# Augmented Reality as a Means of Improving Efficiency and Immersion of Human-Swarm Interaction

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

By
Eric Arthur
Przemek Gardias
Daniel Sullivan

Date: May 11, 2020
Report submitted to:
Professor Carlo Pinciroli
Worcester Polytechnic Institute

# Abstract

Researchers are studying new ways to control robot swarms efficiently and intuitively. A recent approach that has been taken is to control the swarm through an augmented reality application on a tablet. However, the use of a tablet is not only fatiguing, but also adds an extra layer of abstraction, separating the user from directly interacting with the swarm. The user is not fully immersed, as the movement of the tablet restricts the visualization to a subset of the user's field of view. There has been promising research about the use of augmented reality systems to increase immersion in generic tasks such as solving puzzles. Therefore, we developed the first augmented reality application using the Magic Leap headset to allow a user to be fully immersed and to directly interact with and control the swarm. By combining gestures with haptic and visual feedback, we created a more immersive, efficient, and intuitive method of interacting with swarms than existing methods.

## Acknowledgements

We would like to thank our wonderful advisor Professor Carlo Pinciroli for his valuable guidance and support throughout this project.

We would also like to thank Jayam Patel for providing key insights into the basis for our system, assisting us with technical details relating to swarm robotics, and general guidance to our team.

Finally, we are grateful for WPI and the Computer Science and Robotics Engineering Departments for giving us the opportunity to work on such an important project.

# Table of Contents

# 1. Introduction

## 1.1. Swarm Robotics

Swarm robotics is the study of coordinating groups of robots as systems called swarms. This is done in an autonomous and decentralized way[1]. These robots interact to create a collective intelligence or behavior[2]. The inspiration from swarm robotics partly stems from social insects that create complex interconnected systems that accomplish tasks much greater than an individual can[2]. Examples of current robots used in swarm systems include: Khepera, SwarmBots, Kilobots, Alice, etc. Although the domain of swarm robotics is in its early stages, it has applications in exploration, area coverage, rescuing victims from dangerous areas, etc.[3] and can generally be extrapolated to any task which can leverage the utility provided by multiple synchronized robots acting collaboratively.

### 1.1.1. Controlling Swarms

A current area of research within swarm robotics is human-swarm interaction (HSI). It is the study of one or multiple human operator controlling or issuing commands

[1] Iñaki Navarro and Fernando Matía, "An Introduction to Swarm Robotics," ISRN Robotics, vol. 2013, Article ID 608164, 10 pages, 2013. https://doi.org/10.5402/2013/608164.
[2] E. Şahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," in Swarm Robotics, 2004, pp. 10–20
[3] Tan, Ying, and Zhong-yang Zheng. "Research advance in swarm robotics." *Defence Technology* 9.1 (2013): 18-39.

to a robot swarm[4]. HSI allows the swarm's behavior to be augmented and/or defined by a human operator to successfully achieve their desired goal. Combined with autonomous algorithms, HSI allows for complex tasks to be done with simple interaction by the human operator. There are many ways to interact with swarms, such as running commands from a terminal on the swarm's network. In this paper we focus on the use of augmented reality devices in order to interact with a swarm. Furthermore, there are different perspectives through which operators can interact with a swarm. Using the convention from Pinciroli, Patel, Xu, 2019[5] there are three modalities of interaction with a swarm:

- **Robot-oriented:** When a user interacts/commands a single robot within the swarm at a time.
- **Swarm-oriented:** When a user interacts/commands the entire swarm, or a sub-group of it as a single entity.
- **Environment-oriented:** When a user interacts with the environment (in a real or virtual sense), which the system breaks down to tasks which are necessary to achieve this interaction, and sends the commands to the swarm.

These different perspectives each have their own benefits and drawbacks. For example, robot-oriented control provides a greater scale of control to the human operator, but increases the difficulty of executing more complex commands due to the many

---

[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] Jayam Patel, Xu Yicong, Carlo Pinciroli. Mixed-Granularity Human-Swarm Interaction 2019 IEEE International Conference on Robotics Automation (ICRA 2019). IEEE press.

commands necessary for each individual robot[5]. Therefore, the best systems should use a combination of multiple perspectives. Furthermore, since swarms can consist of a great number of robots, the size of a swarm can be problematic for a human operator to understand. In this paper we will focus on the area of HSI involving how to make interaction with a swarm natural and immersive, such that operators can easily use the swarm to complete tasks.

## 1.2 Problem Statement

Robotic swarms have many advantages over single-robot systems in situations that require significant coverage or fine grained control, including exploration or search-and-rescue scenarios. However, swarms are difficult to work with both from an operational and development standpoint as the information and coordination difficulties scale with each robot, as a swarm needs to function synchronously to achieve success in many tasks. Often, it is difficult to understand the state of a swarm, sub-swarm, or even a single robot within the larger context of a swarm. We apply the features of a recent augmented reality headset system, the Magic Leap One, to provide intuitive control options for swarm manipulation and debugging information synthesis through visual indicators, as well as to tackle some of the underlying issues of complex swarm control systems.

## 1.3 Contributions

We create a novel method of controlling a robotic swarm using the Magic Leap augmented reality headset. By combining features of the headset, we provide the swarm operator multiple input methods for control via gestures and controller input, and further explore additional options including control through voice recognition-based commands. Our application aims for feature-parity with existing tablet-based augmented reality swarm control methods in order to be able to evaluate the two systems from a human-swarm interaction perspective. Finally, we propose further development that can improve our application-specific contributions by further expanding and improving our control methods. Additionally, we discuss the state of augmented reality headset hardware, including strengths and limitations when applied to swarm control.

## 2. Background

### 2.1. Augmented Reality

Augmented reality (AR) is a technology which overlays computer graphics on top of one's perception of the real world[6]. Methods of AR come in either of two forms: handheld, where a user holds a device that uses a camera to show the world with augmentations or head-worn, where a user wears a device on their head that augments their view. Generally, AR is used to supplement one's understanding of the physical world with virtual elements that usually correspond to the view which is captured through the AR system’s camera. Current limitations of AR come from the necessity of the system to constantly capture and process the view of the device, requiring

[6] Azuma, Ronald, et al. "Recent advances in augmented reality." *IEEE computer graphics and applications* 21.6 (2001): 34-47.

significant processing capabilities and power supply[7]. Therefore, these systems are often too cumbersome to be worn comfortably for long periods. These limitations can detract from the ideal experience the device provides, however as the technology matures and these limitations are minimized, current applications will only become more effective.

### 2.1.1. Aspects of Immersion

A defining point of AR is the immersion which it enables. However, since poorly developed AR applications can be so detrimental to the user experience that they can be worse than traditional software, development of applications that effectively use the technology is vital. For example, considering an AR headset, poorly designed interfaces can significantly restrict one’s perception of the world. Since we seek to develop an application which attempts to simplify a complex use case, it is imperative that we do not overwhelm the user’s field of view. Another facet of immersion is the means of interaction. AR systems are useful in providing supplemental information to the existing physical environment which the user is interacting with. Therefore, interactions with the AR system should simply be an extension of the natural methods of interaction with the physical environment, as if the system were not there.

[7] N.I.A.M. Nazri, D.R.A. Rambli, "Current limitations and opportunities in mobile augmented reality applications", *Computer and Information Sciences (ICCOINS) 2014 International Conference on*, pp. 1-4, 2014.

## 2.2. Related Work

### 2.2.1. Interacting with Swarms in Augmented Reality

AR allows some of the deficiencies of interactions with swarms to be addressed through the visual feedback that it enables. The latency which is apparent in a swarm's reaction to commands issued can be made less noticeable by the operator due to the visual feedback which can be shown by the virtual object, representing its physical world equivalent, reacting as the local system processes the input before the task is completed. This verification that input has been received by the system allows for some of the complex, multi-step commands to be chained together without the latency of waiting for the entire system to respond.

There are many studies exploring the benefits and drawbacks of HSI using AR. Many come to the conclusion that less gestures are better[8]. This allows human operators to remember less of the nuances of the system and use their previous knowledge of gestures. The use of overloading gestures seems important for expressing the differences in command modality[9]. This allows for users to swipe with their whole hand to send a command to the swarm, or swipe with a finger to send a command to a single robot. This abstraction of commands represents the different levels of modality abstraction and therefore was easy for operators to learn fast. A large

[8] Podevijn, Gaëtan, et al. "Gesturing at Subswarms: Towards Direct Human Control of Robot Swarms." *Conference Towards Autonomous Robotic Systems*. Springer, Berlin, Heidelberg, 2013.
[9] Nagavalli, Sasanka, et al. "Multi-operator gesture control of robotic swarms using wearable devices." *Proceedings of the Tenth International Conference on Advances in Computer-Human Interactions*. IARIA, 2017.

drawback with gestures in general is the ability to remember all the possible actions and therefore one study found that a menu or widget may be necessary to indicate which actions a user can take[10]. This allows for users to forget contextual gestures and just remember the general ones, as the others will be provided when appropriate.

An existing study of human-swarm interaction through augmented reality is central to this paper. It supports three modalities of swarm control: environment, team, and robot-oriented[11], implemented as an app for a hand-held device. The system which the study is built upon compromises of a swarm of robots, tracked through a 10-camera Vicon motion tracking system[12], communicating with the app client through the ARGoS multi-robot simulator[13]. We emulate this system architecture, interacting with the Magic Leap augmented reality spectacles instead of the hand-held device app.

