Diagnosing Robotic Swarms 2 (Dr. Swarm2)

A Major Qualifying Project Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for
The Degree of Bachelor of Science

By
Kent Libby (RBE/CS)
Noah Van Stralen (RBE)
Lingrui Zhong (RBE/CS)

Submitted on May 14th, 2020
To Professor Carlo Pinciroli

Abstract

Robots are envisioned to work alongside humans. However, humans struggle to interpret the state and goals of a robot. The use of multiple robots further exacerbates this issue. To solve this problem, we propose Dr. Swarm 2, an augmented reality (AR) application built on the Magic Leap. The overlay provides concise information in a manner unachievable with existing methods.

Table of Contents

Table of Figures

I. Introduction

Robots are increasingly common in workplaces today. They mainly operate in manufacturing jobs to complete repetitive or boring tasks. However, advances in technology have brought robots into more fields. For example, Amazon uses over 200,000 robots in their warehouses to move shelves of products between different workers[1]. The increase in the number of robots means that human robot interactions will increase in necessity and frequency. One of the main challenges in human-robot interaction is effective collaboration between co-located humans and robots.

Robots currently struggle to effectively collaborate with humans. Effective collaboration requires efficient information sharing between the two parties, and robots struggle to communicate goals and intentions to humans[2]. Information transparency becomes increasingly difficult with large numbers of collaborative robots[3]. Ineffective information sharing prevents the human collaborator from understanding robot behavior. The resultant lack of understanding further complicates human-robot collaboration and robot development.

Consider the example of the shape formation swarm operation[4]. Even when the swarm functions correctly, it takes a significant amount of time for the swarm to complete the shape formation. In shape formation, the robots start from a given configuration and morph into the desired goal configuration. During the transition, external observers struggle to determine what the individual robots want to achieve.

When the robots do not behave as intended, the challenge of determining individual robot goals increases significantly. The failure cascades to functioning robots and complicates finding the root cause of the failure. In shape formation, faulty robots can interfere with operations of operational robots, resulting in an incorrect shape. As the number of robots involved increases, so does the challenge of identifying the faulty robot and troubleshooting its behavior.

In the shape formation operation, each robot relies on information from neighboring robots and sensors to guide its operation. The robot then processes the information and sends control signals to the motors to maneuver itself into position. To troubleshoot robot malfunctions, developers rely on tools to convey “robot state” in human understandable formats. For

---

[1] J. Rey, “How robots are transforming Amazon warehouse jobs — for better and worse.” Vox. https://www.vox.com/recode/2019/12/11/20982652/robots-amazon-warehouse-jobs-automation

[2] A. Bauer, D. Wollherr, and B. Martin, “Human–robot collaboration: a survey,” International Journal of Humanoid Robotics, vol. 5.01, pp. 47–66, 2008.

[3] Julie A. Adams, et al. “Swarm Transparency.” *HRI ’18: Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, Mar. 2018, pp. 45–46, https://dl.acm.org/doi/pdf/10.1145/3173386.3177008.

[4] Mario Coppola, et al. “Provable Self-Organizing Pattern Formation by a Swarm of Robots with Limited Knowledge.” *Swarm Intelligence*, vol. 13, 2019, pp. 59–94, https://link.springer.com/article/10.1007/s11721-019-00163-0.

simplicity, “robot state” refers to the information, its processing, and any control signals for a given robot.

As the information collected by robots becomes more complex, so do the robot states. As such, the tools for visualizations must adapt to display more information in a human-readable format. Robot data visualization tools like ARGoS[5] and Rviz[6] help developers understand the robot’s perception and state.

ARGoS simulates the operations and interactions of numerous robots simultaneously. The novel design of ARGoS enables developers to perform experiments on multi-robot systems in a fraction of the time required for real robots. Simulators like ARGoS create a model of the real world that enables faster prototype construction. This model and other features of simulators allow developers to design in simulation faster than in the real world.

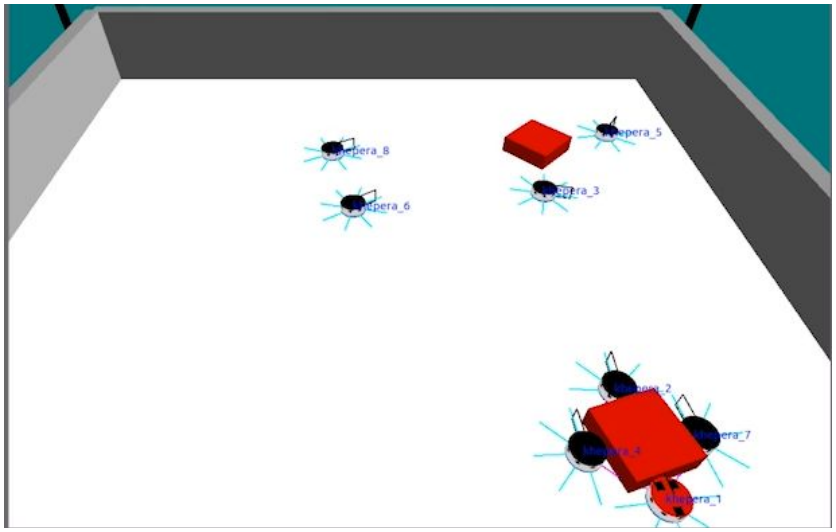

Figure 1.1: Screenshot from ARGoS

Unfortunately simulators cannot perfectly replicate the real world. The discrepancies lead to controls that can exploit the simulator’s limitations. Developers refer to this situation as “reality gap”[7]. Thus errors can manifest on the real robots that never arose in simulators such as ARGoS.

To troubleshoot these errors, H. R. Kam et al. created Rviz to visualize the data of real (and simulated) robots. Rviz graphically represents the sensor data of a robot on the user’s computer display. The displays are a significant improvement over decoding robot log

[5] C. Pinciroli, et al. “ARGoS: A Modular, Parallel, Multi-Engine simulator for Multi-Robot System.” *Springer Science+Business Media*, Nov. 2012, pp. 271–295, https://www.argos-sim.info/stuff/Pinciroli:SI2012.pdf.
[6] H. R. Kam, S. H. Lee, T. Park, T, and C. Kim, C, “Rviz: a toolkit for real domain data visualization,” vol. 60, no. 2, pp. 337–345, 2015.
[7] Sylvain Koos, et al. *Crossing the Reality Gap in Evolutionary Robotics by Promoting Transferable Controllers*. 2010, http://www.isir.upmc.fr/files/2010ACTI1527.pdf.

messages by hand. However, viewing the data may require the developer to divert their attention from the workspace. The rapid attention switching required for certain use cases may complicate robot development.

Due to the reality gap, developers encounter unique issues during real world tests. These unique issues necesitate real world testing, and the limited data gathering techniques available for these tests. Efficient and robust robot development hinges on improved data visualization techniques for real world testing.

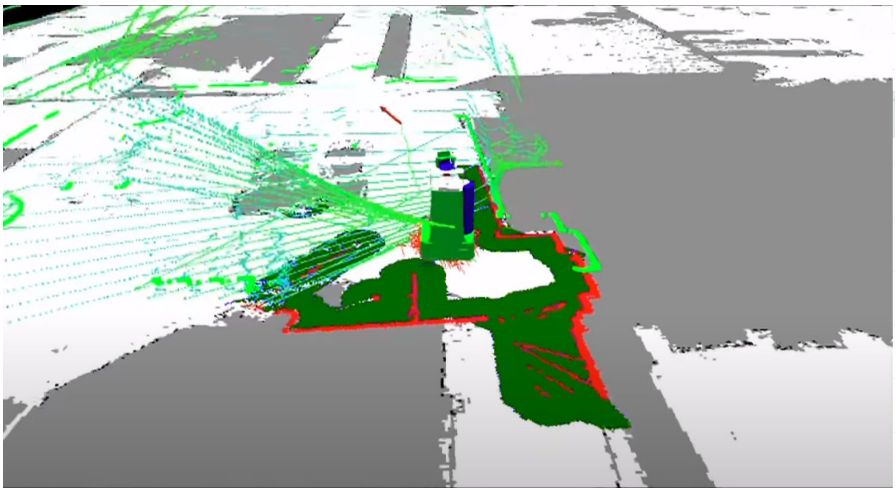

Figure 1.2: Screenshot from Rviz

The same communication shortcomings and performance concerns plague laborers working alongside robots. Collaborative robots, like the ones used in manufacturing, work separated from human counterparts[8]. They use LEDs and other visible markers on the robot to convey basic robot state. The separation between robots and humans minimizes injury risks, as humans and robots struggle to understand each others' intentions. The misunderstanding can lead to injuries and/or damage. The same improved visualizations that help developers can improve human-robot communication, and bolster human robot collaboration. Multi-robot systems compound the shortcomings of existing real-world data visualizations.

Multi-robot systems are large numbers of collaborative robots that accomplish a common task. Swarms are a subset of multi-robot systems where there is no centralized control. Swarm developers suffer the slings and arrows of robot development compounded by the number of robots and the lack of centralized control. Specialized tools like ARGoS enable swarm simulation to help swarm developers conceptualize their ideas. Although optimized for larger numbers of robots, it still cannot perfectly replicate the real world.

[8] "COLLABORATIVE ROBOTS: A COMPREHENSIVE GUIDE TO CAPABILITIES, SAFETY, AND APPLICATIONS." *Force Design Inc*, 2018, https://www.forcedesign.biz/collaborative-robots-a-comprehensive-guide.

Likewise, the real world tests suffer from software limitations. Developers rely on logs, and overt features for their tests. Unfortunately, simple logs struggle to provide context for environmentally situated data. Navigation data and sensor data in particular highlight the shortcomings of logs. Multiple robots on the field also complicate associating robots to sensor readings seen in a log display. Overt features such as LEDs let developers focus on the robot. Although LEDs fix the problem of linking messages to robots, they struggle to provide the breadth of data afforded by logs. Additional overt features may also interfere with the operation of other robots. For instance, the LEDs could be tracked by another robot and confuse it. Similarly, overt features such as LEDs can cause issues if other robots in the system use the same features for robot to robot communication.

Developers can use software like ARDebug[9] to view robots and data in context simultaneously. ARDebug is an augmented reality (AR) debugging tool designed for robotic systems. AR is a technique that overlays graphical artifacts on and around real world objects. ARDebug uses AR to overlay data with the robot workspace to provide environmental context. While using an AR system to provide environmental context overcomes the limitations of logs, systems like ARDebug have a couple drawbacks. The field of view (FOV), viewable area of stationary cameras, bounds the location of overlays. As such, FOV limited tools like ARDebug are ill-suited for operations where robots occupy a large area. The context and message limitations of these softwares further complicate swarms and their development.

The limitations affect the end users as well. The laborers who work alongside robots rely on lights and sounds to convey intention. Laborers move around their workspace, and cannot rely on a stationary display to convey information. The simple communication struggles to convey complex intent intuitively. Thus, laborers must understand the light and sound patterns to communicate with their robot counterparts.

