swARm: Augmented Reality Swarm Control with Multiple Head Mounted Display Operators

A Major Qualifying Project
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment for the requirement for
degree of Bachelor of Science

By:
Harrison James (RBE/CS)
Paul Mara (RBE/CS)
Vlad Stelea (CS)

Date Submitted: May 6, 2021
Report submitted to:
Professor Carlo Pinciroli
Worcester Polytechnic Institute

Abstract

Methods of human swarm interaction are becoming increasingly prevalent in academic literature. Recent approaches have studied how augmented reality applications can facilitate control of robotic swarms. However, studies suggest that robotic swarm systems can overload the operator due to the large number of robots involved. We developed a novel multi-user, head-mounted, augmented reality application, deployed on the Magic Leap One. Our application allows operators to distribute responsibilities among each other when controlling robotic swarms in an immersive and intuitive way. Furthermore, using our modular architecture, new methods of multi-user swarm control can be introduced and evaluated quickly.

Table of Contents

List of Tables/Figures

Chapter 1: Introduction

1.1 Swarm Robotics

Swarm robotics is the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the agents and local interactions among agents [1]. The design of these systems is generally based on collective behaviors commonly found in nature such as flocking of birds, schooling of fish, and nest construction of social insects which share the same main advantages of scalability and robustness to the failure of individual agents. Currently, many different platforms for studying swarm robotics exist with some of the most common being Khepera, e-pucks, and Kilobots. Even with the field in its infancy, research in swarm robotics has shown strong relevance in applications that can take advantage of numerous coordinated agents such as reconnaissance, exploration, environmental monitoring, protection, and even space exploration [2].

1.2 Human Swarm Interaction

With the rise in relevance and usage of swarm robotics, new applications and research areas are being created. Due to the current exploratory nature of the field, autonomy frequently does not cover every case that can arise in a robotic swarm's set of possible states. When autonomy fails, it is important for robotic swarm systems to give humans the ability to correctly react to the situation [3]. This need has led to a new field of study known as Human-Swarm Interaction (HSI) which is the study of human operators controlling or giving commands to a robot swarm [4]. The main advantage of introducing a human supervisor to the swarm system is that an operator could use information unavailable to the swarm to prevent automation errors and improve performance in the system by modifying the swarm's behavior and goals. This effectively allows complicated missions to be completed by combining the autonomous functionality of robot swarms with relatively simple forms of interaction from human operator control.

However, due to the nature of robotic swarms, these systems can easily become complicated due to a large number of interacting parts. As such, these systems can exceed the span of apprehension (cognitive load) of an individual human, which is typically limited to 7(+/-2) entities [5,6], which can limit human performance, negatively impacting the performance of the entire system.

One way to approach the cognitive load problem is to allow operators to choose between a broad and specific level of control. For example, if an operator wants the swarm to

work together to achieve a goal, they should be able to indicate a task at a high level. If the operator noticed one of the robots is performing unexpected behavior, they could manually correct that individual robot. This mixed granularity approach has the benefit of not overloading the operator with information while allowing them to use fine-grained control in certain situations.

Visualization is an important aspect of human-swarm interaction because it enables humans to understand the swarm's intentions. This signals when operators should modify swarm behavior. Analogous to low level control methods, showing every detail of a swarm can overload operators [6]. Because of this, it is important for swarm control interfaces to provide a way to present only relevant data based on the context or user commands.

Even though granularity of control and visualization provide ways to reduce the cognitive overload that operators encounter in swarm systems, it is often not enough [7]. Adding more operators allows humans to split work amongst each other and complete their tasks in parallel. However, operators in a multi-user system will often suffer from the out-of-the-loop performance problem (OOTL) characterized by operators not engaging in the task, not trusting the system or not being aware of the current state [7]. While a previous work has already examined OOTL within swarm control on a tablet, we will be looking at how a head mounted display (HMD) affects this issue.

Head mounted AR displays have three benefits over tablet displays. First, head-mounted displays are more immersive because they allow the operator to look around the environment naturally as opposed to having to point a tablet at interactable elements. Additionally, not holding a tablet allows operators to use their hands for other tasks within the physical world such as looking through physical documentation. Finally, humans are able to communicate with each other much better when they use gestures [8]. By having their hands free, operators will be able to point at elements of the system.

The Magic Leap headset [9] is the AR HMD selected for this work. The Magic Leap pairs a relatively lightweight set of glasses with a wearable compute unit to keep stress on the head and neck low. Magic Leap also comes with a 6-DOF controller which can allow precise selection of a robot or a team of robots and mappable buttons for predefined actions or UI navigation. To our knowledge, no research has studied a team of humans using Magic Leap or other AR HMD headsets with a swarm of robots. While our study has a focus to measure cognitive load, other findings about the system will be recorded as they are novelties.

1.3 Problem Statement

Robotic swarms are useful systems for solving tasks that are either too complex for one robot or inherently distributed. In robotic swarms, multiple robots can solve problems faster through parallelism while being more robust to failure than a single robot. Current robotic swarms typically operate well for situations they are designed for, but they can break and require human intervention in situations they have not been prepared for. In current human-swarm systems, humans need to update goals for an individual agent or for overall swarm behavior. However, swarms can have many parts in the system, exceeding the span of apprehension of a human operator which is limited to 7(+/-2) entities. This can lead to operators being overwhelmed and cause performance issues across the swarm. Therefore, we propose a first of its kind Magic Leap One augmented reality system for distributing relevant information across multiple humans to allow operators to share the cognitive load associated with controlling and understanding swarms.

1.4 Contributions

Our contributions are as follows:

- Created the first multi-user head-mounted augmented reality control system for robotic swarms to balance cognitive load between operators.
- Improved on previous year's major qualifying project control and visualization system by introducing entity locking, marker placement and voice commands.
- Extended previous year's major qualifying project features to move from simulation to real world space.
- Decoupled existing control system architecture allowing us to easily reuse components.

Chapter 2: Background

2.1 Augmented and Mixed Reality

Augmented reality (AR) is a technology encompassing three key aspects: combining real and virtual objects in a real environment, being interactive in real time, and registering real and virtual objects with each other [10]. The definition of *mixed reality* (MR) is not as unified or unanimous as the definition for AR [10], but it includes components from both the physical world and a virtual world. We will use a working definition of MR as a stronger version of AR [10] which includes a stronger link between the virtual and real world, allowing for registration and control of physical objects from the AR environment. AR/MR's physical context contrasts virtual reality (VR), where the entirety of a user's world is virtual.

2.1.1 Affordances of Augmented and Mixed Reality

Functional affordances [11] enable goal-oriented actions for a specified user group. Augmented and mixed reality provide unique functional affordances in a system that may be impossible to experience in physical or virtual reality alone. First, AR and MR allow the user to be free of many constraints of the physical world such as gravity or time, through the use of virtual objects [12]. Virtual reality also allows for the breaking of normal physical laws, but AR/MR do this while maintaining the physical, real word context of the environment [12]. MR provides specific affordances to mobile robots, where robot information and control can be overlaid onto a moving model of the robot [13,14,15], therefore we selected it for this project.

2.1.2 Current Challenges in Augmented and Mixed Reality

AR and MR papers have been published for over thirty years, and significant work has gone into making the technology better in every way, but there are still major challenges that prevent it from being widespread today. AR and MR require precise tracking of the user in all environments [16]. This is to link the view for the user with objects in the real world. Display technologies have also left AR systems with limited field of view (FOV). Controlling AR/MR systems without the use of a keyboard and mouse has proved challenging, and new input methods have yet to be standardized. These current shortcomings are important to design around while implementing an AR/MR system.

2.2 Related Work

2.2.1 Studying Granularity of Control

According to Endsley [17], granularity of control (a system design choice) is a component in determining: (1) the situational awareness of operators, (2) how users build their mental model of the system, and (3) human perceived complexity of the system. Two main granularities of control are considered here: low-level control, and high-level control.

Low-level (or robot-oriented) control focuses on an individual robot. This may mean controlling a single robot manually, or giving a single robot any other robot-specific command. Any actions chosen with this granularity directly impact the selected robot and may indirectly impact the rest of the swarm behavior.

High-level control focuses on high-level goals. It can include controlling an entire swarm directly or stating goals to be accomplished and letting the system determine how to complete the goal on its own. These differences of high- and low-level control have been examined by Patel *et al*. through two approaches to collective transport [7]. Controlling the swarm directly would be commanding the robots to surround a box, and then telling the group to move, moving the box along with the robots. The goal-based approach would instead allow the users to indicate they want the box to be moved from one place to another, and the robotic swarm solves this problem autonomously. Adding robots to a group of robots is also considered high-level control. High-level granularity control commands can directly impact the swarm(s) they are being applied to, and indirectly impact other swarms.

Choosing the right level of control for a task can be a difficult choice. Both high- and low-level granularity are studied in [7], specifically in the context of the out-of-the-loop (OOTL) phenomenon. The OOTL problem occurs when a user becomes disengaged from the work of the system, for a variety of reasons.

If the granularity is too high, such as solely goal-based control, the limited workload can lead to operators becoming bored and disengaged, resulting in the OOTL problem. However, if granularity is too low, as in the case of robot-oriented control, operators might have to keep track of too many different entities and reduce their efficiency because of excessive cognitive load. Because of this, research has shown that allowing operators to dynamically choose their modality of control is important [7].

Mixed-granularity AR swarm control has been previously achieved through the use of a tablet application [7,18]. Patel *et al.*'s tablet application study found that for the purpose of moving a box to a specific location, giving users the option to use both high level and low level

of granularities led to a reduction in mental load, effort and frustration over just low-level control. This can be caused by high cognitive load. This study also found that mixed-granularity control led to fewer interactions and commands [18].

2.2.2 Multi-Human Multi-Robot Interaction

Many research groups have studied multi-human multi-robot interaction which can be generally split up into cases where humans play the role of bystanders, coordination of humans and robots into teams, and supervisory control [18]. However, there exists little research on supervisory control.

In the case where humans play the role of bystanders, most research has been conducted on the navigation of robots in a shared environment and human tracking. One of these studies developed a path-planning algorithm where humans with different mission objectives assigned waypoints for the robot that would imply fair allocation of resources to the users [19]. They found the elitist strategy, where paths cover waypoints selected by an assigned individual or sub-group of users was the most preferred and successful. This is useful as it gives some insight on how swarm resources should be allocated in a multi-human multi-robot system, indicating the preference of users for individual robots to be assigned to a specific user or sub-group of users. Another study developed a behavior system for analyzing human features and classifying behavior. The system allowed robots to identify which user was most likely to interact with them, simplifying the user priority in a multi-human system [20].

Many researchers have also studied the coordination of humans and robots in teams. One study focuses on how to improve the safety, efficiency, reliability, and cost of achieving mission goals by creating an infrastructure to facilitate cooperative and collaborative performance of humans and robots as equal team partners. In this paper the researchers explore the issues that arise from dependency on other teammates, namely when teammates are busy and unable to respond. To solve this dependency issue, the researchers introduce adjustable autonomy as a solution [21]. This paper suggests a relevant method for allocating systems and entities into hierarchical control for successful human robot team operation.

Other research has focused on dynamically formed human robot teams. One such study by Jones *et al.* defines the challenges of pickup teams involving efficient formation, coordination, and interaction of multi-robot teams [22]. In Jones *et al.*'s work, the researchers design and develop the TraderBot allocation system to tackle a search and discover problem in an unknown environment. Another study defines and analyzes various types and levels of interactions between humans and robots in the manufacturing domain by developing an

infrastructure for collaboration based on safety, engagement, and team composition. Their main findings support the claim that higher levels of collaboration in human robot systems lead to improved productivity [23].

Some groups have studied supervisory control within multi-user, multi-robot scenarios [24,25], but these have been limited to resource sharing relating to task performance. These studies presented two control scenarios, one with a pool of 24 robots users shared and monitored, and another where the users split the pool in half and monitored 12 robots each (a third, ‘no discernable strategy’ was also selected by some groups of participants). These papers presented work with multiple robots and users, but all of their work was simulated and focused on human teams and information streams. Most notably, the first study observed different team strategies where operators had access to all robots manually or autonomously and where users had access to only 12 robots each which were specifically assigned to individual users. The best performing teams, in terms of victims found, were those who had autonomous robots that were assigned to individuals. The researchers also found that when both operators had access to all information feeds from each robot, it would begin to interfere with their ability to identify victims. This suggests that limiting the amount of control and information to which an operator has access can improve the overall performance of the team.

