rescue, as the user could direct individual robots to define teams for specific tasks like surveillance. It could also be extended to adjust parameters of coverage in real-time.

Figure 4. Gesture based control of robot swarms. Picture taken from [27].

In contrast to these, our focus is on enabling support of robot swarms, not control. We present a framework that automatically makes informed models about both the robots and the swarm. These models are not hardware specific, like most previous works on control, and assume little prior knowledge that the robots follow Markov decision process. While the OpenMind framework is more suitable for a simple environment, which can be observed by an overhead camera, it may also apply to more complex environments, where only part of the swarm is visible. Moreover, we target non-expert users, who can use the framework to support the robot swarm.

3. DESIGN OF OPENMIND

3.1 SETUP

To demonstrate the concept presented in this thesis, we used Elisa-3 (Fig. 5) robots by the K-Team.

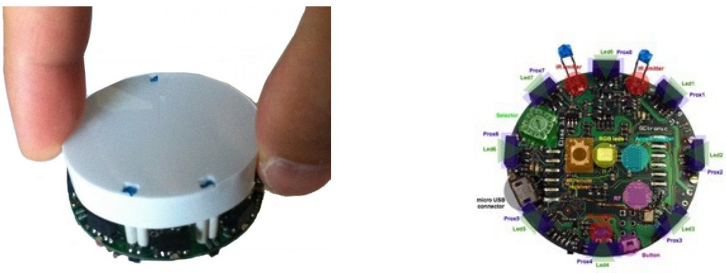

Figure 5. An Elisa-3 robot. Picture taken from: Elisa-3 Mobile Robot in Swarms - Génération Robots. https://www.generationrobots.com/en/203-elisa-3.

Here are some characteristics of these robots, which are also a good general model for robots used in swarms:

- The robots are 50 mm in diameter, small enough for lab operation.
- They have a small processing power (Atmel ATmega2560 @ 8MHz; 8 bit microcontroller) and memory (RAM of 8 KB, flash memory of 256 KB and EEPROM of 4 KB), which is representative for many robots used in swarm robotics.
- They use differential steering. The maximum speed with which these robots can move is 60 mm/s.
- IR sensors are used for local sensing and communication measuring ambient light and proximity of objects (8 IR sensors, measure upto 6 cm of proximity, placed 45 degrees apart). There are 3 IR emitters on the robot, 2 on the front-side and 1 on the back-side of the robot.

- The robots are also equipped with 4 ground sensors on the front, detecting the end of the viable surface.
- For signaling the user, the robots have 1 RGB LED in the center and 8 green LEDs in the periphery, but it would take an expert to interpret the meaning and it doesn't scale well either. We use it for debugging the system in progress.
- The robots have wireless communication, with RF 2.4 GHz, using Nordic Semiconductor nRF24L01, which we use directly for updating the OpenMind database.
- The robots can be programmed using C/C++ with AVR-GCC compiler. There is a 16-position rotating switch/selector for selecting appropriate programs.
- The robots charge themselves using a charger station, which makes operation manageable.
- The micro USB connector can be used for programming, debugging as well as charging the robots.

The robots do have orientation sensor, but the data is very noisy. We utilize AprilTags (Fig. 6) in this work for extracting the position and orientation of the robots more accurately via an overhead camera. They are a type of visual fiducial to provide better localization accuracy. These visual fiducials have a small information payload and are designed to be automatically detected and localized even in low resolution, unevenly lit, oddly rotated, or cornered in a cluttered image conditions [36]. There are two components to it: detection and coding. The detector finds the quads having a darker interior compared to the exterior. The detection starts with detecting lines in the image. The coding system determines if the

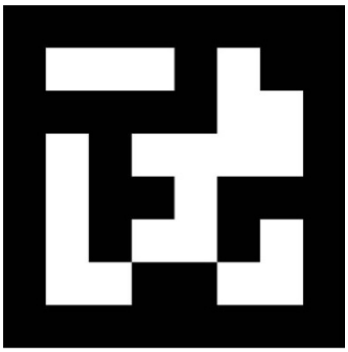

Figure 6. An AprilTag sample from 36h11 family. Picture taken from: https://april.eecs.umich.edu/software/apriltag.html.

data decoded from the quad is valid or not. To facilitate system integration, we use the AprilTags C-library [36] to track the robots and wrap this in Python using ctypes.

Visual feedback for the framework is obtained using a Logitech C920 HD webcam, which is mounted above a 1x1 m^2 arena. The wireless ground station uses Crazyradio PA USB radio dongle, as it is supported by a C-library provided by GCTronic (Fig. 7).

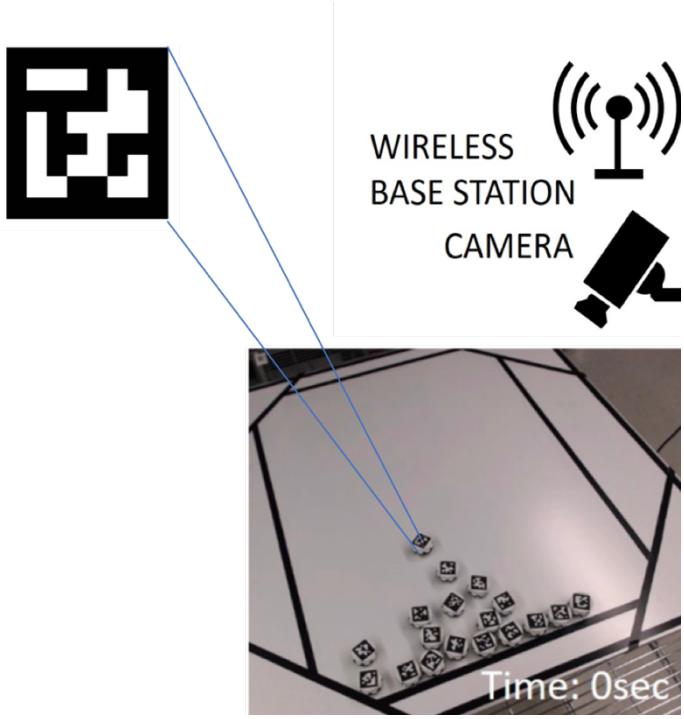

Figure 7. Robot Arena. April tag sample obtained from https://april.eecs.umich.edu/software/apriltag.html.

We modified the C-library for supporting transmission and reception of messages and wrapped it in a Python class. The arena is white in color, with black tapes for the boundaries, which provide contrast to detect the boundary. We used tapes because the robots would mistake physical barriers for other robots and get affected in performance. The arena is large enough for a swarm of 18 robots. For the purposes of

demonstration, we used a platform where all the robots are observable by the overhead camera, but the framework is built to be easily adaptable to other types of robot platforms. As mentioned before, a full view of the arena of swarms is not assumed. The framework can support even a part of the swarm, which is visible.