Recent developments in AR have allowed the creation of mobile AR applications and devices. Vuforia, an AR development tool, allows for the creation of AR applications for tablets and other mobile devices[10]. Vuforia receives environmental data through the camera on the device. This solves the FOV issue present in ARDebugs by allowing the FOV to move with the user.

Companies like Magic Leap took AR to a dedicated mobile headset, making AR displays more immersive. The Magic Leap One[11] (abbreviated as Magic Leap) headset uses a stereoscopic display to project images onto the environment[12]. A stereoscopic display is a display that uses two images, one for each eye, to allow users to perceive depth in the projected images. Additionally, it has its own dedicated processor. The processor allows the Magic Leap

---

[9] A. G. Millard et al., “ARDebug: An Augmented Reality Tool for Analysing and Debugging Swarm Robotic Systems,” Frontiers in Robotics and AI, vol. 5, Jul. 2018.
[10] "Many Devices, Many Platforms", *Vuforia*, 2020, https://engine.vuforia.com/content/vuforia/en/devices.html
[11] Magic Leap. “Magic Leap 1.” *Magic Leap 1*, 2020, https://www.magicleap.com/en-us/magic-leap-1.
[12] "Magic Leap: Spatial Computing for Enterprise," *Magic Leap,* 2018, https://www.magicleap.com/en-us

to perform spatial computations, which includes recognizing gestures in 3D space. Lastly, it comes with its own controller for interaction and pointing.

These new advances in AR allow developers to leverage the spatial intelligence of their users. Spatial Intelligence is the human ability to comprehend three dimensional shapes and objects and link them to other parts of their surroundings[13]. Utilizing spatial intelligence will allow users to understand displays and their relationships to the environment in a faster and easier manner.

For our research on the human-robot communication problem The NEST Lab provided us access to Khepera IV robots[14]. The NEST Lab is a swarm robotics development lab at Worcester Polytechnic Institute. Khepera IV robots are ground based autonomous robots. The information we were able to receive from these robots included wheel actuation value, position and orientation data, sonar values, and log messages.

Problem Statement

Existing methods for human-robot communication are insufficient when dealing with robot swarms. Existing methods either to relay only basic state information, or have a restricted field of view. The limitations force developers to dig through log information to find issues. The safety concerns created by the limitations require workplaces to separate robots from human workers. The goal of Dr. Swarm 2 is to enable efficient communication between humans and robots. Dr. Swarm2 provides augmented reality visualizations that display information received from a robotic swarm. The augmented reality visualizations allow developers and laborers to see the swarm information in real time and quickly identify issues and behaviors.

Contribution

To improve the efficiency and clarity of robot-human communication, we propose and implement a method to display robot data in a spatially situated manner updated in real time. The graphical nature and spatial context will enable users to quickly recognize and diagnose robot actions, intents and issues.

Thus, we created an AR application. Our application leverages the Magic Leap mobile AR headset. The Magic Leap provides a stereoscopic display which allows information to be contextualized spatially. For example, the headset can label robots on the field and display log messages above the robot. Overlaying contextualized information onto the real world removes

[13] Melissa Kelly, "Spatial Intelligence", *thoughtco,* 2019, https://www.thoughtco.com/spatial-intelligence-profile-8096

[14] K team. “KHEPERA IV.” *KHEPERA IV*, 2020, https://www.k-team.com/khepera-iv.

the need for separate displays of robot perception. Additionally, the mobility of the Magic Leap allows users to navigate the workspace while conscious of their robots' intentions and fixes the FOV problem present in systems like ARDebug. The simple graphic representations preserve the users' field of view and allow them to focus on critical tasks.

We developed an extendable application architecture to serve as the base of our application. The architecture consists of three managers that split the functionality of the application between them. These managers are the Focus Manager, the Display Manager, and the Data Manager. The services control, input, display state and information retrieval respectively. The microservices allow the functionality of one manager to be expanded or changed without affecting the other managers.

The application enables a flexible backend setup. By receiving data through UDP packets, the application is independent of the source of information as long as the packets comply with the UDP protocol. To demonstrate the capability of our application, we set up an experiment with Khepera IV robots using ARGoS to supply data for the application.

Based on the data sent and received by Khepera IV, we proposed a taxonomy of individual robot information for ground based robots and recommendations on how to visualize each type for future expansions. The categories are current state information, intended action, sensor reading, actuation inputs, task specific information, and other information. Using this taxonomy, future developers of Dr. Swarm2 can quickly design a robot level information display by classifying the data to be visualized.

Using the taxonomy, we created 5 displays targeted for understanding robot states in a collective object transportation task. These displays are the entity highlight display, the path display, the sonar display, the motor display, and the log display. The entity highlight display overlays the robot or goal object with ID label and orientation indication. The path display visualizes the robot's intended travel path. The sonar display shows the sensor readouts from directional sonar sensors. The motor display visualizes the robots' intended wheel speeds. Finally, the log display presents non-spatial information such as log and debug messages. We then analyzed the displays based on information clarity and extensibility for future applications.

To fit the application onto the Magic Leap and make the application expandable, we redesigned the architecture of Dr. Swarm. We removed the reliance on image tracking capability, opening the possibility of robot tracking to other methods that better fit the experiment setup. Moving the display from a tablet to a dedicated headset makes the displays more immersive, allowing the user to focus on monitoring the experiment instead of positioning the device. Leveraging spatial intelligence provided by the Magic Leap improves the clarity of displayed information, setting Dr. Swarm2 apart from other visualizers.

The remainder of the paper is organized as follows. In Chapter II, we discuss existing methods for robot data visualization, their accomplishments, drawbacks, and areas for expansion. In Chapter III, we formalize the problem and discuss the approach used to develop the application. Chapter IV details the architecture of the final application, the completed

displays, and the taxonomy developed. In Chapter V, we summarize our work, detail lessons learned, and indicate directions for future research.

II. Related Works

Human-robot interactions thrive on communication. Human collaborators must understand the robots' states and goals to work with them effectively. Additionally, communication builds trust between the human and the robot. Humans can use data displayed to them and ensure the robots are operating as expected. As such, researchers have pushed the capabilities of human-robot communication to facilitate collaboration and debugging.

Robot-mounted indicators provide simple information to people in its vicinity. However, these displays require additional robot hardware and real-estate. Thus, these displays scale poorly for small robots and robot swarms. The mounted displays also struggle with information variety given their space constraints. Dashboards can provide additional information but lack portability required by people operating alongside the robots. Dashboards also lose some context due to their placement. Additionally, dashboards can only display data in 2D such as graphs and logs. Dashboards cannot effectively display 3D data, such as sensor clouds. The 2D displays used by dashboards obfuscate some of the depth data conveyed in point clouds. Certain displays circumvent this by showing depth with color or shading, but it is not a perfect representation of the data. As such, they are not sufficient for many operations in 3D space.

As such developers acknowledge the challenge of troubleshooting and visualizing multi-robot systems in the real world. McLurkin et al.[15] used LEDs and sound to communicate with the user. The system relies on complex information encoded into LED blink patterns. The authors note that changes or additions to encoded data would require redeployment of code to all swarm robots. This debugging system allows the user to compare the state with real world context. However, components used only for debug purposes are not ideal. The components could serve different purposes, or interfere with other functionalities, such as using sound as a means to understand the environment[16]. Furthermore, decoding the LEDs and sounds leads to excessive cognitive load, which makes it hard to scale with the number of robots and states. McLurkin et al. proposed that their system could be expanded by incorporating AR. They note AR interfaces would reduce cognitive load and better correlate input and the swarm's response. Additionally, AR would reduce the UI development overhead as the user could simultaneously view the live swarm state and intentions. The improved communication medium would facilitate more complex interactions with robot swarms.

To reduce the effort required by the user to understand the robot state, Daily et al.[17] created a system to display swarm information on an AR headset. The headset could interpret

---

[15] J. McLurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau, and B. Schmidt, "Speaking Swarmish: Human-Robot Interface Design for Large Swarms of Autonomous Mobile Robots," AAAI spring symposium: to boldly go where no human-robot team has gone before, vol. 72, 2006.
[16] F. Arvin, A. E. Turgut, N. Bellotto& S. Yue, "Comparison of Different Cue-Based Swarm Aggregation Strategies", ICSI, Lecture Notes in Computer Science, vol. 8794
[17] M. Daily, Y. Cho, K. Martin, and D. W. Payton, "World Embedded Interfaces for Human-Robot Interaction.," Proceedings of the 36th Hawaii International Conference on System Sciences - 2003, pp. 1–6, 2003.

the blink patterns of IR beacons on the robots and display an arrow guiding the user to a specific location. Users could then follow the path displayed by the robots to arrive at the location. Unfortunately, the AR beacons and the parsing software limited the amount of data the robots could convey. Thus, users could only view directional information from the robots in line of sight of the headset. Daily et al. also outlined plans to expand the work in the future. For example, they wanted to convey more sensor data from the swarm robots. They also wanted to receive data from robots occluded from the camera view. However, while their current system excelled at guiding users to targets, it lacked fine-grained data representation needed for more complex operations. It also required additional robot hardware to convey information to the user.

Collet et al.[18] reduced the robot hardware required for visualizations, and greatly increased the range of data displayed. They integrated with the robot control software to render the information in the robot workspace with a webcam and monitor. The integration brough the information transparency of robot simulators to the real robots' operations. Collet et al. revealed the advantage visual representations of robot data have over text-based logs. However, their system had no 3D capability and obfuscated certain position data as a result. Additionally, the workspace view was limited to the position of the webcam. This could cause the workspace and camera layout to occlude critical robot debug data. Collet et al. also explained future plans for their work such as increasing the camera's view of the workspace. They considered AR headsets as well as maneuverable 2D cameras. They also considered expanding the capabilities of their default visualizations. Additional default visualizations would eliminate the time needed to debug custom visualization as well as the target robot. Overall Collet et al. improved the capability of AR visualizations and demonstrated their superiority in conveying contextual data to users.

Millard et al.[19] circumvented the field of view issue with ARDebug. Their system used a camera placed above the workspace. The system then overlaid relevant data for robots of interest on the display. The visualization technique favors ground based robots and a compact workspace. Robots designed for exploration or foraging require large workspaces, and ARDebug requires camera view of the entire workspace. The need for numerous cameras in a large workspace makes it difficult to scale. The overhead view makes it difficult to observe a robot's perception of 3D spaces. Millard et al. proposed methods that could be used to expand the tracking capabilities to 3D. The system excels at tracking ground based robots with its fixed camera, but struggles with swarms with more than 50 members. The vision-based tracking system also requires a powerful computer to track and update data in real time. They look to expand the system by providing two-way communication between the user and swarm. This will allow ARDebug to perform more complex experiments.

---

[18] T. H. J. Collett and B. A. MacDonald, "An Augmented Reality Debugging System for Mobile Robot Software Engineers," JOURNAL OF SOFTWARE ENGINEERING IN ROBOTICS, vol. 1, no. 1, pp. 18–32, 2010.

[19] A. G. Millard et al., "ARDebug: An Augmented Reality Tool for Analysing and Debugging Swarm Robotic Systems," Frontiers in Robotics and AI, vol. 5, Jul. 2018.