### 2.2.2. Interacting with Swarms with Gestures

Interaction with swarms through the use of gestures is a way of allowing the user to control the swarm in an immersive manner. This can only be achieved when the gestures are simple and natural. There have been many different approaches for

[10] Wobbrock, Jacob O., Meredith Ringel Morris, and Andrew D. Wilson. "User-defined gestures for surface computing." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009.

[11] Patel, Jayam, and Carlo Pinciroli. "Improving Human Performance Using Mixed Granularity of Control in Multi-Human Multi-Robot Interaction." *arXiv preprint arXiv:1909.07487* (2019).

[12] “Award Winning Motion Capture Systems.” *Vicon*, www.vicon.com/.

[13] C. Pinciroli, V. Trianni, R. O’Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, “ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems,” Swarm Intelligence, vol. 6, no. 4, pp. 271–295, 2012.

interpreting gestures in human-robot interaction scenarios, from wearable devices to cameras on the robots tracking the hand movement. Some more advanced methods allow for individual fingers to be tracked, adding more possibilities and combinations for gestures. This also allows for the ability to map more complex actions to natural movements. By limiting gestures to only hand movements, the number of possible gestures is reduced, however the benefit of this restriction is that it prevents the gestures from becoming less natural as complexity increases.

A study conducted by the U.S. Army Research Laboratory summarizes the state of HSI methods in relevant literature or applications as they related to use cases in the field[14]. The study encompasses many systems outside of the relevant domain including camera-based systems and wearable instruments. A large focus of the report reviews gestural control tasks and the quality of how complex commands translate to gesture control, albeit with a focus on controlling a single robot. The study concludes with key points in gesture design, generally outlining the difficulty in abstracting a system of complex commands to acceptable gestures and therefore ruling out a stand-alone gestural system. It lists the following factors, amongst others, which the complexity of the system scales with: member the gesture requires, ease of execution, ease of learning, ease of recognition, distinguishability from normal gesticulations and other work movements, as well as number of gestures, comfort of gestures, and gesture duration. Due to these limitations of a stand-alone gestural control system, it suggests a

[14] Elliott, Linda R., et al. “Gesture-Based Controls for Robots: Overview and Implications for use by Soldiers.” *Army Research Laboratory*, 1 July 2016, doi:10.21236/ad1011904.

deictic information system for pointing gestures that is augmented through a multimodal context such as speech, keyboard input, or visual display.

Due to the near limitless number of gestures, some gesture recognition applications have issues detecting them and therefore limit the number of gestures. For example, the Magic Leap only recognizes 8 default gestures, and extra have to be programmed in. If too many are added the processing takes too long to differentiate one gesture from the next and the immersion and usability of the system is hampered. In conclusion, interactions that are natural and efficient such as pointing or talking are better than many complex gestures.

## 2.2.3. Pointing Gestures

A pointing gesture is a natural indication of an object, but can be used to indicate groups of objects as well through indicating an area in the field of view, comparable to an area selection tool in a classic computer interface[15]. Although the pointing gesture is somewhat limited due to the accuracy of the indicating motion being relative to the view of the user who is issuing a command through the pointing gesture[16], this is an acceptable limitation. A pointing gesture can be used to supplement other control methods to replace the pointing action which would otherwise be performed with a controller. Notably, a multi-user augmented reality system can augment this gesture to

[15] E. Sato, T. Yamaguchi and F. Harashima, "Natural Interface Using Pointing Behavior for Human–Robot Gestural Interaction," in *IEEE Transactions on Industrial Electronics*, vol. 54, no. 2, pp. 1105-1112, April 2007, doi: 10.1109/TIE.2007.892728.
[16] C. Park, M. Roh and S. Lee, "Real-time 3D pointing gesture recognition in mobile space," *2008 8th IEEE International Conference on Automatic Face & Gesture Recognition*, Amsterdam, 2008, pp. 1-6, doi: 10.1109/AFGR.2008.4813448.

other users through visualization of the end point, effectively communicating precise intent.

## 2.2.4. Gestures Combined with Voice Recognition

Richard Bolt suggests a "put-that-there" voice and gesture graphics interface model which combines advances in speech recognition and position sensing accuracy to provide a natural user modality[17]. The usage of dynamically assigned pronouns aids the precision of the pointing gesture to simplify the complexity of modalities which our HSI system requires. Applications with this voice and gesture control schema enable intuitive interactions in augmented reality systems that can enhance gestures with visual feedback, diluting some of the deficiencies which exist in gesture-based HSI which does not include augmented reality. This allows for complex actions to be performed more naturally and to not be restricted by the option of only one modality and non-intuitive and increasingly complex gestures. Combining voice control and gestures is researched by Boris Gromov et al. in an attempt to control a robot swarm[18]. This study utilizes a wearable device to track arm gestures for spatial commands and hand gestures for simple mobility controls, combined with speech for simplifying complex actions.

Enhancing gesture control through voice recognition is supported by neuroscience research which found that human gesture production is highly associated

[17] Bolt, Richard A. *"Put-that-there": Voice and gesture at the graphics interface*. Vol. 14. No. 3. ACM, 1980.
[18] Gromov, Boris & Gambardella, Luca & Di Caro, Gianni. (2016). Wearable multi-modal interface for human multi-robot interaction. 240-245. 10.1109/SSRR.2016.7784305.

with language processing[19,20]. Pointing gestures are naturally paired with pronouns that can define the object which is being pointed to for reference without requiring gestures. This allows the operator to familiarize the swarm in their own context, defined by the task which is being executed. Additionally, using the point which a pointing gesture indicates can allow relative terms to clarify the indicated object or environmental location. Given sufficient hardware and software support from the AR system, this combination of control methods could provide to be the most immersive way to interact with a swarm.

## 2.3. Our Project

In this paper, we expand upon the aforementioned concepts by implementing them in an augmented reality environment through the Magic Leap. The Magic Leap is a head-mounted virtual retinal display which superimposes 3D virtual imagery over real world objects, allowing our application to visualize the swarm, its relevant information, and visual feedback for commands given. The SDK for the Magic Leap includes APIs for developing gesture and voice controls, making it possible to combine these modalities to achieve a fully immersive environment. By allowing the user to interact with the swarm directly through augmented reality and a combination of multiple natural modalities, we create a novel and intuitive method of swarm control.

[19] Kelly, Spencer D., Aslı Özyürek, and Eric Maris. "Two sides of the same coin: Speech and gesture mutually interact to enhance comprehension." *Psychological Science* 21.2 (2010): 260-267.
[20] Hostetter, Autumn B., and Martha W. Alibali. "Raise your hand if you’re spatial: Relations between verbal and spatial skills and gesture production." *Gesture* 7.1 (2007): 73-95.

# 3. Methodology

## 3.1. Problem Formulation

Currently, researchers are devising ways to be able to control robot swarms efficiently and intuitively. A recent approach that has been taken is to control the swarm through an augmented reality application on a tablet[21]. However, the use of a tablet is not only fatiguing, but also adds an extra layer of abstraction, separating the user from directly interacting with the swarm. The user is not fully immersed, as the movement of the tablet restricts the visualization to a subset of the user's field of view. Therefore, we aim to develop an augmented reality application using the Magic Leap[22] headset to allow a user to be fully immersed and directly interact with the swarm. By combining gestures and voice control with haptic and visual feedback, we attempt to create a more immersive, efficient, and intuitive method of interacting with swarms than current methods. To create such an experience, we researched current options available for augmented reality headsets, as well as the best methods of HSI in order to decide the best methods for achieving our goals.

### 3.1.1. Use Cases

The use cases which we aim to cover can be expressed in the following form:

[21] Patel, Jayam, Yicong Xu, and Carlo Pinciroli. "Mixed-Granularity Human-Swarm Interaction." *arXiv preprint arXiv:1901.08522* (2019).
[22] "Magic Leap: Reality Is Just Beginning." *Spatial Computing for Enterprise | Magic Leap*, www.magicleap.com/en-us.

A user wants to

*[action] [object]*

where

*[object]* = one of **robot**, **group of robots**, **swarm**, or non-robot **object**

*[action]* = one of **select**, **move**, or **assign to group**

such that the resulting use cases are

- A user wants to **select** a **robot**
- A user wants to **move** a **robot** to a given location
- A user wants to **select** a **group of robots**
- A user wants to **move** a **group of robots** to a given location
- A user wants to **move** an **object** to a given location
- A user wants to **rotate** an **object**

These use cases can be combined to create increasingly complex use cases such as the following example:

- A user wants to **move** an **object** to a given location using the named group

Since we are combining multiple control methods, these use cases can be done through multiple methods where permissible.

## 3.2. System Architecture and Design Considerations

We use the Magic Leap augmented reality headset to allow a user to control a swarm of Khepera robots[23] through the ARGoS system in a lab setting. Our final system consists of multiple parts as shown in Figure 1. Within this first major section of the methodology, we introduce the parts of our system and provide an explanation of the capabilities of each part. In the later Final Design section, we outline the choices which correspond to each part where necessary.