3.2 SYSTEM ARCHITECTURE

We implemented the OpenMind framework on a MacBook Pro (13-Inch, 2017), 3.5 GHz Intel Core i7, 16 GB 2133 MHz LPDDR3, Intel Iris Plus Graphics 650 1536 MB. We use Python for all the codes (with wrappers for C-libraries), because of its readability, flexibility and popularity. Python supports high quality libraries for data analytics and machine learning, for example Scikit-learn, Numpy.

The OpenMind framework contains four entities:

i) **Data collection:** The data collection is composed of visual feedback obtained using the overhead camera and wireless base radio station. The captured images are used by the base station to automatically find the AprilTags and thus the orientation and position of the robots. Other data like proximity and battery readings is then communicated to the base station over a radio.

ii) **Data storing:** The collected data is stored in a MongoDB database. We chose MongoDB for its user friendliness, Python support, and the absence of a defined database schema. The absence of a defined schema allows for the framework to be adaptable to addition of new information from different robot platforms, for example accelerometer data. This enhances applicability of the OpenMind framework towards a general collective of robots. Fig. 8 shows a sample entry from the database.

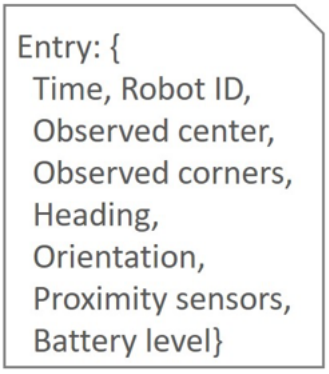

Figure 8. A sample entry from the database.

iii) **Data analysis:** The data stored is then analyzed by the data analysis module, using various statistical methods, described next.

iv) **Graphical User Interface (GUI):** The final module is the GUI. For programming the GUI, we used PyQt5, which is a Python library for the multi-platform application and graphical user interface framework, Qt. The GUI contains a live video of the robot arena. It enables the user to select the relevant information to be overlaid on this live video. The information could be either of the swarm as a whole, or a subset of it. The information from the analysis module is presented on the GUI in a very intuitive manner, keeping in mind the cognitive complexity and a non-expert user.

A key component of the OpenMind framework is the automated analysis, which removes some of the cognitive load for the user. This is an exploratory work and we only present a few methods to demonstrate the idea. More analysis can be added to provide useful insights. The user feedback can be divided into four modes:

i) ***State Monitoring*:** The framework lets the user to have a quick visual assessment of the swarm parameters. For example, the user might want to

know which robots need to be removed based on their battery levels. The other parameters are described in detail later, along with the GUI.

ii) ***State Verification*:** We determine malfunctions by comparing belief and actual outputs of the components in the swarm. This is often related to bias and noise in the sensors. The data from proximity sensors is used to build a model, which is then compared to ground truth data, which is computed by using the visually observed robot positions (Fig. 9). The framework alerts the user by pointing out any sensor value which has output above two standard deviations from the mean. The model improves over time, as more data becomes available.

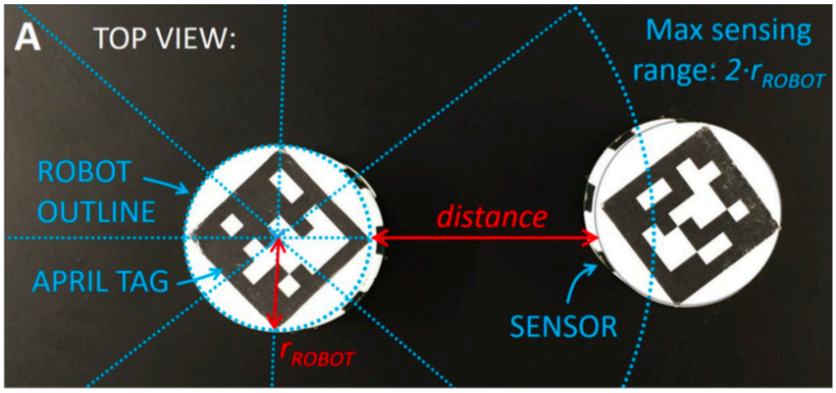

Figure 9. Distance of closest obstacle to a sensor can be measured by knowing the maximum sensing range of the robot and projecting a line from the center of the robot passing through the sensor.

iii) ***Modeling and Analysis of Robot Behavior*:** It is very difficult for a user to perceive if the robots are performing the task optimally. The robots that are causing problems, might not be very obvious to the user. The OpenMind framework uses the information from the database to build a stochastic behavioral model of the robot behavior and is able to highlight outliers.

iv) ***Modeling and Analysis of Swarm Behavior*:** It's difficult to monitor the performance of the swarm as a whole. Having performance measures for the collective is useful in determining if the swarm is working towards the task. Entropy is a measure of randomness in a system. Lower entropy means higher order in the system. Here our system is the swarm of robots, performing flocking. Inspired by [37], we use entropy as a measure of evaluation of the flocking task. We use locational entropy, which is calculated by first clustering robots within a certain distance. The scaled locational entropy is then calculated as follows: $S = -\frac{\sum_{i=1}^{k} p_i \log p_i}{\log R}$, where k is the number of clusters and

p_i is the probability of a robot being in the cluster i, R is the total number of robots in the swarm. The maximum possible entropy is $\log R$, which is when each robot is in a cluster of its own. Thus, maximum entropy is a function of just the number of robots. The denominator in the formula for scaled locational entropy is the maximum possible entropy. We use scaled locational entropy so that we can show it as a percentage, which is more intuitive for the user.

To make the framework work in real-time, and display the live video in the GUI, we used MongoDB GridFS to store images captured by the overhead camera, and then fetched the corresponding images and the robot data. The GUI then displays the images with robot data overlaid. An average of 30 frames per second are processed in the current system. We empirically found that to avoid the lag while displaying the images as a live video in the GUI, using every 3rd frame is efficient. While a live video feed from camera could be used for the GUI, it's not easy to implement, as the camera is also used for the detection of the AprilTags.

The GUI's main features are its ease of use and intuitive nature. It contains several display component options (Fig. 10) that can be overlaid on the live video. The GUI can display the information for the entire robot collective, or for just a subgroup, which can be selected by entering the desired robot Ids in a text field provided, separated by commas or spaces. If this field is left blank, then the information is shown for all robots in view. This only applies to Centroid, Path and Sensor. For the other components, it only makes sense to have the data for the entire collective. Alternatively, the information for all robots can be enabled or disabled by checking or unchecking its respective checkbox. The display components mentioned before can be enabled or disabled by checking or unchecking their respective checkboxes.