Ghiringhelli et al.[20] created a system to view this spatially situated data in a 3D context. Their system uses blinking LEDs on the robots to identify and track them during the debug session. The spatially situated data displayed over its real world context reduces effort to associate data with robot behavior on the developer during debug sessions. It also determines robot orientation to ensure visualizations appear in the robot frame of reference. These features allow for complex visualizations for individual robots and swarms. Once again, the major drawbacks of Ghiringhelli et al.'s approach is the fixed viewing angle and 2D view generated. This again causes loss of context in the transformation from 3D information to a 2D viewport. Additionally, the displays lack expandability. Ghiringhelli et al. focuses on simple information retrieval from various sources, increasing the complexity for modifying display implementation. However, their design choice increases the availability to the average developer. The visualizer exemplifies AR's ability to replicate simulators' capabilities to display pertinent data in context. AR's ability to capture precise robot location allows for complex spatially situated visualizations.

Ghiringhelli et al. based a portion of his work from a system designed by Mark Fiala[21]. Fiala created a system to visualize and control robot swarms. His system relies on multiple cameras tracking fiducial markers on the robots. His design uses the Sony Vaio to present visualizations and controls to human collaborators. The portable tablet PCs provided an intuitive interface between robots and humans. However, the system relies on large fiducial markers, making implementation on smaller robots difficult. Additionally, the system requires the user to see the robot to view the data and provide control input. These limitations complicate implementation on large swarms especially with smaller robots. Fiala planned to expand the system by improving the tracking mechanism to provide visualizations for occluded robots. Moreover, a more portable AR system could facilitate interactions between the robots and human collaborators. Implementation of position tracking for the human collaborators would improve the control capabilities of the system as well.

Andersen et al.[22] created a collaborative robot state visualizer using projections. Their system used 3D object tracking to project robot intentions onto a collaborative workspace. Users reported they could understand robot intentions faster and work alongside them more confidently. Andersen et al. performed a successful user test, finding interaction quality improved with the projection-based approach. However, the system requires a fixed workspace to ensure the projector can relay intentions to the human collaborator. Moreover, the object tracking could fail, leading to mismatched projections and intentions. They plan to further their

[20] F. Ghiringhelli, J. Guzzi, G. A. D. Caro, V. Caglioti, L. M. Gambardella, and A. Giusti, "Interactive Augmented Reality for understanding and analyzing multi-robot systems," in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 1195–1201.
[21] Mark Fiala. "A Robot Control and Augmented Reality Interface for Multiple." *2009 Canadian Conference on Computer and Robot Vision*, 2009, pp. 31–36, doi:10.1109/CRV.2009.32.
[22] Rasmus S. Andersen, et al. "Projecting Robot Intentions into Human Environments." *25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2016, pp. 294–301, doi:10.1109/ROMAN.2016.7745145.

research with increased projection options and additional user tests. Overall, their work highlights the capability of contextual human-robot communications.

Recently, Dr. Swarm[23], an AR mobile application, enabled developers to visualize robot state in context with the workspace. The team used Vuforia and Unity3D to track robots in the workspace and display information adjacent to the associated robot. Their display centers around charts appearing next to the robots.The information passing protocol allows multiple users to run the application and view robot state simultaneously. Unfortunately, the nature of the graphs made displaying text data impossible. Additionally limitations on Vuforia meant visualizations for only five robots could appear simultaneously. Vuforia also imposed a maximum recognition distance for any robot in the workspace. In their future work, the team planned to expand their application to an AR headset to allow for additional tracked objects, and to expand the variety of visualized information. Dr. Swarm's preliminary testing highlighted use cases for portable AR swarm visualizations. As such, they would integrate with existing swarm and robot development software to reduce the overhead for the application's adoption.

Our project will expand the capabilities of Dr. Swarm to the Magic Leap[24]. The Magic Leap, an AR headset and development platform, allows the user to see data and graphics in 3D overlaid with the real world. The two major parts of the endeavor are the visualizations and the navigation between them. Our focus is on creating and displaying the swarm visualizations, but we will be collaborating with our sister team developing a gesture interface for the robot swarm. With the headset we can display meshes and other contextual data seen in Ghiringhelli et al. and Dr. Swarm's work. The user can move through the workspace and use gesture controls from our sister team to create and view 3D data visualizations.

[23] A. Wheeler, E. S. Snow, J. B. Brown, and J. D. Boucher, "Diagnosing Robotic Swarms (Dr. Swarm)," Major Qualifying Projects (All Years), Apr. 2019.
[24] G. R. Bradski, SA. Miller, and R. Abovitz, "Methods and systems for creating virtual and augmented reality," United States US20160026253A1, Feb. 12, 2019

III. Methodology

Problem Formalization

Ineffective communication plagues human-robot teamwork. The lights and sounds robots use to relay only basic states and are difficult for humans to understand. Teamwork hinges on effective communication, so robots and humans currently struggle with effective teamwork. Lacking communication exacerbates safety concerns, requiring robots to operate in separate areas. The separation further inhibits teamwork and effectiveness of human-robot teams.

To address the lack of information transparency, one must understand the types of information that a robot needs to communicate to the users. We classify the information types into six categories:

1. **Current state information:** this tells the user about the robot's current configuration in an experiment or a mission. Some of the examples are instantaneous information, such as the robot's position and orientation, and metadata such as the robot's ID.

2. **Intended action:** this tells the user about the collective result after the robot aggregates both sensor information and information from other robots. One of the examples is the planned path of a robot.

3. **Sensor readings:** this shows the user how the robot perceives the surrounding environment. One of the examples is sonar readings.

4. **Actuation inputs:** this low-level robot control information helps the user to separate hardware issues from software issues. One of the examples is the value given to the wheel motors.

5. **Task-specific information**: this shows the additional information defined by a task, such as foraging, shape formation, and collective object transportation. In the example of an object movement task, the additional information generated include team assignment, robot placement around the object, and the object's movement trajectory.

6. **Other information:** this information is best shown in strings, a human-readable format. The strings often need to be presented as a whole to preserve their significance and integrity. One of the examples is error messages.

In this project, we propose an AR application to improve human-robot communication. We developed displays to address each information category on a robot level tailored towards a Khepera IV[25] swarm. The display features are detailed in the Display Outline section.

[25] K team. "KHEPERA IV." *KHEPERA IV*, 2020, https://www.k-team.com/khepera-iv.

We want the application to accomplish the following goals. We gauge the performance and success of each component based on the following criteria in the Results section.

1. The information presented needs to be accurate, straightforward, and effective. Based on the visualization and the behavior of the robot, the user should be able to estimate the state of the robot--the user should be able to understand the intention of the robot, and if an issue occurs, the user should be able to pinpoint the cause. Beyond the visualizations, the application should also be straightforward and intuitive to use. A user study can measure the time saved using the application, show the effectiveness of the application, and gauge the success of accomplishing this goal.

2. The application must be easily extendable to other swarm experiments and systems. The displays should be customizable to other swarm robots by changing the UI element files and modifying configuration variables. On the architecture level, the user must be able to register and develop easily using features in the development tools. Lastly, the application should function independent of the implementation of the data source.

Display Outline

For each information category defined in the Problem Formalization section, we chose one or more pieces of information to visualize. The mockups show the displays in a hypothetical setup of the robot system.

The robot ID's, a piece of current state information, helps the user to identify a robot. An overlaid label located on top of the robot differentiates the robots on the field and allows the user to troubleshoot robot specific errors.

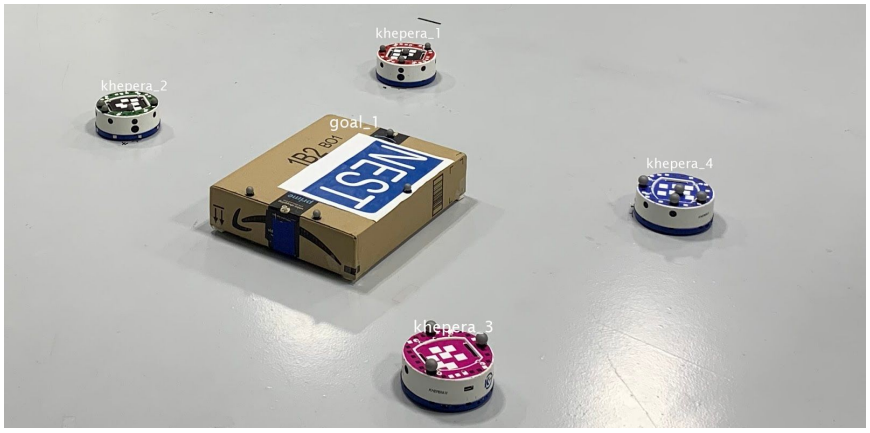

Figure 3.1: Mock up of ID labels on the field

Since the Khepera IV robot looks almost identical from all sides, an indication of the robot's orientation helps the user to identify the robot's travel direction. An arrow on top of the robot can show the robot's heading, a piece of robot current information.

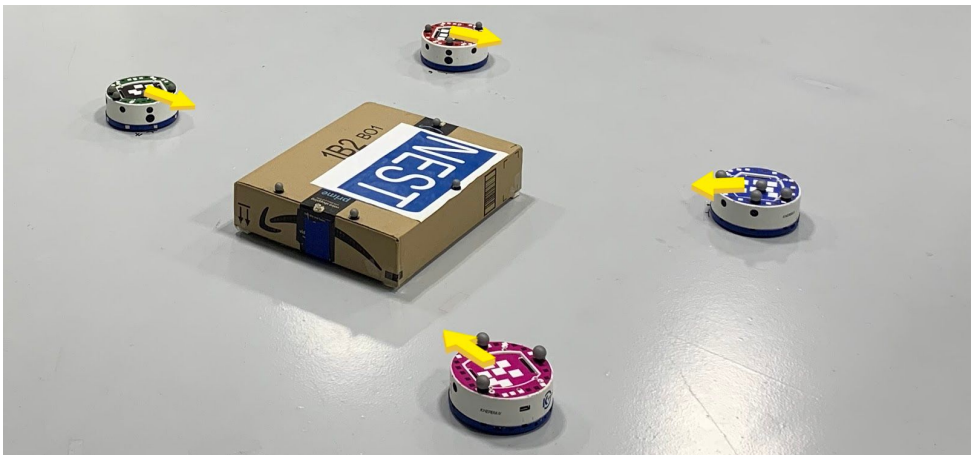

Figure 3.2: Mock up of direction vectors

Since Khepera IV is a ground robot, its planned path gives insight into its intended movement. A list of waypoints connected in sequential order, when displayed on the field, allow the user to contextualize navigation with the environment. If the user is collaborating with the swarm, the user can see where the robot is travelling and plan their own action. In addition to that, the path display can help the user to associate the robot's intention with its physical behavior. Combining sonar, wheel actuation, and path displays allows the user to determine if a faulty behavior is caused by actuation, sensing, or planning part of the controller.

