Fall 2019

Gesture Based Control of Semi-Autonomous Vehicles

Brian Sanders
Old Dominion University, sanderb7@erau.edu

GESTURE BASED CONTROL OF SEMI-AUTONOMOUS VEHICLES

by

Brian Sanders

B.S. May 1985, University of Southern California
M.S. December 1987, University of Dayton
Ph.D. Dec 1993, Air Force Institute of Technology

A Thesis Submitted to the Faculty of Old Dominion
University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

MODELING AND SIMULATION

OLD DOMINION UNIVERSITY
December 2019

Approved by:

Yuzhong Shen (Director)

Rick McKenzie (Member)

Zhanping Liu (Member)

ABSTRACT

GESTURE BASED CONTROL OF SEMI-AUTONOMOUS VEHCLES

Brian Sanders
Old Dominion University, 2019
Director: Dr. Yuzhong Shen

The objective of this investigation is to explore the use of hand gestures to control semi-autonomous vehicles, such as quadcopters, using realistic, physics-based simulations. This involves identifying natural gestures to control basic functions of a vehicle, such as maneuvering and onboard equipment operation, and building simulations using the Unity game engine to investigate preferred use of those gestures. In addition to creating a realistic operating experience, human factors associated with limitations on physical hand motion and information management are also considered in the simulation development process. Testing with external participants using a recreational quadcopter simulation built in Unity was conducted to assess the suitability of the simulation and preferences between a joystick approach and the gesture-based approach. Initial feedback indicated that the simulation represented the actual vehicle performance well and that the joystick is preferred over the gesture-based approach. Improvements in the gesture-based control are documented as additional features in the simulation, such as basic maneuver training and additional vehicle positioning information, are added to assist the user to better learn the gesture-based interface and implementation of active control concepts to interpret and apply vehicle forces and torques. Tests were also conducted with an actual ground vehicle to investigate if knowledge and skill from the simulated environment transfers to a real-life scenario. To assess this, an immersive virtual reality (VR) simulation was built in Unity as a training environment to learn how to control a remote control

car using gestures. This was then followed by a control of the actual ground vehicle. Observations and participant feedback indicated that range of hand movement and hand positions transferred well to the actual demonstration. This illustrated that the VR simulation environment provides a suitable learning experience, and an environment from which to assess human performance; thus, also validating the observations from earlier tests. Overall results indicate that the gesture-based approach holds promise given the emergence of new technology, but additional work needs to be pursued. This includes algorithms to process gesture data to provide more stable and precise vehicle commands and training environments to familiarize users with this new interface concept.

This thesis is dedicated to the idea that
you are never too old to learn.

ACKNOWLEDGMENTS

As with most educational endeavors, the list of people a student owes thanks to is long. I would like to start by thanking Dr. Yuzhong Shen for the steady guidance and many conversations ranging from basic program planning to research concepts and execution. I have never been in a position to discuss programming at this advanced level, and Dr. Shen, as well as other faculty at ODU, have made that aspect of this pursuit highly enjoyable. I would also like to thank my colleagues at Embry-Riddle Aeronautical University (ERAU), to include Dr. Dennis Vincenzi and Dr. Sam Holley, for their continued interactions and involvement with the Human Factors (HF) aspect of this research, and Dr. Ken Witcher for helping me obtain the full financial support from ERAU to obtain this degree. Finally, I like to recognize Lucy Taylor for hanging in there with me as I did some silly stuff like taking multiple courses at a time and spending my weekends catching up on research.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Autonomous Systems

The current autonomous (or unmanned) systems industry, specifically the unmanned aerial system (UAS) market, has been experiencing significant growth in recent years due to maturation and advances in related technology, increased application opportunities, and the availability and affordability of key components and materials. This includes systems that range in size from those that can be held in the palm of a hand to larger, extremely capable military systems. This growth has been accompanied by a solidification of Federal Aviation Administration (FAA) rules and regulations governing legal use and requirements for use of unmanned vehicles within the National Airspace System (NAS) [1] [2] [3], which has enabled a wide range of uses and limitations. With these new systems and emerging technology come opportunities for advancement in command and control of UAS, as well as other autonomous vehicles, and potential support of future operational policy changes, such as the requirement to maintain an unenhanced line of site to the vehicle for recreational and business market applications[1].

1.2 Traditional Control Interfaces and Emerging Opportunities with New Technology

Drones (aka, unmanned aerial systems or UAS) are used for a variety of purposes including aerial videography, photography, and surveillance. Successful accomplishment of these tasks requires the execution of a series of basic maneuvering functions (i.e., take off,

[1] IEEE Transactions and Journals style is used in this thesis for formatting figures, tables, and references.

acceleration, point-to-point navigation) that, when combined, contribute to a mission capable system. Commercially available small unmanned aerial systems (sUAS) have traditionally been designed and controlled using legacy interface approaches to control the basic maneuvering functions of a remote vehicle. These traditional control interfaces are typically one dimensional (1D) or two dimensional (2D) devices that allow the user to interact with a system in a limited manner [4]. For example, keyboards are 1D input devices that allow for text input and activation of preprogrammed functions via a sequence of key/text inputs. Mice have expanded input capabilities into a 2D framework, but input is still limited to menu item selection or "hotspots" on a graphical user interface (GUI). Both of these control devices, while functional and useful, are limited in nature and not very intuitive in terms of control movement, input, and function, and they are often slow and time consuming as control through these devices often requires a series or sequence of inputs to achieve the desired end state.

Other legacy control devices, such as the joystick based one shown in Fig. 1 are better, but still an attempt to translate 2D input into movement through a three-dimensional (3D) space or environment. Integration with touch-sensitive devices such as phones and tablets are beginning to emerge on the market to replace or augment discrete physical controls and information displays [4]. However, these devices are, in many cases, simply electronic or digital versions of the same 2D legacy control devices. These devices typically combine electronic visual displays with touch input, and sometimes electronic input (GPS, accelerometers, and automation for example).

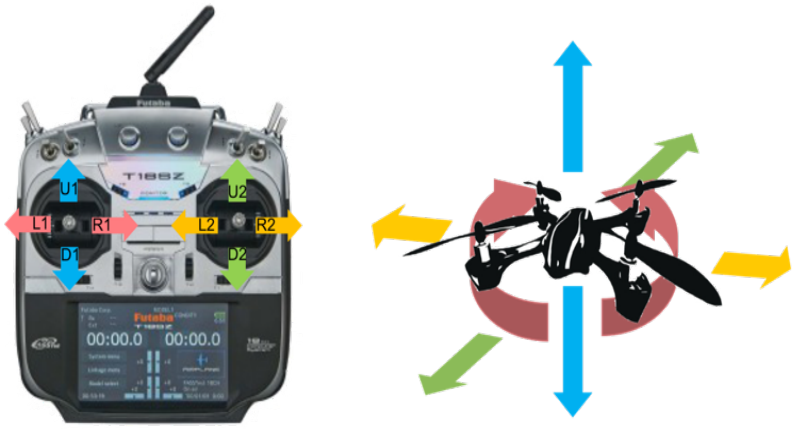

Fig. 1. Typical Legacy Controller and Movement Designations for SUAS Control [4].

An alternative to these traditional command and control approaches is via the use of gestures. Gestures and visual signals are common in military and aviation domains. A series of standard gestures (hand and arm signals) has been in used for many years to transmit information from one person to another [5]. Gestures are movements of human body parts - usually the arms, head, and hands - that provide contextual meaning [6]. Development of a gesture-based approach for sUAS operation may be a viable alternative for implementation into command and control interfaces using technology that is designed to recognize gestures.

A gesture-based approach can free the operator from having to hold and operate a multi-joystick, multi-button-based controller by correlating the UAS operations to a set of fluid, intuitive, natural, and accepted set of hand gestures. This in combination with emerging display configurations could be used to create systems in which the vehicle operator can observe the vehicle position, monitor its operations, stay abreast of vehicle performance parameters, and have the ability to control the sUAS. All of this could be accomplished without the cumbersome necessity of holding a cryptic control device. A simple gesture control interface, such as that possible with technologies such as the Leap Motion Controller, can make the task of piloting

much easier and more intuitive in nature [7]. This in combination with new visual displays can create an entirely new command and control structure.

1.3 Future Possibilities

A significant amount of literature has been written about the need to design better displays for operating UAS (see for examples, [8] [9]) but little effort has been expended to develop controls that are innovative, take advantage of new technology, and are natural and intuitive in design. Instead, sophisticated UAS have relied on legacy displays and controls, such as those mentioned above, and have paid little attention to new technologies that have recently become available for use. The gaming industry recognized the potential to create more robust visualizations and has concentrated on developing environments that move away from the standard 2D environments into richer and more robust 3D environments using 3D stereoscopic displays [4].

Virtual Reality (VR) displays or VR-like displays are now affordable and commonplace, and are regularly used as the display of choice when immersion into a 3D environment is preferred. High resolution displays on phones or Head Mounted Displays (HMDs) using phones have begun to replace and augment visual systems to develop environments that serve as displays for vehicle parameters as well as provide an egocentric view from the UAS camera. Thus, the capability exists for technology to provide more information with realistic visual perspectives similar to looking through a Heads Up Display (HUD) on a manned aircraft.

A vision for such a system can be developed based on Augmented Reality (AR) Systems made available through the use of HMDs. The creation of an AR display that integrates both the real world and computer-generated world into a display that uses relevant parts of both

environments can create a display that is more effective than either one by itself. The goal of this AR type display would be to integrate relevant portions of both the real world and virtual world to produce a display that is efficient, functional, user friendly, and (hopefully) intuitive in design. Utilization of these types of technologies, if designed correctly, can result in a more realistic visual display that provides the information needed for successful operation with minimal training requirements.

1.4 Project Vision, Challenges, and Objectives

New technology, in the form of augmented reality headsets, is emerging and accompanied by a suite of gesture-based systems that can enable a future change in operations, policies, and regulations. These head mounted display systems combined with gesture-based command control interfaces offer a new approach to vehicle control that is unencumbered by traditional handheld devices, and offer unique capabilities for mission planning and vehicle, and even multivehicle, control. The design of these systems will require careful investigation of human factors issues to populate gesture libraries that are natural and intuitive, as well as cognitive loading considerations due to the easy availability of a vast amount of visual information.

To date, studies examining the functionality and effectiveness of virtual interfaces using gestures as the primary method of interaction with a system have been scant. Design of new interfaces that take advantage of emerging technologies is required to complement and enhance the capability of the human component and the system in terms of performance. The above discussion highlights some of the opportunities and considerations for the control of autonomous vehicles using head mounted displays and gesture-based systems. The application investigated

in this study is gesture control of semi-autonomous vehicles. The study will develop the beginnings of a gesture library by conducting a task decomposition for control of a representative, recreational UAV and matching the task to the capability with gesture capture technology. The objectives are to (1) identify and design gestures that are natural and intuitive for incorporation into a gesture-based HMD for vehicle control, and (2) determine the feasibility of using simulation environments to develop verifiable solutions.

CHAPTER 2

BACKGROUND AND PREVIOUS WORK

This chapter sets the stage for this research project by reviewing relevant work and identifying the contribution of this investigation. It begins with a discussion of the readiness and availability of relevant technology to address the feasibility of the proposed effort. It then moves into a detailed review of related gesture-based research (the major research component), and the selected technology capability proposed for this approach. This is then followed by a review of some of gesture-based applications. While it is not the primary focus of this investigation, it is important to have an awareness of the trigger mechanisms that activate working memory. Thus, topics related to cognitive loading are cursorily reviewed since they can drive some design features of the simulation. The chapter ends with an assessment of the potential contribution of the proposed research.

2.1 Is It Science Fiction or Is It Real?

The concept of a reality-virtuality continuum was first introduced by Paul Milgram in 1994. In that paper, Milgram and Kishino (1994) [10] discussed the concept of a reality-virtuality continuum with the real world environment on one end of the continuum and a totally virtual computer generated environment on the other end of the continuum as shown. This image is shown in Fig. 2, which is now a classical and widely used model. Between the two ends of the continuum is a wide range of mixed reality variations between total real environment and total virtual environment. Most advanced interfaces today fall somewhere in the mixed reality section of the reality-virtuality continuum.