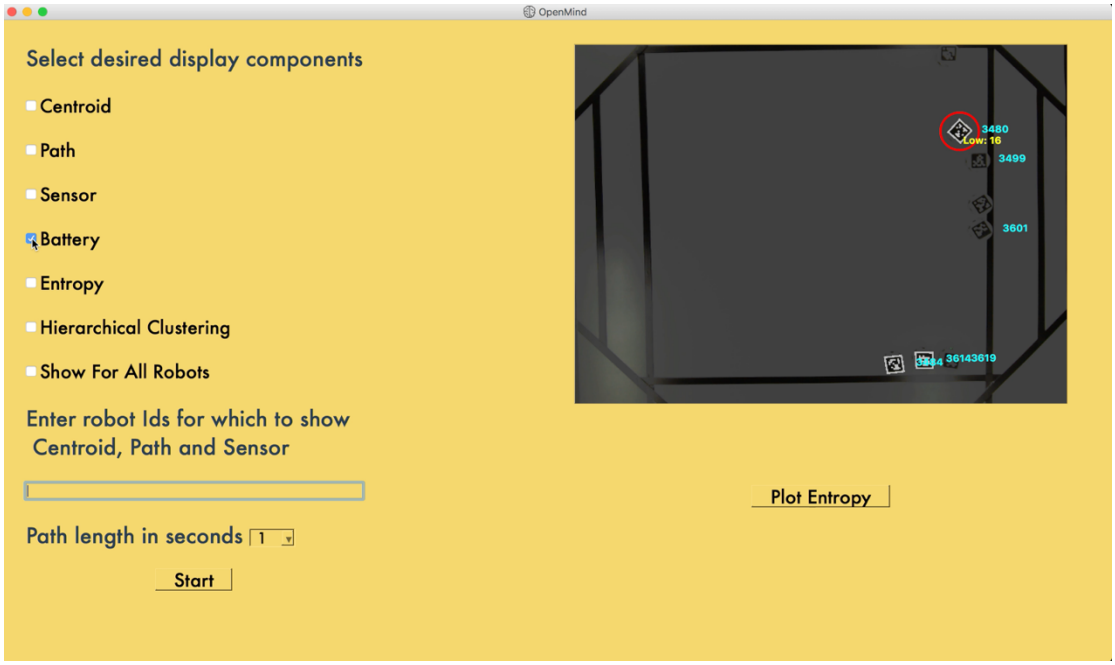

Figure 10. GUI display components selection.

Here are examples showing the different components:

i) **Centroid** of the robots selected (Fig. 11): this feature lets the user observe how a subgroup moves over time. Observing the centroid gives intuitive information about the emerging organization of the group. If the centroid jumps abruptly and looks flickering, it suggests that there is quite a lot of randomness in the group, while if the centroid moves smoothly, it suggests that there is some organization within the subgroup.

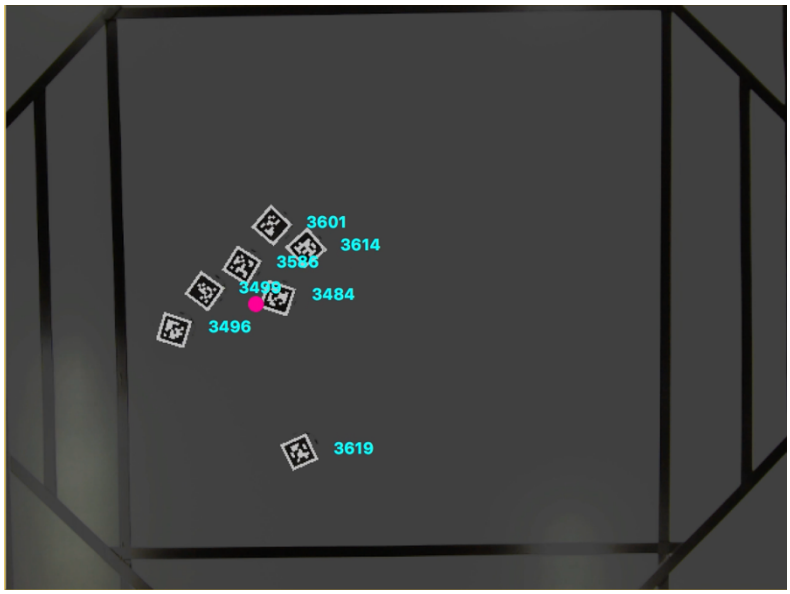

Figure 11. Centroid of the robots.

ii) **Path history** of the robots (Fig. 12), which can be further specified by a drop-down button with path lengths varying from 1-10 seconds. The default is 1 second of path history. This is especially helpful for the user, as it is almost impossible to remember any path history for any robot at all, given the large number of robots. The GUI lets the user observe path histories for multiple robots, while also letting the user select the amount of history to show (in seconds).

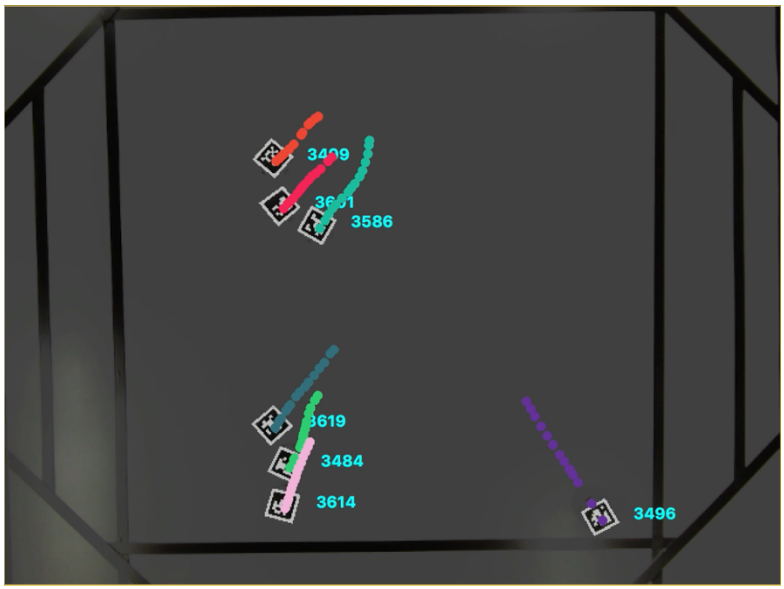

Figure 12. Path histories of the robots.

iii) Color coded IR sensor **proximity readings** (Fig. 13). The proximity readings are in the range of 0-1023; the higher the reading, the closer the object. The values are divided into three classes based on experimental results: None: 0-4, Far: 5-35, Near: 35-1023. The colors used for these classes on the GUI are shades of red, lighter to darker corresponding to lower to higher proximity values. This selection of the