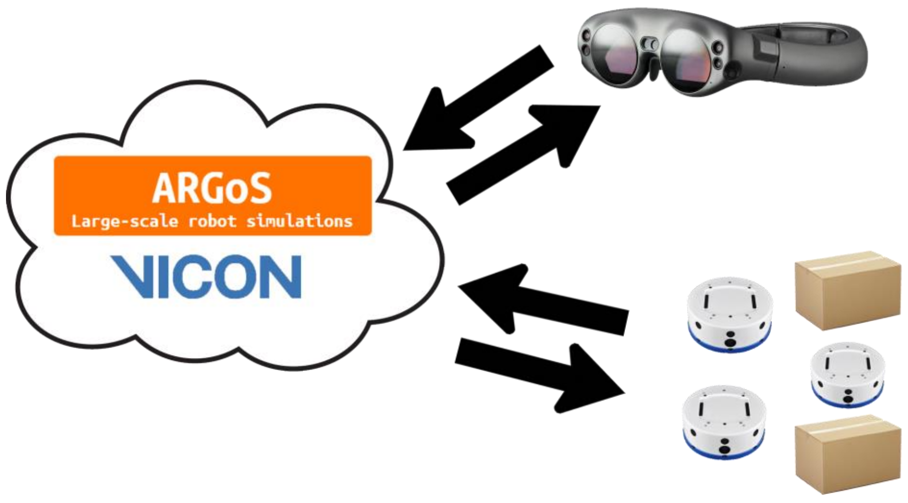


**Figure 1.** Magic Leap integration with ARGoS

The Magic Leap headset displays an augmented version of the real world. The robots are displayed to the user as they appear naturally, but the user can interact with

[23] Kteam. "BEYOND MINIATURE TECHNOLOGY." *K*, Kteam Https://Www.k-Team.com/Dev/Uploads/Logo-K-Team.png, 12 Mar. 2020, www.k-team.com/.

them as if they were virtual objects to issue commands. The Magic Leap receives commands from the user through interaction with the controller or headset. Similar to the aforementioned mixed-granularity study, commands are issued from the Magic Leap to the ARGoS system which in turn sends them to the robots. Because of this multi-step process, response time is a large concern. We define reasonable response time as any action that creates a resulting response within 2 seconds, as shown by an HCI study done on average user waiting time[24].

### 3.2.1. Magic Leap

The Magic Leap One Creator Edition is an augmented reality system released in 2018 that consists of a headset and controller that allow a user to interact with a virtual environment which is displayed over the real world. The headset contains cameras that are used to detect the content of the image which the user sees and a virtual display projected into the lens which enables the headset to draw over the image. The controller which comes with the Magic Leap has multiple buttons and a touchpad that allow various interaction techniques such as swiping, tapping, and holding, on top of having motion and gyroscopic sensors. The Magic Leap is able to run software developed in several environments, and for our software we chose to work with Unity

[24] Nah, F. F. H. (2004). A study on tolerable waiting time: how long are web users willing to wait?. *Behaviour & Information Technology*, *23*(3), 153-163.

using C#. Magic Leap offers an API with documentation[25] that provides examples of different functionalities and guides that aid developers.

The Magic Leap has the ability to mesh out the environment, and using raycasting it is possible to interact with the environment. Raycasting, provided by the Unity engine, allows for a ray to be projected from a specific starting point and orientation and returns the object it hit. For our purposes we may raycast from the controller position, the cameras on the headset and the pointing gesture. This allows us to, depending on the circumstances, determine where a user is pointing with the controller, where they are looking, or what object they are actually pointing to. Therefore, we are able to determine what the user is focusing on in multiple ways. The most intuitive way will be evaluated and discussed in the respective areas of the design.

### 3.2.2. Controller

The Magic Leap controller provides multiple forms of user interaction and input. As shown in Figure 2, the controller has three buttons and a touchpad. The bumper button state and home button state are binary, either pressed down or up, and can be mapped to a function call on press. The trigger is a continuous value between 0 and 1 that provides information on how far down the trigger is pressed, with 1 representing the trigger pressed down. The touchpad is manipulated by the user's finger and tracks which gesture they input and its direction and speed. There are 10 default trackpad

[25]Developer Portal | Magic Leap. (n.d.). Retrieved May 5, 2020, from https://developer.magicleap.com/learn/reference

gestures that the controller can recognize ranging from tap, hold down, or radial scroll. The home button cannot be used for anything that requires prolonged duration as a long press on the home button is always bound to exiting the program.

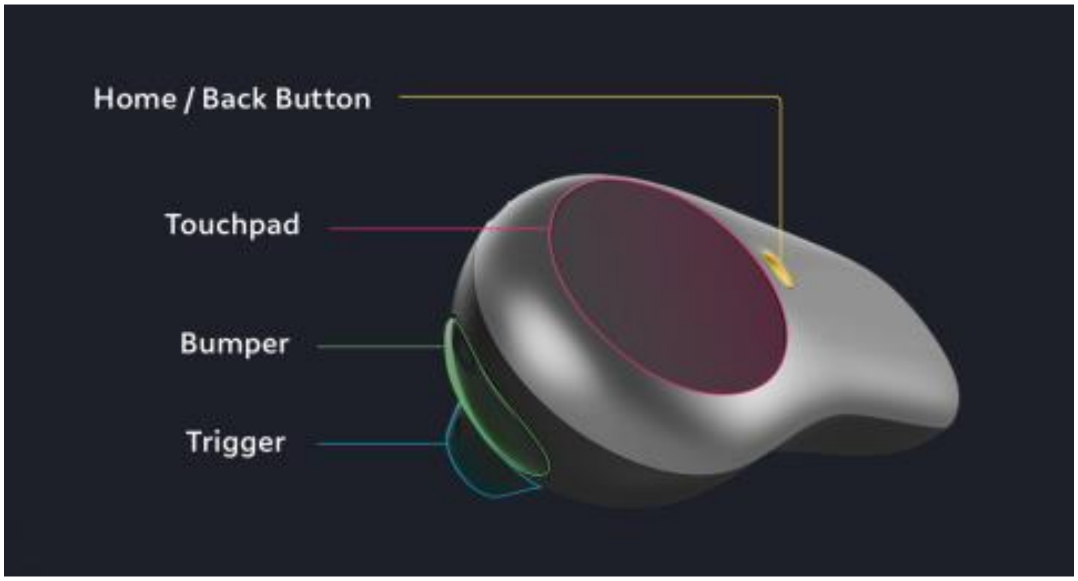


**Figure 2.** Magic Leap controller layout labelling[26].

The Magic Leap controller has a gyroscopic sensor that determines where the orientation of the controller is. Furthermore, the controller contains actuators that allow for haptic feedback to be given to the user. These can provide different levels of intensity allowing for multiple meanings to be encoded this way. Haptic feedback can indicate when a user does a task well or does something wrong or the system does not support it. Haptic feedback has been shown to increase the immersion of the system by giving response information to the user[27].

[26] Magic Leap Human Interface Guide. (2018, September). Retrieved December 13, 2019, from https://medium.com/@davecancode/magic-leap-human-interface-guide-part-two-d3cbe6d0a853.
[27] Henrysson, A., Billinghurst, M., & Ollila, M. (2006). AR tennis.

### 3.2.3. Voice Control

Voice as a method to control the robot swarm could be intuitive or simple due to how humans intuitively combine natural speech with corresponding hand gestures. We believe voice recognition allows us to create one of the easiest controls for human swarm interaction and wished to evaluate this claim further. A user should be able to name an object or group of objects, and from here be able to access that object or group of objects through their voice and give them an action to perform. Once the objects or groups are named, the user should be able to perform the same actions as the controller can through voice.

Our approach mainly focused on two possibilities of recognizing voice, using a wake word or constantly streaming audio. Each method of voice recognition has their own advantages and disadvantages. Using a wake word may be less intuitive and takes more time, but would reduce the amount of data we are streaming and splicing, as well as ensures when a command is being delivered. Constantly streaming audio runs into the issue of background noise and constantly streaming and splicing data, but allows for a more intuitive and immersive experience.

The use of voice is an integral method of interacting with the swarm, as it can feel natural and be efficient when used properly. We aim to transcribe audio and look for keywords to trigger the desired commands. The keywords are defined as the names assigned to an object or a group. Once one of these names are recognized, our system will expect an action. For example, to move a robot named “Alice” to a location that the user is pointing at with the controller, the user would say “Alice move here.” Our system will then get the location the user is pointing at, the robot named “Alice”, and the given action and send a UDP message to ARGoS such that the given command can then take effect through the system. This will allow for the user to have a large amount of functionality at ease, without having to worry too much about selecting the correct objects once they are already named and/or grouped.

Names can also be given to or removed from objects or robots. Similar to the commands to move an object, a command can be given to rename or remove a name given to an object. In this way users have control over what the robots are named, which provides a more immersive experience as it is easier for them to remember compared to arbitrary default names. The commands are expected to be delivered as:

1. [Object or group] move here
2. [Robot or group of robots] move [non-robot object] here
3. [Object] rotate X degrees
4. [Object or group] select
5. [Robot or group of robots] help [other robot]
6. Name X

### 3.2.4. Gestures

Using the Magic Leap, we have eight default hand gestures that can be recognized very quickly and map those to actions, shown in Figure 3. The Magic Leap also has the ability to create custom gestures through checking key points on the hand, as well as at what position the points are, although the processing time and accuracy of these gestures is worse compared to the default ones.