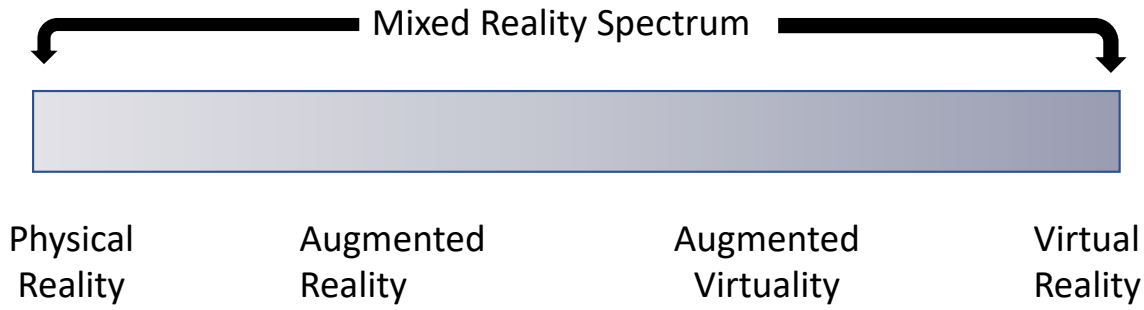

Fig. 2. Classical Illustration of Mixed Reality Spectrum.

Although Virtual Reality (VR) and Mixed Reality (MR) type displays have been available since the 1980s and 1990s, MR interfaces that include control components have not. Interactive, MR display and control interfaces have only recently appeared on the consumer market in a usable and affordable form. Typically, a combination of technology can be integrated and utilized to create an inclusive human-machine interface that can be used to both display information in a VR environment while designing a control interface which can be used to manipulate objects in the real or virtual worlds. This combination of technology provides the means to design a MR display and a VR control interface for use in a real or virtual world.

For purposes of this research the one-dimension spectrum view shown in Fig. 2 is expanded to three-dimensions (3D) as shown in in Fig. 3 to include the mixed reality spectrum, visual interface spectrum, and the interaction spectrum. Where the visual interface spectrum is defined to span the range of 3D images on a two-dimensional space (2D) space (aka, screen space), mixed reality in the form of handheld devices (such as tablets) and head mounted augmented reality (AR) devices to fully immersive VR headsets. The interaction spectrum includes human-machine interfaces such as the mouse, touch screen, handheld controllers, and

gestures. This 3D space is useful in describing the potential system capability from a given set of technology. For example, the HoloLens by Microsoft [11] could be represented on this diagram as a combination of gesture-based interface and an HMD to produce an AR capability. While a handheld tablet would fall in another region of touch and a flat screen display to produce a different AR experience. This chart can be populated with more technology examples, so this indicates the proposed research topic of this thesis is feasible. Albeit, the utility of it is still unanswered. Further, to fully answer the question requires investigations across each of the spectrums in Fig 3. This study focuses primarily on a component at the far end interaction spectrum, which can be considered to include gestures, and along the visual interface spectrum to include the screen space and VR headsets.

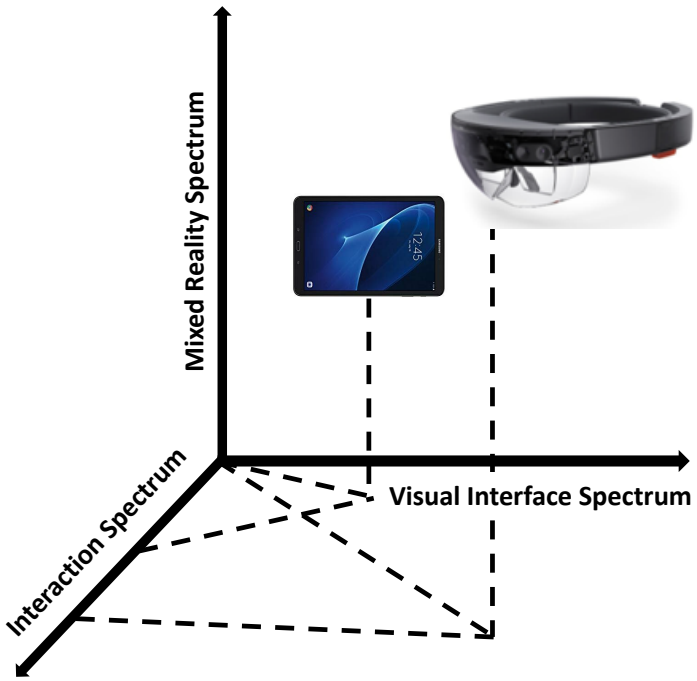

Figure 3. 3D Virtual Technology Spectrum.

2.2. Gestures and Gesture Capture Technology

As described by Hamilton et. al. [6], gestures are movements of human body parts - usually the arms, head, and hands - that provide contextual meaning. Gestures are used in several daily activities ranging from personal interactions to the military operations. For example, effective military operations depend on clear and accurate communication among ground units and supporting aviation units [12], and a standard set of visual signals have been defined for use in combat operations [5]. Limitations of visual signals include range and reliability which are dependent upon visibility, and this may affect the degree to which these visual signals are understood or misunderstood. The same is true in computer environments. The degree of recognition is dependent upon many factors including noise present in the environment, accuracy and consistency of the gesture, and resolution of the receptor.

Gesture-based control, as well as traditional control technology, pose a unique challenge to remote operations of unmanned vehicles. To begin with, the term "unmanned system" is a misnomer at this point in time; since there is a human operator present in the system, the system will always be "manned" in some way. The only difference in the case of unmanned systems is that the operator is not collocated with the vehicle. Thus, placing the operator in a unique position and providing a different operational perspective since many of the environmental cues normally present in manned scenarios are no longer present and available to the human operator.

Research has suggested that while separated from the vehicle gestures can help mentally connect with it. Cauchard et al. [13] investigated how to interact with flying robots (aka drones). They conducted a study to learn how to naturally interact with drones. In this investigation gestures were made by a participant even though actual vehicle control was achieved by a remote operator in a separate location. Results show strong agreement between participants for many

interaction techniques, such as when gesturing for the drone to stop. They discovered that people interact with drones as with a person or a pet, using interpersonal gestures, such as beckoning the drone closer. It should be noted that these where typically large motion gestures, and not the small ones proposed in this project. This suggests that given a suitable gesture library and capable technology an alternative command and control experience can be developed.

Some previous gesture related research centered around development of computer algorithms that would allow robotic systems to recognize gesture commands in the field as part of military teams. Other research has focused on virtual reality environments integrated with optical sensors to recognize and measure movement, velocity, and patterns of movement, of fingers and hands, and then translates those gestures into commands. Hamilton, et al. [6] conducted research that focused on developing the ability for robotic systems to understand military squad commands. The long-term goal was to develop the capability to integrate robots with ground forces as seamless teammates in combat operations. Their research focused on creating a recognition model that understands 12 squad-level commands, such as rally, listen, stop, and come here. The input into the model was collected using Microsoft Kinect's skeletal model and processed with a logistic regression activation function to identify the gesture. The logistic model showed an overall 97% effectiveness when discriminating if the datasets are from a given member set. The decision model was 90% effective in determining the gesture class a given dataset represents.

Lampton et al. [12] conducted investigations into using a gesture recognition system integrated with a virtual environment. Their goal was to measure the accuracy and effectiveness of a VR based gesture recognition system. The system consisted of two video cameras, software to track the positions of the gesturers hands, and software to recognize gestures by analyzing the

position and movement of the hands. The researchers selected 14 basic and accepted hand gestures commonly used in the field by U.S. Army personnel. In general, the results were mixed in terms of recognition and accuracy. Many of the gestures were problematic in terms of tracking, recognition, or both.

Recent advancements in hardware and software processing has resulted in large strides in the ability to capture gestures. As mentioned above the Microsoft Kinect's is one example. Another one is the Leap Motion Controller (LMC) [14]. It is a relatively recent technology that can capture and track hand motion with a sensor just slightly bigger than a standard USB flash drive. As is discussed in Chapter 3 it is the selected technology for this study, so it warrants a detailed discussion on previous work to support this decision.

Being a new technology, limited literature is available on the LMC performance. But a few studies have emerged over the last few years. For example, Weichert et al [15] evaluated the reported accuracy and repeatability of the LMC. A novel experimental setup was developed making use of an industrial robot with a reference pen allowing a position accuracy of 0.2 mm. A deviation between a known 3D position and the average LMC measured positions below 0.2 mm was obtained for static setups and of 1.2 mm for dynamic setups.

Guna et al. [16] investigated the performance of the LMC with a professional grade, high-precision, fast motion tracking system. A set of static and dynamic measurements were performed with different numbers of tracking objects and configurations. For the static measurements, a plastic arm model simulating a human arm was used and measurements were made at 37 reference locations covering the controller's sensory space, which is about the size of a standard beach ball. For the dynamic measurements, a special V-shaped tool, consisting of two tracking objects maintaining a constant distance between them, was created to simulate two

human fingers. In the static scenario, the standard deviation was less than 0.5 mm. The results of the dynamic scenario revealed inconsistent performance of the controller, with a significant drop in accuracy for samples taken more than 250 mm above the controller's surface. They conclude that due to its rather limited sensory space and inconsistent sampling frequency, in its current configuration it cannot currently be used as a professional tracking system. These two studies suggest that the LMC is a highly accurate system. While it may have some limitations of sensory space it is considered a good model technology to use for the current investigation.

In another validation effort Smeragliolo et al. [17] compared the LMC with an accepted markered motion capture technology. Their goal was to assess the use of the LMC for possible health care applications. Participants were instructed to perform three clinically relevant wrist (flexion/extension, radial/ulnar deviation) and fore arm (pronation/supination) movements, which were tracked with each technology and compared results by performing Pearson's correlation and root mean square error. Wrist flexion/extension and radial/ulnar deviation showed good overall agreement between the two approaches. However, when tracking forearm pronation/supination, there were serious inconsistencies in reported joint angles. Hand posture significantly influenced the quality of wrist deviation and forearm supination/pronation, but not wrist flexion/extension. They concluded the LMC is capable of providing data that are clinically meaningful for wrist flexion/extension, and perhaps wrist deviation, but not for measuring forearm pronation/supination. In additional to another validation of the LMC performance this investigation also provides meaningful insight into the range of physical motion applicable for gesture library development since it showed some natural limitations of hand motions.

Some studies have emerged addressing comparisons with a mouse. Bachman, Weichert and Rinkenauer [18] present a Fitts' law-based analysis of the user's performance in selection

tasks with the Leap Motion Controller compared with a standard mouse device. With an error rate of 7.8% for the LMC and 2.8% for the mouse device, movement times twice as large as for a mouse device and high overall effort ratings, the Leap Motion Controller's performance as an input device for everyday generic computer pointing tasks is rather limited, at least with regard to the selection recognition provided by the LMC. This suggests it is not suitable as a direct replacement for a mouse. However, as suggested by Wigdor and Wixon [19], new touch and gesture devices may require new interface designs. It appears that the LMC falls under that basic premise. Following this line of thought, Scicali and Bischof [20] argue that a 2D mouse is not very useful in a 3D environment and that the LMC may be a better fit. They developed several games to gauge user performance in different 3-D environments. They obtained excellent general information about several usable gestures and information feedback such as auditory and visual features that accompany desired gesture interaction with the virtual environment.

A few actual applications have been reported too. Staretu and Moldovan [21] used the LMC to control an anthropomorphic gripper with five fingers. Following the creation of the prototype, performance tests were conducted under real conditions to evaluate the recognition efficiency of the objects to be gripped and the efficiency of the command and control strategies for the gripping process. It was found that the command and control system, both in terms of capturing human hand gestures with the Leap Motion device and effective object gripping, is operational.

There has also been documented efforts to control drones with the LMC and multi-modal approaches. Sarkar et al. [7] used the LMC to control some basic motions of a UAV. They present the implementation of using the LMC to control an off the shelf quadcopter via simple human gestures. The drone was connected to a ground station via Wi-Fi and then the LMC was

connected to the ground station via USB port. The LMC recognized the hand gestures and relayed it on to the ground station. They wrote interface scripts in Python to interpret the hand gestures captured by the LMC and transmit them in order to control the motion of the drone. Some basic tests were accomplished to document the feasibility of the LMC based system to control the vehicle motion.

There have been reported efforts to extend the application of gesture-based control to include other modalities too. Chandarana et al. [22] explored a multimodal natural language interface that uses a combination of speech and gesture input modalities to build complex UAV flight paths by defining trajectory segment primitives. Gesture inputs (measured with the LMC) were used to define the general shape of a segment while speech inputs provide additional geometric information needed to fully characterize a trajectory segment. They observed that the interface was intuitive, but the gesture module was more difficult to learn than the speech module.