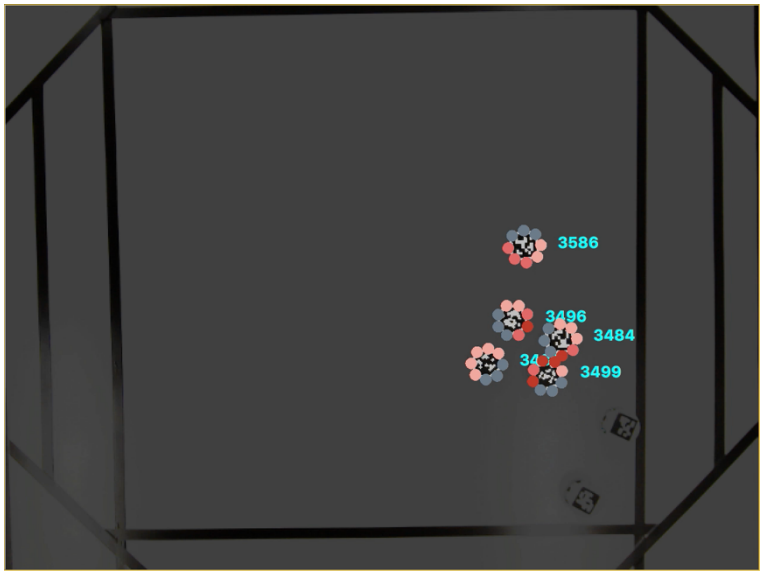

Figure 13. Color coded proximity values.

representation lowers the cognitive load, as it's not as important for the user to know the exact proximity reading, but just the range it falls in. The color coding allows the user to easily determine if a sensor is giving the expected proximity reading or not. As can be seen in the figure, some sensors are represented with gray color. These are the back sensors of the robots and are grayed out because they do not provide reliable readings, as will be mentioned later.

iv) **Battery**: selecting this option highlights the robots with battery level lower than 20% (Fig. 14). This feature directs the user's attention to the robots running on a critically low battery level, so that the user can remove the robots from the swarm if desired. As can be seen in the figure, the AprilTag detection is not perfect and some tags are not registered in a given frame. However, over 3 frames of data, almost all tags are present. We use the data over past three frames to get rid of such discrepancies.

Figure 14. GUI displaying robot with low battery.

v) **Hierarchical Clustering** (Fig. 15) shows the robot clusters based on behavior. This feature lets the user see the robots that behave differently from the rest of the collective. Details on this are omitted as it was implemented by Ryan O'Hern, and only the big picture is mentioned. Each proximity sensor has 1024 possible values. As will be described later, the back three sensor values are ignored, as they are mostly zero. Thus, we have a total of $(2^{10 \text{ bits}})^{5 \text{ sensors}} = 2^{50}$ possible input states. We manually restrict this to 27 input states by combining sensor values: three left-most sensor values are combined into one by taking their maximum, and three right-most sensor values are combined similarly. The forward-facing sensor is considered alone. The output values are separated into three bins: 'none', 'far', and 'near'. Turning is also discretized: 'left', 'right', 'none'. Robot speed is combined into three classes as well: 'slow', 'normal', 'fast'. Using this input-output model, the aggregate behavior of the robot as a combination of probability distributions over the output states is learned. The probability distribution matrices for the robots are collapsed into vectors, and cosine similarity is used to compare the behavior of two or more robots (Fig. 15A).

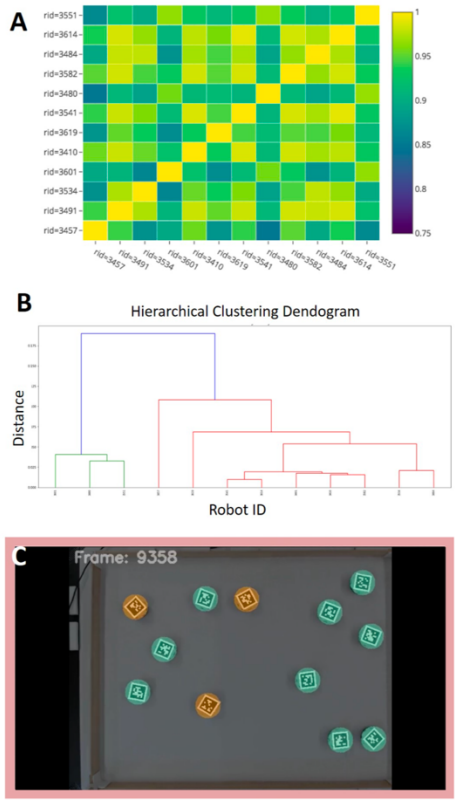

Figure 15. Hierarchical clustering. A: similarity scores, B. dendogram, C. GUI.

vi) **Entropy** selection starts the computation of the scaled entropy of the swarm as a whole (Fig. 16). Enabling this also shows the robots which have not been in any cluster for past 45 samples, which is about 4.5 seconds, determined empirically for this task. In addition, the clusters are shown with links and with robots in the same cluster colored with the same color. The GUI also provides

an option to plot the entropy, through a button, which pops up a graph with the frame number on the X-axis and the Entropy % on the Y-axis.

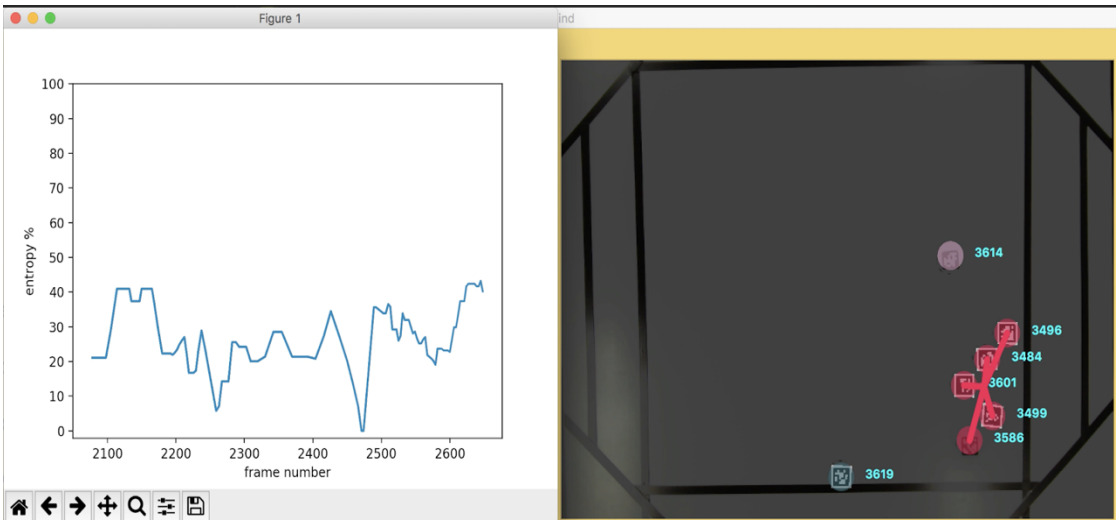

Figure 16. Entropy, showing the clusters used for calculation.