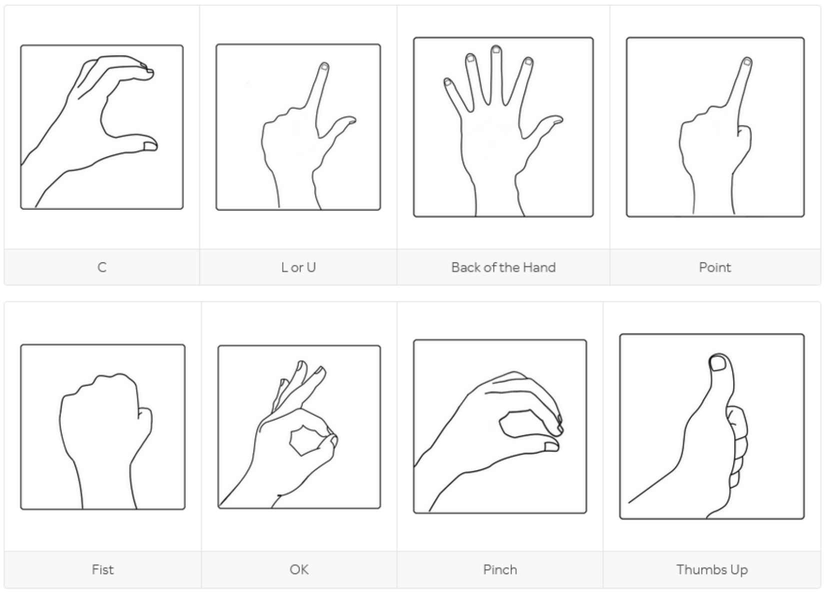


**Figure 3.** Magic Leap Default Hand Gestures[28].

However, using any Magic Leap gestures, whether preset or custom, comes with downsides. Unfortunately, the field of view offered by the Magic Leap is not large, which is necessary for visualizing and interacting with the swarm. To use gestures, the hand needs to be in the field of view of the Magic Leap so that it is able to map the points to the user's hand, which takes up a large portion of the field of view, which may be disruptive to the experience and unintuitive. However, we still believe that gestures are

[28] Magic Leap. (2018, June). Hand Tracking Example. Retrieved December 13, 2019, from https://developer.magicleap.com/learn/guides/sdk-example-gesture-input.

an important way of interacting with the swarm, and decided that they should still be included.

#### 3.2.4.1. Pointing Gestures

An implementation which uses pointing gestures unfortunately includes the same downsides that come with using standard gestures, as shown in the previous section, but also has its own drawbacks that come from the information requirements to raycast from a pointing finger. The pointing gesture, which is defined as part of the default list of key hand poses within the Magic Leap API, is referred to as “index” also has a similar key hand pose that contains an extended index finger, referred to as “L” because the thumb is also extended. These two gestures can both be recognized as pointing gestures because they contain the points required for the origin and direction vectors of the raycast, down the length of the user’s index finger. To get the location of these key points, we are able to refer to the Magic Leap’s API and get the location of the index tip and Metacarpophalangeal (MCP) joint. An example of tracking pointing is shown in Figure 4.

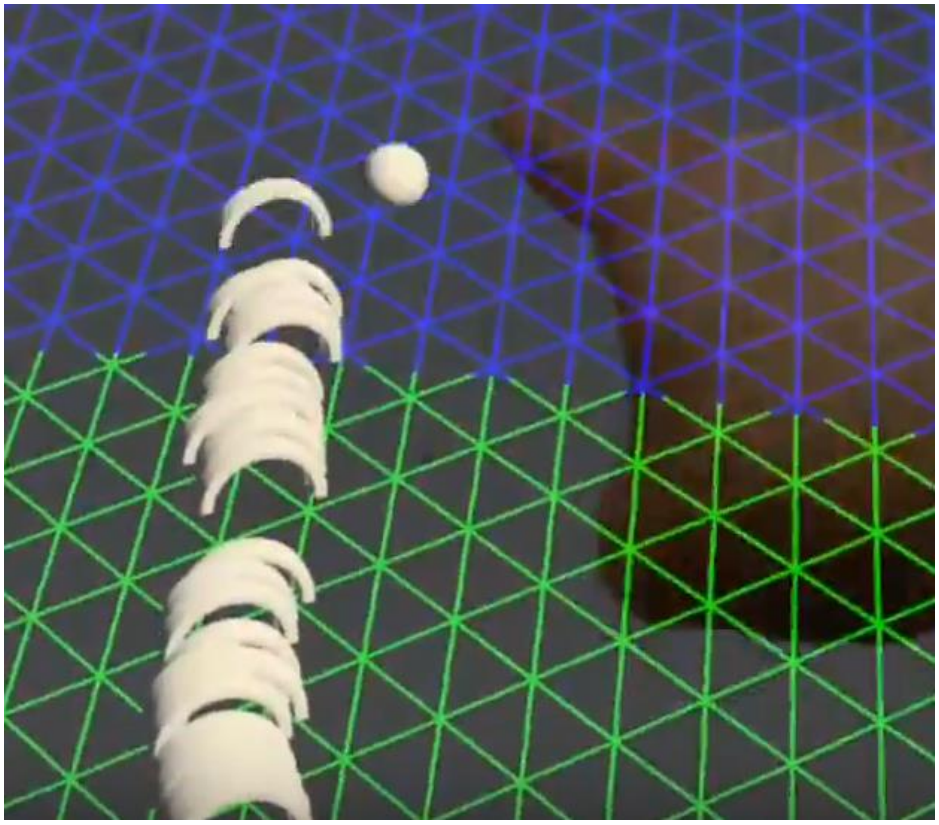


**Figure 4.** Tracking of pointing gesture.

As stated earlier, if we have the location of the index tip and MCP joint, we are able to raycast from the index finger and determine the raycast collision with the mesh or objects as necessary. Additionally, we should be able to combine this information with gesture recognition to only trigger this raycasting upon recognizing a pointing gesture. However, the reality of the gesture and key hand point recognition is that the Magic Leap must recognize the hand within its field of view, the gesture past a predefined confidence, and the key hand points that are required for raycasting in order to successfully provide the described pointing functionality.

Additionally, the aforementioned requirement of having the user's hand within the Magic Leap's field of view may also cause the user to believe the system is not working properly if the hand is within the user's field of view in the headset, but not captured by the Magic Leap's camera. This is another consideration that should be taken into account when designing feedback for pointing gesture input, such as in the form of a cursor indicating the position of pointing when done successfully, differentiating proper input and incorrect hand states/locations.

### 3.2.5. Visualization

In combination with the object API which is native to Unity, the Magic Leap is able to create, manipulate, and remove objects with definable meshes and textures. This meets the visualization requirements that come from the need to match much of the visualization available in the mixed-granularity swarm control tablet application which we compare to. For example, drawing objects over the robots, and updating the robot's respective virtual object's location and orientation when using any of the control methods. Furthermore, we can expand much of the visualization to add visual feedback to effectively band-aid the intrinsic system latency which comes from working with a complex swarm system. We create virtual objects to overlay the real-world Khepera robots which are interactable by the user in real time. These models reflect the shape of the robot to emulate real interactions in a way that is clear to a first time user.

### 3.2.6. Environment

To work with the Magic Leap, it is best to set up in a large, open indoor space. We utilized a large, open lab space dedicated for a Vicon system. The tile floor is covered by a mat that is all the same shade of a matte gray to reduce the reflection of light. This mat is there primarily for the Vicon system, but is also very useful for our purposes. The Vicon system is a motion capturing system that consists of several cameras surrounding an area to track clusters in a controlled environment. For our purposes, we will be using the Vicon to set the origin for our system so that all the objects in our environment have a relative position to it.

It is best that, especially when the mesh is being created, the desired interaction area is clear of objects. If any objects move after the original mesh is rendered, the Magic Leap may have difficulty adjusting to the changes. By having an environment set up such as this, it allows for the Magic Leap to create a mesh as easily as possible, with as little error as possible, which is essential for the success of our system. It is the optimal situation for being able to recognize and interact with the objects in the view with as little environmental disruption as possible.

## 3.3. Final Design

Familiarizing ourselves with the components of the Magic Leap, which we require to achieve the functionality that has been outlined so far, only gives a certain amount of insight into what the headset is capable of, and to what degree. While applying this research to the implementation, we made decisions that were affected by unforeseen factors. In the sections below, we attempt to clarify the reason behind the decisions, justify our choice, and address drawbacks that may come as a consequence of this choice. Finally, we outline the results of these decisions and conclude how the section affects our final application.

### 3.3.1. Magic Leap

Our software implementation on the Magic Leap is divided into specific classes that each handle a different facet of our overall design. The main class *ControlFocusHandler* handles all logic with user input to move objects. Figure 5 shows an example flow of how user input is handled, ending with commands being sent to the robot. The class also handles logic for reading user's gestures, rotation of objects, calculating raycasts, adding or removing for multiple selection, and using helper classes that send UDP commands to the robots. This script is attached to all of the robot objects and therefore only comes into scope when a user hovers over one of them. Because of this the helper classes are singletons in order to enable data continuity. Since this script is constantly created and deleted as the user's focus changes, access to the same data is necessary for each call across instances.