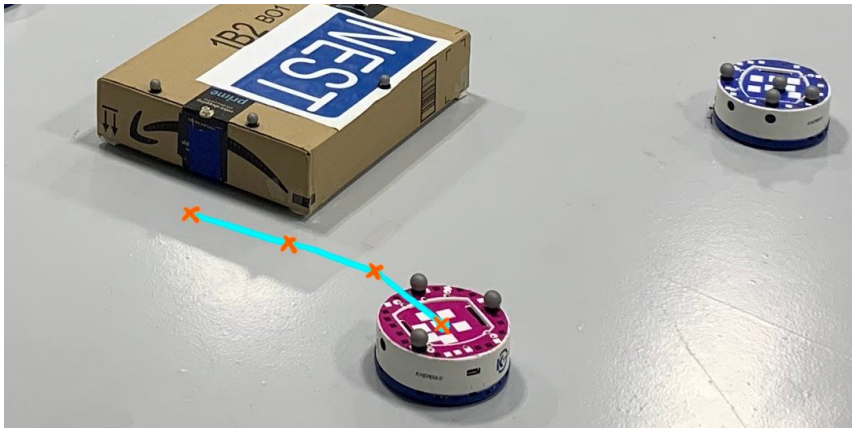

Figure 3.3: Mock up of path display

The only cause of motion on Khepera IV robots comes from the two wheel motors. Wheel actuations, an actuation inputs information, reflects the robot's intended behavior on the hardware control level. The user can compare the vector with the robot's behavior to determine

if an error is mechanical or software. An arrow attached to each of the wheels can indicate the magnitude and the direction of the motor force applied.

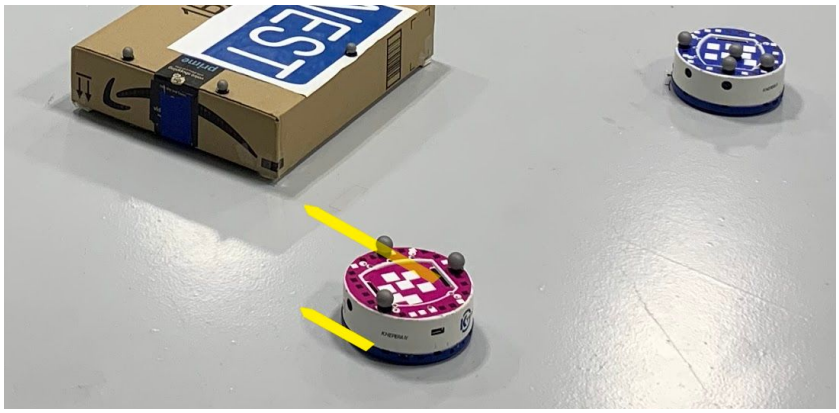

Figure 3.4: Mock up of path display

We chose to visualize sonar sensor readings because the readings directly impact the robot's perception of its immediate surroundings. A robot's incorrect perception of its surroundings leads to aberrant behavior. An overlaid mesh on the floor can show where the robot thinks it can go, providing a clear relation between the robot's perception and its environment. A user can quickly find errors and troubleshoot them with the overlay.

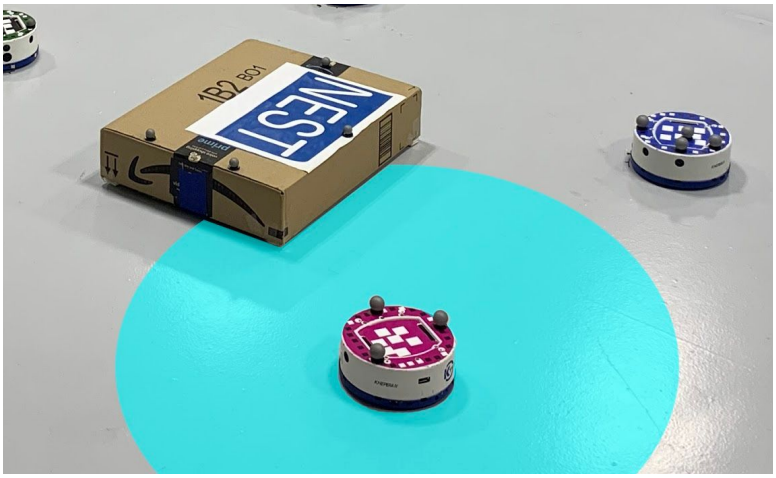

Figure 3.5: Mock up of proximity sensor display

In a collective object moving task, it is key to understand how the robots perceive the goal object. An overlaid mesh over the goal object can show the location of the goal detected by the swarm. If instead a robot struggled with finding the goal, this view could compare where the swarm thought the goal was with its actual location.

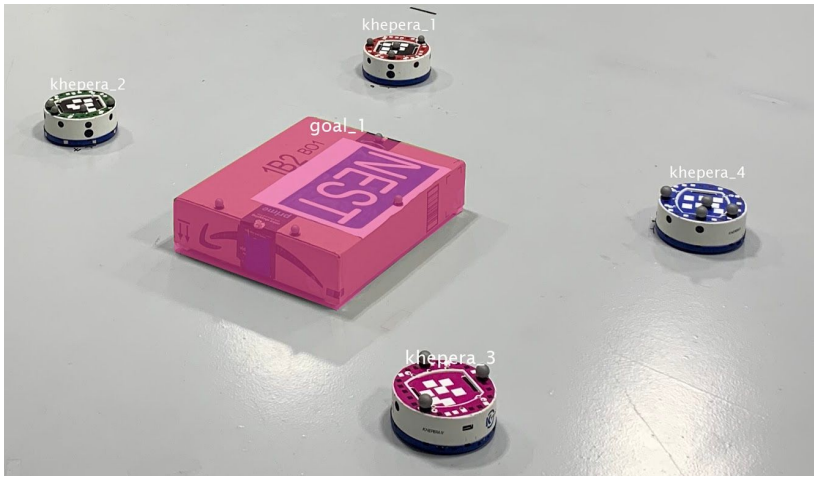

Figure 3.6: Mock up of a goal label on the field

Log information is critical for users to understand the robot's intention, and the information is especially useful when viewed in context of the robot’s behavior. Having the information right next to the robot lets the user focus on debugging its behavior within the AR environment. Since the information is best presented as a string, a text box that appears in the proximity of the robot in question can save the user a trip going back to the controller station.

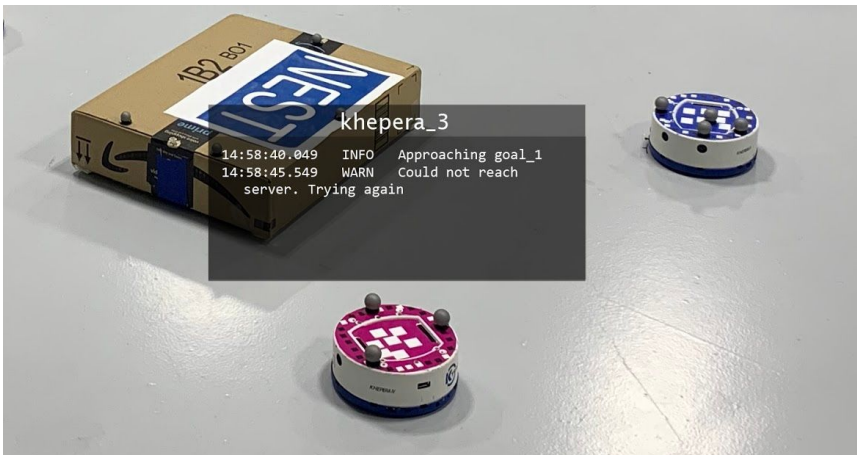

Figure 3.7: Mock up of log messages in proximity of the robot

System Setup

Our system is an AR application running on a Magic Leap. The application overlays the state, movements and intention of robots with the environment. The overlay provides transparency and context to robot actions. Dr. Swarm2 was developed using Magic Leap Unity API. It talks to the robot controller through UDP packets using LAN, receiving aggregate data from the swarm in the form of json packets. The robot controller then controls either the robot

swarm or the swarm simulator. We used ARGoS as our simulator throughout the development process and to capture the samples for this manuscript. Dr. Swarm2 can also send control commands back to the robot controller using features developed by our sister control team. The extensible design of Dr. Swarm2 means different, or even multiple UDP servers can send data to the app. The implementation even allows decentralized swarm robots capable of sending UDP packets to provide information to Dr. Swarm2. It can then render the data with its various displays.

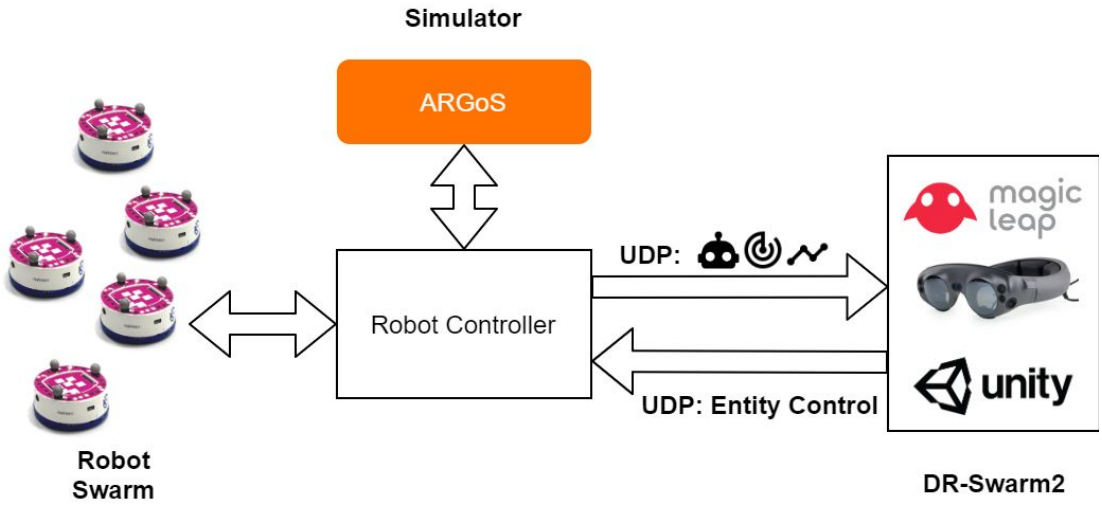

Figure 3.8: Communication architecture

Approach

Controlling and utilizing various displays requires a robust and easily expandable architecture for displays. We accomplish the expandability using microservices[26]. IBM defines microservices as an application-scoped architecture that enables internal components of the application to scale and change independent of each other. The loosely coupled services mean we can add functionality without breaking other services. Dr. Swarm 2 relies on three major service managers to run the displays and handle user input.

The first manager is the Focus Manager. As the name suggests, the Focus Manager focuses user input on the selected control. The Magic Leap provides 3D input, so the UI must exist in 3D as well. In a standard application, the UI is a 2D screen overlay or world object selected with a mouse click. In our application the user interacts with controls with laser pointer style selection from the controller. The user then uses the bumper instead of a mouse click to

[26] IBM. “SOA (Service-Oriented Architecture).” *SOA (Service-Oriented Architecture)*, 17 July 2019, https://www.ibm.com/cloud/learn/soa.

interact with the control. Unity does not natively support 3D input for canvas objects so we created the Focus Manager.