Fernandez et al. [23] implemented a multimodal control system based on a Graphical User Interface (GUI) and several Natural User Interface (NUI) concepts, along with computer vision techniques in a single software framework to control aerial drones operating in a GPS-denied environment. These strategies include speech, body position, hand gesture and visual marker interactions used to directly command tasks to the drone. The NUIs were based on devices like the LMC, microphones and small size monocular on-board cameras which are unnoticeable to the user. Users were able to select the most intuitive and effective type of interaction for their application. This and the other studies cited above highlight the possibilities of alternative command and control approaches with the emergence of new technology.

2.3 Cognitive Loading Considerations

An example of what is possible for a command and control system, and in particular methods to reduce cognitive loading, was discussed by Zollmann et al. [24]. They investigated the application of micro aerial vehicles (MAVs) equipped with high-resolution cameras to create aerial reconstructions of selected locations. They identified that a challenge is that automatic flight path planning and autonomous flying is often applied but so far cannot fully replace the human in the loop for supervising the flight on-site to assure that there are no collisions with obstacles. They went on to discuss that this workflow yields several issues in cognitive loading, such as the need to mentally transfer the aerial vehicle's position between 2D map positions and the physical environment, and the complicated depth perception of objects flying in the distance. They presented an AR supported navigation and flight planning of micro aerial vehicles by augmenting the user's view with relevant information for flight planning and live feedback for flight supervision. Additionally, they introduced depth hints supporting the user in understanding the spatial relationship of virtual waypoints in the physical world and investigated the effect of these visualization techniques on the spatial understanding. While this paper did not encompass the entire spectrum as described by Fernandez et al. [23] above it did highlight the possibilities of an AR system and specific challenges related to cognitive processing.

Zollman et al. [24] highlighted a few of the cognitive loading issues related to the design of an AR based command and control system. There are several more that need to be considered [25] [26]. For example, Dodd et al. [27] investigated touch screen capability in aircraft cockpits and stated that as elements and workload increase in number and complexity, increased cognitive loading will follow. For the current effort this will drive the number and complexity of gestures we expect a participant to initiate for controlling the vehicle. As the research progresses beyond

the flat screen additional factors come in to play. As AR capability is added, issues of switching views between the operator real-world view and a virtual framework need to be considered. Recent evidence indicates that very different brain processes are involved in comprehending meaning from these sources [28].

The above discussion highlights some of the complexities that can quickly emerge in a command and control system. So careful consideration must be given to the design of the simulation and real-life demonstration. The approach taken in this investigation is to minimize the load on the working memory. This will result in limiting the information transmitted to the user to include basic vehicle status (i.e., speed, altitude) and visual information to improve perception and vehicle component control. Taking this approach will keep the focus on the control aspect and suitability of the basic simulation environment. Finally, as described Chapter 4, the idea of gradually introducing control factors into the simulation, as suggested by Antoneko et al. [29], helped to improve user performance.

2.4 Contributions of This Work

The objective of developing new technology and new approaches to Human Machine Interfaces (HMIs) is to increase system efficiency and reduce cognitive workload on the individual operator. Improvements in communication capabilities coupled with development of new virtual-friendly environments have created the potential for a HMI that both takes advantage of technological innovations and encourages the development of new types of interfaces [4]. In order to be successful and readily accepted by UAS users in general, the new technology and HMIs must be easier to use, more intuitive in nature, and possibility provide additional capability. They must produce a combined system performance that is better than existing HMIs

in that workload is reduced, situation awareness is increased, and system safety and reliability is enhanced.

Developing this next generation of command and control systems will require codification of effects across a spectrum of technologies, such as that illustrated in Fig. 3, and human factors and limitations. This investigation will add to the somewhat limited literature on gesture-based control of drones by developing suitable gesture-based libraries and mechanisms (i.e., hand positions and virtual visual aids) suitable for command and control of semi-autonomous systems, and also the applicability of a commercially available game engine to develop simulations with interactive interfaces for which to observe human performance, use as a training aid, and communicate with the remote vehicle.

CHAPTER 3

RESEARCH OBJECTIVES AND APPROACH

This chapter discusses the basic strategy for executing this investigation. It begins with a description of the research goals and objectives, which is then followed by a general discussion for how they are achieved. This includes a description of the two-phase approach the study followed going from gesture identification to simulation development to a physical demonstration. Requirements that drove the simulation are discussed as well as the selected equipment. This includes the gesture capture technology, head mounted devices, and the simulation software. Finally, the test approach and procedures are described. Details of how these were implemented are discussed in Chapter 4.

3.1 Research Objectives and Challenges

There are two research objectives for this project. One is related to the human factors component while the second addresses the suitability of a simulated environment as an assessment and training tool. They are stated as follows:

1. Investigate the application of gesture-based control of semi-autonomous systems to identify capability, challenges, and limitations to assess the feasibility (can you do it) and viability (does it add value) of the approach.
2. Assess suitability of a simulation environment to (1) support assessment of human performance and interface preferences for vehicle control and (2) provide a training environment for transition to a real-world system.

Given these objectives the project challenge then is to design an investigative approach that enables the assessment of human performance to control a semi-autonomous vehicle based on hand gestures. The approach selected is a combination of simulation and real-life demonstrations.

3.2 Research Plan

Fig. 4 illustrates the two-phase approach taken to address the program challenge. Phase 1 involved simulation only. It centered around the idea of observing a user's ability to control a recreational quadcopter. The steps in this phase started with the identification of hand gestures to control the vehicle and the selection and accuracy validation of a gesture capture technology. With these fundamental building blocks in place the simulation development began and followed an evolutionary approach (akin to agile software development) where participants where brought in three times to exercise the simulation and provide feedback on the basic gesture-based concept and simulation features. Modifications and additions were made after each of these events. The objective of Phase 2 was to add validity to the findings from the pure simulated environment. In this phase a small ground vehicle was selected as the control model. This phase included a virtual reality simulation to train and familiarize the participant with the controls and vehicle performance. It was then followed by a physical demonstration of navigating an actual vehicle around a room with obstacles.

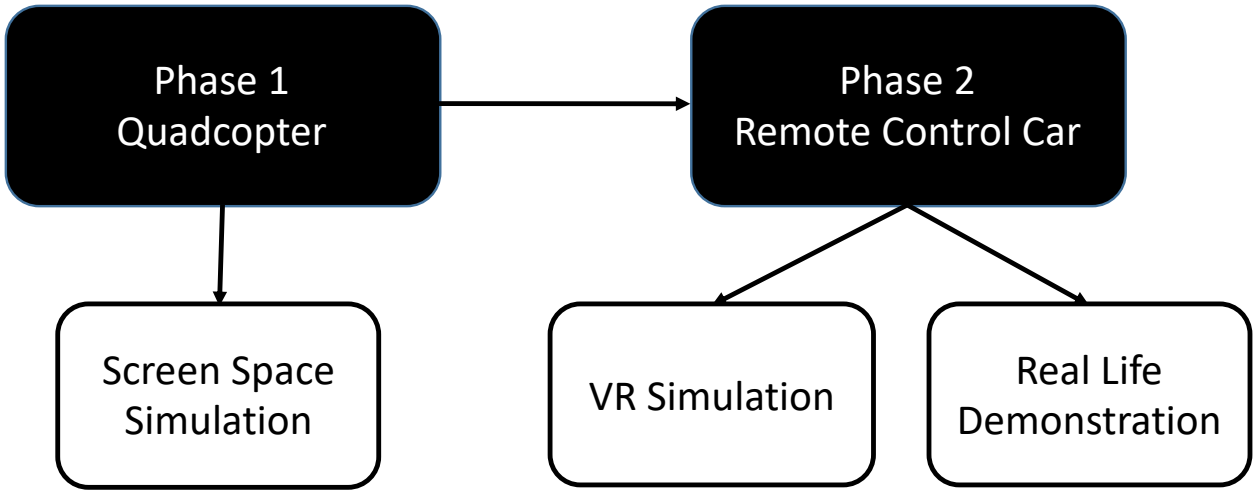

Fig. 4. Block Diagram of the Two-Phase Research Approach.

3.2 Simulation Development and Validation

As previously mentioned, the goal of the simulation was to provide an environment to observe user performance of gesture-based command and control. Before starting on creating the simulation, a gesture library that matches vehicle command and control requirements to common and natural gestures was developed. As shown in Fig. 5, the general idea was to find the "sweet spot" when considering natural gestures, the gesture capture technology capability, and the desired vehicle response.

For Phase I a typical recreational hovercraft, show in Fig. 6, was selected as the model vehicle for which to develop the command and control gestures. Key performance parameters of this vehicle are shown in TABLE 1. These were used as a guide for the simulated vehicle to approximate. This vehicle type was selected due to its simplicity but yet multifunction capability such as on-board camera. A task breakdown analysis was conducted to identify tasks associated with control of the vehicle and onboard equipment. These tasks were then matched to potential gestures that can be detected by selected gesture capture technology, which is described next.

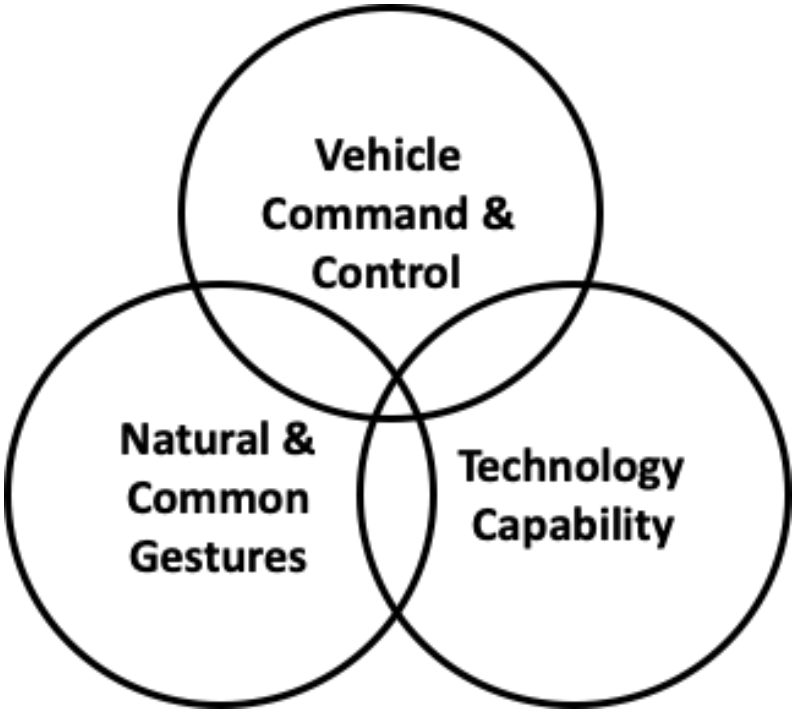

Fig. 5. Metrics for Gesture Selection.

Fig. 6. Quadcopter Vehicle Modeled in Simulation.

TABLE 1

KEY QUADCOPTER PERFORMANCE SPECIFICATIONS

Weight	1216 g
Characteristic Dimension (distance between propeller hubs)	25 cm
Max Vertical Speed (decent-accent)	3-5 m/s
Max Translational Speed	16 m/s
Max Angular Speed	150°/sec
Max Tilt Angle	35°

An important building block for this research was the selection of a gesture capture technology. There are at least three methods to do this such as gloves, handheld controllers, and touchless sensors. The Leap Motion Controller (shown in Fig. 7) was selected for this effort. At less than $100 it is an affordable device, and previous research by the investigator and others ([7] [30]) have demonstrated its potential for this investigation.

Fig. 7. Leap Motion Controller.

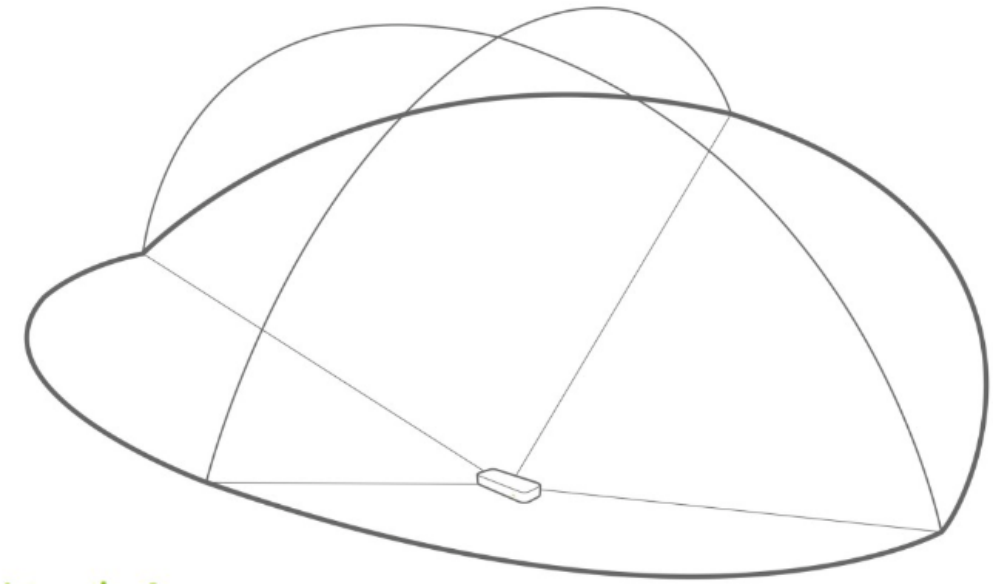

Fig. 8. Schematic of LMC Field of View [14].