In [7], supervisory control was used to study the OOTL problem. This study used a tablet interface, unlike our proposed head mounted display interface. Task engagement, trust in other users, and trust in the system shared between users (all parts of the OOTL problem) may change within the new hardware and interface.

2.2.3 Human Swarm Interaction in Augmented Reality

To our knowledge, there are no established methods for controlling swarms in cooperative mixed reality with a head-mounted display. There has been a previous attempt which involved using an AR interface on a tablet to instruct swarms how to construct structures from cardboard boxes [7].

One proposed method of controlling swarms in augmented reality was through the use of hand gestures such as pointing [26]. This research found that gestures alone did not provide a robust enough control method as an operator's hand would have to constantly be visible within the field of view. Additionally, they noted that the Magic Leap had a small field of view which exacerbated the problem. Because of this, they found that the optimal way to control

swarms was by using the provided Magic Leap controller. They also suggested the use of voice commands, but they also noted the Magic Leap's lack of native support for them.

Additionally, being able to understand the state of multiple devices is a common problem which has been researched within both robotic swarms and interaction with IoT devices. For example, one paper found that communicating information through graphics synthesized data in a simpler form than textual or numerical displays [27]. Limiting view complexity is of particular importance with augmented reality headsets because of their small field of view [26].

One way of limiting the size of views within Augmented Reality is through the use of Proximity based control. Proximity based control involves dynamically changing how much information is displayed to the user based on how close the user is to a certain interactable item [28]. It relies on the assumption that users are more interested in nearby interactables than far items.

Another problem with Augmented Reality headset's limited FOV is that it is easy for important objects to exit a user's field of view. One way this problem has been solved has been through the use of "Picture in Picture mode" [28]. This solution overlaid small icons that represented entities of interest on the edges of the screen to let users know where out of view entities were.

2.3 Non-verbal Communication

Non-verbal communication is a method for humans to convey information without using speech. It can take many different forms including the use of space, body movements, gaze and the placement of artifacts. In fact, gestures can sometimes fully replace the use of verbal communication altogether [29].

One place where non-verbal communication has been implemented is in the online game League of Legends where developers have introduced "pings." Pings are a form of spatial communication where players place markers representing different information such as danger, stop, or the need for help. In fact, one 2016 study found a positive relationship between the number of pings a player activated and their performance throughout the game [29]. One hypothesis on why this is the case is that League of Legends is a fast-paced game in which having to process text information distracts the player. However, pings are not a perfect system. In the same Ping to Win study, researchers found that when users spammed pings their team actually performed worse than if they did not spam them. The researchers claim that this was

caused by the continuous interrupting of work that each ping did, without actually adding new information [29].

Our project involves multiple operators participating in an experiment in an augmented and mixed reality environment. There are many cases in which pointing to a robot, or a 3D location through the system may aid or replace the need for a verbal cue between operators. We included non-verbal communication elements to improve human to human communication in our project.

2.4 Our Project

Our project leverages the technology of two Magic Leap augmented reality headsets to enable an immersive augmented reality, mixed granularity control of a robotic swarm. The Magic Leap enables 3D virtual objects that can give users intuition about the system to build a coherent cognitive model of the system. The infrastructure that currently supports one Magic Leap to control the swarm will be modified to allow for two users to view and control what is occurring in the system. Our project gives users the ability to collaboratively interact and control a robotic swarm while simultaneously providing intuitive non-verbal communication features. By allowing two users to interact with the swarm through augmented reality with HMDs, we create a novel method of cooperative swarm control.

Chapter 3: Methodology

3.1 Problem Formalization

Controlling and understanding complex robotic swarms is a difficult task for one human operator. Because individual robot's states can become complex, it may require the complete attention of one operator. This narrowed view of the swarm can lead to an operator's inability to interact quickly and effectively with other robots within the swarm. With slower responses, severe problems arise: increased safety risk, decreased swarm efficiency, and lowered operator trust in the system.

To address the single user attention span problem, we propose a system in which multiple users (two studied within this project) may share the cognitive load of interacting with the swarm. Through this multi-user application, we want to accomplish the following goals: (1) intuitive human swarm interaction and (2) improved inter-human communication. To measure these goals in the future, we have created an experiment template with performance metrics derived from the survey in Appendix A (following from Patel *et al.* [18]).

Human swarm interaction: Each user in the system should be able to control and understand the robots they are working with. We desire both high- and low-level control for the robots. Our sub-goals for human swarm interaction include the following.

1. Perform low level control of robots. This includes moving a robot from one location to another.
2. Perform goal-oriented control of robots. For our experiment, this means issuing a command to move the box, and allowing the system to work out which robots will perform the move.
3. Understand robot behavior. Why are the robots moving the way that they are? The displays in the system should allow our operators to understand: (1) what a robot is doing, (2) if this differs from what it is trying to do, and (3) if either of (1,2) differ from the expected behavior.

Inter-human communication: Each user in the system should know what other users in the system are doing and be able to communicate this in the MR environment, with the following information.

1. Intended Actions - users should be able to understand the actions that other users are taking.
2. Locked Resources - users should be aware of what resources (robots, movable items) they can and cannot use so that they are able to negotiate with other users.
3. Important elements - users should be able to indicate on what other users should focus.

3.1.1 Measurements of Goal Success

Human Swarm Interaction

Success for control commands will be measured by the robustness and reliability of control commands for both high- and low-level control. To evaluate the operators' understanding of robot behavior, survey questions regarding the swarm's actions and its intended actions can be administered.

Inter-Human Communication

The measures of success for inter-human communication rely heavily on survey questions. Possible survey questions that could evaluate the success for inter-human communication include teammate trust and system trust questions. We expect that increased uses of inter-human communication (some verbal, some non-verbal) would result in increased trust in teammates.

3.2 Use Cases

The full use cases, including entry/exit conditions as well as the flow of events are attached in the Appendix B: Use Cases. The following lays out a brief generalization of the use cases this project aims to cover. These include command use cases, such as:

- Issue Collective Transport Command
- Issue Create Marker Command
- Issue Move Single Robot Command

Users should also be able to see information about a currently selected robot so that they can make inferences about the state of the swarm and inform future commands. These include:

- Highlight Selected Robot
- View Selected Robot Direction of Travel

We believe that users need to be informed of their teammates actions and intended actions, including:

- Highlight Teammate Selected Robot
- View Created and Teammate Created Markers

3.3 System Hardware

3.3.1 Magic Leap Augmented Reality Headset

The Magic Leap One Creator Edition [9] is a wearable, light weight, spatial computer that provides the user with the ability to work and interact in both augmented and mixed reality. These features are made possible by overlaying virtual elements and interactions over the physical world in real-time. The real-time overlays are made possible through the use of infrared transmitters and receivers which are integrated with the Magic Leap's cameras to track both the room and objects within the space. This allows the system to then project overlays onto the lens of the Magic Leap providing the augmented and mixed reality experiences. With a horizontal FOV of 40 degrees and a vertical FOV at 30 degrees the Magic Leap One provides one of the largest FOV currently available for augmented reality headsets [9]. Previous contributions and infrastructure from the aforementioned DRSwarm-Gestures [26] study also provides considerable reuse of work and documentation for application development for the Magic Leap headset [26].

3.3.2 Magic Leap Controller

The Magic Leap controller (Figure 1) provides a user the ability to point at and interact with virtual objects. The controller provides two digital buttons with on/off states, the bumper and home button. The controller also has a trigger with analog states, ranging from 0-1 giving information of how much the trigger is pressed down. Also, the touchpad on the controller can be used for different modes of control. Magic Leap default libraries recognize 10 default gestures on the touchpad, with the ability to expand this to custom gestures.

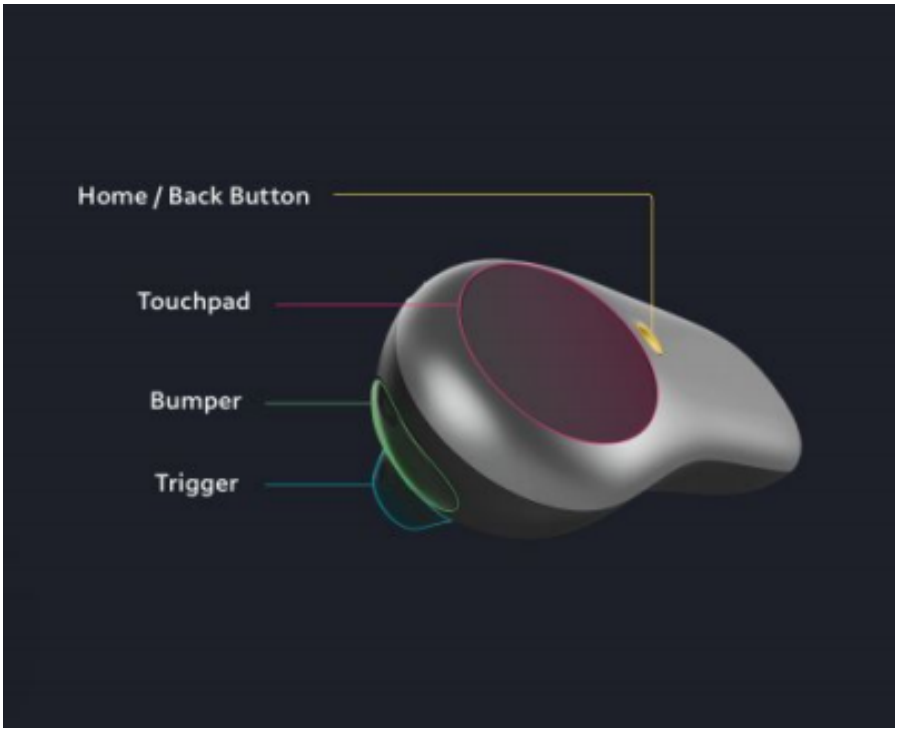

Fig 1. Magic Leap Controller Layout [30]

The physical controller was chosen for our implementation due to its high level of accuracy and speed of updates compared to pointing gestures with the Magic Leap. In addition, the controller can be held outside of the Magic Leaps already limited FOV while pointing at objects within the FOV. The design of the controller fits well in users' hands and can handle simple button presses well, with the possibility of extending into more complex combinations of interactions.

Utilizing the advantages of the controller, the Magic Leap is able to reliably interact with the environment through ray casting. Using the Unity Engine [31], ray casting allows for a ray to be projected from a specific point and orientation and returns any Unity Game object that intersects with that ray. Ray casting from the controller allows for accurate identification of where a user is pointing and what object the user is pointing at.

3.4 Networking Layer

3.4.1 ARGoS for Robot Positions and Commands

To keep track of both robots and sensor data, we use the ARGoS [32] system. ARGoS is a multi-physics robot simulator with the ability to simulate large-scale swarms of robots and the

ability to be customized with ease through the addition of plug-ins. ARGoS is able to achieve this by incorporating the design features of modularity, multiple physics engines, composability, and parallelism. For our purposes, ARGoS was used as a server which receives commands, sent from the Magic Leap, processes them, and generates high-level motion goals for the robots of the swarm. ARGoS then sends these motion goals to the robots which execute these goals [26]. This can be seen in Figure 2.

Fig 2. DRSwarm-Gestures Magic Leap, ARGoS, Swarm Communication System [26]

3.4.2 Web Services

We used web services to support the functionality that could not be done solely on the magic leap. We have two main services in our project: *SwarmServ* and *SwarmSpeechServ*. SwarmServ is responsible for handling all of the multi-user communication functionality (markers, highlights, locking). SwarmSpeechServ is able to transcribe spoken commands and is the backbone of our speech processing functionality.

In order to support shared visualizations of non-verbal communication features, SwarmServ is responsible for validating that the sent commands are valid (especially important for locking features) and for distributing that information to all connected clients. SwarmServ has three easily swappable layers: networking, data, and handlers. The handler layer is responsible for doing all the business logic (validating that a user can take an action). The networking layer is responsible for communicating between clients and the server (our current implementation uses a mix of HTTP and TCP). The data layer is responsible for storing all the state information that the handler will use to do its processing. Because these layers are all easily swappable, if in

the future a developer identifies a bottleneck in one of the layers, they will be able to quickly swap them out without worrying about affecting the rest of the system.