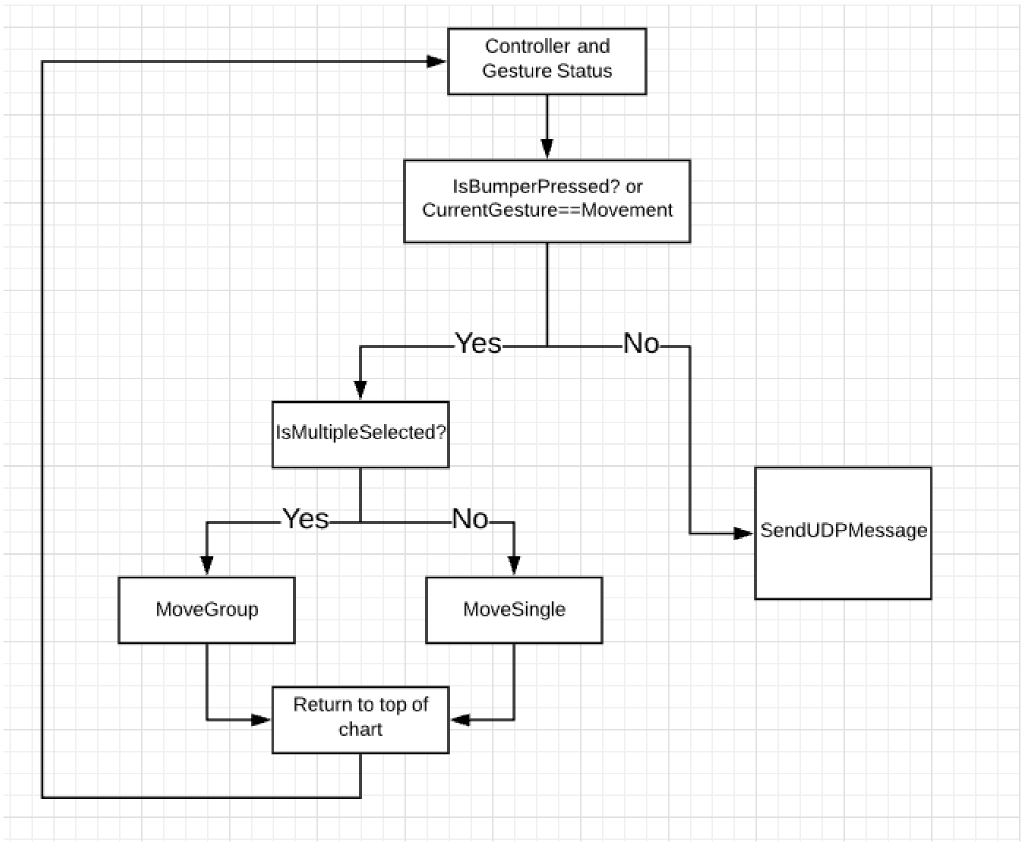


**Figure 5.** Flow chart for movement of objects.

The first helper class, *UDPSenderInstance,* is a singleton that provides access to the socket that sends UDP messages to ARGoS. This calls on a private internal class that contains the actual UDP implementation, and the instance wrapper class is simply a singleton access point. This class provides simple utility functions that generate UDP messages for a robot by using its current position and orientation in accordance with the guidelines provided by Patel et al.[29], and then sending these UDP messages through the socket.

[29] Patel, Jayam, and Carlo Pinciroli. "Improving Human Performance Using Mixed Granularity of Control in Multi-Human Multi-Robot Interaction." *arXiv preprint arXiv:1909.07487* (2019).

The second helper class, *MultipleSelectionInstance*, handles the logic and monitoring of multiple robots being selected. As described in the controller section robots are added or removed from the currently selected group. Then moving any member of the group moves them all with relative positions. The handling of selected robots is simply a list of currently selected robot IDs. Movement of a group of robots is done by assigning selected robot prefabs as children of the one being moved, which is a feature of Unity that allows objects to move in groups and keep their internal distances constant. Handling of creating, maintaining, and emptying this selected list and associated prefabs is done in this class.

### 3.3.2. Gestures

Our primary concern with integrating gestures to be used alongside the controller for certain inputs is to maintain immersion for the entire set. We want to avoid mapping a gesture to something that is completely unrelated to the hand position. It is also important to not chain several gestures together, as it is difficult to remember and becomes less natural. Because of this, we decided to map individual gestures to actions. In testing, the Magic Leap proved to be extremely responsive to the preset gestures that it comes and less so to custom gestures, as points may get lost or take longer to be recognized. For our purposes, we have chosen to use the thumbs up as an alternative way to select an object, and the L hand shape to rotate a selected shape to the left. The use of gestures severely limits the field of view of the user, which is integral to our system. Therefore, we decided to limit the number of gestures, as they seem as if they would improve the user experience in interacting with the swarm only slightly.

Since we focused on tight integration of gestures, we accompanied a successful recognition of a gesture with an auditory chime. Similar to our goals with presenting movement of virtual objects immediately as the user manipulates, this immediate feedback is to remove one of the frustrations which may be met with using the gesture recognition of the Magic Leap. The Magic Leap relies on a recognition confidence threshold for detection, which is the probability which the gesture recognition API returns for each cycle of the application. We set our confidence threshold fairly high, per the documentation recommendations. However, since there are often issues with recognition of hand points and hand poses, which relate to gestures, we believed it essential to provide feedback when a gesture is successfully is detected so the lack of a chime also provides the user an indication that the Magic Leap has *not* detected a gesture should they be doing one.

### 3.3.2.1. Pointing Gestures

As mentioned earlier in the corresponding section which outlined the limitations of how the Magic Leap recognizes gestures and key hand points, there are multiple points of failure which may affect the ability of raycasting down the index finger. The following are points of failure which need to be considered:

- Failure to recognize the user’s hand
- Failure to recognize a key gesture above the predetermined confidence
- Failure to identify any or all key hand points on the index finger which are required for raycasting

The first point of failure, in which the user’s hand is unable to be recognized, cannot be fixed without trying to improve on the Magic Leap’s native API. The second point of failure may be partially fixed through a few approaches. The confidence of the gesture recognition can be lowered to allow the pointing gestures to be recognized more easily. However, alternatively the pointing raycasting could simply not depend on the pointing gesture recognition. By eliminating the need to recognize one of the two pointing gestures, this fixes the issue of the user pointing with their finger occluded by the back of their hand. This does not, however, address the third point of failure, in which the Magic Leap is unable to identify any or all key hand points.

As stated before, if both the index finger tip and MCP joint are visible to the Magic Leap and identified properly, then we are able to simply able to raycast to detect if the user is pointing at an object by using the origin vector as the index tip location and the raycast direction vector as the difference between the index tip and index base. Using these vectors means the raycast will continue the line created by the index finger, which means that if the index finger is bent but still visible the resulting raycast will be drawn in an unexpected angle since we no longer require a pointing gesture to be recognized. However, as the third point of failure states, the required index points may not be available or properly identified, even when the user is pointing at an object, such as when the index finger is occluded by the back of the user’s hand. In this case, there are a few alternative approaches which we can use to estimate a raycast assuming a pointing gesture. Since we are able to use the Magic Leap API to raycast the user’s gaze and get a point of focus, we can instead revert to this input method as input of the pointing gesture when the index points are not available. Alternatively, we can also use the vector from the user’s eyes to the base of their index finger if we are able to identify the MCP joint. This secondary approach makes the assumption that the index finger tip is directly behind the index MCP joint, which often may be incorrect. Therefore, we needed to evaluate the consistency of these two alternative approaches, with the goal of maximizing the consistency of any of the raycasts at any object, while also minimizing noise and discrepancies between switching between our multiple pointing inputs.

In order to reduce noise, we applied a noise filtering method known as the Kalman filter[30]. Because the filter works on multiple incremental time-contextual inputs, the normalization will work only after multiple frames after the beginning of recognizing pointing input. This means that our application also has to recognize pauses between pointing, which means that by default naïvely reverting to using raycasting from the Magic Leap eyes API will not consistently function well.

Because of these previous points, primarily the poor native hand point detection of the Magic Leap, indicating an object or interface through the use of a pointing gesture simply isn’t feasible. In order for it to be as immersive as controller input the detection needs to be consistent and precise. Although the preciseness of the selection can be improved through smoothing methods as discussed before, there is simply no way to overcome poor detection consistency by the underlying system.

[30] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

### 3.3.3. Feedback

To make the experience more intuitive and immersive, we added different forms of feedback to make the user feel that they are directly interacting with the swarm and to confirm the action they have performed has been recognized. To do this, we use haptic feedback through the controller and audio feedback through the Magic Leap headset. When the user hovers over an object with the controller, the controller quickly vibrates to let the user know that they have hit an object. The same vibration happens when the user completes moving an object. This is done because the virtual robot that is created when moving is deleted that the user knows their action went through. Audio feedback is used when a user performs a gesture. If a gesture is recognized, then the user will hear a chime to let them know that their gesture has been recognized and that action will be performed.