As shown schematically in Fig. 8. , the LMC sensor has a field of view of about 150° and an effective range of approximately 25 to 600 millimeters from the sensor [LMC website], so the FOV is around the size of a standard beachball. The LMC is capable of tracking dynamic and static finger and hand positions with mm accuracy [6]. The LMC application programming interface (API) enables tracking of palm and finger position and orientation for each hand. Fig. 9 shows the normal and direction vectors associated with a hand defined in the API. Using these vectors, it is possible to track hand rotation (pitch, roll, and yaw) and other "touch-like" functions associated with finger movement. In the next chapter these capabilities are matched to potential vehicle control tasks and further investigated for accuracy of tracking these motions.

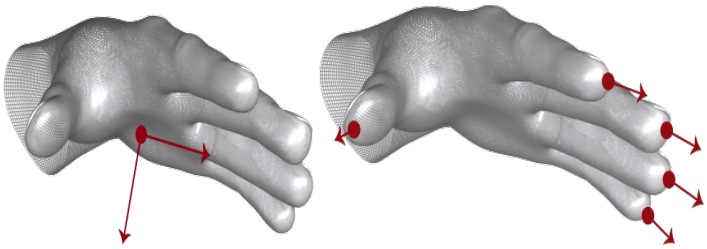

Fig. 9. Palm Normal and Direction Vectors and Finger Vectors [14].

3.3 Simulation Development and Validation of Findings

Simulations were built using Unity [31], which is a cross-platform game engine with the capability of simulating 3D rigid-body motion kinematically or via forces and moments. The engine has a C# based application programming interface for its scripting language. All development and testing were performed using an Alienware Aurora R7 workstation and a Dell Precision T3610 workstation. The general requirement for the simulations was to provide an environment where human performance of a gesture-based control system could be demonstrated and observed. It needed to have properly scaled features and include only the basic visual information related to vehicle control so as to minimize loading up working memory. Finally, and maybe most important, the gestures needed to be translated to force and moment-based commands on the vehicle.

Phase 1 simulation was designed around the idea of observing a user's ability to control a 3D model of the quadcopter shown in Fig. 6. Gestures were captured with the LMC. Some early testing also included control using an Xbox 360 Controller. The latter is a traditional technology and used to compare control approaches during the first round of participant testing. In both configurations, control commands were translated to force and moment vectors on the vehicle. During testing the simulation was displayed on a on a 55" in flat screen TV as shown in Fig. 10 below.

Fig. 10. Phase 1 Participation Test Configuration.

Fig. 11. Oculus Rift VR Headset.

As mentioned above the purpose of Phase 2 is to validate findings from the simulation tests conducted in Phase 1. This is accomplished by demonstrating that the skills developed in the simulated training environment translates to control of an actual vehicle. This was accomplished via a Virtual Reality simulation using the Oculus Rift head set in conjunction with the LMC. The simulation was of a laboratory room meant to emulate the environment where the actual vehicle would be controlled. In this case the control object was a remote-control car. This was selected as opposed to the air vehicle based on considerations of cost, accidents, process for gaining approval, and finding a suitable test location.

The model car selected for this application was an Adeept Smart Car Kit [32] shown in Fig 12. It consists of a rear-wheel drive, electric vehicle and a handheld controller. The communication subsystem is based on NRF24L01 2.4G Wireless communications module with a transmission range reported up to 250m [33]. The vehicle control system is based on Ardunio microcontrollers that are programmable using C++ scripts. Having access to these basic scripts enabled their manipulation to tailor it to the current research objectives.

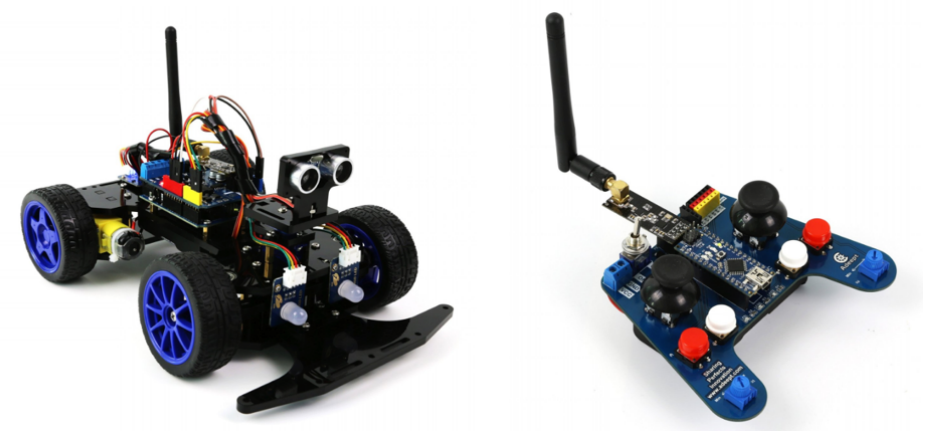

Fig. 12. Remote Control Car and Controller.

3.4 Test Methodology and Simulation Evolution

To achieve the research goals, it is desirable to go beyond the developers self-testing and obtain data from multiple participants once a design iteration is completed. This adds validity to the solution and helps to uncover previous unforeseen implementation issues as well as new ideas. The simulation development approximated an agile development model cycle as shown in Fig. 13. After each round of tests, the simulation was modified, and in some cases, basic requirements added, such as including rigorous simulation training modules.

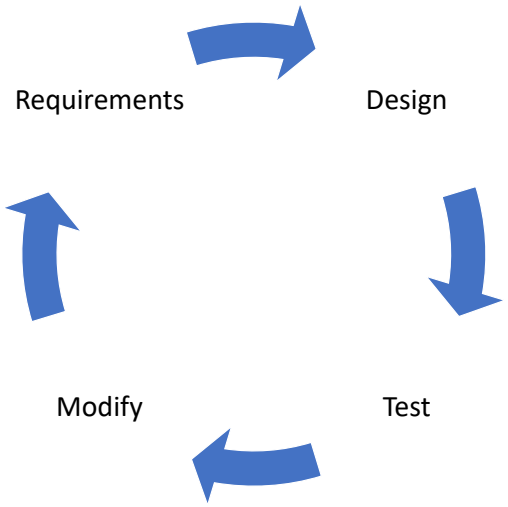

Fig. 13. Simulation Development Cycle.

In the first round of tests participants spent time in a play environment and a mission environment. The play environment was meant to allow the participant to become familiar with the basic controls and sensitivity of the vehicle in an unconstrainted setting. In the play environment there were no assigned tasks. Rather the user would just navigate the vehicle around the simulation or actual environment as they so desired. This was then followed by a

more directed series of tasks such as getting to a position, finding targets, changing views, etc. As will be discussed in the next chapter, the approach of starting in an unconstrained but complex play environment led to low vehicle control performance by the participants. In an attempt to improve this performance a training sequence was added to the simulation in follow-on tests. It was designed to gradually familiarize participants with vehicle control gestures starting with limited degrees (i.e., just vertical motion).

This approach provided insight into how well a user could control the vehicle without being overly burdensome by asking them to fly specific flight paths. The flying of precision flight paths will be reserved for future investigations. The quantitative data gathered were the amount of time spent in play and the amount of time to complete the assigned task in the operational environment. Qualitative data gathering included post-test interviews using the questionnaires are shown in the Appendix A. Each one consisted of a series of questions measured on a Likert scale and constructed to uncover the perceived suitability and advantages of the gesture-based control approach, as well as other simulation features and gesture commands. They are designed around the idea to first ask a top-level assessment question, such as which system did you prefer. Then follow-up questions helped to explore why this choice was made. For example, it asked questions about which system was easier to control or more intuitive to use.

3.5 Summary

This chapter described the overall research objectives and strategy for conducting this investigation. It discussed two research objectives that addressed the feasibility and viability of the gesture-based control approach as well as assessing the value of using simulation as an

investigative and training tool. Basic requirements for the simulations were described as well as the equipment and software used to develop the test environments. Finally, the cyclic approach to test and evaluation was described. The next two chapters take each of these topics into more detail by discussing the implementation of each phase.

CHAPTER 4

PHASE 1 IMPLEMENTATION, RESULTS AND DISCUSSION

This chapter describes details of the development and execution of the quadcopter simulation and how it evolved as a result of testing and feedback from participants. Basic command and control actions for a representative quadcopter are first identified and then matched to potential gestures measurable with the Leap Motion Controller (LMC). Next, details of the quadcopter simulation are described. This is followed by a series of two tests. In the first test, participants compared control preferences between joystick control and gesture-based control. Lessons learned from this event are described as well as modifications implemented in the simulation to improve user performance. Then a second set of tests was conducted and the results are presented and discussed. Knowledge gained from this development and testing formed the building blocks of the Phase 2 activities that included a VR simulation and a real-life control scenario, which are discussed in the next chapter.

4.1 UAV Control Task Breakdown and Gesture Matching

The first step in the task breakdown and gesture matching is to identify the functions associated with flying and operating the representative recreational hovercraft (aka., quadcopter) shown in Fig. 6 from Chapter 3. These are then matched to the LMC capability. This task breakdown is shown in the first column of TABLE 2. It is partitioned into categories of flight control and camera control. There are 7 potential actions in the flight control category related to the movement of the vehicle in the airspace. While the camera actions refer to the view (i.e., a first-person view from the operator or vehicle) and direction (pitch and yaw). The description of

the flight control is an abstraction and describes what the operator wants to make happen rather than how the vehicle does it. For example, the desired action is for the vehicle to climb or descend, translate in a horizontal plane, or yaw around its vertical axis. This motion is enabled through the application forces and moments on the vehicle. Those forces and moments in turn are determined by the internal control logic of the air vehicle, or in this investigation a C# script, and are transparent to the operator.

TABLE 2

QUADCOPTER CONTROL ACTIONS AND CORRESPONDING GESTURES

Vehicle Action	**Gesture**
Flight Control	
Climb/Descend	Left Hand Pitch
Translate Left/Right	Right Hand Roll
Translate Forward/Aft	Right Hand Pitch
Yaw	Left Hand Yaw or Roll
Increase/Decrease Speed	Controlled by Vehicle Pitch and Roll
Stop	Make a Fist or Remove Hands from Control Environment
Control Initiation	Open Hand
Camera Control	
Switch View	Tapping Motion
Pitch	Right and Left Index Finger
Yaw	Left Hand Yaw or Roll

These desired actions were then matched to potential gestures. The three basic hand motions of pitch (flexion/extension), yaw (radial/ulnar deviation), and roll (pronation/supination) are shown in Fig. 14. Gesture selection was based on consideration of natural association and common association. For example, pitching the hand up and down or rolling it left and right is a natural hand motion tied to control of those vehicle flight maneuvers. On the other hand, common gestures are defined as those accomplished on most touch screen interfaces such as increasing and decreasing the distance between the thumb and forefinger to represent zoom-in and zoom out actions. A comfortable range of motion needs be considered too and is discussed later in the chapter as well as natural neutral positions. The natural neutral position is defined as the position most comfortable to the user in terms of applying minimal muscle stress yet can also achieve the desired range of motion for vehicle control.