Because our system needs to run on a local network without internet access, we are not able to contact external services. This meant that we have not been able to use any of the major speech processing services such as GCP on which the Magic Leap's built-in speech to text relies. Because of this, to handle voice commands, we had to build a locally deployed service which can process WAV files of given commands and return 1 of 4 supported commands (place marker, delete marker, lock and unlock). To avoid training a speech model from scratch, we used the *CMUSphinx* [33] speech recognition library. Because out of the box this library supports full sentence transcription which would potentially pick up non-supported commands, we configured CMUSphinx to only support pre-determined key phrases.

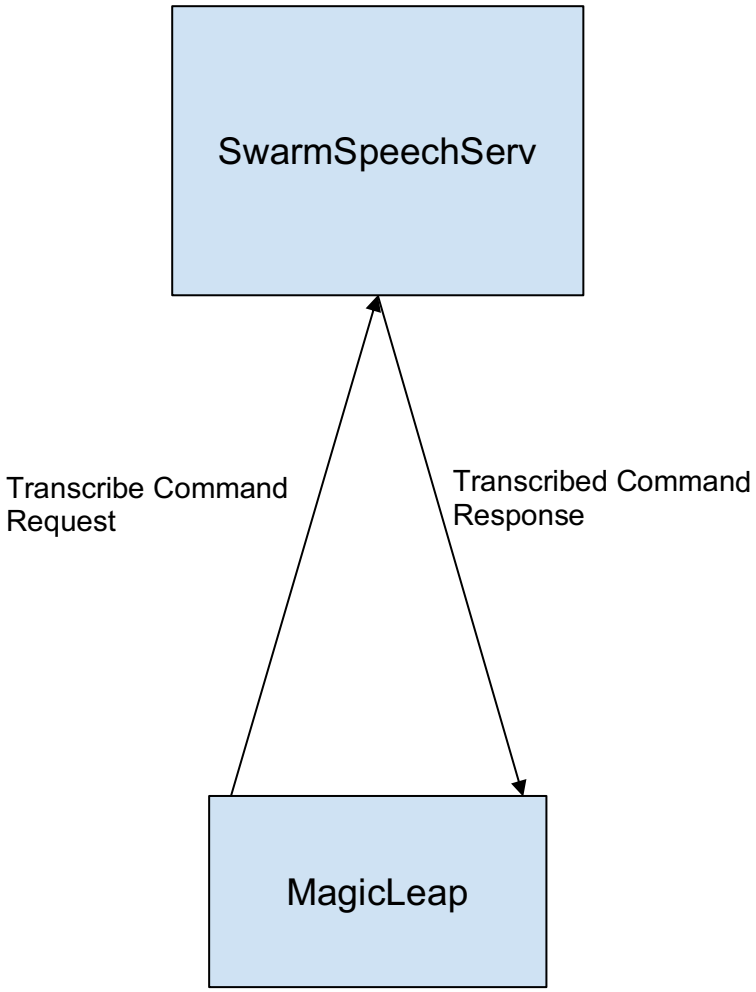

Fig 3. Request Response pattern for SwarmSpeechServ

3.4.3 gRPC: Ideal SwarmServ Solution

In order to communicate with SwarmServ, we looked into the use of the *gRPC protocol*. gRPC is a networking protocol where clients are able to call methods directly on a server as if the server was a local object [34]. The request response model used by gRPC can be seen in Figure 4.

gRPC uses *Protocol Buffers* as the default form of message serialization [34]. A protocol buffer specification can be written within a .proto file in a declarative language. Because when we declare fields within a protocol buffer message we also have to specify the data type, a protocol buffer message can be optimally encoded. This is different from a JSON message which is encoded as a large string and then parsed by the receiver. After a developer writes a *.proto* file, they pass it into a language specific compiler which will generate the bindings required to use gRPC method stubs.

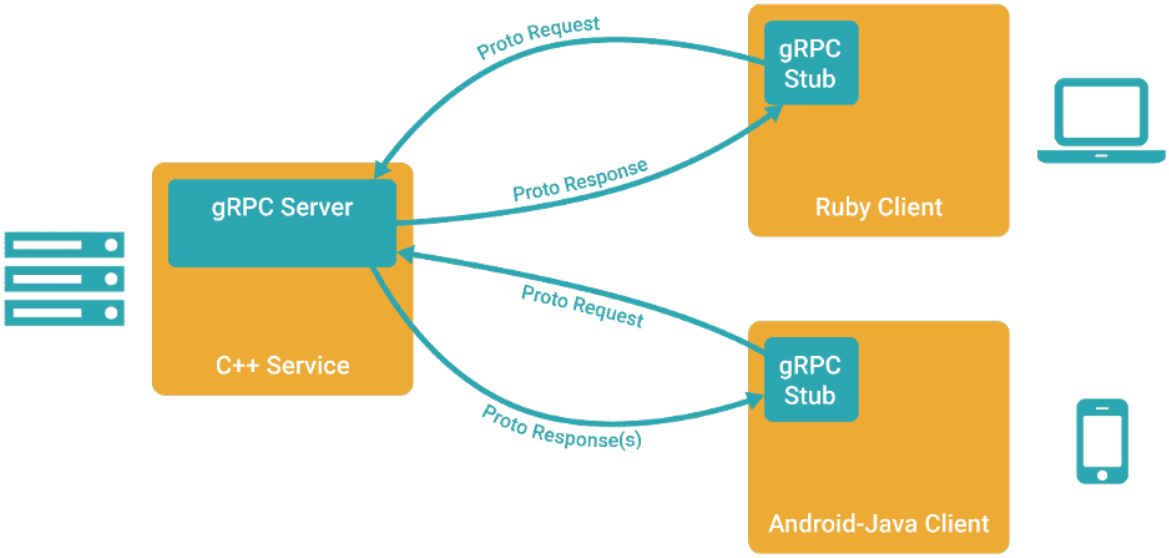

Fig 4. gRPC client calling gRPC server stub [34]

gRPC also provides four combinations of service method types: *Unary*, *Server Streaming*, *Client Streaming* and *Bidirectional Streaming*. In a Unary call, the client sends one request and receives a response. A Server Streaming call involves the client sending an initial request to the server and reading a stream of data from the server. In a Client Streaming call, the client publishes a sequence of messages to the server. When the client is finished with this

sequence, the server will respond with a single message to the client. Within a Bidirectional Streaming call, both the client and server can send messages to each other until the stream is closed. A visual representation of the four service method types can be seen in Table 1. Additionally, unlike in UDP, all the streaming method types preserve message order [36]. It is important to maintain message order because messages may contain information such as a certain feature being on/off, and if these arrive out of order, it may cause incorrect state representation of the receiving end of the gRPC messages.

Table 1: gRPC service method types

	Server Single Response	Server Streaming
Client Single Request	Unary	Server Streaming
Client Streaming	Client Streaming	Bidirectional Streaming

Knowing this information, we initially chose to use the gRPC protocol for the following reasons:

1. Protocol buffers are able to provide a smaller payload than if we had used a JSON string;
2. The four possible service method types account for the entire range of connection models we would need; and
3. The standardized API between the server and client means that we are forced to update both when the specification changes.

However, gRPC had one major issue which prevented us from using it: the inability to compile for the Magic Leap. Because gRPC required the use of native binaries not available pre-built for the Magic Leap, the IL2CPP tool that is generally used to package native code during builds would fail. As such, we had to remove the use of gRPC within our application after already writing most of SwarmServ. If moving to another AR headset platform in the future, utilizing gRPC could provide great benefits, unfortunately, we had to find an alternative.

3.4.4 Replacing gRPC: HTTP/TCP Hybrid

In order to replace the gRPC backed networking layer, we needed to find a networking solution which would be able to work out of the box with the Magic Leap as well as easily support both unary calls and server streaming. To solve this, we use a HTTP-TCP hybrid server where the

HTTP portion supports unary calls and the TCP is able to efficiently support server streaming. We chose HTTP because it had the built-in request/response structure as part of the protocol, useful for issuing requests and receiving responses about the results of our request. We chose TCP for the streaming side because we needed something fairly lightweight, but also something that guaranteed the order in the packets it sent and guaranteed their arrival. Both of these properties were important for the teaming features. For example, if User A's un-highlight message is lost or sent out of order from SwarmServ to User B, then the robot would appear highlighted to User B until User A re-highlights and un-highlights the entity again. Using TCP to guarantee delivery resolves these issues.

In contrast to gRPC's protofiles, HTTP and TCP do not have inherent client/server contracts. We used *Swagger* [37] to document the HTTP client/server contracts. Swagger uses YAML syntax to define HTTP contracts and displays service names and details in a UI. This visual element (see Figure 5) was helpful during development and will be easy to read for future teams.

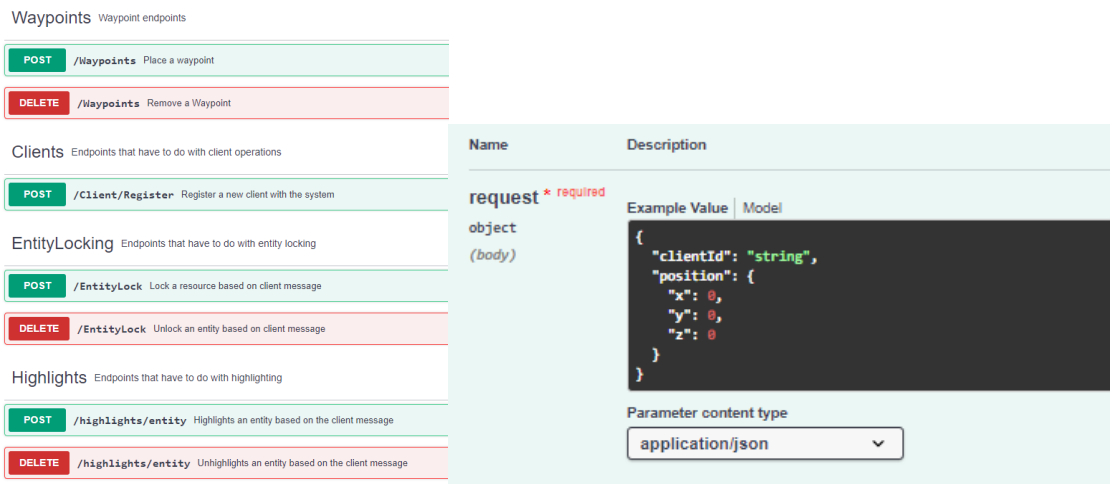

Fig 5. Swagger List of Methods and Marker HTTP Post Example

Clicking on each method from the left image expands it to show image on the right, as well as HTTP message codes

For the TCP connection, we used a wrapped JSON technique to deserialize multiple objects on a single connection. Once per frame the TCP connection checks for updates from the socket connection to SwarmServ and then performs deserialization in a two-step process. Figure 6 shows the structure of our JSON objects.

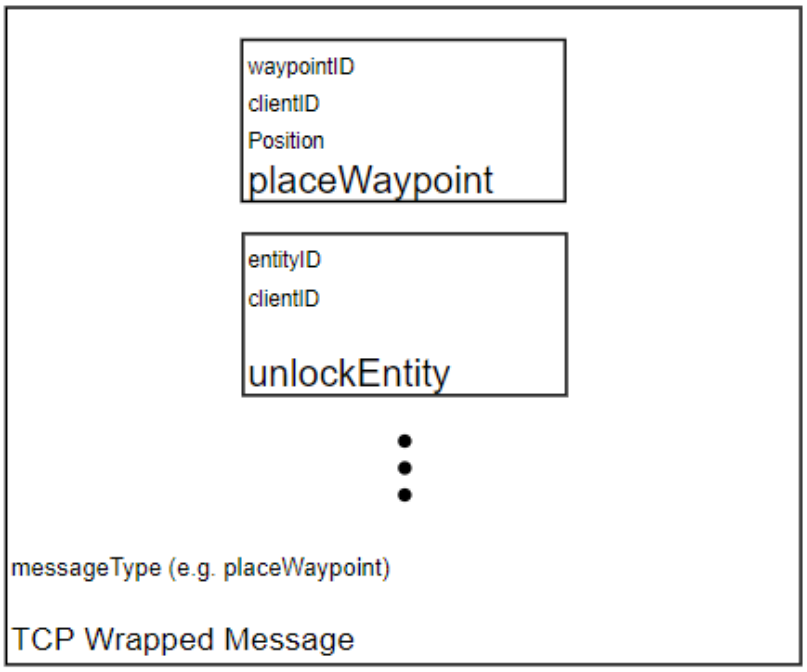

Fig 6. TCP JSON Structure

We stripped off the `messageType` header (e.g., "*`placeWaypoint`*" for placing a marker) from the original JSON by processing the `messageType` then serialized further according to the message type for this specific message.