OpenMind framework is adaptable to the needs of a robot platform and can display more or less components. It would take someone with coding skills to modify the code and change the information displayed on the GUI. However, it should be noted that, the operation of the framework doesn't require such a user. For that matter, we assume a non-expert user. To add a new component, one needs to add the corresponding checkbox, the corresponding method required to display that information, and to add a line in the main loop that determines whether or not to display the information based on the checkbox state. To add checkbox, add the relevant code in the method named 'checkboxes' (Fig. 17). Then call the new method from the method 'onStart', after checking the state of the checkbox. Similarly, modifications to the color choices can be made as well. Buttons can be added for displaying more type of graphs, as desired. We keep our focus on the proposed idea, and don't dive deep into different possible

graphs such as battery levels over time, or centroid position over time, which can be used to perform more analysis on the data.

```
305
306     def checkboxes(self):
307         self.checkboxCentroid = QtWidgets.QCheckBox(self.leftWidget)
308         self.checkboxCentroid.setText('Centroid')
309         self.checkboxCentroid.stateChanged.connect(
310             lambda:self.checkboxState(self.checkboxCentroid))
311
312         self.checkboxPath = QtWidgets.QCheckBox(self.leftWidget)
313         self.checkboxPath.setText('Path')
314         self.checkboxPath.stateChanged.connect(
315             lambda:self.checkboxState(self.checkboxPath))
316
317         self.checkboxSensor = QtWidgets.QCheckBox(self.leftWidget)
318         self.checkboxSensor.setText('Sensor')
319         self.checkboxSensor.stateChanged.connect(
320             lambda:self.checkboxState(self.checkboxSensor))
321
322         self.checkboxBattery = QtWidgets.QCheckBox(self.leftWidget)
323         self.checkboxBattery.setText('Battery')
324         self.checkboxBattery.stateChanged.connect(
325             lambda:self.checkboxState(self.checkboxBattery))
326
327         self.checkboxEntropy = QtWidgets.QCheckBox(self.leftWidget)
328         self.checkboxEntropy.setText('Entropy')
329         self.checkboxEntropy.stateChanged.connect(
330             lambda:self.checkboxState(self.checkboxEntropy))
331
332         self.checkboxHierarchical = QtWidgets.QCheckBox(self.leftWidget)
333         self.checkboxHierarchical.setText('Hierarchical Clustering')
334         self.checkboxHierarchical.stateChanged.connect(
335             lambda:self.checkboxState(self.checkboxHierarchical))
336
337         self.checkboxShowAll = QtWidgets.QCheckBox(self.leftWidget)
338         self.checkboxShowAll.setText('Show For All Robots')
339         self.checkboxShowAll.stateChanged.connect(
340             lambda:self.checkboxState(self.checkboxShowAll))
341
342         listCheckboxes = [self.checkboxCentroid, self.checkboxPath, self.checkboxSensor, self.checkboxBattery, self.checkboxEntropy
343         [self.vbox1.addWidget(x) for x in listCheckboxes]
344
```

Figure 17. Code snippet showing where to add more checkboxes.

4. EVALUATION

4.1 ROBOT TASK

For the development of the OpenMind framework, we implemented two types of algorithms on the robots as when it comes to evaluating a swarm performance, both individual and collective behaviors are crucial. To evaluate the use of the framework for collective behavior, we used flocking, which was implemented by Ryan O'Hern, and I integrated it with the framework. To evaluate the performance on individual behavior, we used obstacle avoidance.

Obstacle avoidance is important in any swarm application, as the robots need to prevent collision with each other and with foreign bodies in the environment they are operating upon. Almost all robots, individuals or swarms, depend on obstacle avoidance. The proximity sensors have values ranging from 0-1023. We treat proximity values of less than 5 as noise. Proximity values are combined and the speed of the motors is set according to the situation: if the obstacle is straight ahead, the robot stops, if it is on the right side, it turns left, if it is on the left side, it turns right.

Flocking has been studied in the field of swarm robotics, inspired by flocks in nature, such as that of birds and fishes (Fig. 18).

Flocking (Fig. 19) is comprised of three components:

i) ***Repulsion:*** to prevent collisions between robots.

ii) ***Cohesion:*** to attract robots toward the average center of all other robots in the view.

iii) ***Alignment:*** to match the heading of a robot to its neighbors.

Figure 18. A flock of birds. Picture taken from: "LIKE THE BIRDS." Positivity Practices. July 13, 2011. Accessed April 13, 2018. http://positivitypractices.com/like-the-birds/.

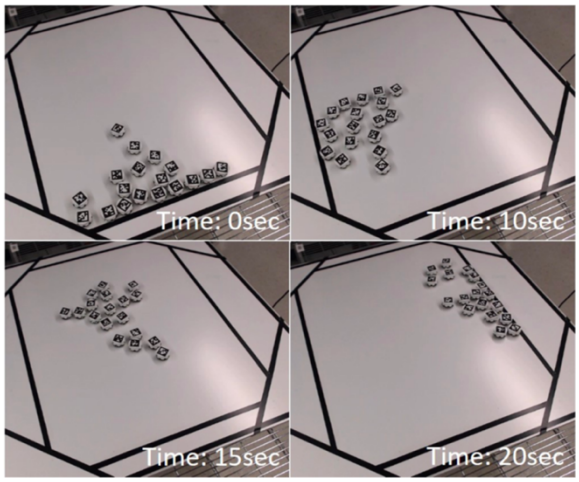

Figure 19. Flocking performed by Elisa-3 robots.

Attraction and repulsion were generated by combining the values of forward five proximity sensors into one parameter, where the front facing sensor had the most weight. Alignment is achieved in reasonable time using the AprilTags detection. The robots get informed of the orientation of their neighbors (located within one robot diameter) over the radio, which they then use to adjust their flocking parameters accordingly. Use of AprilTags becomes especially important for the purpose of flocking because it allows for accurate sensing of neighbor orientations. This is because the location and orientation accuracy are important for an effective flocking of the swarm, as the robots rely on knowing the orientation and positions of their neighbors.