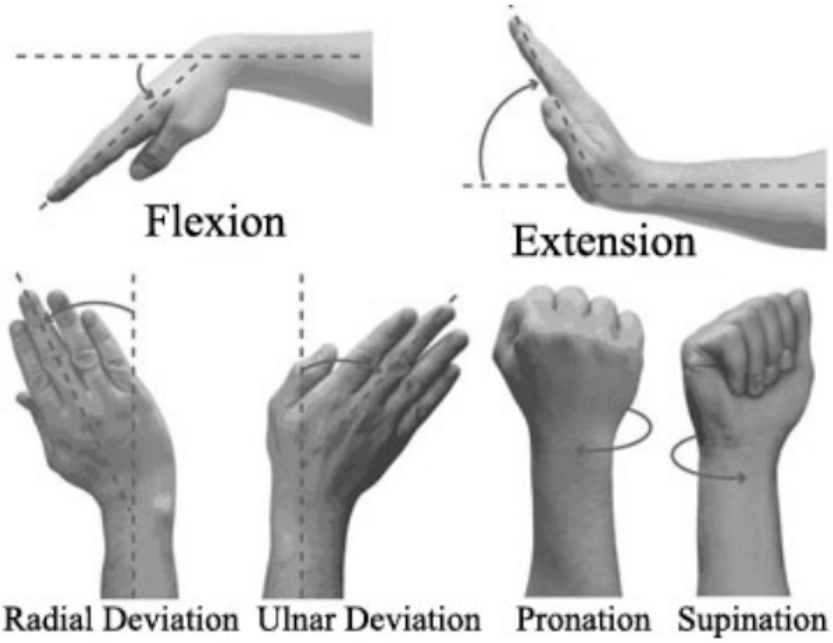

Fig. 14. Hand Motions.

While not strictly required, it was decided to align these gestures with the left and right hand as typically applied in multi-joystick controllers and feedback from helicopter pilots that participated in early demonstrations. They used two hands to control similar degrees of freedom in helicopters. For example, the left hand is used for altitude control, so that pitching of the left-hand palm controls that motion. Originally, yaw was controlled with a roll of the left-hand palm. This was selected since gestures would then be limited to pitch and roll motions only resulting in less gestures to remember. However, as will be described later, based on user feedback this was changed to yaw of the left palm, which is more naturally associated with the yawing motion of the vehicle. Left and right translation of the vehicle is controlled by a rotation of the right hand while forward and back motion is controlled by a pitching motion of the right hand.

There were two methods explored for initiating and stopping the vehicle control commands. As will be shown later one was to place the hands in a region of the field of view (FOV) identified with the assistance of a visual reference. The second involved making a fist. This latter approach did not limit where the user placed the hands in the FOV. While in this position the vehicle did not respond to any motion of the hand. Once the hand was unclenched the vehicle would response to hand gestures. As will be discussed later this made for an overly sensitive system since it would respond as soon as the hand was unclenched. To overcome this a dead zone was included in Phase 2 so as to enable some freedom of motion in the FOV without activating a vehicle response. For the camera view control a tapping motion was implemented. The tapping motion can be associated with selecting a target, such as a button, that will initiate some action. As will be described later there were two methods implemented based on a tapping motion. One was a dynamic user interface while the second was a single touch-like motion. The rotation of the camera view was achieved by yawing the vehicle.

Once the gesture commands were identified it was of interest to assess the reliability and fidelity of the LMC to capture these gestures. Weichert et al. [15] reported the LMC was able of capturing hand motions with sub-millimeter accuracy. They used a mechanical hand setup to measure this. It was desired to build on this and assess how smoothly the human performed gestures shown in Fig. 14 are captured by the LMC. This was achieved through the use of a C# program to capture the motion and then analyzing the data related to hand motion and gestures about the coordinate system, shown in Fig. 15, of the LMC. Rotations of the hand are measured by a roll around the z-axis, pitch around the x-axis, and yaw about the y-axis. Fig. 16 through Fig. 18 show the angle vs sample number captured by the LMC for each of the three motions of interest. These were produced by performing the gestures with the right hand at a natural speed so as not be excessively slow or fast.

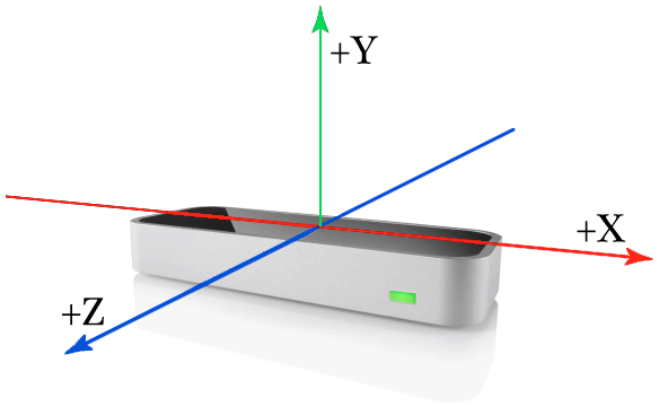

Fig. 15. Leap Motion and Coordinate System [14] .

From these figures it can be observed that the LMC captured the gestures with a high degree of fidelity. The slopes variations are a result of minor changes in rotational speed of the hand, indicating again the highly accurate nature of this sensor. This exercise demonstrated the precise

results produced by the LMC algorithms used to processes the captured images. As will be discussed in the next chapter filtering methods are explored to smooth out and dampen the commands to the vehicle to account for these observed human performance variations. Otherwise it can be an overly responsive system producing unstable results (i.e., hard to control the vehicle position). This as well as a dead zone were implemented in Phase 2 and are discussed in more detail in the next chapter. One final point to observe is the maximum and minimum values in each graph. They were on the order of ±30°. While not specifically targeted as a data point, this came about as a natural, ergonomic limits based on feel and roughly correlates to the findings of Smeragliuolo et al [17]. These assessment results and observations added validity to the selection of the LMC to produce the desired result and provided the baseline for how to define the range of motions to capture and build the control scheme in the simulation.

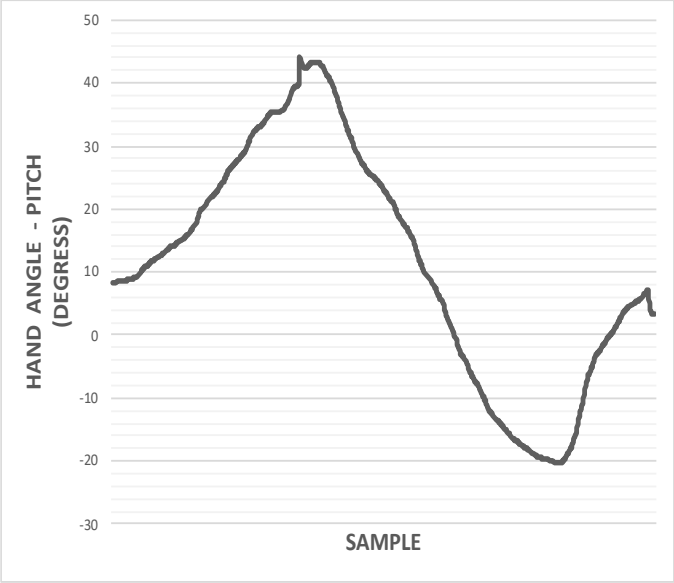

Fig. 16. LMC Pitch Gesture Tracking.

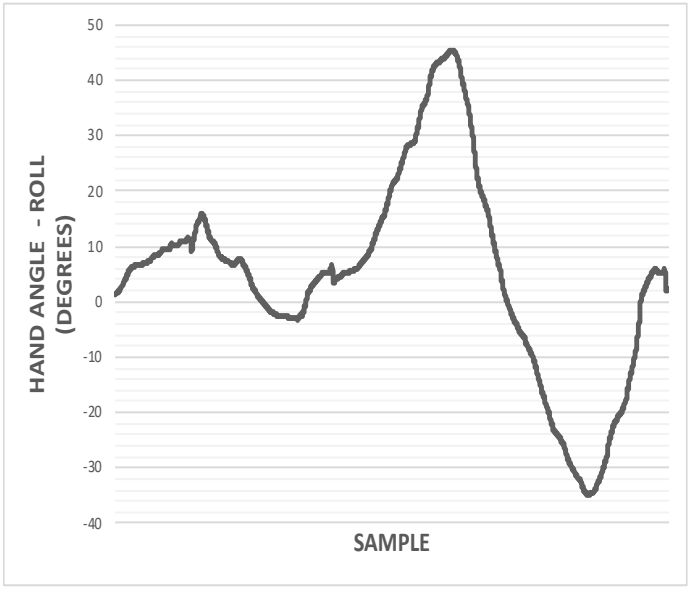

Fig. 17. LMC Roll Gesture Tracking.

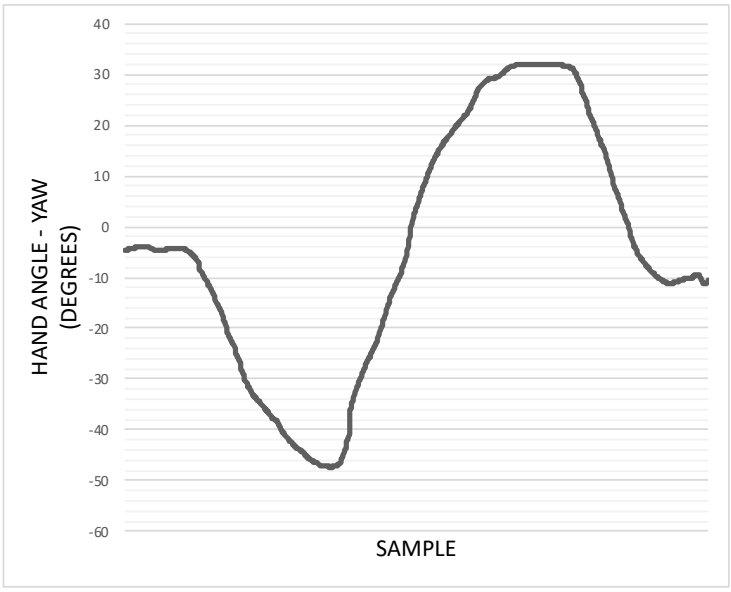

Fig. 18. LMC Yaw Gesture Tracking.

So far, only the dynamic hand motion has been addressed, but the LMC is capable of tracking each individual finger too. This in combination with Unity's collider feature can result in an effective button interface design, which as discussed above is the primary technique

identified to switch the camera view. This can be using either a static menu, button interface that is always present, or a dynamic menu where a particular motion or position of the hand or one of its digits triggers the display of a menu. As reported by Sanders et al. [30], the effective design of either of these approaches is based on the individual button size and arrangement in 3D space. For example, Fig. 19 shown below depicts a test scenario where users were asked to press a button based on a random prompt from the computer. This was presented as a verbal instruction as well as a visually as shown in the figure.

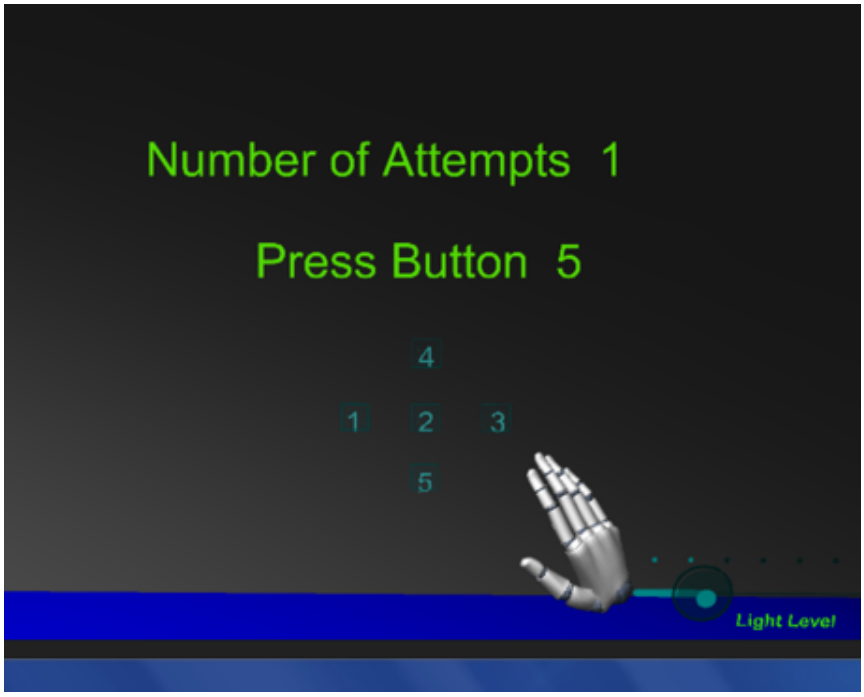

Fig. 19. Representative Button Interface Configuration Test.

It was found that when using the LMC the arrangement into the plane of the screen was as important as the spacing on the plane of the screen. The most effective configuration was an inverted stairstep setup such as when buttons higher in a vertical arrangement (i.e., button 4) were on a plane closest to user while the lower ones were deeper into the screen plane. This was

due to other parts of the virtual hand inadvertently coming into contact with those targets. Note, another approach is to designate that this inadvertent contact be avoided by indicating that only the index finger triggers the response. However, it was thought that this unnecessarily limits the interaction. TABLE 3 shows some of the measurements from this experiment. The first column is the square button side dimension, the second column is the space in between the buttons and the last column is the measured accuracy of the five participants. This result may improve once users become more familiar with the interface, but for these early stages it is recommended to keep the button side dimension larger 5 cm and/or the spacing no less than 10 cm.