We found the transition process between the original gRPC implementation and the HTTP/TCP hybrid relatively effortless because the networking functionality was already designed to be decoupled. This meant that we were able to reimplement the same interfaces we had used to abstract the networking functionality and the code within our displays did not need to change at all. This allowed us to replace our networking layer in only two weeks.

3.5 Transitioning from the Virtual World to the Real World

For the visualizations of the augmented reality application to properly overlay with the real swarm and objects we utilize the Magic Leap One's built-in image tracking combined with two transformations about the headset's current position. When these transformations are applied to the robot and object positions, provided by ARGoS and the fiducial tracking system Vicon, we are able to correctly offset the headset and controller within the unity application matching the world origin and scale of the Vicon system [38].

Using the mixed reality XR library, built for unity [39], the camera position is directly updated as a result of the magic leap's provided head pose. The axis alignment was conducted by placing the application's main camera as a child of an origin object which is used for

offsetting the camera and controller's position. Utilizing the Magic Leap's built-in image tracking, an image can be placed in the same location and orientation matching the Vicon system's world frame. We are then able to update the parent origin's position and rotation through the following set of equations:

Equation 1,2

$$\begin{bmatrix} originXPos \\ originYPos \\ originZPos \end{bmatrix} = \begin{bmatrix} -imageXPos * cos(imageYRot) + imageZPos * sin(imageYRot) \\ -imageYPos \\ -imageZPos * cose(imageYRot - imageXPos * sin(imageYRot) \end{bmatrix}$$

$$\begin{bmatrix} originXRot \\ originYRot \\ originZRot \end{bmatrix} = \begin{bmatrix} 0 \\ -imageYRot \\ 0 \end{bmatrix}$$

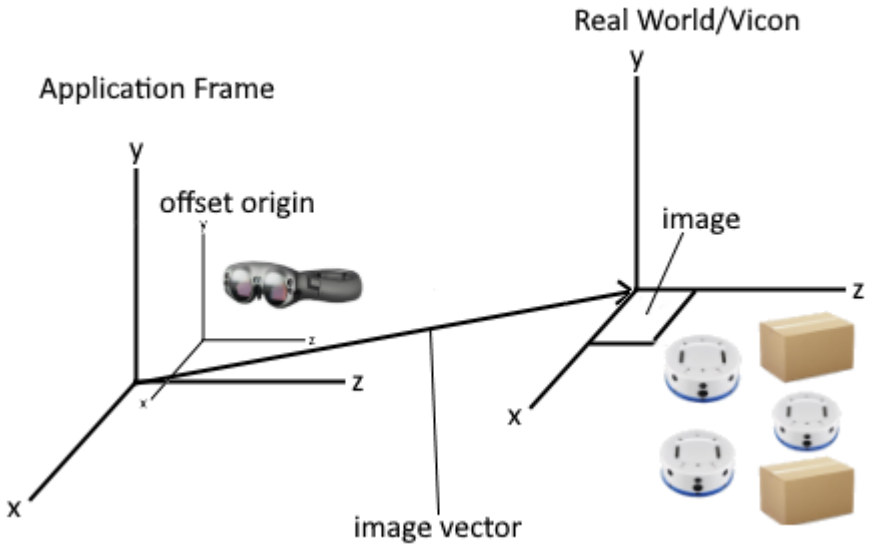

Fig 7. Axis Alignment between Application frame and Vicon frame

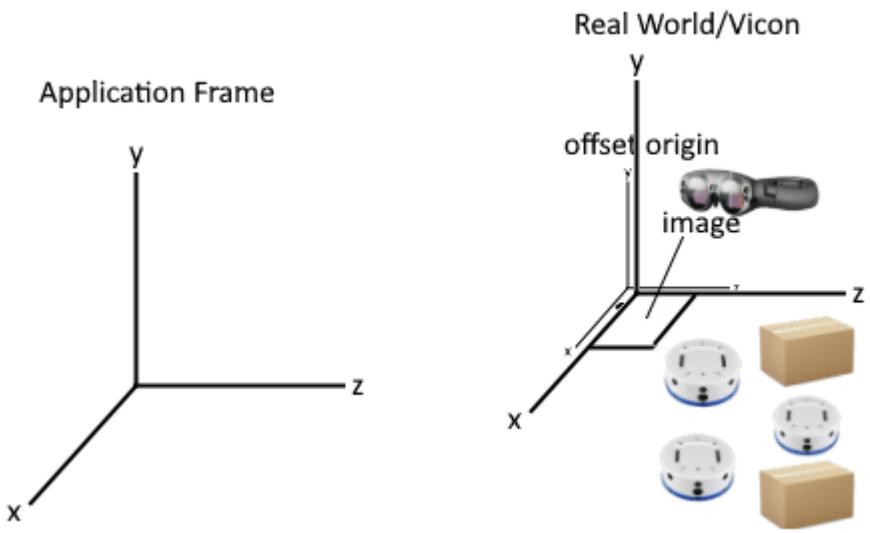

Fig 8. Axis Alignment offset origin frame being set based off previous image vector

For the offset origin's position, we are updating its position to be the subtracted vector from the origin image's position with respect to the origin image's rotation around the *y* axis in the application frame. Fortunately for the rotation, it is only necessary to set the offset origin's *y* rotation as we are only working on a two-dimensional field and the Magic Leap initializes to proper *x* and *y* rotation at the start of the application.

3.6 Legacy Architecture

A vital section of this project is managing the virtual objects within the Magic Leap client application. The framework was built on Libby *et. al.*'s work [26], where a comprehensive review can be found. A brief outline of this legacy architecture is included in Figure 9.

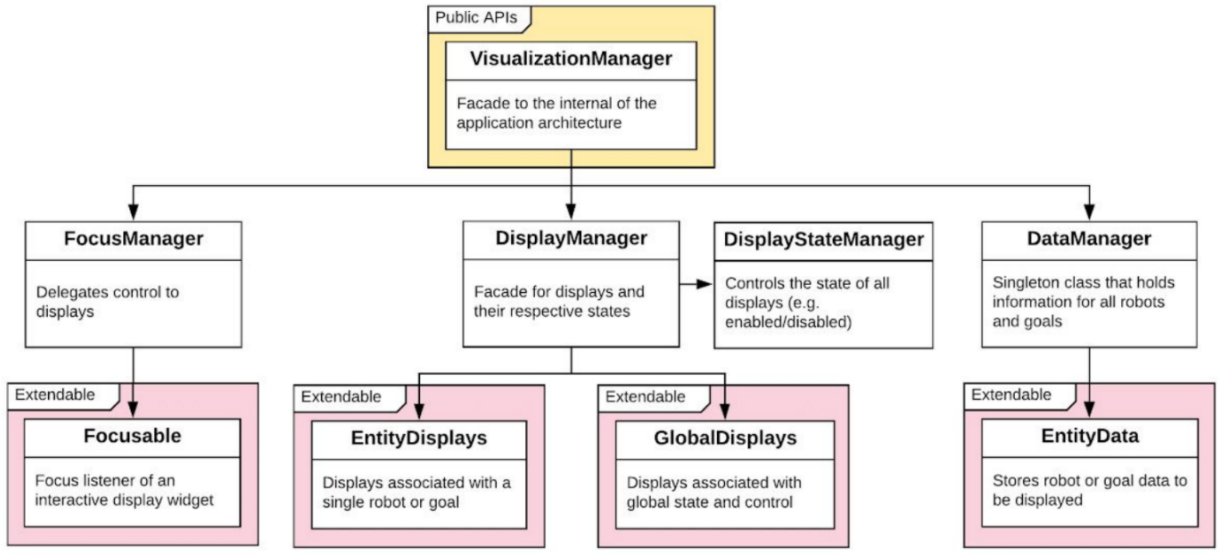

Fig 9. Legacy Architecture System [28]

3.6.1 Data

Robot position and state are stored in and accessed with the `DataManager`. The `DataManager` is coupled with the networking layer of the system. Real life changes pass to the client through this channel. The `DataManager` takes in this data and applies it to the proper entities as necessary. Packets from the connection to ARGoS are parsed and operations are applied to whatever entities the packet relates to.

3.6.2 Displays

Displays can be one of two types: `GlobalDisplays` or `EntityDisplays`. `GlobalDisplays` contain information like the Magic Leap IP address, objectives, and a global event log. `EntityDisplays` work on a per-object basis. `EntityDisplays` are one of the following:

- `LogDisplays`, which contain a log of an individual robot actions and messages;
- `PathDisplays`, which show the vectors along which the robot is attempting to navigate;
- `SonarDisplays`, which show the sonar proximity sensor information of individual robots; and
- `WheelVelocityDisplays`, which show the velocities of the wheels of robots.

Each Display is created on startup and stored in the client, but all of the entity displays are set to hidden. The functionality to hide and view different displays is kept track of in the

`DisplayStateManger` class. It handles the logic of when to draw Displays or not depending on what the user has enabled.

The `DisplayManager` holds references to all of the Global and Entity Displays and calls each Display's update function each frame that the magic leap renders content. Each Display will check its state and sometimes the state of the controller and update the rest of the system accordingly.

3.6.3 Focus

The `FocusManager` keeps track of what the user has selected. It is important for updating display states and for making sure commands are sent to the proper robots. Importantly, it defines an interface for the other displays to implement: `OnFocusStart`, `OnFocusContinue`, and `OnFocusEnd`. Each display contains an implementation of these, with different control functionality depending on the display. Our team has deprecated the `FocusManager` from future development, although some objects in the system still rely on it.

3.7 New Architecture: Event Based Programming

Our main idea in architectural changes was to change to an event-driven approach to system input. In our event-based system, the Controllers drive the changes rather than Displays checking on controller state every frame. The main reason behind this change was to decouple control code from the Displays themselves to enable different control inputs in the future. A wrapper (`Controller`) on the Magic Leap controller libraries capture when a controller event occurs notifying the `ControlScheme` of this event. The `ControlScheme` will then notify all the event handlers registered to it through the `EventRegistrar` allowing displays to be decoupled from control schemes. Figure 10 outlines the event registration and invocation process.