The manager uses a physics raycast from the controller to look for intersections with box colliders in the scene. The manager then sorts the hit objects by distance from the controller and determines if the first object hit has a Focusable component. The Focusable component has event handlers to start, continue and stop focus. The manager invokes the start event when the control first receives input, then invokes continue until the control loses focus. The manager then invokes the stop event. Additionally, scripts can attach handlers to global focus events that fire when no other GameObject has focus. The manager uses the following decision tree to invoke focus:

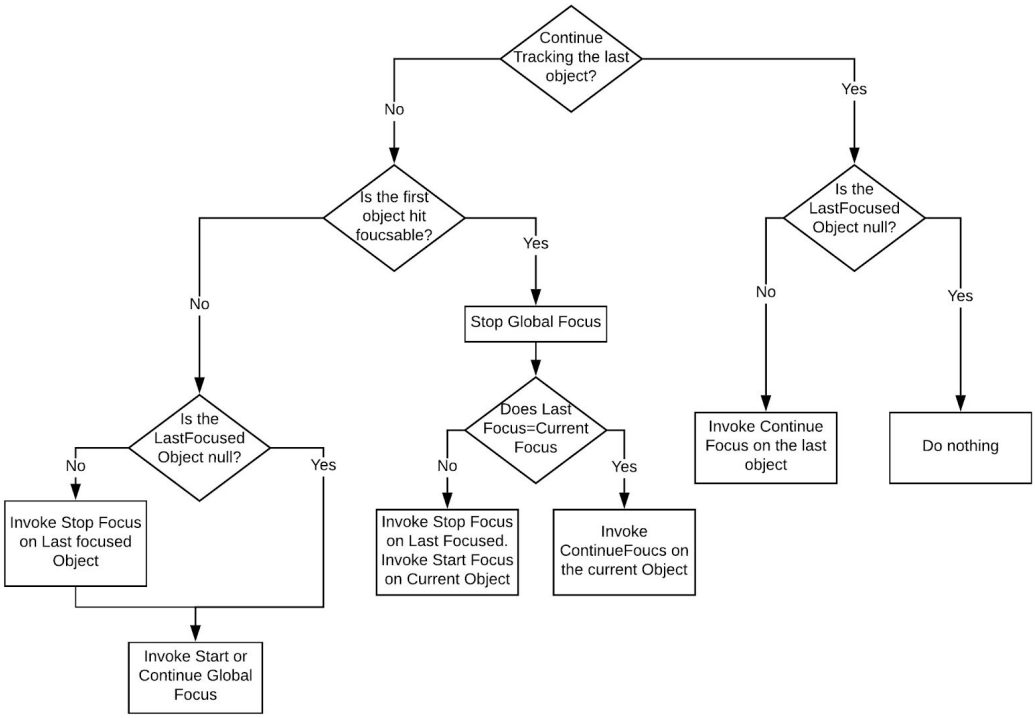

Figure 3.9: Focus Manager decision tree

To ensure the Canvas Objects could receive focus, we attached sphere colliders to all interactable objects. The colliders ensure objects can receive physics raycasts from oblique angles and act like physical objects. Through the 3D colliders and Focus Manager we created a system to manage and direct 3D input.

The second manager is the Data Manager. The purpose of the Data Manager is to collect data from all contextual entities (robots and goals). The Data Manager then stores the data in an easy access format. The manager collects the data through a UDP server. The UDP server is run through the UDP Packet handler class. The packets sent over the server are JSON packets containing the entityID and updated information. JSON stands for JavaScript Object

Notation and it provides a structure to store and send data in a human readable format[27]. The entity ID is used to access the Entity Data of the entity the message is associated with. Any displays that utilize the data also need the entityID. The information received from the UDP server is used to update the Entity Data of the identified entity.

The last of the three managers, the Display Manager, is designed to provide an intuitive interface for controlling each display element. It also keeps the interface extensible to support the variety of display implementations mentioned above. Each display element in the application is categorized as an entity display, a contextual display associated with an entity, or a global display, a display that is not associated with a set of Entity Data. To maintain the extensibility of the display interface, each display only exposes the OnDraw method. The method is used for updating the display each frame. All components are hidden within the implementation classes.

However, to design a smooth user experience, the display elements need to interact with each other. For example, we can present control buttons for turning a display on and off. This necessitates a Display State Manager. It stores the state information of all displays and provides a safe interface for inter-display communication. The structure of the Display State Manager mirrors the Display Manager. It separates the display states into robot display states and global display states. Each of the display state implementations expose control properties, such as the enabled status. In order to avoid race conditions when updating the display elements when the state is changed, each display class exposes an UpdateState method. The method updates the state of the display. The UpdateState method is called before the OnDraw method.

The final piece of the architecture is the Visualization Manager. The Visualization Manager owns the three service managers and serves as the main entry point into the application. This architecture provides a low friction method for creating new displays without disrupting existing displays. The architecture also allows for new types of data to be tracked. During the implementation of the architecture we utilized dependency injection to ease the creation of unit tests. Dependency injection allows classes to easily be mocked and increased coverage. These unit tests allow us to ensure the code is robust.

[27] "The JSON Data Interchange Syntax", *ECMA International,* 2017, http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

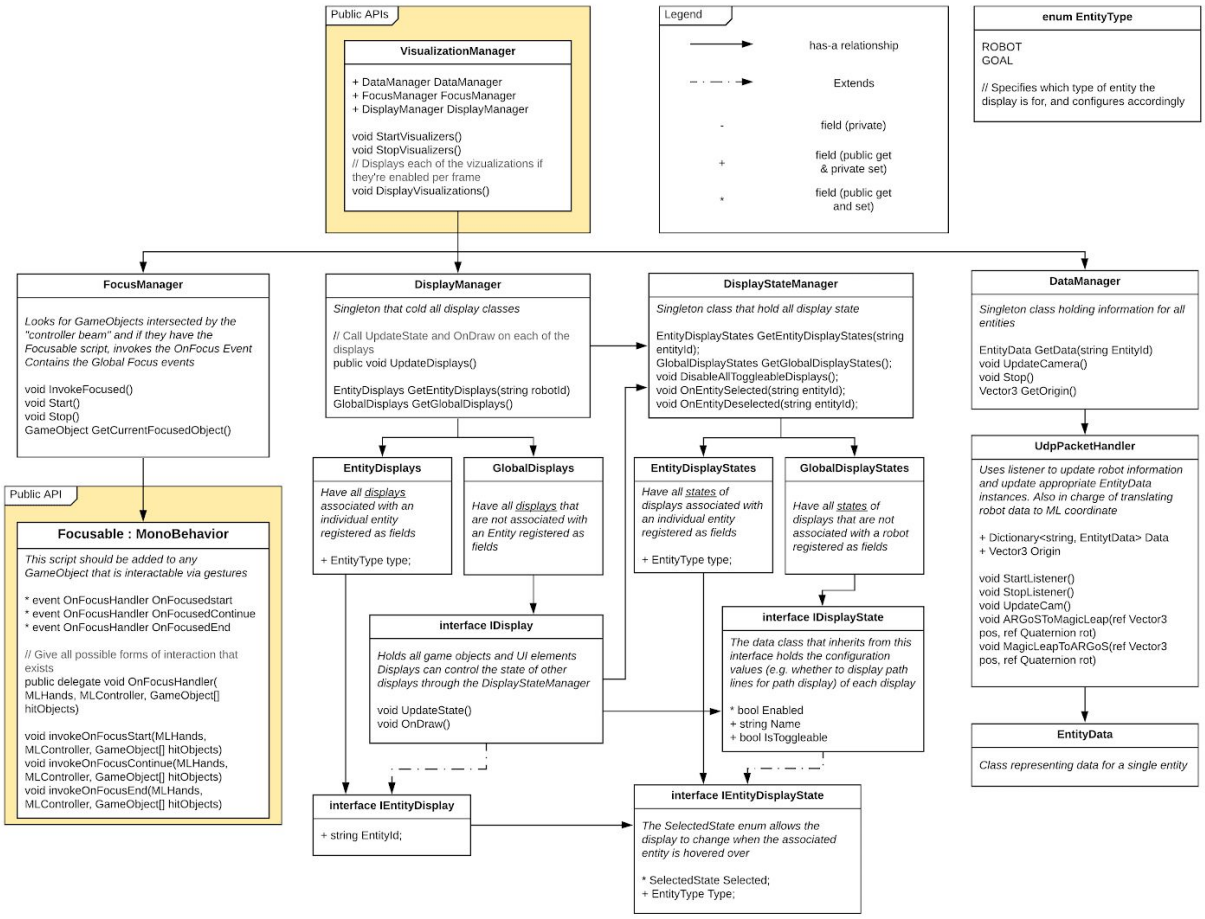

Figure 3.10: Detailed service diagram for the Visualization Manager

IV. Experimental Evaluation

Experimental Setup

We anticipated running a user study to gauge the effectiveness of our project. However, due to Covid-19 we were unable to conduct the study. Thus, we documented our intended experiment setup here. We hope future work on Dr. Swarm2 will utilize our planned experiment once circumstances allow for it.

We planned to gauge our project's success by the time users save in troubleshooting and collaborating with robots. We planned to run two tests, one for debugging and one for collaboration. The users will start with traditional methods and incorporate our technology piecemeal. We would then see how the process changes with the application of AR (see Appendix A for setup).

Results

Our design for Dr. Swarm2 has two criteria for success. We need to present the entity data in a straightforward manner. We also need an expandable and simple platform for display creation. We centered our implementation and features around the Khepera IV robot[28]. Our experiment would have evaluated the capability of our displays with the Khepera IV. Instead, we will evaluate the effectiveness of the managers based on their contributions to these criteria.

Application Architecture

Data Manager

The Data Manager stores information from UDP packets sent to the application. Over the course of implementation we needed to make changes to the Data Manager. In the original design the Data Manager would receive entity position information from the Magic Leap's Image Tracking API. However during implementation we discovered that the image tracking was too limiting for our purposes. In our experimentation with Magic Leap image tracking, we found it limits users to a maximum 2 meter range and 70 degree angle of incidence. In swarm robot use cases, robots can be tens of meters away, making image targets unusable. As such we rely on the server to relay the location of robots in the workspace. To improve tracking distance, we designed the Data Manager to receive robot locations through UDP messages. The Magic Leap receives positions as vectors relative to the position of the headset. Dr. Swarm2 then converts the vectors to the Magic Leap's frame of reference. ARGoS handles the larger axis conversions,

[28] K team. "KHEPERA IV." *KHEPERA IV*, 2020, https://www.k-team.com/khepera-iv.

and Dr. Swarm2 does the final rotations required for axis alignment. After the axes are aligned, the position vectors are converted to world coordinates.