TABLE 3

RESULTS OF BUTTON ACCURACY TESTS

Button Side Dimension (cm)	**Button Separation (cm)**	**Accuracy (%)**
5	10	81
5	5	68
2.5	5	71
2.5	2.5	65
1.5	3	74

4.2 Quadcopter Simulation

Discussion of the simulation begins with a description of the visual component of the virtual environment (VE) developed for testing. This includes the basic setting, information displayed, and vehicle control mechanisms. It then subsequently explores details of key

components that make the simulation functional. For example, details for how the rigid body feature of Unity is applied to model the forces and torques on the vehicle are described. Then two of the fourteen C# scripts developed to enable the simulation are explored. The first is the Gesture capture script while the second is the UAV Controller script. The former captures the gestures from detected by the LMC while the second interprets these data to provide command and control of the drone.

A representative view of the initial VE design is shown in Fig. 20. Basic features were guided by (1) suggestions from two recreational drone pilots to include comparative items to help with scale and distance measurements and basic vehicle data such as altitude, and (2) minimize cognitive workload. It is of a generic rolling hills environment that contains some natural environment features such as trees, a lake, and several manmade features such as a jeep, tents, and recreational vehicles. These features helped to provide depth and scale perception. They also are easily describable and identifiable targets for participants to find and navigate the vehicle to (i.e., fly over the jeep).

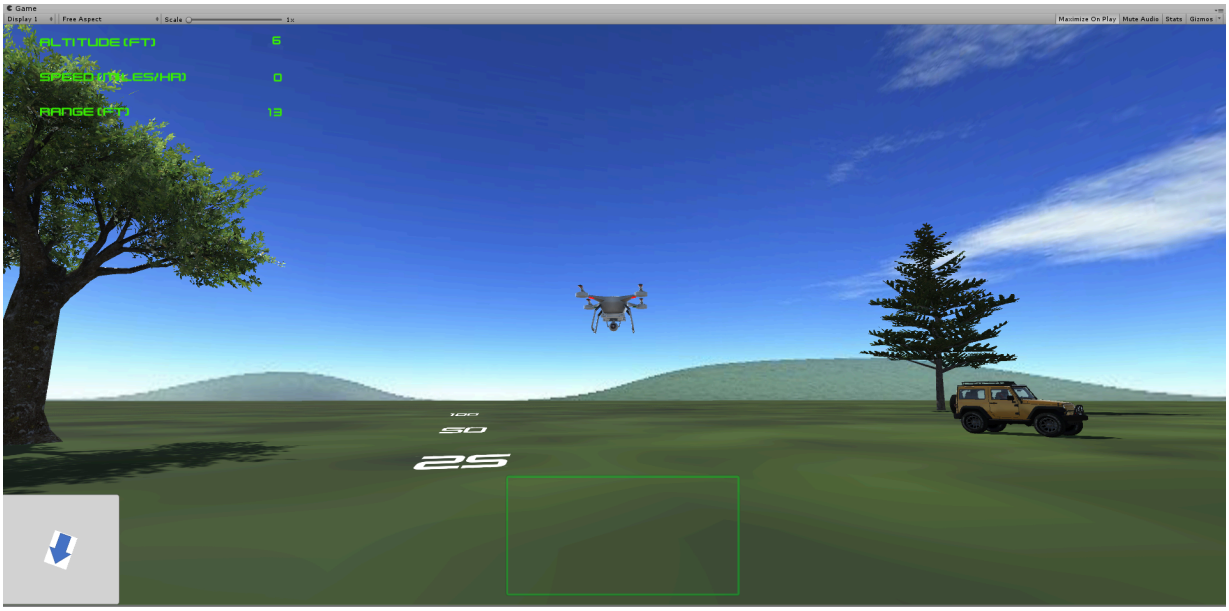

Fig. 20. Initial Design of Virtual Environment.

As discussed in Chapter 3, the drone is a generic representation of a recreational quadcopter. It models a 1kg drone with nominal dimensions of 30 cm × 30 cm × 10 cm and has red lights indicating the forward part of the drone and blinking green lights in the rear of the unit. The arrow in the left-hand corner serves as an orientation aid for users to determine the vehicle direction when it is too far away to clearly distinguish the lights. This is best understood by rotating the arrow 90 degrees so it is on a parallel plane with the vehicle. For the case shown in the figure the arrow indicates it is coming at the user from the right. The vehicle information displayed is altitude, speed, and range to vehicle and is shown in top left of the figure. An alternative concept for the vehicle data was to have it follow the vehicle in a fixed position as shown in Fig. 21. However, it would tax working memory unnecessarily and so not selected for the final design since this was not a focus of the research at this point in time. Therefore, the side position was decided so the user could quickly glance at the data when needed.

Fig. 21. Alternative Vehicle Data Display Concept.

It was desired to provide a mechanism for when a user's hands could be within the field of view of the LMC but the vehicle would not respond to commands. The green box is meant to provide this target region. Inside this region the vehicle will interpret hand motions as flight control commands but ignore those basic commands outside of it. Outside of this area other commands can be given, such as turning the vehicle camera on and off, which will be discussed next.

For the first round of tests, control commands were given by either a traditional joystick/button controller, or gestures captured by a leap motion controller. The joystick/button controller is shown in Fig. 22. It is a typical Xbox 360 controller. The left stick controls the attitude and yaw of vehicle, and the right joystick controls the fore/aft and left/right translation, respectively. Note that fore/aft and left/right translation is in reference to direction of the lights

with the red light being forward. Button A changes the perspective from the operator to a view from the drone camera, and the B button resets the vehicle to the starting position. The assigned hand gestures together with the virtual hands selected for this test are shown in Fig. 23. Multiple hand models come with the LMC. These range from basic hand models shown in the figure below to robotic and humanoid looking hands.

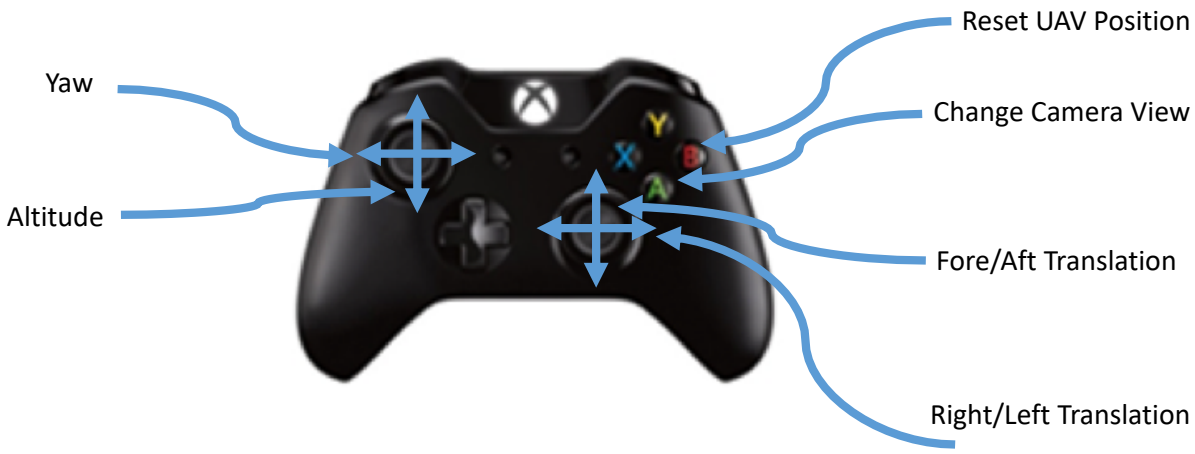

Fig. 22. Xbox 360 Controller Showing Mapping to Vehicle Actions.

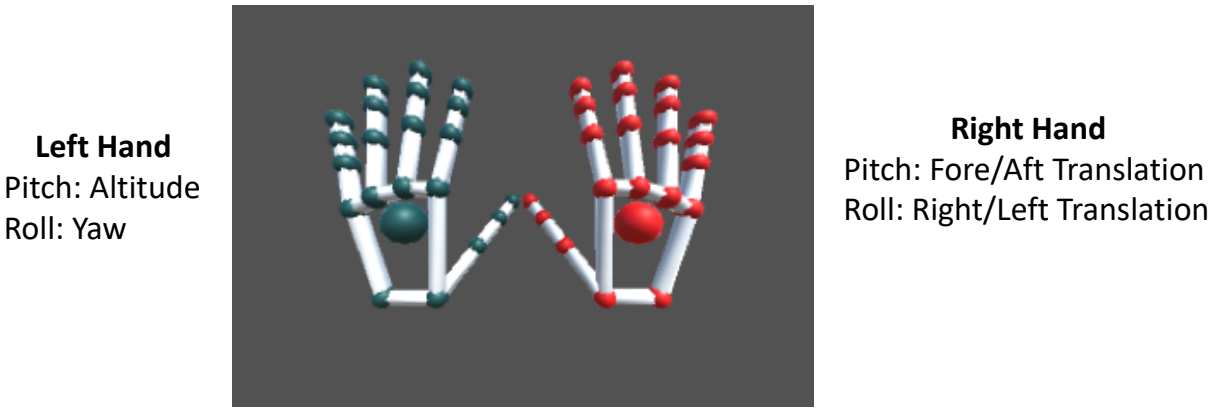

Fig. 23. Leap Motion Virtual Hands and Mapping to Flight Commands.

A dynamic UI was used to switch the camera view using gestures. It is a capability available in the Orion Version of the LMC API [14]. In this case a dynamic UI is attached to the left hand and is visible when that hand is rotated toward the user as shown in Fig. 24 below. It contains two buttons to enable the user to switch the view between the operator or vehicle camera. This type of dynamic UI is an attractive feature for the proposed system. It has the potential to lower can working memory load since it is not always in the field of view.

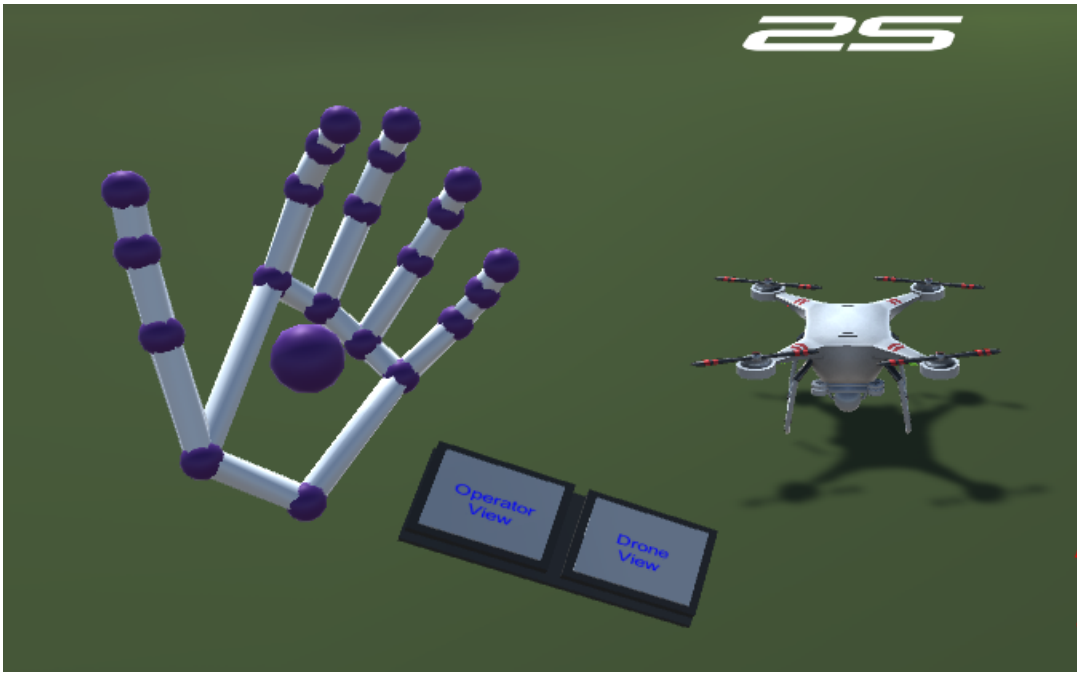

Fig. 24. Leap Motion Dynamic UI for Controlling the Camera View.