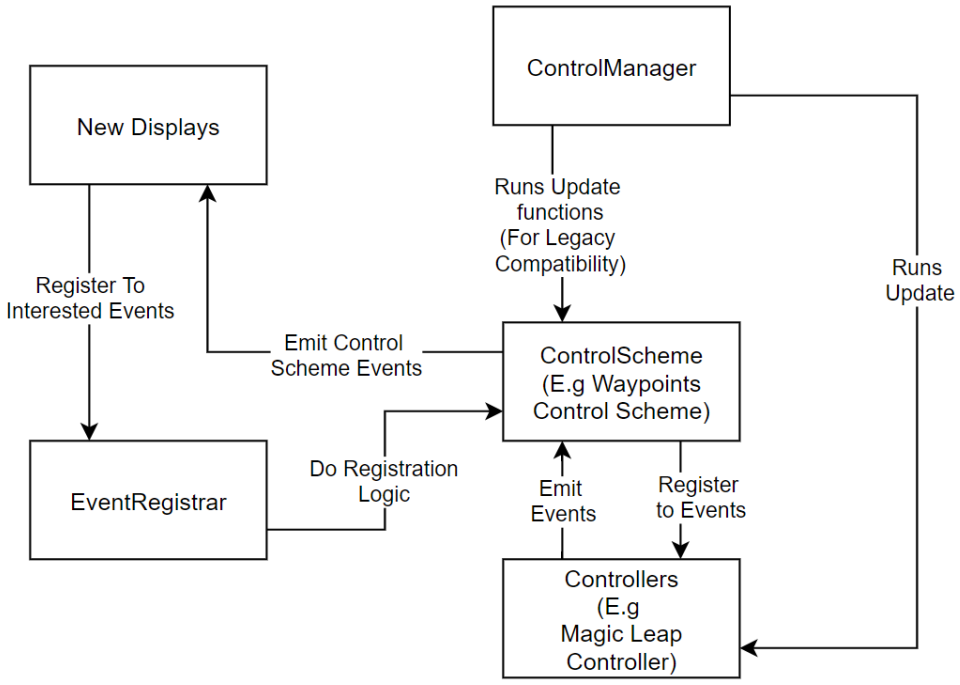

Fig 10. Event Driven Architecture

In this process, first the `Controller` and `ControlManager` are created. Next, the `ControlSchemes` are created and register their method implementations for certain controller events directly with the `Controller` class. Then, the event registrar is created with `ControlSchemes` passed in. Finally, new displays are created and register their method implementations with different control schemes through the `EventRegistrar`. The new displays never deal directly with controller input, and they only have implementations that can be invoked by `ControlSchemes`. In this way, different control schemes can be hot swapped during build or even runtime. For example, when we wanted to add a combined controller-voice control method, we would simply make sure that that `ControlScheme` emits all required events and set that as the `ControlScheme` in the `ControlManager`.

3.7.1 Attaching Networking Code to the Event Driven Process

In a single user scenario, events simply cascade from the controller down to the selected display at the time of the event. Similarly, when a user sends a message to SwarmServ, we can react to these messages within the receiver's displays through an event-based system. This allows our displays to avoid polling for new external changes on every frame update.

To achieve this, we define a repo interface for each networking functionality that Displays can keep references to and call send commands when necessary. These interfaces

also allow the Displays to register event handlers for when new information comes from an external source. We then defined a concrete implementation called `SwarmServClient` that calls the event handlers when necessary. We decided on using the same class for the concrete implementation of all our networking repo interfaces as we wanted to reduce the number of concurrently open channels to SwarmServ.

Ideally, all of the legacy code base should be transferred to this new system to allow for more uniform development processes. Another benefit from updating the system is extended controller functionality becoming available across the entire application. In this new architecture, the `DataManager` may be entirely the same, but each Display implementation and `FocusManager` would require expansive changes.

3.8 Non-Verbal Communication Features

At the time of writing, there is no known research that has studied multi-user collaboration for robotic control using an augmented reality head mounted display. Therefore, we chose to experiment with new features to gather information for this area of research. We designed three main features to aid in collaboration: *Highlights*, *Markers*, and *Locking*. For use in this section, we selected two terms: *Activator* and *Observer*. While in reality, both users are Activators and Observers, it makes sense to look at an individual interaction to separate the two users and what occurs when one of these features is activated.

Highlights were a tool we implemented to aid in non-verbal communication between two users. We imagined a set of possible use cases that we planned to look for if an in-person experiment would have been possible. One case we expected was the Activator points at a robot while conversing with the Observer, saying “Something is wrong with this robot” or “Look at this robot”. Pointing with the controller or a hand can be ambiguous when robots are close to each other. Another advantage of the highlights is that there will be a visual change on the Observer’s headset, immediately drawing their attention to it. Highlights contrasts using a robot’s number or unique identifier to call attention to it, because the Observer would then have to initiate a search rather than just observing a change to the current view. This would also scale well because using an identifier could lead to long searches if there are many ids in the system to check. One potential drawback to evaluate is that the user must be looking at the robot to see the highlight show up, and the field of view of many AR headsets can be small.

Markers are similar in many ways to the Highlights feature. Markers are meant to augment parts of the system, but they focus on marking spaces where there is no

corresponding physical object in the system. The originally imagined use case for markers is the Activator indicating a goal for the Observer. This is useful to analyze how far along a robot's operation is from reaching their goal, or for planning purposes so that the Observer does not also plan a path in between a set of robots and their goal. The implementation of markers we use is more flexible than just a ‘moving here marker’ and is not tied to the movement functionality. We chose to do this because we wanted to see for what other uses people ended up using the marker functionality.

The Locking feature is meant to prevent two users’ workflow from interfering with one another. In the current system, users can interrupt a robot’s move command by assigning another move command while the first command has not yet completed. When an Activator initiates a lock, the Locking feature will not allow the Observer to use that resource until the Activator unlocks it. The Observer can still view the locked object and the different displays it has associated with it but cannot perform a move command. The server deals with concurrency issues so that only one user can lock a resource at one time. This is a useful feature because one user may have access to knowledge that indicates the robot is better suited for its current course of action rather than changing tasks. Again, this feature and others would have been tested in a laboratory environment if possible.

3.8.1 Non-Verbal Communication Feature Implementation

Highlights

The system will highlight robots whenever one is considered selected. This can be done by emitting an `OnHoverStart` event with the *Entity ID* of the Entity to be highlighted. All entities in the system are notified but only take actions if the *Entity ID* matched its own. Figure 11 shows the sequence diagram for this method call.

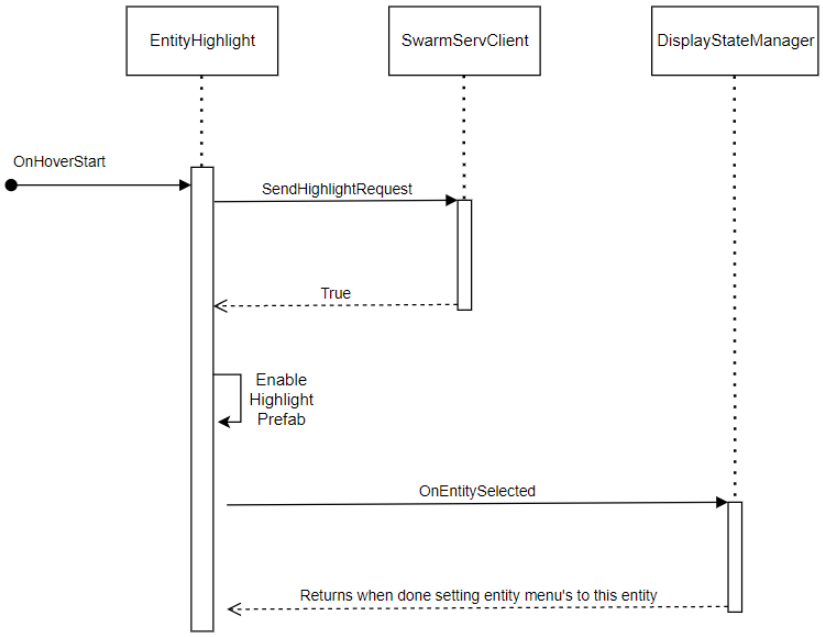

Fig 11. Entity Highlight Display Sequence Diagram

One potential improvement is the ability to selectively turn on and off this feature, as sometimes the two users did not need to use this feature and the highlights could be distracting. One natural intricacy of highlights is when both users are highlighting a robot. For example, what happens when one user is highlighting a robot, then another user tries to highlight it? We chose a simple approach: local highlights are prioritized over remote highlights. The second user to highlight the robot will have it shown red before the robot is selected, then blue after they select the robot. The first user to deselect the robot will see it shown red for the other user's current highlight.

Markers

Shown in Figure 16 is a picture of an operator placing a marker on the ground. Markers can be used to attract the attention of another user to a specific location not inhabited by an entity such as a robot. By emitting a ping sound when they are placed, they also attract the operator's attention that a marker was placed. Furthermore, when a marker is no longer needed, the initial placer is able to delete it from all operators' views. The sequence diagram for Markers is shown below in Figure 12. (Note: ‘Marker’ is called ‘Waypoint’ throughout the code base)

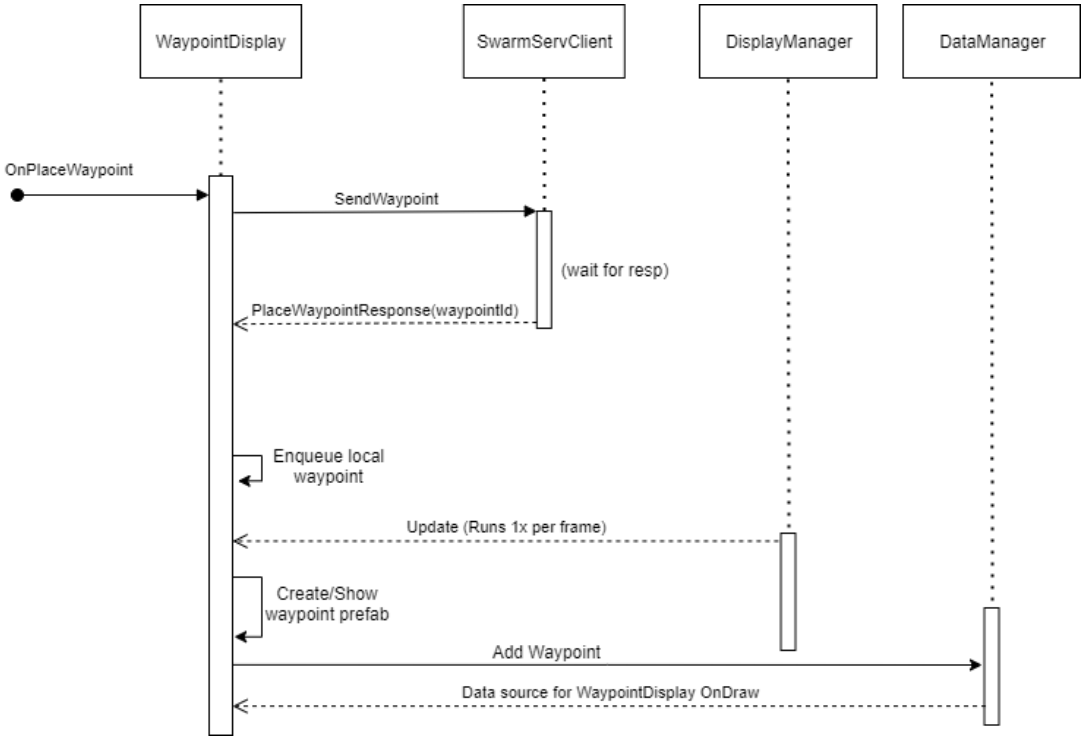

Fig 12. Waypoint (Marker) Sequence Diagram

Locks

Resource locking is used to prevent users from interrupting each other's workflow by only allowing the user who locked a specific resource to interact with it. Figure 13 shows the sequence diagram for this method call.

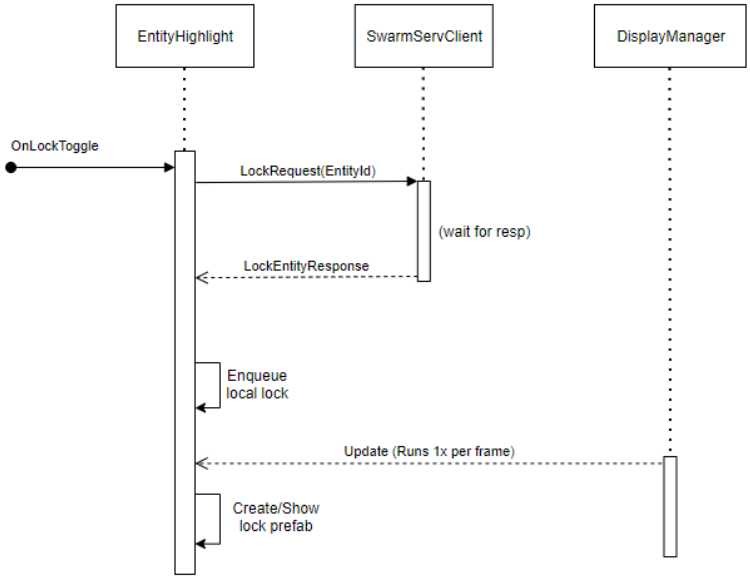

Fig 13. Entity Locking Sequence Diagram

This is done by having the user hover over a resource followed by the user emitting an `OnLockToggle` event. If the selected resource is not currently locked locally, a locking request is initiated to the SwarmServ service which if successful will broadcast to all users of the lock on the designated resource. For the user who placed the lock, a green lock icon will appear above the resource and the user will be able to continually interact with the resource. For all other users in the system, a red lock will appear above the resource and they will be prevented from interacting with that resource, specifically preventing move commands. This feature aims to prevent other users from taking resources away that are currently important to the users placing the locks.