All of the data received by the Data Manager is stored in an Entity Data instance. A single instance of the Entity Data structure exists for each entity, either a robot or a goal object. When Dr. Swarm2 receives a UDP packet, it first checks the entity that owns the information. The app then accesses the Entity Data instance for this entity and updates it with the new information. This structure provides an easy method to expand the data that Dr. Swarm2 can track. The only restriction is what information the ARGoS server can send about the objects on the field.

The Data Manager facilitates the range of entity data displayed by Dr. Swarm2. The information in the Entity Data class allows for concise access and update code. The simple string to data association allows for unique and verbose entity names. Additionally, it makes the entity names easy to update and amend. Thus, the Data Manager passes the first criterion for straightforward data presentation. Its server-agnostic design allows for various and even multiple simultaneous sources of information. Developers can easily modify the Entity Data class used to determine information gathered from robots. The modifications facilitate a broad range of displayed data as a result. Between the expandable Entity Data class and server-agnostic design, the Data Manager fulfills the expandable platform criterion.

Focus Manager

The observer pattern implemented by the Focus Manager provides fine-grained user control with low overhead. Developers can attach listeners to the Focus Manager that are agnostic of other listeners. The Focus Manager automatically calls the corresponding start, stop or continue function in the focus listeners when a GameObject receives controller focus. Thus, GameObjects can wait for invocation from the Focus Manager, and avoid the Unity update loop. We could reduce the size of buttons on screen to 5cm in Unity world space by using 3D physics raycasts for input. The raycasts interact with colliders defining the bounds of user-interactable components. The precise interactions reduce the required size of colliders and buttons. The extra screen space allows users see more relevant contextual information. The Focus Manager mitigates controller shakiness by requiring the user to move the controller a short distance before updating the beam position. The adjustment further facilitates use of smaller buttons.

The Focus Manager succeeds in the first criterion for straightforward entity data presentation. The small required size of buttons and display controls allows displays to consume a majority of the screen space. The extra space afforded to displays means entity data becomes more apparent to the user. Thus, the Focus Manager successfully facilitates the straightforward display of entity data.

The Focus Manager also allows for an expandable and simple platform for data display. Developers can attach controller-state agnostic reactive elements to displays. The Focus

Manager controls when elements receive inputs, and developers can focus on how to respond to input instead. The simple and expandable interactions provided by the Focus Manager fulfill the second criterion for success as well.

Display Manager

The Display Manager provides another wrapper around the Unity update loop. The state update followed by display updates makes sure displays update the same frame that changes occur. The displays show live data about an entity, so the looping update ensures the most recent data appears on screen. Custom state information means developers can create multiple states for their display via the Display State Manager.

The Display State Manager separates display state from display information. The separation helps developers access display states without accessing the actual display. Global displays like the HUD or entity menu display access the display state associated with a specific robot ID. The global display can then modify the state for a specific robot with one line of code. The fast access means the application can support complex state manipulation for various displays.

The Display Manager and Display State Manager synergize to meet both criteria for success. The Display Manager provides state update and draw functions to simplify the display render process. The streamlined display update functions mean the Display State Manager and Display Manager pass the first criterion for success. The Display State Manager allows displays to check the state of multiple displays and allows for complex state-display interactions. The diverse interactions provided by both managers also fulfill the expandable and simple criterion.

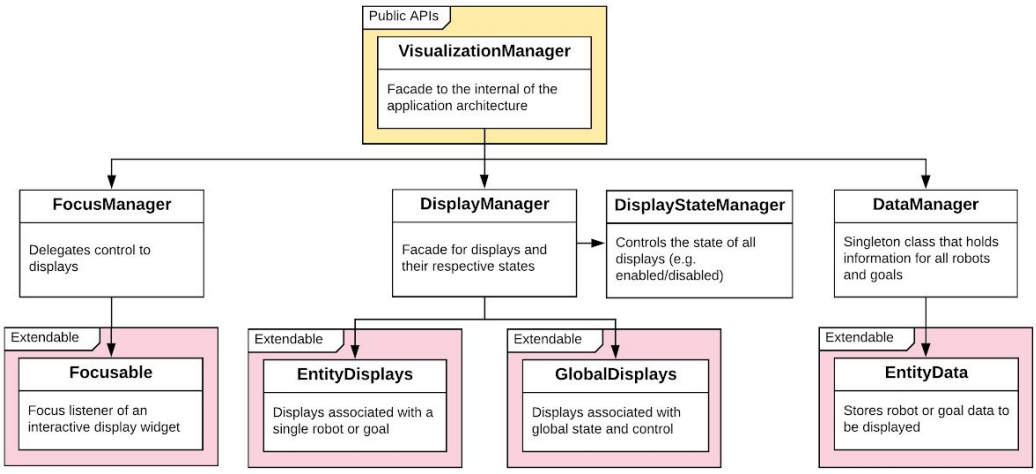

Figure 4.1: Abridged service diagram highlighting how the application can be extended

Overall, we implemented all managers successfully. Each component contributes to the straightforward data displays and the simple yet expandable nature of Dr. Swarm2. The loose coupling between managers allows developers to augment the architecture and capabilities to further suit their needs.

Display Taxonomy

Dr. Swarm2 innovates on contextual data display for robot swarms. We created five main displays: log display, path display, sonar display, motor display, and entity highlight display. The displays provide contextual information for robot operations in a variety of situations. Our implementations of the displays focused on dispersion and team movement experiments using ARGoS. We also created a heads-up display (HUD), an entity highlight display and an entity menu display to allow users to control the appearance of visualizations. We will evaluate the displays' success based on how well a user can understand the displayed data. The ease of understanding hinges on two criteria. The first being clarity of data presented. The second is the preservation of data accuracy when presented in the display.

Log Display

The log display provides non-spatial vector or non-scalar information. The location and size of the display compliment the nature of the information. The log overlay with the real world facilitates data association. Users can root cause and troubleshoot faster when the information and action appear in the same context.

The log display fulfills both criteria necessary for success. For the first criterion, the data presented are string log messages. The log display affords ample space for verbose messages. The blue line and title clearly associate the log messages to their owner. The buttons on the side of the display sacrifice some screen space for display appearance control. However, the effect on data clarity is minimal. Thus the log display passes the first criterion. The log display also preserves the accuracy of the messages perfectly. The strings appear in the display exactly as they appear in the controller. Thus, the log display fulfills both criteria and is a successful display.

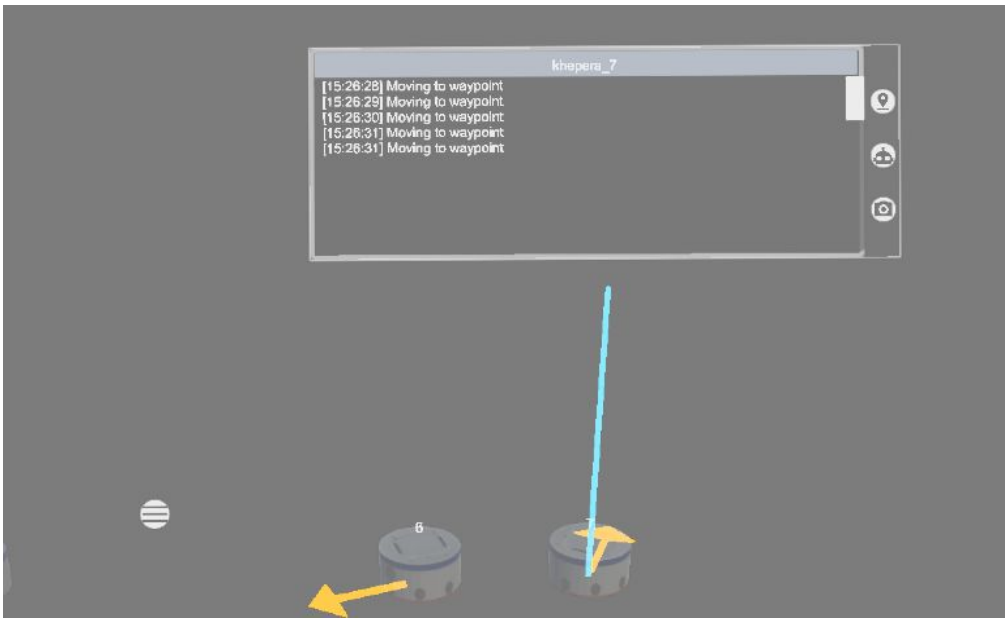

Figure 4.2: Log Display

Path Display

The path display provides a contextual display for entity movement. Users can see a robot's intended path and analyze entity movement. The color change on highlight helps users differentiate paths when multiple entities move into the same area. The information buffering means developers can send packets natively on waypoint generation. Thus, the path display simplifies rendering and parsing movement information.

The path display also fulfills the criteria to be a successful display. The data presented in the path display is the entity's anticipated position at a given time. The path display appears on top of the associated entity, clarifying its association. Additionally, it changes color when the user highlights the associated entity. The points on the path display represent waypoints in the entity's travel plan. The lines show the entity's route between waypoints. As such, the path display provides a clear picture of an entity's intended position over time. Thus, it satisfies the first criterion for clarity of presented data.

The path display also preserves the data accuracy needed to fulfill the second criteria. The robot controller passes the waypoints as numerical position data. The path display renders the waypoints at the position they appear in the real world. If the entity deviates from the path, the discrepancy becomes apparent. Thus, the path display preserves data accuracy, and fulfills both criteria to be an effective display.

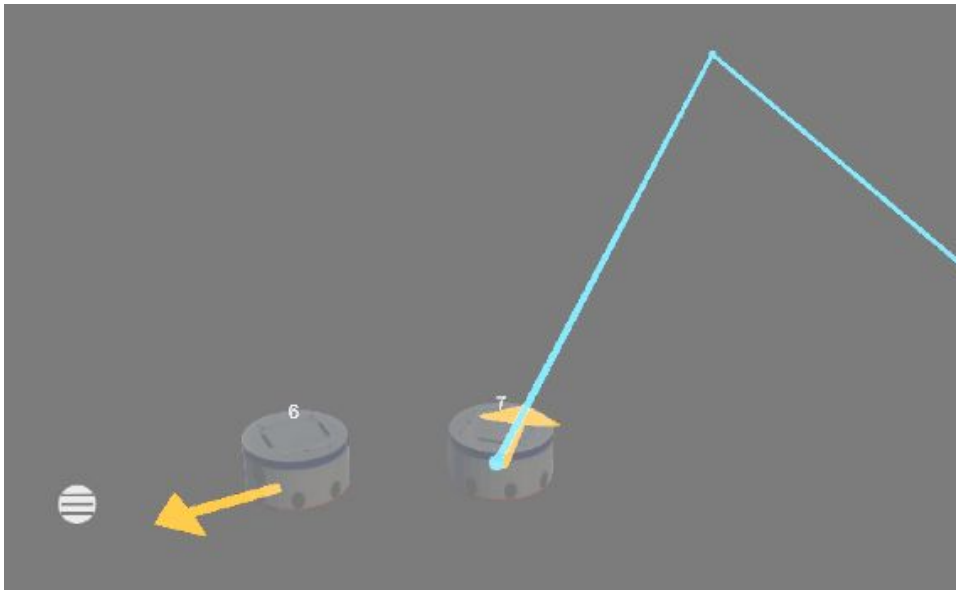

Figure 4.3: Path Display

Sonar Display

The sonar display provides similar contextual information for robot sensor information. Users get active feedback when robots see obstacles, and can quickly debug sensor failures. The packet protocol for sensor displays means developers can render sensor information with only an array of floats. The display assumes a radial arrangement of sensors, optimizing it for the Khepera IV robots.