Now that the visual component of the VE has been described let us dive into some of the mechanics that made it work starting with a discussion of how the vehicle motion was controlled. There are two approaches to determine how the vehicle responds to hand gestures. One is through the control of basic kinematics and orientation. For example, if the user's hand rolled right the vehicle could be commanded to match that hand angle and be assigned a speed based on

a given mathematical model ranging from a pure linear relationship to some higher order function. However, that is not how the vehicle operates. Unity also provides a physics engine to apply forces and torques to an object via its Rigidbody class, which controls the object's linear motion via forces and angular motion via torques. This was the approach selected for this effort.

Fig. 25 shows two free body diagrams of the model vehicle. The top one shows the four forces produced by each propeller. By adjusting individual propeller forces a force-torque combination will be applied to the actual vehicle to produce the desired flight behavior. For this simulation the vehicle was modeled with a rigidbody component attached to it. This enables the application of a single 3D force vector and a single 3D torque vector to the vehicle. They can be applied in either the world or local coordinate system in Unity. In this case they are applied with respect to the local coordinate system. For the simulated vehicle the four propeller forces are then modeled as single force in y-direction relative to the orientation of the vehicle (i.e., perpendicular) and a single torque vector as shown in lower freebody diagram in Fig. 26, where $\dot{H}_G$ is the rate of change of angular momentum (i.e., sum of the moments). Maximum forces and moments applied to the vehicle were adjusted so that the simulated vehicle performance closely approximated that of the real vehicle as documented in TABLE 1. To achieve this the maximum allowable applied force was set to 20 Newtons, which is equivalent to 2 times the weight of the vehicle, and the maximum torques were set to 1 Newton-meter. The Rigidbody class also contains properties of drag and angular drag that are calculated using recursive algorithms such as that shown here:

$$V_{new} = (V + a * dt)(1 - C_d dt) \tag{1}$$

where V is the current velocity, a is the acceleration or force/mass, dt is the Unity time interval and C_d is the coefficient of drag. The latter is set by the user.

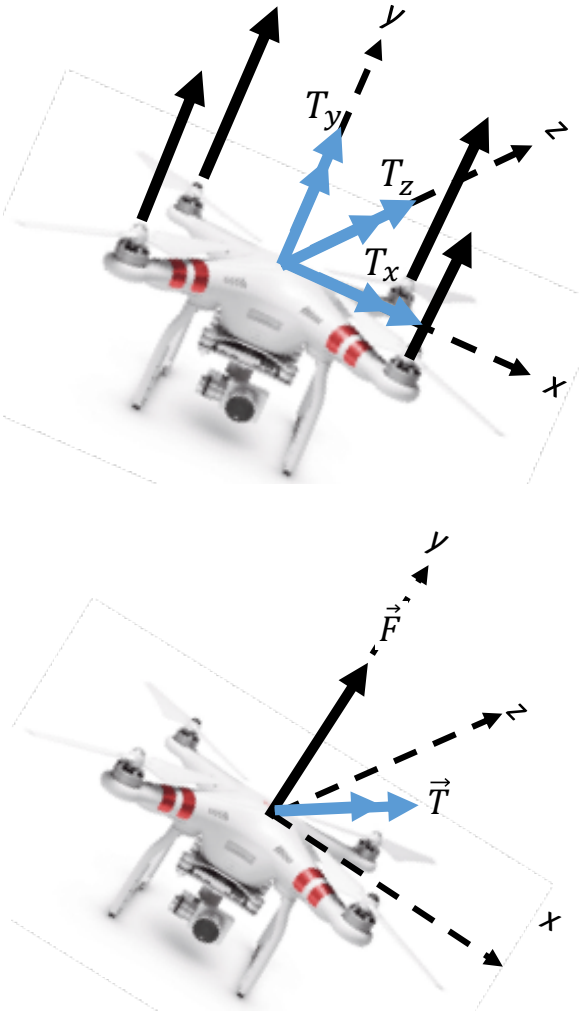

Fig. 25. Schematic of Forces on Actual Vehicle (top) as Modeled in the Simulation (bottom).

The shape of this curve is dependent on the drag coefficient. Fig. 26 show representative velocity calculations for $a = f/m = 20\ m/s^2$, $dt = 0.2\ sec$, *and* $C_d = 1.0$ *and* $C_d = 0.75$. It can be observed that while the maximum values changed the time it took to reach them was about the same. A similar concept is applied to compute the angular drag. These coefficients of drag were

adjusted to approximate the flight speeds of the actual vehicle. For example, maximum vertical speed of the vehicle in the simulation is 15 m/s while that of the actual vehicle is reported to be 16 m/s.

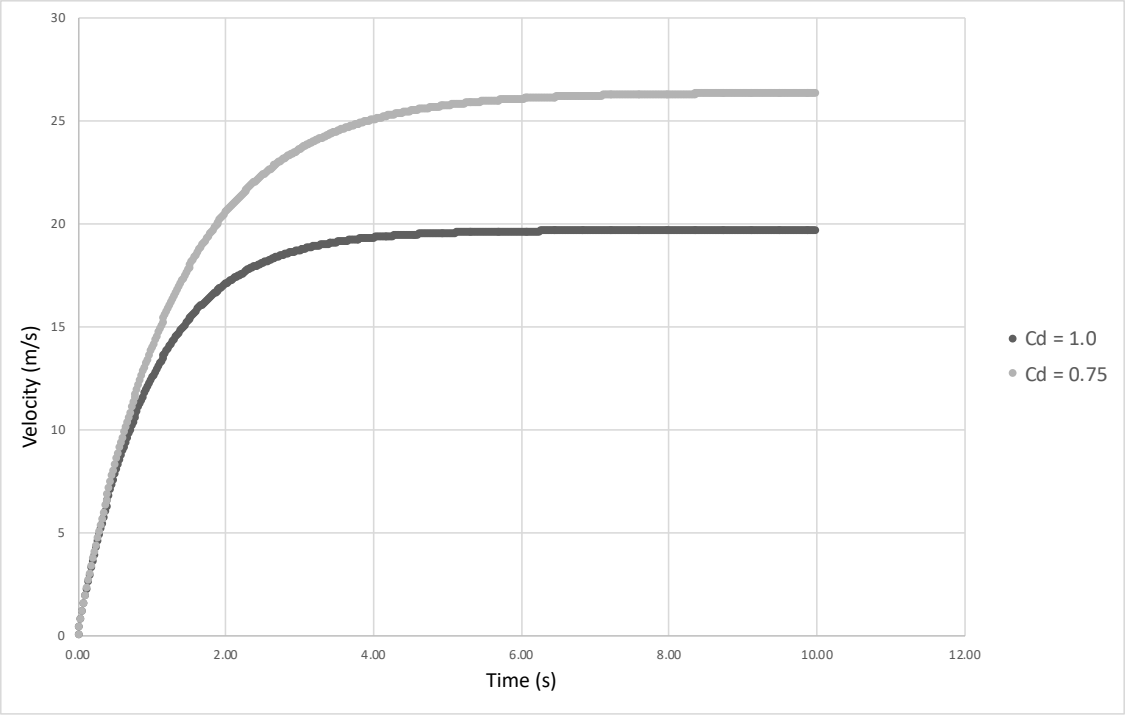

Fig. 26. Representative Velocity Calculation Using Recursive Drag Algorithm.

There were fourteen individual scripts written to enable the functionally of the simulation. This includes scripts to display visual features such as tracking and displaying the vehicle information to capturing gestures and controlling the vehicle. The list below provides the title of each script along with a brief description. Two of these, the UAVController and GestureListener scripts, are expanded upon in more detail since they are the most complex and basic to the primary objective of the simulation.

- Primary Simulation Functions
    - UAVController – controls flight behavior of the drone
    - GestureListener – captures gestures from the LMC
    - PIDController – determines multiplier for a given error
    - FollowPlayer – tracks drone from the operator's position
- Drone Functions
    - CameraController – changes the camera view and orientation of drone camera
    - BlinkingLight – blinks the red drone navigation lights
- Information Display
    - DirectionalArrow – controls the orientation of the navigation indicator
    - DronePerformance – displays drone altitude, speed, and range on the canvas

Fig. 27 shows a class diagram for the GestureListener script. This diagram represents the final development that will be discussed over the reminder of this chapter. The function of the GestureListener Class was to initialize the LMC controller and gather hand status and position data. It contained nine private fields. Each publicly accessible through the use of properties. There are four three dimensional vectors to track the direction and palm orientations for each hand. Five boolean datatypes were defined too. One to track if any hands were in a frame, one to determine if it was present in the frame, and one for each hand to determine if the hand was making a fist. This last feature was determined by the number of fingers the LMC detected. For example, if less than three fingers were detected then the GestureListener would set the boolean variable to true since this was the condition to identify a fist.

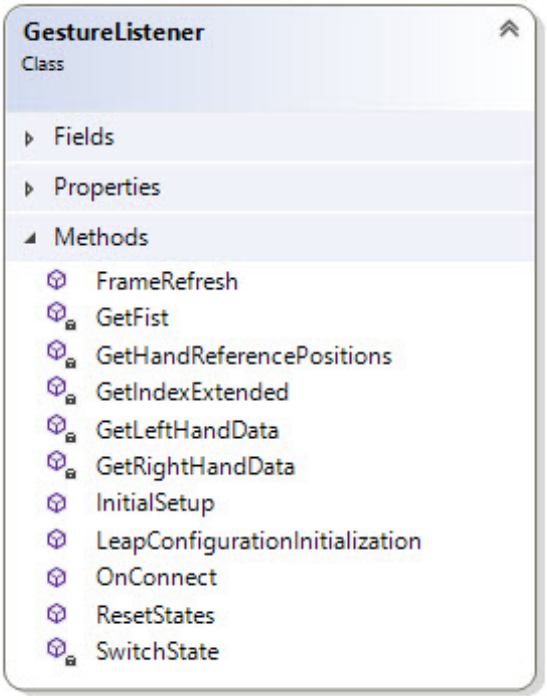

Fig. 27. GestureListener Class Diagram.

There are eleven methods contained in the GestureListener script. Three to initialize the LMC (InitialStepup, LeapConfigurationInitialization, OnConnect) and the remaining to capture the orientation of the hands and process gestures, such as fist and pinching motions. Early iterations of the simulation implemented the initialization methods in additional to the following methods: FrameRefresh, GetFist, GetIndexExtended, GetLeftHandData, GetRightHandData. In each frame the controller determines if a hand is present and if so which one or both. Then orientations of the hands, positions, and gestures are updated. The three remaining three methods (ResetStates, SwitchState, GetHandReferencePositions) are discuss in more detail in a later section of this chapter.

Flight control of the drone was accomplished by the UAVController class. Fig. 28 shows a class diagram for which the LMC is implemented. A class diagram for the joystick setup is similar with the exception of methods to capture the hand motion. There are thirteen methods in this class. Most of the key processing for flight control is done in the VerticalForce, Pitch, Roll, and Yaw methods. In these method forces and torques were determined and applied to the drone based on input from the joystick or the gestures retrieved from the GestureListener Class. Updates to the hand positions (GetHandUpates) are done every frame via the Update method. While updates to vehicle control are implemented via the ActiveControl method which is called from the FixedUpdate method. Update and FixedUpdate are abstract methods in the Unity Monobehavior Class. The difference between them is that the Update method is called every frame. On the other hand, the FixedUpdate method is based on a specific fixed time interval. This results in smoother motions of game objects when using the Unity physics engine.

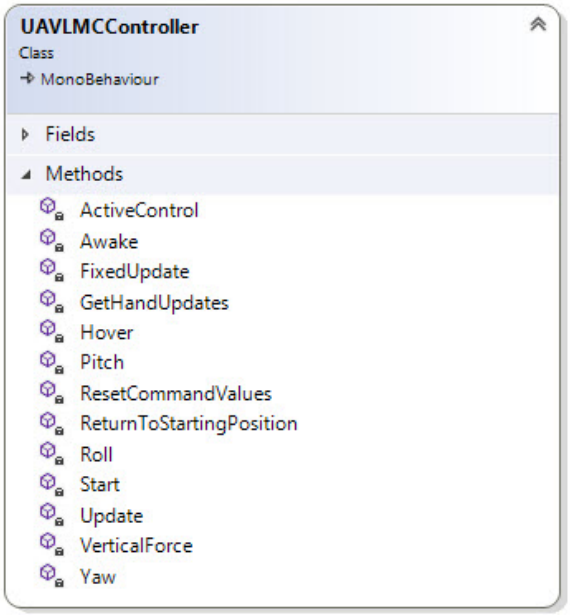

Fig. 28. UAVLMCController Class Diagram.