# 4. Experimental Evaluation

## 4.1 Summary

Unfortunately, due to COVID-19 we were unable to perform a user study as intended to get analysis on the immersiveness and usability of our system. In Table 1 we present each use case that would have been tested in the user study and whether or not it is implemented in the final application. Furthermore, we explain how each action is performed, following this table we have a discussion on each individual use case alongside figures showcasing the application. Finally, we have a brief discussion on how the user study would have been performed for future work on this application.

| Use case | Implemented? | How to perform action? |
|---|---|---|
| Select a robot | Controller: Yes | Hold bumper down when hovering over robot (see Figure 8) |
| | Gesture: Yes | Thumbs up gesture when hovering over robot (see Figure 8) |
| | Voice: No | See discussion on voice |
| Move a robot | Controller: Yes | Hold bumper down when hovering over robot, move robot, then release bumper (see Figure 8) |
| | Gesture: Yes | Thumbs up gesture when hovering over robot, move robot, then Thumbs up gesture again (see Figure 8) |
| | Voice: No | See discussion on voice |
| Select a group of robots | Controller: Yes | Trigger down when hovering over a robot to add to group (see Figure 9). |
| | Gesture: Partially | Implemented but not tested/used. Done with a back of hand gesture when hovering over the robot to add to the group. |
| | Voice: No | See discussion on voice. |
| Move a group of robots | Controller: Yes | Hold bumper down when hovering over the robot after multiple selection has been initiated (see Figure 9). |
| | Gesture: Yes | Thumbs up gesture when hovering over the robot after multiple selection has been initiated (see Figure 9). |
| | Voice: No | See discussion on voice. |
| Move an object | Controller: Yes | Hold the bumper down when |

| | | |
|---|---|---|
| | | hovering over the robot (see Figure 8). |
| | Gesture: Yes | Thumbs up gesture when hovering over a robot (see Figure 8). |
| | Voice: No | See discussion on voice. |
| Rotate an object | Controller: Yes | Rotate finger on trackpad when hovering over the box (see Figure 10). |
| | Gesture: Yes | L hand gesture when hovering over the box (see Figure 10). |
| | Voice: No | See discussion on voice. |
| Assign names to objects | Voice: No | See discussion on voice. |

**Table 1.** Our use cases, their implementation status, and how to perform them.

## 4.2 Discussion

Our system was able to implement individual and multiple selection on robots and boxes. The user is able to select either individual objects and move them to a desired location, as well as groups of objects. Due to the implausibility of testing with the actual robots we have proven this to work successfully using the Magic Leap and ARGoS simulators, which were able to send and receive our UDP messages as shown in Figures 6 and 7.

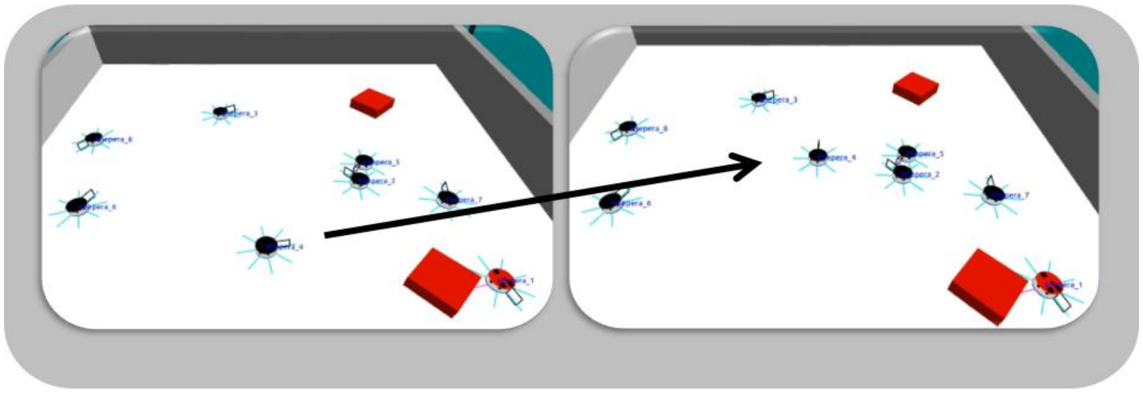


**Figure 6.** Movement of an individual robot in ARGoS using UDP messages.

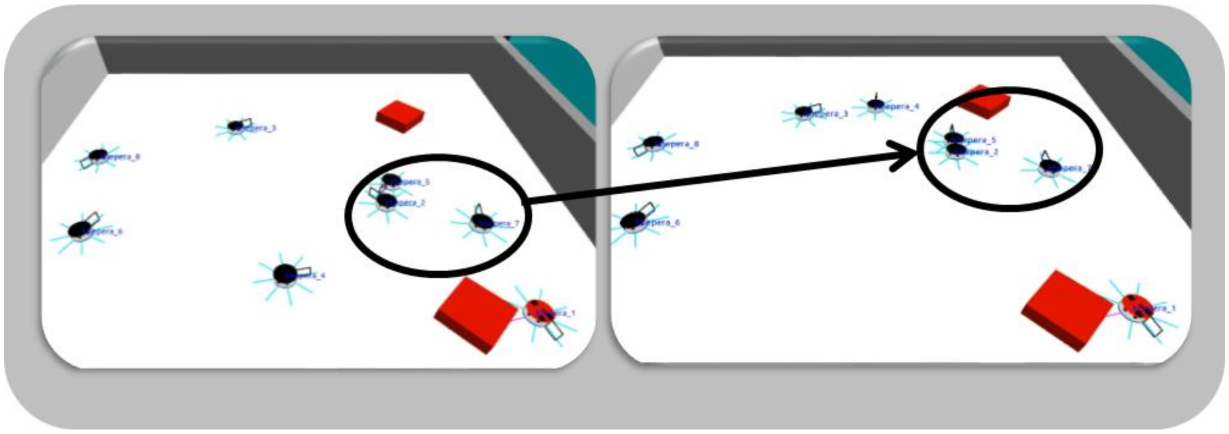


**Figure 7.** Movement of a group of robots in ARGoS using UDP.

The input for selection on the controller is arbitrary, since there are two buttons that both work quite well. We decided to use the bumper for single selection, since it has a binary state it is easy to determine if the user is pressing it down or not. Compared to the trigger which instead provides continuous values. A user presses the bumper down to select an object they are pointing at and then holds it down to move the object as shown in Figure 8. This is done by ignoring the raycast hit on the selected object and getting the new point of where the raycast hit. Since the world is meshed the raycast will hit the ground and therefore objects can be moved by simply pointing and dragging. When the bumper is released the object is translated to that new position using Unity, and in the process described above, a UDP message is sent to move the robot or box there.

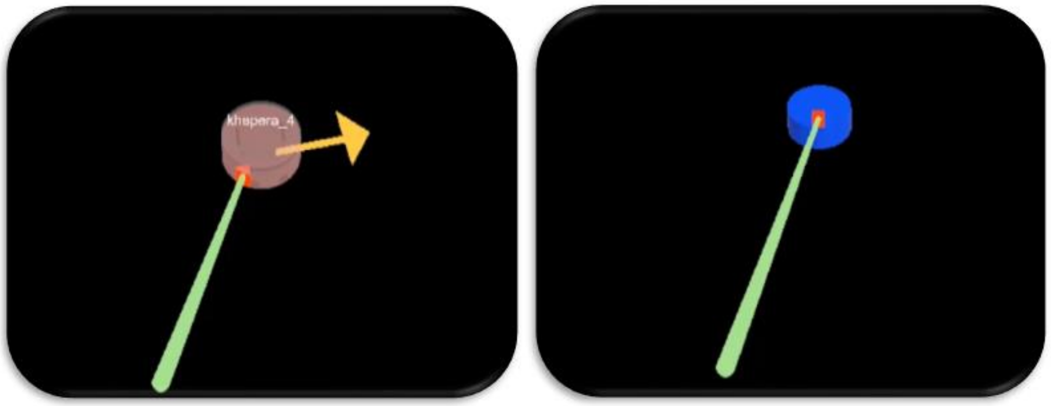


**Figure 8.** Single selection and movement of a single robot.

Similarly, single selection can also be initialized using the thumbs up gesture as described in our initial design. Using a thumbs up is a universal symbol for a confirmation or response of “good”. When the Magic Leap recognizes this gesture, it will make a chime noise to confirm that the gesture has been recognized, and will select the object just as if you had pressed the bumper on the robot. Once the gesture is recognized and the chime is heard, the user may stop holding the gesture and perform their movement with the object. Once the user is finished performing that action, they may show the thumbs up gesture again, and the selection of that object will stop, working the same as releasing the bumper. Due to the inability to record actual footage using the Magic Leap, and the fact that the simulator has no difference in appearance for controller versus gestures we are unable to showcase this

Since the trigger does not provide simple access points for logic, we use it for multiple selection. If the trigger is pressed down the user may select multiple robots with the bumper and by holding down the bumper can move them all. Individual robots are added or removed from the selection group by pressing trigger as shown in the left image of Figure 9. Then moving any robot in the group moves all robots in a cluster relative to their starting positions and clears the selected list in the process. Then UDP messages are sent to each robot exactly the same as single selection.