Flocking is an emergent behavior, which ranges from fully ordered to disordered motions. It would put a high cognitive load on the user to monitor this behavior. The global behavior of the swarm may be altered by a large amount if there are errors in multiple robots due to low battery levels, proximity sensors, or faulty information regarding the positions of their neighbors. Noise in sensor readings could lead to oscillatory behavior over long time spans. In general, all the components are prone to failure in unexpected ways, which lead to counter-intuitive and non-obvious global behaviors. The robots that completely malfunction will be ignored by other robots in the swarm, not affecting the global behavior much. However, in the other more likely cases, mentioned above, OpenMind helps the user to support the swarm, and flocking is an interesting algorithm to look at when designing such a framework.

4.2 PERFORMANCE

In the initial stages of development, the analysis module of the OpenMind framework helped us discover an issue with the back-sensor proximity values by comparing it with actual states – distances, which can be measured as shown in Fig. 9. When we

looked at the plot of the proximity values vs. the distance (Fig. 21), we noticed that there is far more noise than expected. There were a lot of zero proximity values for lower distances, which is unexpected as closer obstacle should give higher reading. We determined that this was because the robots tilt towards the back while moving and therefore the proximity values for the back three sensors are almost always close to zero as the light emitted by the back emitter is reflected off the surface (Fig. 22). This is a good example of spurious errors that would have been hard to detect without the framework.

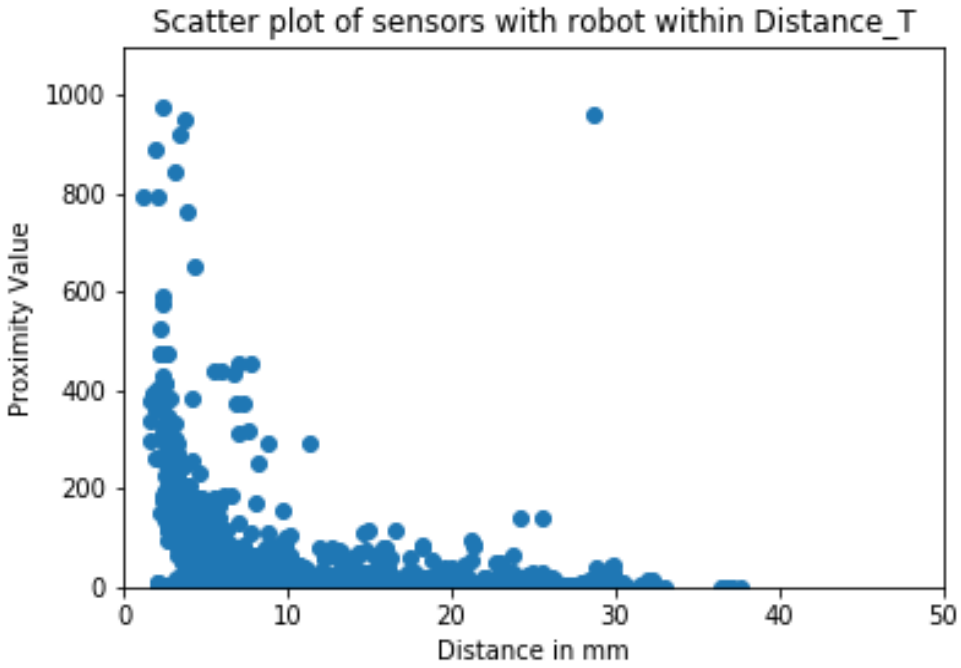

Figure 21. Plot of proximity values (belief states) and distances (actual states).

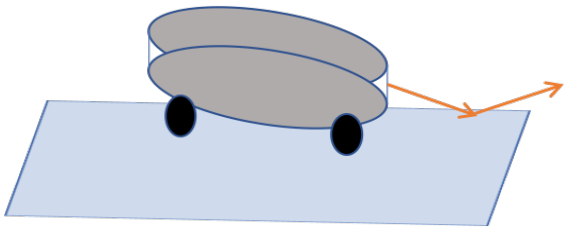

Figure 22. Back IR emitters emit light that is reflected away from the surface of the arena.

We ran the swarm to test the usability of the system. As shown in Fig. 10, the GUI highlights the robot with low battery level: less than 20%. The user just needs to select the battery checkbox to get notified about any robot that is running critically low on battery. User can then decide to remove that robot from the swarm.

To test hierarchical clustering, we used a swarm of 12 robots, out of which the value of the left-most sensors of three robots were turned off and another three robots had 10 added to the value of the right-most sensors, while the rest of them performed normally. As can be seen in Fig. 23, the GUI highlights the robots that perform differently from the rest of the collective.

Figure 23. Hierarchical clustering shown on GUI.

The flocking performance can be measured in terms of how the swarm organizes over time and how long it takes for it to achieve a low entropy. Here is a sample of a graph from the GUI (Fig. 24). It represents about 120 seconds of the flocking from the beginning. As can be seen from the graph, there is quite a lot of randomness associated with the emergent behavior. It's not a smooth plot, even if some noise has been filtered out by using past frames. The swarm goes through varied degrees of randomness before achieving movement as a flock. It is difficult for a user to observe the state of the collective in such a condition. OpenMind framework assists the user to determine if the swarm is moving towards the goal of flocking by providing an intuitive graph, which gets updated as the swarm moves.

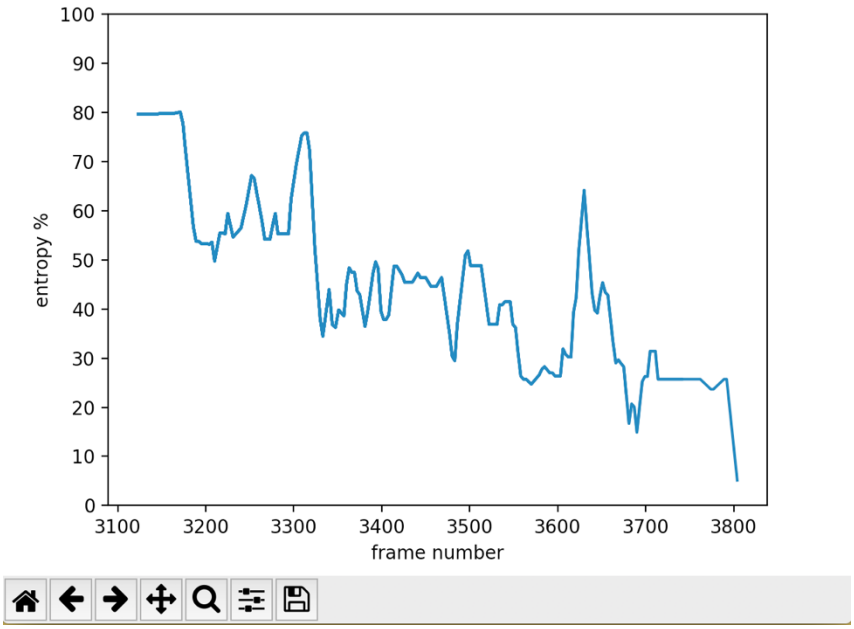

Figure 24. Graph of entropy of the robot collective.