The sonar display provides the largest transformation between the data received and rendered. The display receives data as a float, specifying the distance to the nearest obstruction. The rendered sonar data centers around its associated robot. The “x” symbols highlight a sonar sensor’s reading and clarify the mesh construction. Additionally, when the user highlights a robot, the associated sonar display changes color accordingly. The clear mesh display, and highlighting capabilities mean the display fulfills the clarity criterion.

Based on the index of a given data point, the display determines location to render one vertex of the mesh. Because the display knows the sensor layout, the vertices will appear at the correct location for any given robot. Thus, the resultant mesh shows what space the robot thinks is open. Therefore the display fulfills the second criterion by preserving the accuracy of the sonar readings in the mesh.

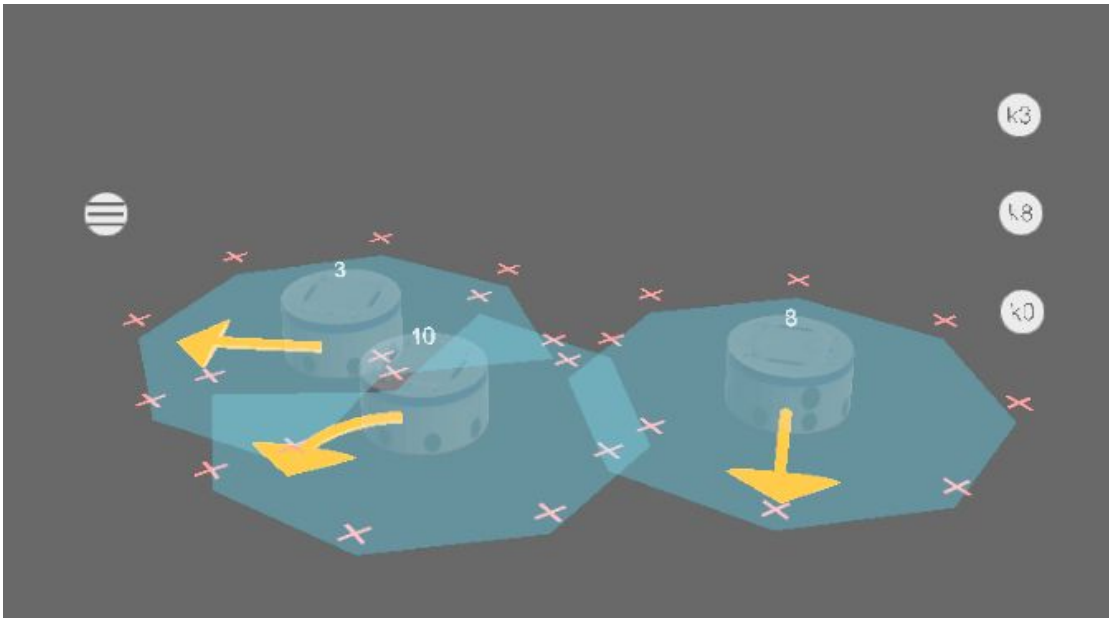

Figure 4.4: Sonar Display

Motor Display

The motor display provides information on the robot actuation. The motor speed is presented as a bar graph in green. It changes color to yellow to show a sign of warning if the motors actuated near zero or its maximum speed. Users can use this information to compare the behavior of the robot and determine whether the robot travels according to plan.

The motor display renders the speed of a motor using a bar extending in the relevant direction. The contrast between the bars and environment, as well as its location at the robots' center conveys data clearly. Thus, the motor display fulfills the clarity criterion for a successful display. The display conveys relative wheel speed instead of absolute rate. As such, the bars do not require a particular scale factor to represent the data. Because the bars scale proportionally, differences in wheel speed become apparent. Thus the display fulfills the second criterion for displaying data accurately.

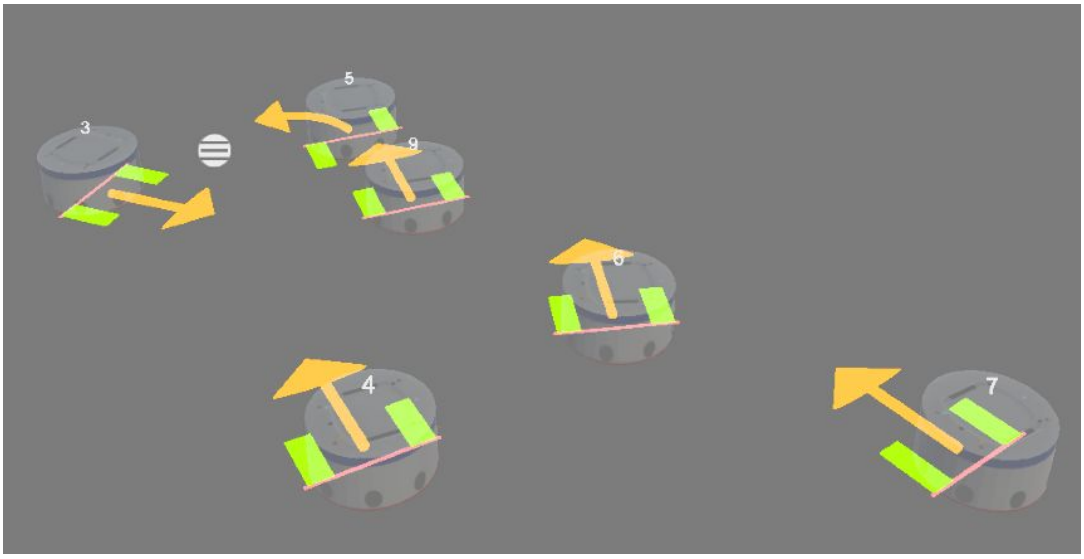

Figure 4.5: Motor Display

Entity Highlight Display

The entity highlight display provides functionality similar to the labels in ARGoS. The highlight display provides functionalities to both robots and goal objects. It shows the user the ID of the entity below it. A bendable arrow is attached to the robot to indicate the direction it is facing and its projected trajectory. The display also provides a clickable surface so users can move the entity or adjust display settings. When the entity highlight display is clicked, an entity menu display is brought up.

The entity highlight display presents entity ID, the highlighted entity, and travel direction. The entity ID appears clearly above the entity and the travel direction appears in front of the entity. As such, both elements are visible at all times when the entity is in view. When highlighted by the controller, a red mesh will appear over the entity. The high contrast means users can easily tell when they highlight an entity. If the entity is a robot, the directional arrow appears in yellow, a contrasting color against both the mesh and the environment. As such, the entity highlight display fulfills the first criterion for a successful display.

The display renders the entity ID as text from the data sent to Dr. Swarm2. Thus, Dr. Swarm2 can guarantee an accurate representation of ID labels from the packets sent by the server. The directional arrow bends based on the difference between motor speeds to indicate travel direction. Thus, for skid-steered robots like the KheperaIV, the arrow will accurately indicate travel direction. The highlight mesh appears at the location of the robot reported by the server. As such, the entity highlight display preserves data accuracy presented in the display. The entity highlight display fulfills both criteria, making it a successful display.

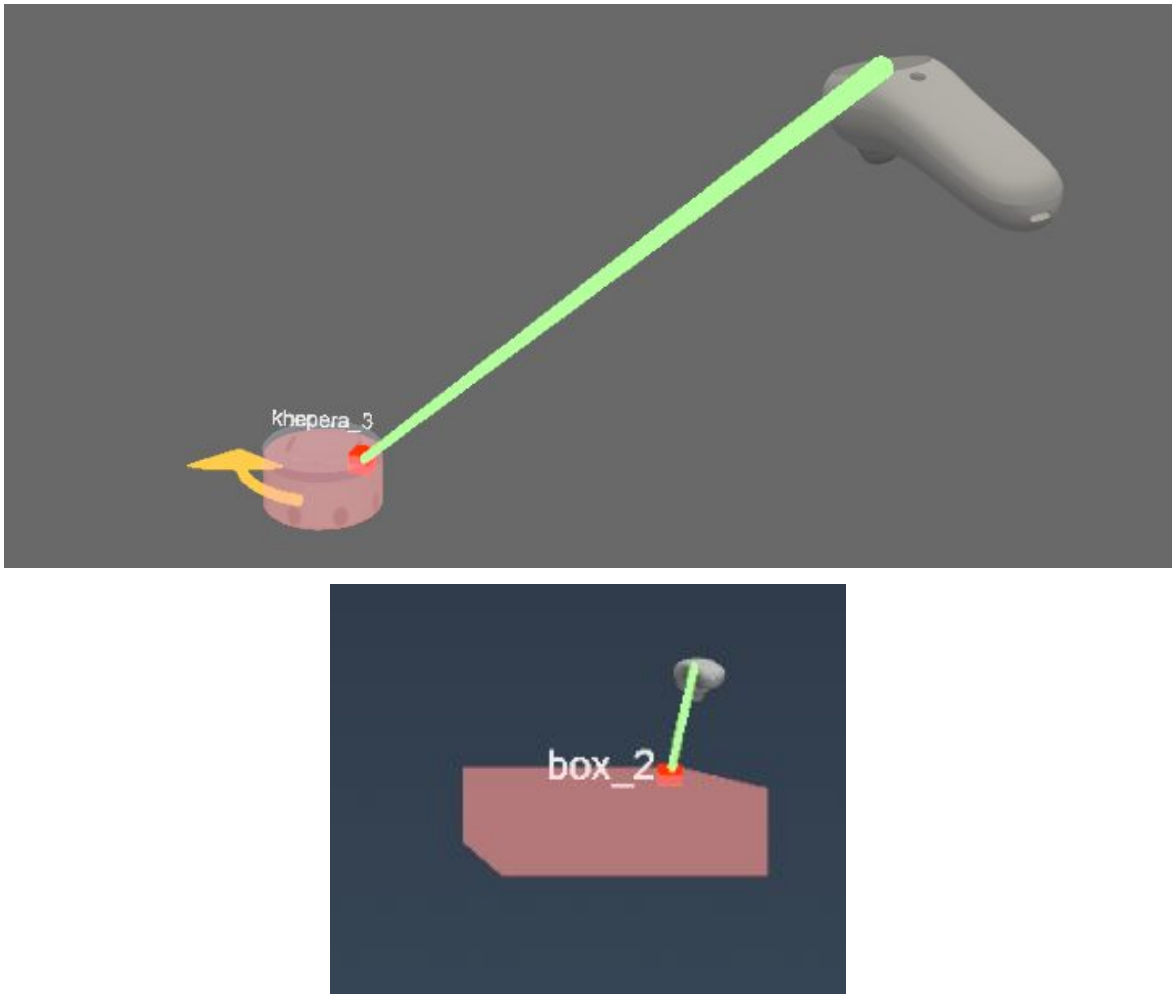

Figure 4.6: Entity Highlight Display (featuring robot and goal entity highlights)

All of the above displays meet both criteria for success. As such, the displays we provide for robot data provide an accurate overview of entity state and intentions. Developers can extend and modify these displays to preserve their accuracy in a variety of use cases.