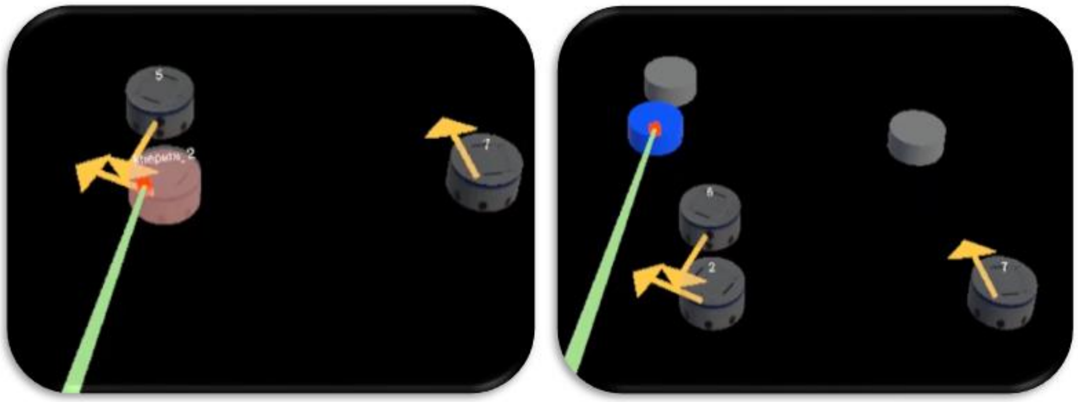


**Figure 9.** Multiple selection followed by movement of a group.

The touchpad is used to handle rotation. Compared to the possibility of rotating the controller to signal rotation, the touchpad is more intuitive and easier to use because of the granularity which enables it to function in parallel with other controller functions. Using the touchpad the user can move and rotate objects at the same time in a way that feels simple (see Figure 10). The touchpad checks for input gesture, direction, and speed. Therefore, we use the radial scroll functionality of the touchpad, alongside its direction to determine rotation. If a user scrolls counterclockwise then the object will rotate counterclockwise relative to the speed they do the gesture at.

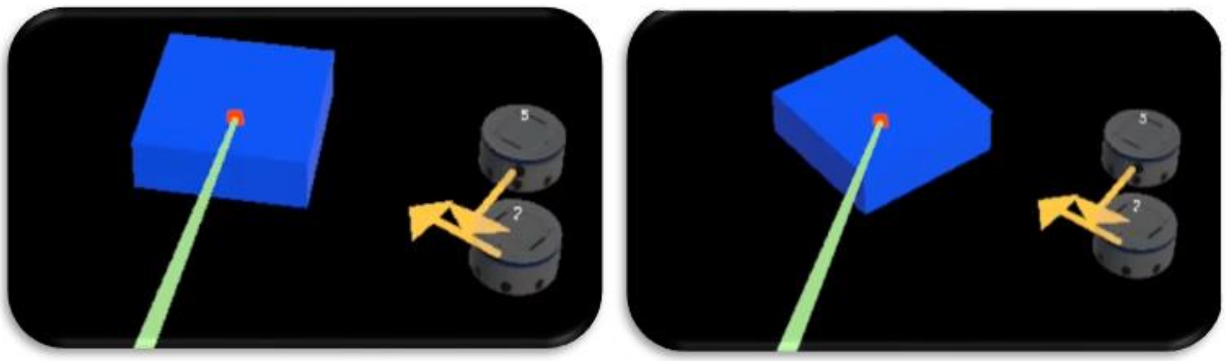


**Figure 10.** Rotation of a single box using the touchpad.

The usage of the L hand shape as a gesture provides an alternative way to rotate an object. When an object is hovered over, the user may hold up the L hand shape. Once it is recognized, the Magic Leap will play a chime. As long as the gesture is held up, the object will rotate. Once the gesture is no longer recognized, the object will stop rotating. This provides a secondary option to perform object rotation

The method of voice control which we discussed earlier was shown to be a fantastic supplemental method to the other modes of control which the Magic Leap offered. The internal microphone allowed us to be able to record audio from the user, which we would have begun recording on the press of the home button and end recording by pressing the home button again. The audio between those two presses would be stored as an audio clip. The audio clip is then made into an .wav file through

Calvin Rein's SavWav script[31], which takes a unity audio clip and converts it to a .wav file and saves it. This file is then sent to IBM's Watson API[32] to be converted to text. The call is typically returned in about half the length of the audio clip, so a 7 second clip will be returned in about 4 seconds. From here, we would parse the returned text for the action requested and the objects that it affects, and then send the UDP command to ARGoS to have that action occur. A flow chart of this detailed process is shown in Figure 11, and the Watson output is shown in Figure 12. While we did not have the ability to fully test this within our system, we do believe it would be an efficient and intuitive way of controlling a robot swarm.

[31] https://github.com/steelejay/LowkeySpeech/blob/master/Code/SavWav.cs

[32] "Watson Speech to Text - Overview." *Overview* | *IBM Cloud*, www.ibm.com/cloud/watson-speech-to-text.

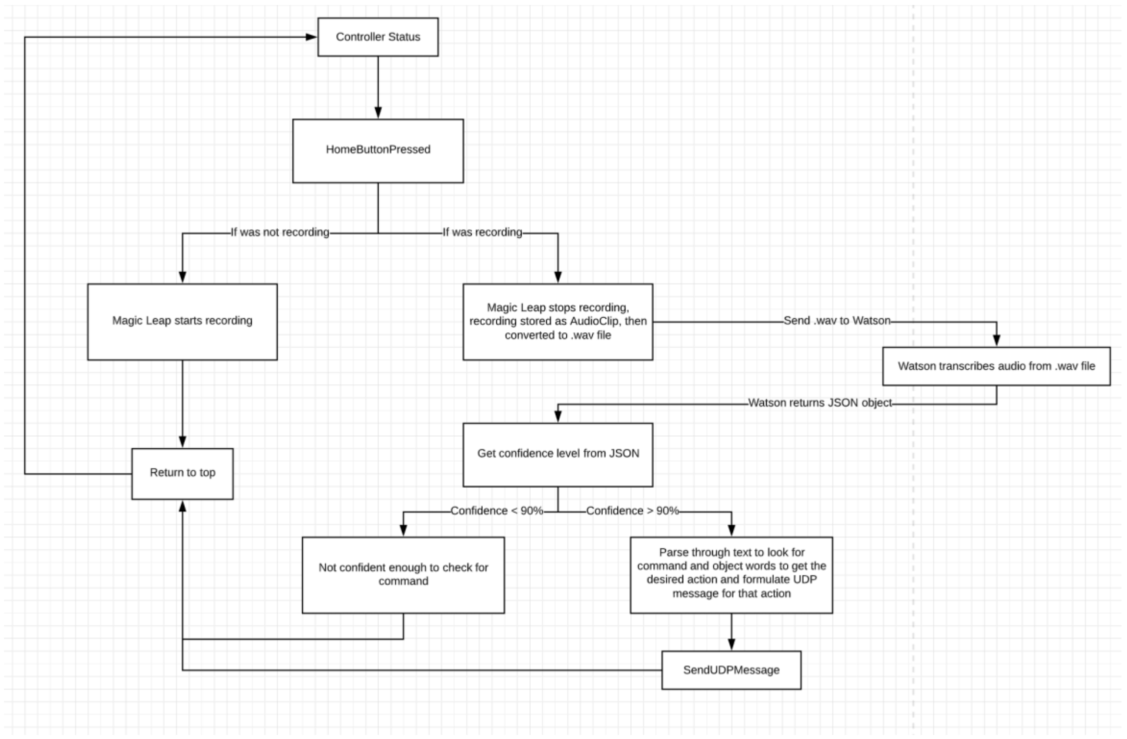
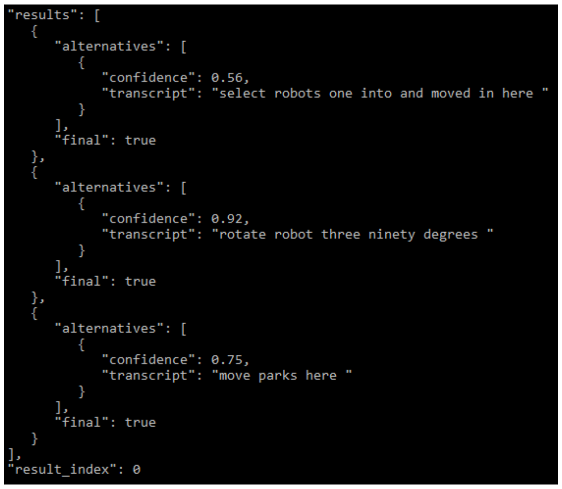

Controller Status
HomeButtonPressed
If was not recording
If was recording
Magic Leap starts recording
Magic Leap stops recording, recording stored as AudioClip, then converted to .wav file
Send .wav to Watson
Watson transcribes audio from .wav file
Watson returns JSON object
Get confidence level from JSON
Return to top
Confidence < 90%
Confidence > 90%
Not confident enough to check for command
Parse through text to look for command and object words to get the desired action and formulate UDP message for that action
SendUDPMessage

**Figure 11.** Voice control data flowchart.



```
"results": [
   {
      "alternatives": [
         {
            "confidence": 0.56,
            "transcript": "select robots one into and moved in here "
         }
      ],
      "final": true
   },
   {
      "alternatives": [
         {
            "confidence": 0.92,
            "transcript": "rotate robot three ninety degrees "
         }
      ],
      "final": true
   },
   {
      "alternatives": [
         {
            "confidence": 0.75,
            "transcript": "move parks here "
         }
      ],
      "final": true
   }
],
"result_index": 0
```

**Figure 12.** Watson's response to three example commands.

### 4.2.1 User Study

To do measure the efficiency and intuitiveness of our system, we would assign a set of tasks that would be able to measure these factors in individual parts of our system. After the user performed these tasks, we would ask them questions on their opinions on the individual parts of our system that we would like to assess. To measure our system specifically, we would ask the user to assess the intuitiveness and efficiency of our usage of feedback, gestures, and voice.