Since OpenMind is designed for a swarm of robots, it's important to look at scalability of the memory requirement, bandwidth, computation time and analysis. We use a swarm of 12 robots to discuss these parameters.

- Memory: This is the MongoDB data that is accumulated by the robots. 12 robots accumulate around 0.3 GB of data in 1 hour. Frames from the overhead

camera are not stored, but only used to get the position and orientation of the robots and to display the live feed for the user. The history of the frames from the camera is therefore not needed. Old data points could also be discarded to improve the memory requirement.

- Computation:
    - Communication: This is dependent on the capability of the base-station. In current system, 20 updates per second are received from all the 12 robots. Without the need for inter-robot communication, the number of transmitted messages decrease linearly with the number of robots, as in the case of flocking. With a larger number of robots, noise and interference may become a problem, but can be handled by adding more receivers.
    - Image Processing: The image processing time taken by AprilTag detection is dependent on the size of the swarm. As the number of robots increase, the frame rate will decrease as the processing of images for getting robot position and orientation will become slower, as more tags have to be detected. In the current system with 12 robots, we can process 30 frames per second on average, which is 3 times faster than what we need to present the live video. A single processor can handle between 18-36 tags to be able to process fast enough for live feed. But the detection can be parallelized to make the framework compatible with larger swarms.
    - Analysis:
        - Accuracy: The accuracy of the analysis depends on the amount of available data. It improves with the number of robots and length of the execution. This is because the state verification and behavioral modeling analysis are based on statistical inference. Hence, with a very small collective or a short run, the model might not be useful.

- Time complexity: The time complexity for both behavioral clustering and entropy calculation is $O(n^2)$

Scalability improvement would require more work in this area. For OpenMind, we don't require a full view of the arena. Moreover, it can handle some missing frames. Scalability issues can thus be combated by only listening to and processing data for a part of the swarm. It may also be possible to apply OpenMind in scenarios where swarm operates in a large geographical area, where getting a view of the entire collective at once is impractical. Or it might also be applied to larger swarms, using multiple overlooking cameras.

5. DISCUSSION

This work provides the building blocks for a non-expert user to support the robot swarm. Autonomous swarms are increasingly being studied for real world applications, but their operation and monitoring remains difficult. Human support will prove highly beneficial in aiding the swarm to goal completion. Due to the high cognitive load, it is difficult however for a user to monitor the swarm. OpenMind is a decision support system that helps the non-expert user in supporting the swarm by overlaying relevant information onto a live video of the swarm, while also automatically analyzing the data. The user can see the internal state of the robots, any discrepancies in the sensors and actuators, state level variances of the robots within the collective, and the performance of the collective over time.

We leave user studies to determine the effectiveness of the system for future work. User studies would improve generalization and practicality of the framework. We aim for further improving the framework to help move robot swarms from lab to real world scenarios. Things we target for improvements are:

i) automatic model generation for robots, sensors and actuators.
ii) classification of a wider set of malfunctions and discrepancies.
iii) analysis and prediction of more complex swarm behaviors on more complicated platforms.
iv) prioritizing information shown to the user, further reducing the cognitive load.
v) implementing the framework on a hand-held device so that the user can move amongst the collective while assessing its state.

In addition, we could also focus on making the GUI more user-friendly by conducting user studies on the choice of colors, fonts and layouts. For usability by a general non-expert user, we could focus on picking colors that are friendly for color blind individuals. The choice of display components can be improved so that it's easy to distinguish the different categories of information even when they are all selected together. More intuitive controls, instead of the checkboxes, can be used to show the information on the GUI. Enabling human support of robot swarms by overcoming cognitive limitations is a significant step towards aiding real world application of autonomous swarms.

BIBLIOGRAPHY

[1] Brambilla, Manuele, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. "Swarm robotics: a review from the swarm engineering perspective." *Swarm Intelligence*7, no. 1 (2013): 1-41. doi:10.1007/s11721-012-0075-2.

[2] Rubenstein, M., A. Cornejo, and R. Nagpal. "Programmable self-assembly in a thousand-robot swarm." *Science*345, no. 6198 (2014): 795-99. doi:10.1126/science.1254295.

[3] Penders, Jacques, Lyuba Alboul, Ulf Witkowski, Amir Naghsh, Joan Saez-Pons, Stefan Herbrechtsmeier, and Mohamed El-Habbal. "A Robot Swarm Assisting a Human Fire-Fighter." *Advanced Robotics*25, no. 1-2 (2011): 93-117. doi:10.1163/016918610x538507.

[4] Fields, Maryanne, Ellen Haas, Susan Hill, Christopher Stachowiak, and Laura Barnes. "Effective robot team control methodologies for battlefield applications." *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009. doi:10.1109/iros.2009.5354188.

[5] Miller, George A. "The magical number seven, plus or minus two: Some limits on our capacity for processing information." *Psychological Review*101, no. 2 (1994): 343-52. doi:10.1037//0033-295x.101.2.343.

[6] Kolling, Andreas, Katia Sycara, Steve Nunnally, and Michael Lewis. "Human Swarm Interaction: An Experimental Study of Two Types of Interaction with Foraging Swarms." *Journal of Human-Robot Interaction*2, no. 2 (2013). doi:10.5898/jhri.2.2.kolling.

[7] Dudek, G., M. Jenkin, E. Milios, and D. Wilkes. "A taxonomy for swarm robots." *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 93)*. doi:10.1109/iros.1993.583135.

[8] Knepper, Ross A., Todd Layton, John Romanishin, and Daniela Rus. "IkeaBot: An autonomous multi-robot coordinated furniture assembly system." *2013 IEEE International Conference on Robotics and Automation*, 2013. doi:10.1109/icra.2013.6630673.

[9] Petersen, Kirstin, Radhika Nagpal, and Justin Werfel. "TERMES: An Autonomous Robotic System for Three-Dimensional Collective Construction." *Robotics: Science and Systems VII*, 2011. doi:10.15607/rss.2011.vii.035.

[10] Mondada, F., A. Guignard, M. Bonani, D. Bar, M. Lauria, and D. Floreano. "SWARM-BOT: from concept to implementation." *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. doi:10.1109/iros.2003.1248877.