The displays also demonstrate AR's capability to display types of robot data. To assist future work and guide our design principles, we created a taxonomy of displays (see Table 4.6). We developed the taxonomy by analyzing the data types provided by Khepera IV's and selecting a 3D display to represent the information. We chose a display type for each data type that is human-readable and preserves information fidelity in the display.

Entity Data Type	Display Type	Our Example
Current state information	Effective graphics or labels around the entity highlight display that the user can access through simple interactions	Entity ID label on the entity highlight display
Intended action	Repetitive elements with indication of time sequence	Path display
Sensor readings	Meshes or graphs that response to the change in environment	Sonar display
Actuation inputs	Graphs reflecting input magnitude	Motor display
Task-specific information	Variable depending on the defined task. If the data type fits in other information category, the developer can reuse the display by modifying a existing display for a different entity type	Goal object modification of the entity highlight display
Other information	Text box in proximity of the entity	Log display

Table 4.7: Summary of the display taxonomy

Application Control Displays

We also provide displays and controls to effect the enabled displays at run time. We refer to these displays as “global displays” given they are always visible from any location during run time.

Heads-Up Display

The heads-up display (HUD) provides a unified interface for disabling cluttered displays. Multiple displays opened during the same session can crowd pertinent information. The HUD displays at the peripherals of the user's vision. The display location sacrifices ease of use for extra visualization screen space.

We use the same criteria for success to evaluate the global displays. In the case of the HUD, it visually represents the state of displays in the visualization. It synthesizes the information from the Display State Manager, and represents the toggleable enabled displays from the Display State Manager as buttons in the sidebar.

The white buttons and grey background that appears when the user hovers over the sidebar of the HUD make it visible in any conditions. When the user clicks a button representing an entity's display, they see a flyout containing all enabled displays for the robot. The flyout allows users to easily view enabled displays for any robot. As such, the HUD satisfies the clarity of data criterion for a successful display.

Because the Display State Manager is the ultimate authority on enabled displays, the information displayed in the HUD will always be accurate. As such, the HUD satisfies both criteria for a successful display.

Figure 4.8: Heads Up Display

Entity Menu Display

Lastly, the entity menu display appears in the middle of the HUD when a user selects an entity, currently a robot or a goal. This allows users to quickly enable and disable displays for a specific entity. The display shows the options in a radial menu. The user can then toggle the enabled displays, with the touchpad or bumper. We chose the robot-level granularity to streamline visualizations for a single entity.

The radial menu, and its location in the center of the screen reduces the number of interactions needed to enable a display for any entity. We use the same buttons in the radial menu as the sidebar to improve visibility. As such, the entity menu display fulfills the criterion for clarity of data.

When a user toggles a display, the button color changes to represent an enabled or disabled display. Once again, the entity menu display retrieves and updates information with the Display State Manager. As such, it guarantees data accuracy, fulfilling both criteria for a successful display.

Figure 4.9: Entity menu display for a robot entity

Discussion

Dr. Swarm2 strives to quickly communicate robot intents to the human users. By overlaying the information in the workspace, users can compare various forms of robot data with robot's behaviors and make decisions for debugging, controlling the robots, and predicting their behaviors. Furthermore, since the backend of the application is flexible, developers can modify the displays to suit the need of different systems. The five displays we created serve as a good foundation for future AR robot visualizations and the two control displays serve as guidelines on how users can interact with the visualizations.

Connecting the state of a robot with its behavior in a multi-robot environment is challenging. Log messages in the robot control program convey state information in human readable form. We created the log display to associate the state information with individual robots on the field. When the robot behaves abnormally, the state information helps locate the issue quickly. In addition, seeing error messages in real time can prevent a cascade failure. The user can adjust the log display's location in space for additional visibility. When the display is set to robot follow mode, the user can closely monitor the robot and log. Pinning the display outside

of the field of view reduces clutter while keeping the information available. Multiple log displays help convey the overall state of the swarm. However, the display is limited to text information. Other analytic information, such as task progression, can be presented in similar displays made using canvases.

Knowing a robot's planned path is critical in predicting its behavior. The path display graphically represents the planned movement of the robot. During collaboration, users can see where the robots will move and avoid collisions. If the user sees the robot intends to move into prohibited areas, users can stop the robot and isolate the problem. Conversely, if the robot deviates from the intended path, developers can check the drive code using our visualizations as well. However, the path display only applies to robot controllers using a waypoint system for path planning. For other control systems, such as vector summing, Dr. Swarm2 indicates the path with a directional arrow on the entity highlight display.

We created the sonar display to showcase an effective method for contextualizing sensor data. The main advantage of the display is the fusion of robot location with the sonar readout. By creating a mesh, inconsistencies in readouts with the real world are easy to see. By extension, the display facilitates debugging drive issues by quickly exposing sensor issues. Additionally when collaborating with robots, users can confirm a robot sees them with the display. For drones and other robots that move in 3D, developers can extend the mesh calculator to render 3D sensor readouts.

Similar to sonar sensors, raw motor speeds can be difficult to monitor alongside location data. By displaying the motor speed in context with position, users can find issues in drive code faster. Additionally, users collaborating with robots can confirm a robot is moving out of their way so they can continue to work. Unfortunately, absolute speed is hard to render on the display. As such, users may incorporate the log display to get accurate readouts.

The entity highlight display is one of the few non-toggleable displays. It is enabled for all robots by default and identifies robots for the user. The highlight display provides a long press functionality for developers to attach functionality. For example, our sister control team expanded the highlight display to allow for entity movement. Short-clicking the entity highlight display brings up the entity menu display for that entity. Developers could also override the long click to close all displays or toggle a preset state. Like the other systems in Dr. Swarm2, we built the entity highlight display to be easily expandable for any situation.

Even though entity menu display and the HUD do not directly communicate robot state with the users, they make information easily accessible. They show the state of the application and allow the users to organize displays.

The entity menu display enables the user to toggle the displays available to a specific entity. Its location on the HUD makes the options directly visible to the user immediately after opening it. However the entity menu reduces visibility for other displays in the background, but the user can quickly dismiss it. The tooltip in the center of the menu explains how to toggle the displays, simplifying adoption of Dr. Swarm2. The user can control the menu with the pointer or

touchpad. The various selection modes minimize wrist and eye movements. The radio menu automatically adds and removes options depending on the displays registered for an entity, making it easily expandable. Currently, a robot entity menu display can enable log, path, sonar, and motor displays. Alternatively, a goal entity menu display can enable log and path displays.

We built the HUD using the head tracking canvas provided by Magic Leap. The HUD appears on the right side of the screen, making user interaction difficult in some situations. However, it allows for the display of large amounts of data without interfering with the workspace. Conversely, moving it to the top or bottom of the screen can improve usability, but distracts the user. The scrolling again increases the amount of data that it can display, but makes navigation to a specific robot difficult. Overall, the HUD exemplifies the trade offs necessary for a successful AR experience.

Overall, we successfully designed seven basic displays that allow the user to interpret and control the visualizations and the application. All displays have viable use cases and a clear path for modification in other swarm systems. The application centers around the Khepera robots, and the implementation and can serve as a foundation for future expansions.

V. Conclusion

The project aimed to improve the communication between robot swarms and humans. We accomplished the goals by expanding the functionality of the Dr. Swarm AR mobile application to the Magic Leap AR headset. The Magic Leap headset provides the ability to overlay contextualized data with the real world and allow users to navigate the workspace. We created four visualizations to showcase the three dimensional visualization capabilities of the headset. We created a taxonomy of 3D AR visualizations in place of the user study.

Throughout the implementation of the Dr. Swarm2 application we learned three key lessons. First, we learned to communicate data through graphics wherever possible. Graphic displays simplify synthesis of robot state, take less space than textual or numeric displays. Furthermore, graphics reduce cognitive load for the users. Secondly, being able to quickly access the graphical visualizations is also an important part of human-swarm communication. The major challenge of visualizing a swarm system is the large amount of information it generates--not only does it generate robot specific information, the swarm also has shared information. Only when the users can view the data in time as the error occurs, the information is presented effectively. Finally, a UDP packet structure allows for multiple and varied data sources as the application is server-agnostic. Being independent of the server implementation allowed us to finish the project during the pandemic.

Future Work

We accomplished the scope of the project in our 28 weeks of work. Our progress continued even in light of the COVID-19 outbreak limiting the physical testing possible. We switched our focus to implementing a robust backend and key visualizations to facilitate additions to the project.

Future teams should complete the physical user testing precluded by the pandemic. The users tests will reveal bugs and improvements critical to the project's ongoing success. Users studies will also accurately assess the benefits and drawbacks of the current application. Physical tests and user studies will highlight additional paths for future development.

Moreover, additional displays will help overcome the limitations of Dr. Swarm2. The complete and extensible backend will allow future teams to create complex displays to address user needs found in studies. For future experiments, the team should implement more swarm level visualizations tailored to the experiment. Future teams can leverage A/B testing to optimize existing displays and discover additional features for Dr. Swarm2.

VI. Appendix

Appendix A: User Study Experiment Setup

The first experiment will simulate a user-robot collaboration assignment. In the experiment, the user and robot must move a number of boxes from one side of the room to the other. The time required for the team to complete the task will depend on the efficiency of the collaboration. The robot will select boxes to move randomly, so the user must not move the same box as the robot. The methods of communication between the robot and user are as follows:

- Test 1: Only log statements from laptop
- Test 2: Only contextual log display
- Test 3: Contextual movement display
    - Test 3a: Path display
    - Test 3b: Directional arrow
    - Test 3c: Path with label on destination

Users will then evaluate their experience with the following rubric:

Test	Did you complete the task? (yes/no)	How cognitively challenging was the task? (1-10) 1 being easy, 10 being impossible	After using all methods, would you use this one again? (yes/no)	How long did the task take? (seconds)
1.				
2.				
3.				
3a.				
3b.				
3c.				

We will compile rubric responses with the task time to determine effectiveness. The results will tell us the most useful displays and the time saved with our method.

The second experiment simulates a robot debugging session. In the session, the robot has a faulty path planner and the user needs to find the solution. We will change the solution for each test so results from one test do not bleed into another. The users will perform the tests with the following methods:

- Test 1: Only log statements from laptop
- Test 2: Only contextual loggers
- Test 3: All displays available
- Test 4: Only path and obstacle view

As with the last test, the user will record their feedback with the following rubric:

Test	Did you complete the task? (yes/no)	How cognitively challenging was the task? (1-10) 1 being easy, 10 being impossible	After using all methods, would you use this one again? (yes/no)	How long did the task take? (seconds)
1.				
2.				
3.				
3a.				
3b.				
3c.				