Given enough time to get user feedback, we would set up a user study with six tasks that utilized our control system in order to get an assessment by a number of users. We would perform these tasks first on the original tablet application and then on our Magic Leap application. Below is a list of tasks we would ask the users to complete given both systems. Tasks which can only be completed on the Magic Leap system are denoted with an asterisk[33].

1. Select and move robot X to a new location
2. Select, move, and rotate box X to a new location
3. Select, move, and rotate box X using gestures*
4. Select robot X using voice commands*
5. Select multiple robots and move them to a new location
6. Make robot X and robot Y on the same team

[33] For tasks which can only be performed on the Magic Leap application, we would collect feedback specifically considering the intuitiveness and simplicity of gesture- and voice-based commands. Further research into evaluating these is required to properly develop this section of the user study.

Considering haptic feedback, we would prioritize collecting feedback on haptic feedback intensity and frequency. Similar goals apply to auditory feedback, focusing on the clarity of relating the chime to a successful gesture recognition. We want to verify if the inclusion of haptic and audio feedback was an improvement from the tablet application experience, and if user’s have feedback methods which they may have expected to be associated with certain inputs.

Considering gestures, we want to collect feedback regarding the usefulness of each gesture, considering the controller input counterpart. We want to take user input of suggesting preferred gestures for current input methods. Additionally, we want to also get the opinion of the user if they would ever consider using these gestures, or if they would solely use the controller.

Finally, we want to ask how the controller input was received and control methods felt when interacting with the system. Since the controller is a more traditional control method, we believe users will more frequently feel comfortable with the controller. We want to compare the functionality and usability of the controller compared to that of the tablet application.

# 5. Conclusion

## 5.1 Summary

We propose and demonstrate a Human-Swarm Interaction system built for the Magic Leap One augmented reality headset. We utilize the headset's capabilities to display virtual objects over the interactable Khepera IV robots and allow multiple control methods of the swarm through gesture input and a physical controller that allow the user to command the swarm to complete collective transport tasks efficiently and intuitively. Additionally, we combine the visual feedback of the graphical overlay with haptic feedback through the controller and auditory feedback through the headset. We also attempted integration of intuitive voice control and pointing gestures as additional control methods to remove the need of a physical controller and to make controlling the system more immersive. We found the pointing gesture was unreliable for control, especially when looking over a system, as the headset would not be able to track the end of the finger. Voice control showed much promise, but we were unable to complete this feature due to the Magic Leap implementing its functionality late in our project and time lost to COVID-19. Our system is combined with the parallel work of Kent Libby, Eula Zhong, and Noah Van Stralen to provide additional visualizations for the entire swarm including details of the task assigned, such as robot status and path of movement.

## 5.2 Lessons Learned

Magic Leap One suffers from a significantly under-documented developer API, and many features of the headset were classified as in progress. As an example, an article outlining voice recognition on-device led to a page describing a demo but no code nor links to associated API documentation[34]. Sound playback was similarly undocumented. Attempts to contact Magic Leap One support by email resulted in a redirect to pose the question on the product forums, which were not helpful. This lack of documentation, and community support meant often roadblocks in development. Although a large set of tutorials exists for getting started with the Magic Leap, they cover minimal functionality often dedicated to exhibiting a single feature.

We also learned about how to deal with and adapt to technical limitations through using the Magic Leap. Our initial idea of using pointing gestures as a reliable control mechanism would require more than just the headset in our case. While gestures are able to be easily recognized by the headset, our pointing gesture requires the tip of the finger to be seen when pointing at an object. However, since in our situation we look down at our robot swarm, the headset could not track the tip of the finger, making the pointing unreliable. While gestures were initially supposed to be at the forefront of our project, we learned that the Magic Leap had a limited field of view that is heavily disrupted by gestures, as they need to be placed directly in front of the headset. Due to this, as well as our research and conversations with others about how too many

[34] "Developer Portal: Magic Leap." *Developer Portal* | *Magic Leap*, developer.magicleap.com/en-us/learn/guides/sdk-example-audio-capture.

gestures become confusing and unintuitive for users, we decided to limit the gestures to simple gestures for common actions.

We learned a lot about the development process while making this application. By working in groups, we learned how to properly split up and delegate tasks that we needed to get done weekly. By splitting up our work, we were able to get more work done faster rather than having the whole group work on one problem at a time. Since we also had to work with another group of students on the same project, we also learned a lot about working with other groups. While our project was the same, our code often did not overlap or conflict, but the functionality would be dependent upon each other’s code. We would meet weekly with the other team to discuss their needs from us, as well as our needs to them, and then merge our code together so we could solve any possible conflicts in person.

Working on this project also taught us a lot about the importance of research prior to attempting to solve a problem. Since this application is an extension upon already existing work, we had to spend a significant amount of time understanding how it currently worked and all of its existing functionality so that we could replicate it and expand upon it. We then had to do research and requirements gathering of our own to see what approaches have been taken to solve problems similar to ours, as well as see what people would desire in our application. This was very beneficial, as we were able to see what people had already tried and determined were worth pursuing in the future, as well was what people did not like and what we should avoid doing in our application.

## 5.3 Future Work

Primarily, if we had more time, we would have performed a user study of our system as we were planning to. We want a user study of our system to be tested against the original tablet-based application mentioned earlier to see which is preferred, as well as to evaluate the usability and intuitivity of our system and its features alone. The experiment would be run in a large open and flat room with a Vicon system, six Khepera robots, and the Magic Leap headset and controller. To ensure that we kept the same functionality as the previous system, we would run the same tasks as the original system. These tasks include moving and rotating robots, team swapping, multiple selection, and box pushing. To test our new implemented features, we also would include tasks to move and rotate robots and boxes using gestures and voice commands.

Voice control is one of the important features that if we had more time we would have further expanded upon. Unfortunately, due to Magic Leap implementing voice recording late in our project and COVID-19 occuring, we were not able to test this on the Magic Leap device. However, we were able to prove that we could successfully turn an audio clip into a wav file, send that to Watson and have it returned in a reasonable amount of time, which would return several possibilities of what was said, as well as a confidence level. If a phrase is within a certain confidence interval, such as 90% or higher, that command can be considered accurate, and we could parse the returned text to deliver the command to ARGoS and have the action take place.

The Magic Leap would have been a much better device to work with if it had better support. As mentioned previously, not only was the API lacking information needed to implement features that they had mentioned were possible, but also emailing the support team would lead to a suggestion to post on the forum which is not used by many people, so you are unlikely to receive an answer. Because of this, it would be better in the future to continue this project with a more established augmented reality headset that will have more support from the community and the business that owns it, such as the HoloLens. Another addition that would make the Magic Leap a great product would be a larger field of view. However, this issue is primarily an issue for all augmented reality headsets today, as it would allow for a fully immersive experience that is not restricted by a small field of view.

# Appendix

## Appendix A: Application Setup

The application is a Unity application, and therefore must be built in Unity. With the project open in Unity, it can be built by going to File → Build Settings… Once in here, ensure that the platform is set to Lumin and that the scene in the build is LogScene. Then build the application and load it onto the Magic Leap as described in Appendix B.

## Appendix B: Magic Leap Setup

The Magic Leap comes with many different components, and must have our application loaded onto it in order to run it. In order to load the application onto it, it must be connected to the computer. To connect it to the computer, you can directly do it from the lightpack to computer by either USB-A or USB-C. If the Magic Leap needs to be charged, it should be connected to the computer as shown below.

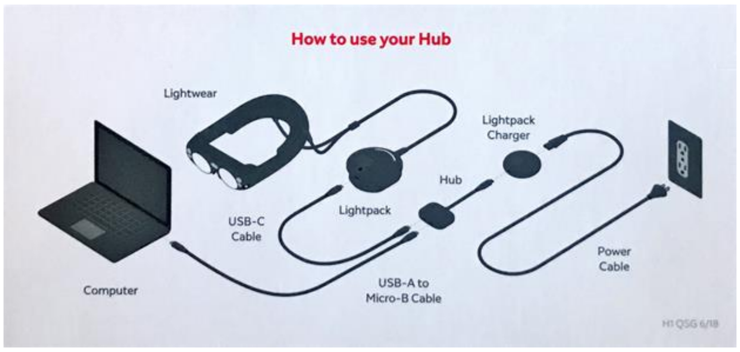


**Figure 13.** Magic Leap setup diagram[35].

Assuming the application is already built in a .mpk file, the user should be able to open the Magic Leap application called “The Lab” to access the Magic Leap through their computer. If the application is not yet built, Appendix A explains how to build it. The user then can go into the Device Bridge option and then select “Load” to load the application onto the Magic Leap. Once the application is loaded onto the Magic Leap, the user can just go to the home screen of the Magic Leap and it should be one of the icons. The user can click this icon, and the application will begin.

[35] 05, February. “Hub Computer Connection Troubleshooting.” *Care*, www.magicleap.care/hc/en-us/articles/360021672052-Hub-computer-connection-troubleshooting-.