3.9 Voice Commands Implementation

Another major feature we implemented quickly using the new modular event driven architecture was that of voice control. This feature allows the operators to use voice commands to perform the features listed above. Figure 14 outlines the flow of events for the SwarmSpeechServ voice command service.

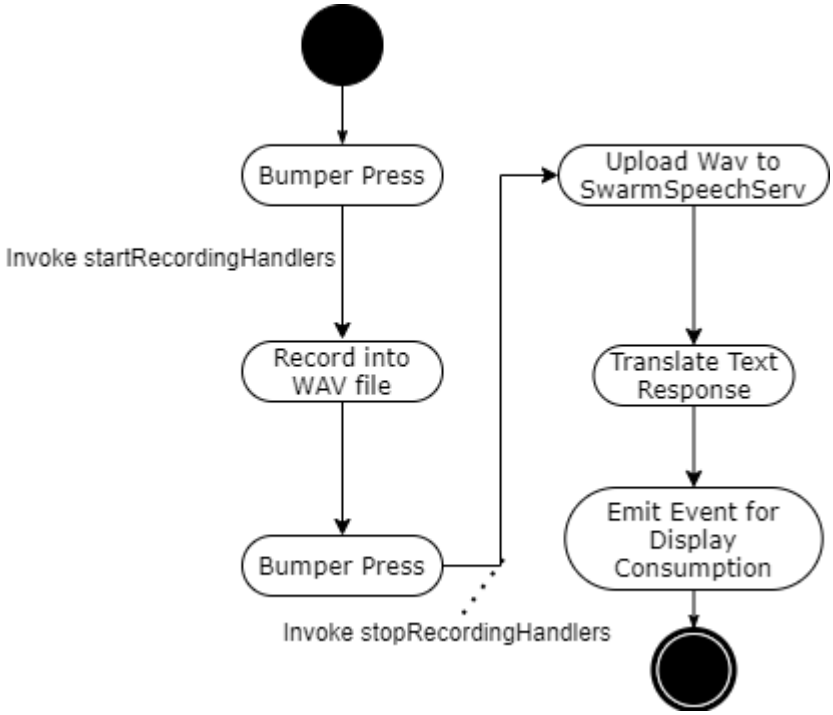

Fig 14. Voice Service Activity Diagram

The flow of this system begins with the user moving the cursor over an entity, followed by pressing the bumper to initiate a recording. The user then speaks the voice command and presses the bumper again to halt the recording and send the audio file to be interpreted by

SwarmSpeechServ service which decodes the audio and returns the text to the client. The returned text is then used to emit locking, unlocking, marker placement and deletion events. Voice commands aim to allow for a more intuitive and accessible style of control while simultaneously showing the modularity of the modularity of the system by allowing control schemes to be quickly hot-swapped as a result to the event-driven architecture.

3.10 Experimental Design for Future Use

The system is to be tested in 3 groups:

- Single User
- Multi-User with collaboration tools turned off
- Multi-User with collaboration tools turned on

In between Single User and both Multi-User tests, we planned to examine mental load, effort, and how users perceive information about the system. We planned to compare the Multi-User tests with collaboration tools off and on to see how it affects users' trust in the system and the other humans using the system. Furthermore, we planned to obtain qualitative data about how our interface worked to allow future work on developing a better AR system for controlling swarms. For additional information on our User study design, see Appendix A.

Chapter 4: Experimental

4.1 Feature Testing

4.1.1 Highlights

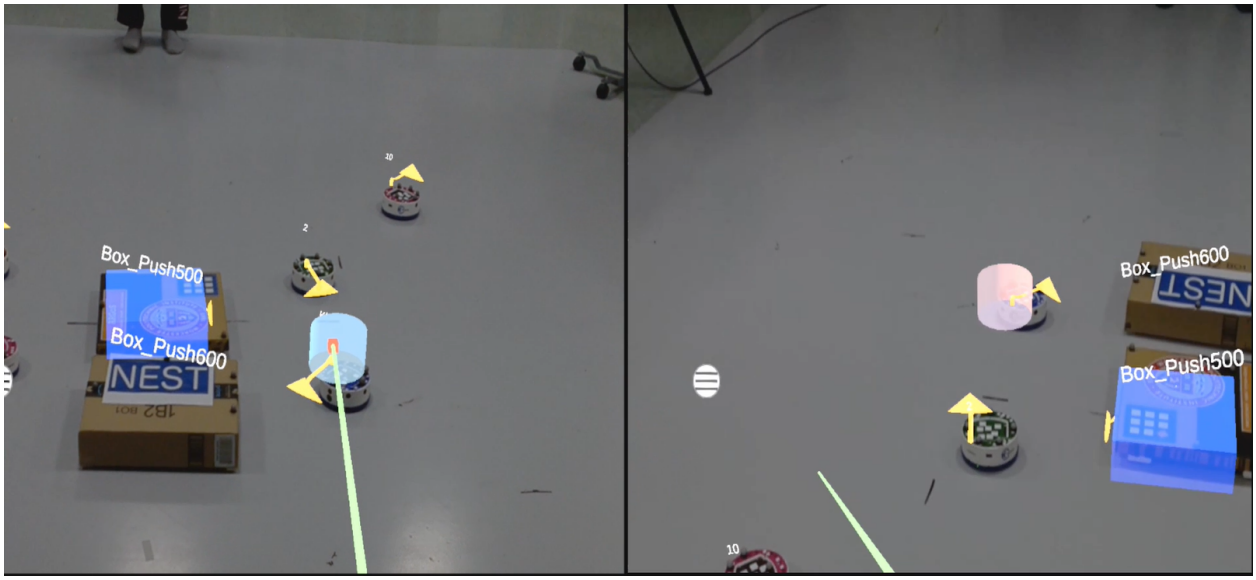

Fig 15. Highlight Display Between two users

One user (left) points the controller at a robot, highlighting it blue. Second user (right) sees this as a red highlight.

During our testing of the system, we did find Highlights (shown in Figure 15) to be a useful indicator for referring to different entities. Often, the names of the entities show up quite small on the AR display and the large color change did succeed in drawing attention to a robot. One potential improvement is the ability to selectively turn on and off this feature, as sometimes the two users did not need to use this feature and the highlights could be distracting.

We ran a series of 20 highlights tests (10 each between two users) and marked the results in Table 2 below. The “% highlights shown” column is how many times the highlight shows up (locally and remotely) when a user places their marker on the entity. The “% un-highlights” column shows the percentage of times the highlights actually disappeared after the highlighting user stopped highlighting the entity. The response time column measures the time from when the highlighting user moves their controller to the entity until the red highlight shows on the remote user's display. The response time was taken by recording a video on the Magic Leap, and measuring after by analyzing the video timestamps when the actions occurred in the video.

Table 2: Entity Highlight Feature Test Results

User #	% of highlights shown (local)	% of highlights shown (remote)	% of un-highlights (remote)	Response time (cursor move to highlight on remote user)
User 1	100	100	100*	μ=0.370 σ=0.119
User 2	100	100	100*	μ=0.464 σ=0.12

Our results showed a consistent standard deviation, while having a varying mean from 0.37-0.464 seconds. Given these times are almost all under 0.5 seconds, we consider these results satisfactory. The standard deviations are also smaller than average human reaction time. The * in the un-highlights column is a discrepancy discovered outside of testing. We found that sometimes highlights would become stuck on for a remote user if the cursor passed very quickly over a robot. We believe this issue has something to do with threading, but we did not have time to investigate. Since we were doing significant times on our highlights, the issue was not uncovered here.

4.1.2 Markers

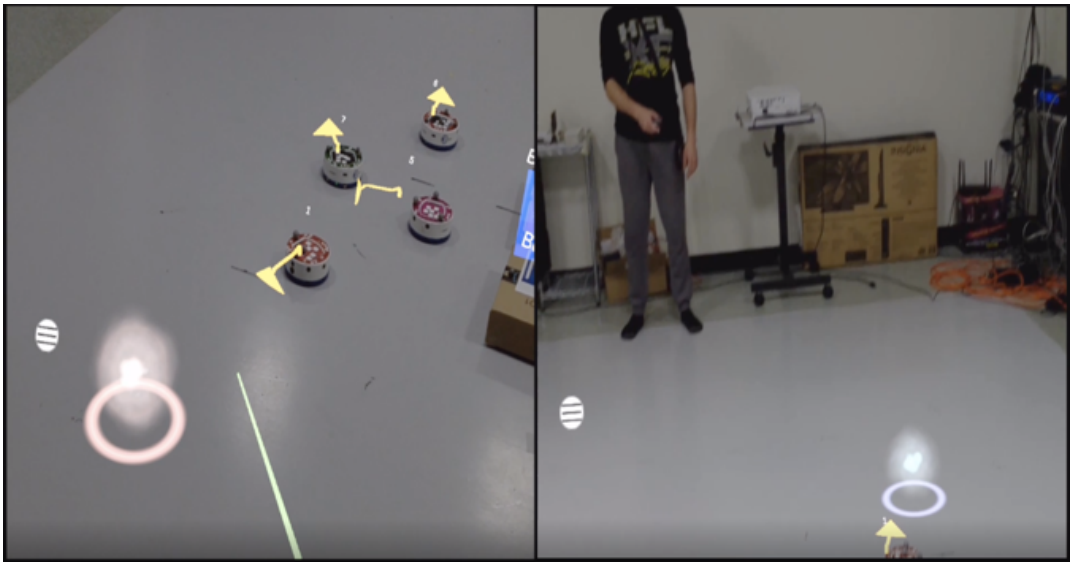

Fig 16. Marker Display between two users

One user (right) places a marker on the ground. Second user (left) sees this as a red marker.

Shown in Figure 16 is a picture of an operator placing a marker on the ground. Markers can be used to attract the attention of another user to a specific location not inhabited by an entity such as a robot. By emitting a ping sound when markers are placed, they also attract the operator's attention that a marker was placed. Furthermore, when a marker is no longer needed, the initial placer is able to delete it from all operators' views.

To validate the markers behavior, we ran 20 tests (10 for each user) and recorded our results in Table 3 below. The "% of markers placed (local)" column corresponds to the percentage of placed markers shown on the initiator's display. The "% of markers placed (remote)" column corresponds to the percentage of placed markers that became visible to the other operator. The "Place Response Time (ms)" column corresponds to the number of milliseconds it took for the other operator to see a placed marker. The “Delete Response Time (ms)” column corresponds to how many milliseconds it took before the other operator saw a marker be deleted from their field of view after the other operator initiated the command.

Table 3: Marker feature test results

User #	% markers placed (local)	% markers placed (remote)	Place Response Time (ms)	Delete Response Time (ms)
User 1	100	100	μ=0.821 σ=0.206	μ=0.451 σ=0.166
User 2	100	100	μ=0.655 σ=.173	μ=0.595 σ=.223

Our data shows that all the markers placed and deleted were successful. Furthermore, we can see that both deleting and placing markers takes less than one second. An important note is that placing a waypoint takes more time to show up than it does to delete one. One reason this might be happening is that the time it takes to render a new waypoint is larger than it is to remove it.

4.1.3 Locks

Fig 17. Locking display showing a locally placed lock (green) and remote lock (red)

Shown in Figure 17 is a picture capture of the locking feature. Resource locking is used as a means to prevent users from interrupting each other's workflow by only allowing the user who locked a specific resource to interact with it. This is done by having the user highlight a resource followed by the user emitting an `onLockToggle` event. If the selected resource is not currently locked locally, a locking request is initiated to the SwarmServ service which, if successful, will broadcast to all users of the lock on the designated resource. For the user who placed the lock, a green lock icon will appear above the resource and the user will be able to continually interact with the resource. For all other users in the system, a red lock will appear above the resource and they will be prevented from interacting with that resource, specifically preventing move commands. This feature aims to prevent other users from taking resources away that are currently important to the users placing the locks.