[11] Xiang, Li, M. Fikret Ercan, Zhou Yi, and Yu Fai Fung. "Algorithm for swarm robot flocking behavior." *2009 4th International Conference on Autonomous Robots and Agents*, 2000. doi:10.1109/icara.2000.4803943.

[12] Oikawa, Ryotaro, Munehiro Takimoto, and Yasushi Kambayashi. "Distributed formation control for swarm robots using mobile agents." *2015 IEEE 10th*

Jubilee International Symposium on Applied Computational Intelligence and Informatics, 2015. doi:10.1109/saci.2015.7208181.

[13] Sand, S., S. Zhang, M. Muhlegg, G. Falconi, C. Zhu, T. Kruger, and S. Nowak. "Swarm exploration and navigation on mars." *2013 International Conference on Localization and GNSS (ICL-GNSS)*, 2013. doi:10.1109/icl-gnss.2013.6577272.

[14] Kwon, Ji-Wook, Jin Hyo Kim, and Jiwon Seo. "Consensus-based obstacle avoidance for robotic swarm system with behavior-based control scheme." *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, 2014. doi:10.1109/iccas.2014.6987879.

[15] Mandal, Paramita, Ranjit Kumar Barai, and Madhubanti Maitra. "Collaborative Coverage Using Swarm Networked Robot."

[16] Sugawara, Ken, Nikolaus Correll, and Dustin Reishus. "Object Transportation by Granular Convection Using Swarm Robots." *Springer Tracts in Advanced Robotics Distributed Autonomous Robotic Systems*, 2014, 135-47. doi:10.1007/978-3-642-55146-8_10.

[17] Nunnally, S., P. Walker, A. Kolling, N. Chakraborty, M. Lewis, K. Sycara, and M. Goodrich. "Human influence of robotic swarms with bandwidth and localization issues." *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2012. doi:10.1109/icsmc.2012.6377723.

[18] Kolling, Andreas, Phillip Walker, Nilanjan Chakraborty, Katia Sycara, and Michael Lewis. "Human Interaction With Robot Swarms: A Survey." *IEEE Transactions on Human-Machine Systems*46, no. 1 (2016): 9-26. doi:10.1109/thms.2015.2480801.

[19] Brown, Daniel S., Sean C. Kerman, and Michael A. Goodrich. "Human-swarm interactions based on managing attractors." *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction - HRI 14*, 2014. doi:10.1145/2559636.2559661.

[20] Bashyal, Shishir, and Ganesh Kumar Venayagamoorthy. "Human swarm interaction for radiation source search and localization." *2008 IEEE Swarm Intelligence Symposium*, 2008. doi:10.1109/sis.2008.4668287.

[21] Kolling, Andreas, Steven Nunnally, and Michael Lewis. "Towards human control of robot swarms." *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction - HRI 12*, 2012. doi:10.1145/2157689.2157704.

[22] Goodrich, Michael A., Sean Kerman, and Shin-Young Jung. "On Leadership and Influence in Human-Swarm Interaction."

[23] Vasile, Cristian, Ana Pavel, and Catalin Buiu. "Integrating human swarm interaction in a distributed robotic control system." *2011 IEEE International Conference on Automation Science and Engineering*, 2011. doi:10.1109/case.2011.6042493.

[24] Schoof, Eric, Airlie Chapman, and Mehran Mesbahi. "Bearing-compass formation control: A human-swarm interaction perspective." *2014 American Control Conference*, 2014. doi:10.1109/acc.2014.6859380.

[25] Hayes, Sean T., and Julie A. Adams. "Human-swarm interaction: Sources of uncertainty." *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction - HRI 14*, 2014. doi:10.1145/2559636.2559827.

[26] Walker, Phillip, Steven Nunnally, Mike Lewis, Andreas Kolling, Nilanjan Chakraborty, and Katia Sycara. "Neglect benevolence in human control of swarms in the presence of latency." *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2012. doi:10.1109/icsmc.2012.6378253.

[27] Alonso-Mora, J., S. Haegeli Lohaus, P. Leemann, R. Siegwart, and P. Beardsley. "Gesture based human - Multi-robot swarm interaction and its application to an interactive display." *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015. doi:10.1109/icra.2015.7140033.

[28] Nagi, Jawad, Alessandro Giusti, Luca M. Gambardella, and Gianni A. Di Caro. "Human-swarm interaction using spatial gestures." *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014. doi:10.1109/iros.2014.6943101.

[29] Giusti, Alessandro, Jawad Nagi, Luca M. Gambardella, Stéphane Bonardi, and Gianni A. Di Caro. "Human-swarm interaction through distributed cooperative gesture recognition." *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction - HRI 12*, 2012. doi:10.1145/2157689.2157818.

[30] Daily, M., Youngkwan Cho, K. Martin, and D. Payton. "World embedded interfaces for human-robot interaction." *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, 2003. doi:10.1109/hicss.2003.1174285.

[31] Naghsh, Amir M., Jeremi Gancet, Andry Tanoto, and Chris Roast. "Analysis and design of human-robot swarm interaction in firefighting." *RO-MAN 2008 - The 17th IEEE International Symposium on Robot and Human Interactive Communication*, 2008. doi:10.1109/roman.2008.4600675.

[32] McLurkin, James, Jennifer Smith, James Frankel, David Sotkowitz, David Blau, and Brian Schmidt. "Speaking Swarmish: Human-Robot Interface Design for Large Swarms of Autonomous Mobile Robots."

[33] Haas, Ellen, Maryanne Fields, Susan Hill, and Christopher Stachowiak. "Extreme Scalability: Designing Interfaces and Algorithms for Soldier-Robotic Swarm Interaction." 2009. doi:10.21236/ada498162.

[34] Gancet, Jeremi, Elvina Motard, Amir Naghsh, Chris Roast, Miguel Munoz Arancon, and Lino Marques. "User interfaces for human robot interactions with a swarm of robots in support to firefighters." *2010 IEEE International Conference on Robotics and Automation*, 2010. doi:10.1109/robot.2010.5509890.

[35] Reina, Andreagiovanni, Alex J. Cope, Eleftherios Nikolaidis, James A. R. Marshall, and Chelsea Sabo. "ARK: Augmented Reality for Kilobots." *IEEE Robotics and Automation Letters*2, no. 3 (2017): 1755-761. doi:10.1109/lra.2017.2700059.

[36] Olson, Edwin. "AprilTag: A robust and flexible visual fiducial system." *2011 IEEE International Conference on Robotics and Automation*, 2011. doi:10.1109/icra.2011.5979561.
[37] Folino, Gianluigi, and Agostino Forestiero. "Using Entropy for Evaluating Swarm Intelligence Algorithms."