We conducted 10 locking focused tests and marked the results in Table 4 below. The “% locks appear/disappear (local)” and “% locks appear/disappear (remote)” columns report how many times the lock shows up locally and remotely, respectively, when a user places a lock on an entity. The Resp. Time lock columns measure the time from when the locking user presses the trigger on the controller, while hovering over the entity, until the lock shows on the local user’s display and remote user’s display correspondingly. This process is repeated for the unlocking columns as well. The response times were taken by recording a stopwatch alongside both local and remote users while placing and removing locks on an entity.

Additionally, following each lock we attempted to issue move commands from the user not holding the lock to ensure the command is successfully blocked, then moved the locked entity from the user holding the lock to ensure the resource could continue to be used. We found this feature to be successful.

Table 4: Locking unit testing results

User #	% locks appear /disappear (local)	% locks appear /disappear (remote)	Resp. Time lock/unlock
User 1	100*	100*	μ=0.408 σ=0.085

The results showed 100% locking placement and removal, locally and remotely, with a cumulative response time average of 0.408 seconds from initiating the locking event to the actual event being activated with a standard deviation of 0.085. It is important to note there were discrepancies when using the locking feature while recording through the Magic Leap. When using the recording feature for data collection, a phenomenon occurred where, when attempting to lock an entity, it would take an additional trigger press to activate the command. This was only present while recording, but it is worth noting along with potential further investigation.

4.1.4 Voice Commands

We conducted 10 voice command tests and marked the results in Table 5 below. The “% lock” column reports how many times the lock feature was triggered after issuing a lock voice command. The “% unlock” column shows the number of successful unlocking events triggered after an unlock voice command with “% place marker” and “% delete marker” displaying the number of successful actions upon receiving their respective voice commands. The “Response Time Cumulative Average” column measures the time it took after the command was finished, i.e., after the second bumper press was finished indicating an end of the command recording, to the time it took for the actual event to be emitted.

Table 5: Voice command test data showing voice command success rates and response times

User #	% lock	% unlock	% placeMarker	% deleteMarker	Response Time Cumulative Average
User 1	80	100	90	100	μ=2.306 σ=0.165

Our data shows that our voice control system currently struggles with the lock voice command along with the place marker voice command with their successful activation rates at 80% and 90% respectively with 100% effectiveness when issuing unlocking and delete marker commands. We also found after the second button press, the average response time for a command to be processed and executed came to 2.306 seconds with a standard deviation of 0.165. This variance in response time could potentially be a result of network traffic along with the variance in average activation times for performing each command separately.

Chapter 5: Conclusion

5.1 Summary

Our work developed the first multi-user, head-mounted, augmented reality swarm control system. Our system allows two users to collaboratively interact with the swarm, sharing control responsibilities over multiple human operators. We transitioned the previous swarm control application from simulation to the physical world. We carefully built a modular architecture that allows for future work to branch off or expand upon our work. We demonstrated this through the introduction of voice commands. In the future, the modular architecture will allow different human-swarm and human-human experiments to be developed quickly using our system.

5.2 Future Work

While our MQP group believes our teaming features will be a success, we recognize these features and ideas need to be validated in one or more user studies. Our proposed user study evaluates the features and ease of use on a headset, it would also be interesting to compare these results to a tablet study. Do the headsets provide the immersion or better results? How do users become more or less efficient due to what platform they are on? The number of possible experiments is very high.

One other feature we did not fully explore was extending voice commands to allow for users to communicate with the system more naturally. We believe that allowing variation in the voice commands will allow for a more affordant control system which reduces the learning curve to use this system. Furthermore, being able to selectively filter shown displays through a voice command would allow users to traverse through less menus.

5.3 Lessons Learned

The Magic Leap platform is not mature enough to be used as a commercial product. Magic Leap's libraries are severely undocumented and forces users to go through their forums [40] in order to solve issues. This was especially important for when we were trying to include the gRPC binaries within the build system.

Having to migrate from gRPC to the HTTP, TCP hybrid solution taught us the importance of designing a decoupled architecture from the beginning of the project. It also taught us about the importance of testing on the actual device before putting in a lot of

development time. Being able to modularly swap the classes which did the logic for this allowed us to pivot quickly and minimize the amount of code we rewrote. Furthermore, we noticed this architecture's benefits through how quickly we were able to add new features.

We also learned about some of the quirks of Unity that are well documented but do not present themselves in a visible way. One issue that continually popped up (in part due to the event-based architecture) was that of creating and destroying GameObjects. Unity only allows GameObjects to be created and destroyed in the main thread, and it will fail this operation silently unless wrapped in a try/catch block. This behavior occurs when splitting a thread to wait for a server's response to a request (as in Locking, when we wait to see if the resource is available to lock before moving forward). Our solution to this problem (used in Waypoints and Locks features) is adding objects to thread safe queues that update once per frame. One other smaller unity quirk to note is the use of .meta files. To summarize what is important about these, they mark resource IDs for different items in the project structure.

5.4 Covid Considerations

One major issue we faced while working through this project was a lack of lab access, as well as a lack of access to the physical Magic Leap One device due to the COVID-19 pandemic. For the entirety of the first semester, the NESTLab, our projects research space, was completely closed. This forced us to work solely in the simulation space through the use of Magic Leap's *The Lab* application. We were extremely grateful for this simulation tool, however, as it still allowed us to continue development and perform rudimentary tests during the first semester.

Another major drawback to the COVID-19 pandemic was the prevention of conducting a user study. As subjects would have had to come into an enclosed area for long periods at a time and place headsets on themselves, we were unable to successfully obtain IRB approval for our user studies. In the near future, as the crisis of the pandemic recedes, we hope for a proper user study to be conducted by a future Major Qualifying Project team to study the effectiveness of head mounted, augmented reality, multi-user swarm control.

Citations/Bibliography

[1]: Hamann, Heiko. *Swarm Robotics: A Formal Approach*. Springer International Publishing, 2018.
[2]: W. Truszkowski, M. Hinchey, J. Rash and C. Rouff, "NASA's swarm missions: The challenge of building autonomous software", *IT Professional*, vol. 6, no. 5, pp. 47-52, 2004.
[3]: A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, “Human interaction with robot swarms: A survey,” IEEE Transactions on Human-Machine Systems, vol. 46, no. 1, pp. 9–26, 2015.
[4]: 4 Kolling, Andreas, et al. "Human-swarm interaction: An experimental study of two types of interaction with foraging swarms." Journal of Human-Robot Interaction 2.2 (2013): 103-129.
[5]: G. A. Miller, “The magical number seven, plus or minus two: Some limits on our capacity for processing information.” Psychological review, vol. 63, no. 2, p. 81, 1956.
[6]: M. Lewis, H. Wang, S. Y. Chien, P. Velagapudi, P. Scerri, and K. Sycara, “Choosing autonomy modes for multirobot search,” Human Factors, vol. 52, no. 2, pp. 225–233, 2010.
[7]: Patel, J., & Pinciroli, C. (2019). Improving Human Performance Using Mixed Granularity of Control in Multi-Human Multi-Robot Interaction. arXiv preprint arXiv:1909.07487.
[8] Billinghurst, M., Weghorst, S. & Furness, T. Shared space: An augmented reality approach for computer supported collaborative work. *Virtual Reality* 3, 25–36 (1998). https://doi.org/10.1007/BF01409795
[9]: “Magic Leap: A Thousand Breakthroughs In One.” Magic Leap, https://www.magicleap.com/en-us/magic-leap-1
[10]: Maximilian Speicher, Brian D. Hall, and Michael Nebeling. 2019. What is Mixed Reality?. In CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland, UK. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3290605.3300767
[11]: Gibson, J.J. The Ecological Approach to Visual Perception. New York, NY: Psychology Press, 1986.
[12]: Steffen, J. H., Gaskin, J. E., Meservy, T. O., Jenkins, J. L., & Wolman, I. (2019). Framework of affordances for virtual reality and augmented reality. Journal of Management Information Systems, 36(3), 683-729
[13]: F. Ghiringhelli, J. Guzzi, G. A. Di Caro, V. Caglioti, L. M. Gambardella and A. Giusti, "Interactive Augmented Reality for understanding and analyzing multi-robot systems," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, 2014, pp. 1195-1201, doi: 10.1109/IROS.2014.6942709.
[14]: J. A. Frank, S. P. Krishnamoorthy and V. Kapila, "Toward Mobile Mixed-Reality Interaction With Multi-Robot Systems," in IEEE Robotics and Automation Letters, vol. 2, no. 4, pp. 1901-1908, Oct. 2017, doi: 10.1109/LRA.2017.2714128.
[15]: Reardon C., Lee K., Rogers J.G., Fink J. (2019) Augmented Reality for Human-Robot Teaming in Field Environments. In: Chen J., Fragomeni G. (eds) Virtual, Augmented and Mixed Reality. Applications and Case Studies. HCII 2019. Lecture Notes in Computer Science, vol 11575. Springer, Cham. https://doi.org/10.1007/978-3-030-21565-1_6
[16]: RT Azuma - Presence: Teleoperators and Virtual Environments, 2016 - MIT Press

[17]: M. R. Endsley, “From here to autonomy: lessons learned from human– automation research,” Human factors, vol. 59, no. 1, pp. 5–27, 2017.
[18]: Patel, J., Xu, Y., & Pinciroli, C. (2019, May). Mixed-granularity human-swarm interaction. In 2019 International Conference on Robotics and Automation (ICRA) (pp. 1059-1065). IEEE.
[19]: J. C. G. Higuera, A. Xu, F. Shkurti, and G. Dudek, “Socially-Driven Collective Path Planning for Robot Missions,” in 2012 Ninth Conference on Computer and Robot Vision. Toronto, Ontario, Canada: IEEE, May 2012, pp. 417–424. [Online]. Available: http://ieeexplore.ieee.org/document/6233171/
[20]: K. S. Ong, Y. H. Hsu, and L. C. Fu, “Sensor fusion based human detection and tracking system for human-robot interaction,” in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Vilamoura-Algarve, Portugal: IEEE, Oct. 2012, pp. 4835–4840. [Online]. Available: http://ieeexplore.ieee.org/document/6386222/
[21]: A. Freedy, O. Sert, E. Freedy, J. McDonough, G. Weltman, M. Tambe, T. Gupta, W. Grayson, and P. Cabrera, “Multiagent Adjustable Autonomy Framework (MAAF) for multi-robot, multi-human teams,” in 2008 International Symposium on Collaborative Technologies and Systems. Irvine, CA, USA: IEEE, May 2008, pp. 498–505. [Online]. Available: http://ieeexplore.ieee.org/document/4543970/
[22]: E. Jones, B. Browning, M. Dias, B. Argall, M. Veloso, and A. Stentz, “Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks,” in Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006. Orlando, FL, USA: IEEE, 2006, pp. 570–575. [Online]. Available: http://ieeexplore.ieee.org/document/1641771/
[23]: A. A. Malik and A. Bilberg, “Developing a reference model for human-robot interaction,” International Journal on Interactive Design and Manufacturing (IJIDeM), Jun. 2019. [Online]. Available: http://link.springer.com/10.1007/s12008-019-00591-6
[24]: M. Lewis and K. Sycara, “Effects of automation on situation awareness in controlling robot teams,” p. 7.
[25]: M. Lewis, H. Wang, Shih-Yi Chien, P. Scerri, P. Velagapudi, K. Sycara, and B. Kane, “Teams organization and performance in multi-human/multi-robot teams,” in 2010 IEEE International Conference on Systems, Man and Cybernetics. IEEE, pp. 1617–1623. [Online]. Available: http://ieeexplore.ieee.org/document/5642379/
[26]: Arthur E., Gardias P., Sullivan D. (2020) Augmented Reality as a Means of Improving Efficiency and Immersion of Human-Swarm Interaction
[27]: Libby, K. J., Zhong, L., & Van Stralen, N. M. (2020, May 14). Diagnosing Robotic Swarms 2 (Dr. Swarm2). Retrieved December 13, 2020.
[28]: Ke Huo, Yuanzhi Cao, Sang Ho Yoon, Zhuangying Xu, Guiming Chen, and Karthik Ramani. 2018. Scenariot: Spatially Mapping Smart Things Within Augmented Reality Scenes. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). Association for Computing Machinery, New York, NY, USA, Paper 219, 1–13. DOI:https://doi.org/10.1145/3173574.3173793
[29]: Leavitt, A., Keegan, B. C., & Clark, J. (2016). Ping to Win? Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems.
[30] Magic Leap Human Interface Guide. (2018, September). Retrieved December 13, 2019, from https://medium.com/@davecancode/magic-leap-human-interface-guide-part-two-d3cbe6d0a853

[31] https://unity.com/
[32] Pinciroli, C., Trianni, V., O'Grady, R. et al. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intell 6, 271–295 (2012). https://doi.org/10.1007/s11721-012-0072-5
[33] "CMUSphinx Open Source Speech Recognition." CMUSphinx Open Source Speech Recognition. Web. 04 May 2021.
[34]: "Introduction to GRPC." *GRPC*, grpc.io/docs/what-is-grpc/introduction/.
[35] "GRPC Server Client Relationship." *GRPC.io*, grpc.io/img/landing-2.svg.
[forums] https://forum.magicleap.com
[36]: "Core Concepts, Architecture and Lifecycle." *GRPC*, grpc.io/docs/what-is-grpc/core-concepts/.
[37] https://swagger.io/
[38] https://www.vicon.com/
[39] https://docs.unity3d.com/Manual/XR.html
[40] https://developer.magicleap.com/en-us/home

Appendix A: Survey Questions

User ID

Task ID

Complexity of Situation: How complicated was the experiment? Was it complex with many interrelated components (7) or is it simple and straightforward (1)?

Alertness: How alert were you throughout the experiment? Were you alert and ready for activity (7) or did you have a low degree of alertness (1)?

Division of Attention: How much was your attention divided throughout the experiment? Were you concentrating on many aspects of the situation (7) or focussed on only one (1)?

Overwhelming Factor: How overwhelmed did you feel throughout the experiment? Were your tasks completely manageable (7) or did you feel overwhelmed (1)?

Information Quantity: How much information have you gained about the situation? Have you received and understood a great deal of knowledge (7) or very little (1)?

Information Quality: How useful information have you gained about the situation? have you received and understood useful (high quality) knowledge (7) or useless (low quality) knowledge(1)?

Familiarity with AR: How familiar were you with Augmented Reality before the experiment? Did you have a great deal of relevant experience (7) or was it a new situation (1)?

Familiarity with Robotic Swarms: How familiar were you with controlling robotic swarms? Did you have a great deal of relevant experience (7) or was it a new situation (1)?

Mental Load: How mentally challenging/demanding was the task? How much did you have to think?
less challenging/demanding - 1
very challenging/demanding - 7

Physical Load: How physically challenging/demanding was the task? How physically tired because of the task are you?
less challenging/demanding - 1
very challenging/demanding - 7

Temporal Load: How fast did you have to think to complete the task? How fast did you have to respond to control the system and perform the task?
very slow - 1
very fast - 7

Performance: How successful were you in accomplishing what you were asked to do?
not successful - 1
very successful - 7

Effort: How hard did you have to work to accomplish your level of performance?
not at all - 1
very much - 7

Frustration: How frustrated, discouraged, irritated, stressed, and annoyed were you?
not at all - 1
very much - 7

Competence: To what extent does the system perform a given task effectively?
not at all - 1
very effectively - 7

Predictability: To what extent can you anticipate the system's behavior with some degree of confidence?
can't anticipate at all - 1
can anticipate perfectly - 7

Reliability: To what extent is the system free of errors?

full of errors - 1
completely free of errors - 7

Overall Trust: To what extent do you trust the system overall?
don't trust at all - 1
completely trust - 7

Accuracy: How accurate/correct the responses of the system were?
not at all accurate - 1
very accurate - 7

Human Trust Questionnaire - Only run on Experiments with Teams

My teammate is trustworthy:
7 - strongly agree
1 - strongly disagree

If I have a problem my teammate is always able to help me:
7 - strongly agree
1 - strongly disagree

My teammate is always willing to help:
7 - strongly agree
1 - strongly disagree

I felt accepted as a member of the team:
7 - strongly agree
1 - strongly disagree

Teammate's Intention: Did you understand your teammate’s intentions? Were you able to understand why your teammate was taking certain action?

Didn't understand at all	Understood less often	Moderately understood	Understood more often	Fully understood
○	○	○	○	○

For the above question, why did you select that answer? Write a short explanation

Teammate's Actions: Could you understand your teammate’s actions? Could you understand what your teammate was doing at any particular time?

Didn't understand at all	Understood less often	Moderately understood	Understood more often	Fully understood
○	○	○	○	○

For the above question, why did you select that answer? Write a short explanation

Task Progress: Could you follow the progress of the task? While performing the tasks, were you able to gauge how much of it was pending?

Not at all	Mostly No	Half the time	Mostly Yes	Always Yes
○	○	○	○	○

For the above question, why did you select that answer? Write a short explanation

Robot Status: Did you understand what the robots were doing? At all times were you sure how and why the robots were behaving the way they did?

Not at all	Mostly No	Half the time	Mostly Yes	Always Yes
○	○	○	○	○

For the above question, why did you select that answer? Write a short explanation

Information Clarity: Was the information provided by the interface clear for accomplishing the task?

No clear at all	Less often clear	Moderately clear	More often clear	Very clear
○	○	○	○	○

For the above question, why did you select that answer? Write a short explanation

Information Necessity: Was the information provided by the interface necessary in accomplishing the task?

Not at all necessary	Less often necessary	Moderately necessary	More often necessary	Very necessary
○	○	○	○	○

For the above question, why did you select that answer? Write a short explanation

Appendix B: Use Cases

Issue Collective Transport Command
Participating Actor - Magic Leap User

Entry Condition:
- Object is not currently being moved/rotated
Exit Condition:
- Nearby robots will collectively work to transport the selected object
Flow of Events:
1. User selects object to be transported
2. System displays this object is selected
3. User Indicates final position for object to be translated/rotated
4. System displays final position

Issue Move Group Command
Participating Actor - Magic Leap User
Entry Condition:
- Group is not currently executing move group
Exit Condition:
- Group will begin moving to specified destination
Flow of Events:
1. User selects robots to be grouped
2. System displays selected robots
3. User indicates final position for robots to move to
4. System displays final position

Issue Move Single Robot Command
Participating Actor - Magic Leap User
Entry Condition:
- Robot is not currently being moved
Exit Condition:
- Robot will be moving to designated destination
Flow of Events:
1. User Selects Robot to move
2. System displays selected Robot
3. User Indicates final position for Robot to move to
4. System displays Robot's final position

Issue Assign to Sub-group command
Participating Actor - Magic Leap User
Entry Condition:
- Robot is not currently in the subgroup
- Robot is not currently in any other subgroup
Exit Condition:
- Robot will be part of the selected subgroup
Flow of Events:
1. Magic Leap User selects robot they want to assign to a subgroup
2. System displays selected robot

3. Magic Leap User indicates subgroup robot should be assigned to
4. System displays updated group

[Robot Oriented]Switching
Participating Actor - Magic Leap User
Entry Condition:
- Magic Leap User is not currently using Robot-Oriented Control
Exit Condition:
- Magic Leap User will be using Robot-Oriented Control
Flow of Events:
1. Magic Leap User Indicates that they want to use Robot-Oriented Control
2. System indicates that it is now in Robot-Oriented Control

[Swarm Oriented]Switching
Participating Actor - Magic Leap User
Entry Condition:
- Magic Leap User is not currently using Swarm-Oriented Control
Exit Condition:
- Magic Leap User will be using Swarm-Oriented Control
Flow of Events:
1. Magic Leap User Indicates that they want to use Swarm-Oriented Control
2. System indicates that it is now in Swarm-Oriented Control

[Robot Oriented] Battery Level
Participating Actor: Magic Leap user
Entry Condition:
- In robot oriented control mode
Exit Condition:
- Magic Leap User leaves robot oriented control mode or hide battery / status selected OR Magic leap user deselects robot
Flow of Events:
1. Magic leap user selects a robot
2. Battery level is shown

[Robot Oriented] Robot final position
Participating Actor: Magic Leap user
Entry Condition:
- Interface in robot oriented control mode
Exit Condition:
- Robot moves to designated position
Flow of Events:
1. Magic leap user selects a robot
2. Interface displays selected Robot
3. Magic Leap User indicates final position for robot

4. Interface displays final position

[Robot Oriented] Assigned Subgroup
Participating Actor: Magic Leap user
Entry Condition:
- Interface robot oriented control mode
Exit Condition:
- Leave robot oriented control mode
- Hide Assigned Subgroup / status selected
- Magic leap user deselects robot
Flow of Events:
1. Magic leap user selects a robot
2. Interface displays assigned subgroup

[Robot Oriented] Selected robot should be highlighted
Participating Actor: Magic Leap user
Entry Condition:
- Interface in robot oriented control mode
Exit Condition:
- Magic Leap user deselects robot
Flow of Events:
1. Magic Leap user selects robot
2. Robot model indicated to be selected

[Robot Oriented] Selected robot direction of travel
Participating Actor: Magic Leap user
Entry Condition:
- Interface in robot oriented control mode
Exit Condition:
- Leave robot oriented control mode
- Hide direction / status selected
- Magic leap user deselects robot
Flow of Events:
1. Magic leap user selects a robot
2. Robots heading is shown

[Swarm Oriented] View number of robots in selected group
Participating Actor: Magic Leap user
Entry Condition:
- Interface in swarm oriented control mode
Exit Condition:
- Leave swarm oriented control mode
- Hide Number of Robots / status selected
Flow of Events:
1. Magic leap user selects a sub group

2. The number of robots in the selected sub group is shown

[Swarm Oriented] Indicate robots part of Subgroup
Participating Actor: Magic Leap user
Entry Condition:
- Interface in swarm oriented control mode
Exit Condition:
- Leave swarm oriented control mode
- Magic Leap user deselects subgroup
- Hide highlight subgroup / status selected
Flow of Events:
1. Magic Leap user selects subgroup
2. All robots in the assigned subgroup are indicated to be selected

[Swarm Oriented] Visualize Object Final Position
Participating Actor: Magic Leap User
Entry Condition:
- Interface in swarm oriented control
- Selected subgroup has a collective transport command running
Exit Condition:
- Object has arrived in final position
Flow of Events:
1. Magic Leap user selects a subgroup
2. Interface displays final position of object being transported

[Swarm Oriented] Visualize Sub-group Final Position
Participating Actor: Magic Leap User
Entry Condition:
- Interface in swarm oriented control
- Selected subgroup has a move command running
Exit Condition:
- Subgroup has arrived in final position
Flow of Events:
1. Magic Leap user selects a subgroup
2. Interface displays final position of subgroup

[Swarm Oriented] Visualize dead/unhealthy robots
Participating Actor: Magic Leap user
Entry Condition:
- Interface in swarm oriented control mode
Exit Condition:
- Leave swarm oriented control mode
- Robot resumes communication
Flow of Events:

1. Robot loses communication with ARGoS system
2. Five seconds passes
3. Robot becomes indicated that it is dead/unhealthy

[Swarm Oriented] Place virtual markers
Participating Actor: Magic Leap user(s)
Entry Condition:
- In swarm oriented control mode
Exit Condition:
N/A
Flow of Events:
1. Magic Leap user indicates they want to place a virtual marker
2. Virtual marker shows up where the controller is pointed
3. User may choose where marker is placed
4. User indicates they want to release and place, marker places and indicates it has been placed

[Robot Oriented] Dynamic information scaling
Participating Actor: Magic Leap user
Entry Condition: In robot oriented control mode
User has a robot/s selected
Exit Condition:
User deselects robot
Flow of Events:
1a. Indicators on robot grow in size as users move closer to that robot
1b. Indicators on robot shrink in size as users move away from that

[Swarm Oriented] Assign Robots to Subgroup
Participating Actor - Magic Leap User
Entry Condition:
- Robot is not currently in the subgroup
- Robot is not currently in any other subgroup
Exit Condition:
- Robot will be part of the selected subgroup
Flow of Events:
1. Magic Leap User selects robot they want to assign to a subgroup
2. System displays selected robot
3. Magic Leap User indicates subgroup robot should be assigned to
4. System displays updated group