A linear relationship was used to interpret the data from the control source in determining the applied force and torque. For the joystick-controlled drone, the applied force or torque is proportional to joystick output, which is a value between -1 and 1, multiplied by either the maximum thrust value or the maximum torque value. A similar methodology is applied in the gesture-based system except the scale factor is a function of the ratio of the current hand orientation (i.e. roll or yaw) and the maximum allowable hand angle. A limit on the hand rotation was based on the observations on the range of motion of natural hand gestures discussed previously. For example, the maximum wrist rotation was set to 30°. Even though the user may rotate the hand to a larger angle the control input was maxed out at this condition.

One of the requirements was for the drone to maintain altitude when conducting a purely translational motion. This required determining the orientation of the concentrated propeller force in 3D space and using the fact that the vertical force must equal the weight of the vehicle. Given these two data points, the force in the horizontal plane can be found and each of these forces can then be applied to the vehicle. In the first iteration of the simulation this was implemented by applying the force and moments in a piecemeal fashion where the command could only be for pitch or roll and not a combination of both. This was in part due to the approach for tracking the vehicle orientation using 2D coordinates and not spherical coordinates. In the second iteration of the simulation quaternions were implemented to help determine the vehicles orientation in 3D space. It turned out to be a very efficient technique to track the local vehicle orientation to determine the direction of the vertical force vector.

One last component of the UAV class to discuss was the application of a control loop. The control technique implemented was a proportional-integral-derivative (PID) controller. A block diagram of a PID controller is shown in Fig. 29. The PID controller minimizes the

difference between the target position (aka setpoint) and actual position (aka, process variable) as a function of the error, *e(t)*. The error is the difference between the setpoint and the process variable. The proportional component, *P*, is determined directly from the error and a constant K_p. The integral, *I*, and derivative, *D*, components minimize the error over time and the settling time, respectively and are proportional to constants K_i and K_d, respectively.

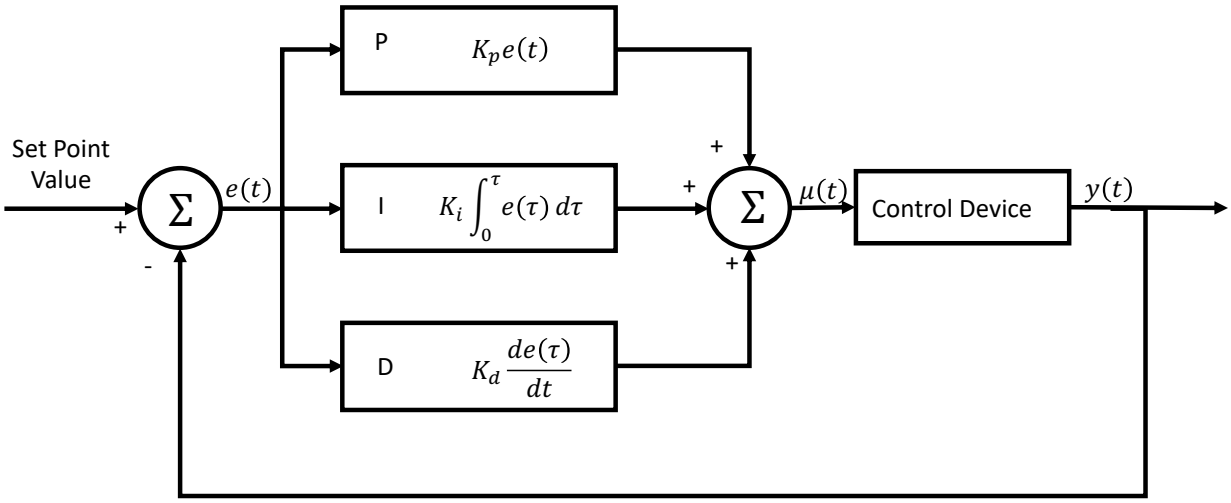

Fig. 29. Block Diagram of a PID Controller.

The PID controller was activated when calculating forces and moments to ensure the vehicle did not exceed the prescribed maximum pitch or roll angle and when transitioning to hover. For the pitching and rolling motions only the proportional component was implemented to ensure the maximum pitch or roll angle of the drone is not exceeded. While it did prevent the vehicle from toppling over in flight, this approach resulted in a slight oscillation of the vehicle when the maximum angle was reached. A full PID controller was implemented for the hover mode. The parameters for each element (i.e., K_p, K_i, K_d) were determined based on minimizing the error in the desired hover altitude and oscillation settling time while still keeping the

simulation realistic looking. This was based on observations of the flight behavior of a Phantom Standard drone, which the simulated drone is modeled after.

4.3 Demonstration, Results, and Discussion

The purpose of the first round of tests was to make a comparative assessment between joystick/button device (the Xbox 360) and gesture-based control. Four participants took part in the testing. Each participant engaged in two scenarios with each control approach. The first was play time and the second was a search mission. In the first activity the users were not asked to do anything specific. It was just meant to give them time to explore the response of the vehicle to the flight control inputs via the two techniques and also become familiar with operation of the dynamic UI for controlling the camera view. In the second scenario they were asked to locate and navigate the vehicle to a location, such as looking for and traveling to the RV park and land in an identified landing zone. The landing zone had a 10 m radius, so much larger than the drone, and was white so easily identifiable. There was no prescribed path at this point but rather just a destination. After this the participants were asked to engage in a short post-test interview. The total time to complete the test and post-test interview typically took just under an hour per participant.

In general, the participants preferred the Xbox controller over the gesture-based control system. Several observations and comments support this position. For example, on average twice as much time (11 mins vs 22 mins) was spent in play mode with the gesture-based system. This is an indication that the users felt more comfortable with the Xbox controller. A typical user's ability to control the vehicle significantly improved over the play period, but they still did not feel as comfortable with gesture system as compared to the joystick device at the end of the

play session. Finally, mission times when using the Xbox were on the order of three minutes while the missions using gesture control were rarely completed due to fatigue and frustration with the system.

In the post-test interviews participants reported feeling fatigued, mostly due to using the gesture system. This is most likely from a combination of physical and mental fatigue. Even though only minimal hand movement is required to control the vehicle, it was observed that the participants used large hand gestures requiring more energy compared to the small thumb motions that can be used with the joystick. Also, the vehicle did not respond as accurately to these gestures since they did not fall into the detection region (i.e., the green box) and were not the subtle motions expected by the processing algorithm. These observations coupled with the consideration that gesture control is a new approach probably led to a higher level of mental engagement and thus fatigue.

The users also commented that they preferred the joystick for making small command inputs. As shown in Fig. 16, Fig. 17, and Fig. 18, the LMC is highly accurate when it comes to detecting the gestures. Any inaccuracy or inconsistency comes from the operator's ability to perform such gestures. Achieving the same control precision with gestures will require additional processing to translate this information into stable and precise command actions. Some simple techniques to demonstrate this are discussed in the next chapter.

For the most part the visual content was satisfactory for the participants. The location and amount of the textural information was enough, and the user's responses did not indicate they were overly taxed with processing that information. In fact, they were typically so focused on the vehicle that they needed to be told this information was available. On the other hand, the virtual hands were distracting. This concern was somewhat alleviated by making them smaller.

They still provided a point of reference that could be viewed when needed rather than constantly in the visual processing path.

Other comments and observations centered around the use of the dynamic UI and visual aids. Participants were not able to consistently use the dynamic UI to change the camera view. They could not consistently produce the menu and often could not make the selection once the menu was available. Restricting the region where the vehicle control was activated received unfavorable comments too. The control box made them feel constricted, and it led to lack of control because they frequently had to check where they were in the field of view. Finally, they had difficulty processing vehicle orientation using the arrow. One final observation that all of the participants made was that they liked how the gesture-based system made them feel more connected to the vehicle response. These comments and observations from this set of tests led to several modifications of the simulation and is discussed next.

4.4 Simulation Modifications

The first round of tests uncovered some undesirable characteristics of the simulation and flaws in the training concept. This resulted in several modifications to the VE. First, the idea of introducing an unconstrainted play environment did not result in effective condition for the participants to learn the new gesture interface. A building block training environment was implemented to address this shortfall. Other concepts included a different approach for control initiation, and the introduction of a different user interface to control the camera view, as well an alternative visual aid to help the user process the vehicle orientation.

Hand positions are tightly connected making it difficult to give just one command input. For example, it would be difficult to just increase altitude without imparting yaw. This can be

overcome to some degree with training, so environments were added that build the participants skill set one degree of freedom at a time. TABLE 4 correlates six levels of training with the vehicle degree of freedom. So, for example, Level 1 training restricted vehicle control to altitude changes. Note that with the exception of Level 1, the vehicle remained in the same location. For instance, in Level 6 the vehicle can pitch and roll but did not translate as a result of forces and moments changing the vehicle orientation. It was anticipated that progressing a user though training in this manner will lead them to become more aware of the small range of motion required to control the vehicle. Thus, leading to a lower fatigue level and more confidence in their ability to control the vehicle.

TABLE 4

TRAINING LEVELS

Training Level	**Degree(s) of Freedom**
1	Altitude
2	Yaw
3	Yaw and Altitude
4	Roll
5	Pitch
6	Pitch and Roll

In the first round of tests, participants had a difficult time processing the correlation of the vehicle orientation with the 2-D direction indicator. As described above this was attempted

using a direction arrow, which is similar to what is on devices for controlling recreational vehicles. In the updated version of the simulation a 3D representation of the vehicle was included. This is shown in the bottom of Fig. 30 as a semitransparent sphere containing a small-scale version of the drone model. This drone matches the pitch and roll orientation as well as the direction the vehicle is flying. Again, it is anticipated that this will reduce the cognitive loading and thus fatigue since it is a more direct representation of the vehicle's orientation and will require minimum processing to understand the vehicles position.

Fig. 30. Updated Phase 1 Simulation Configuration.

Components of the user interface (UI) were also updated based on feedback from the first round of tests. Users reported they did not like seeing the hand nor restricting them to a region in the field of view. A neutral command was programmed into the simulation. If a hand was detected to be in the shape of a fist then no control command would be transmitted to the vehicle.

Also, the virtual hands were made out of clear material, so it was less distracting to the user but still available for reference. Next, the dynamic UI was hard for participants to control. So, this was replaced by simply performing a task that appears as if the user was touching the vehicle to change the camera view. When viewing from the camera a small semitransparent square in front of the viewer is the target interface. In addition to being a bit more intuitive it is also a simpler technique. A command (rotating the index finger) was also added to rotate the camera pitch angle 90°. This let the users scan from a position parallel to the flight path and straight down, which was useful for searching an area and landing.

To assess the effectiveness of these modifications, two participants from the previous test were brought back. It was conceded that the joystick approach far exceeded the gesture-based control at this time, so the users were asked only to engage in the gesture-based control. Each participant was first led through the six training environments. As anticipated, this aided in helping them develop a feel for the limited range of motion required to control the vehicle. Then they again went into the play and mission scenarios. In general, the feedback from the users was much more positive and it was observed that they had better control of the vehicle, able to complete the requested missions, and switch camera views. They also demonstrated a lower level of fatigue and frustration.

Finally, the participants motions were more limited in this second round of tests, and in general they were more relaxed. This enabled a new observation. When the users formed a fist so not to control the vehicle, their hand would frequently move to a similar position. This position was a rotation of the wrist of approximately 30° in roll and slightly pitched forward. It was a natural and relaxed position. This in combination with post-test discussions lead to the concept of control via a virtual trackball, which is implemented and discussed in the next section.

4.6 A Fundamental Redesign of the Vehicle Control Algorithm

Up to this point vehicle control is based on a direct connection between the hand gestures and the applied vehicle forces and torques. For example, a change in pitch of the left hand is directly proportional to a change in the vertical force. While the improvements described above aided in better control, it is still difficult to fly the vehicle in a stable and consistent manner. The training environments aided in informing participants about the limited range of motion required and consequently led to less fatigue too. These shortfalls are addressed in a redesign of the control approach to further stabilize vehicle control and reduce fatigue. This is accomplished by increased usage of the PID controller, incorporation of nonlinear hand gesture interpretation algorithm, and a state machine. Each of these is discussed next.

Previous implementation of the PID controller was limited to transiting to hover mode and ensuring the vehicle did not exceed its maximum rotation angle in pitch and roll. This approach was expanded to include more control setpoints. These setpoints include the vertical climb rate, yaw rate, and the pitch and roll angle. Having this structure results in the hand gestures determining the setpoint and then the PID controller determines the required force and torque vector to maintain the vehicle in this condition until an additional command is given. So, it is still a kinetic based simulation.

The setpoint is determined based on a cubic relationship using the normalized change in hand orientation. This approach can be clarified by studying Fig. 31. This figure illustrates a cubic relationship between the normalized gesture command and a control parameter. For this

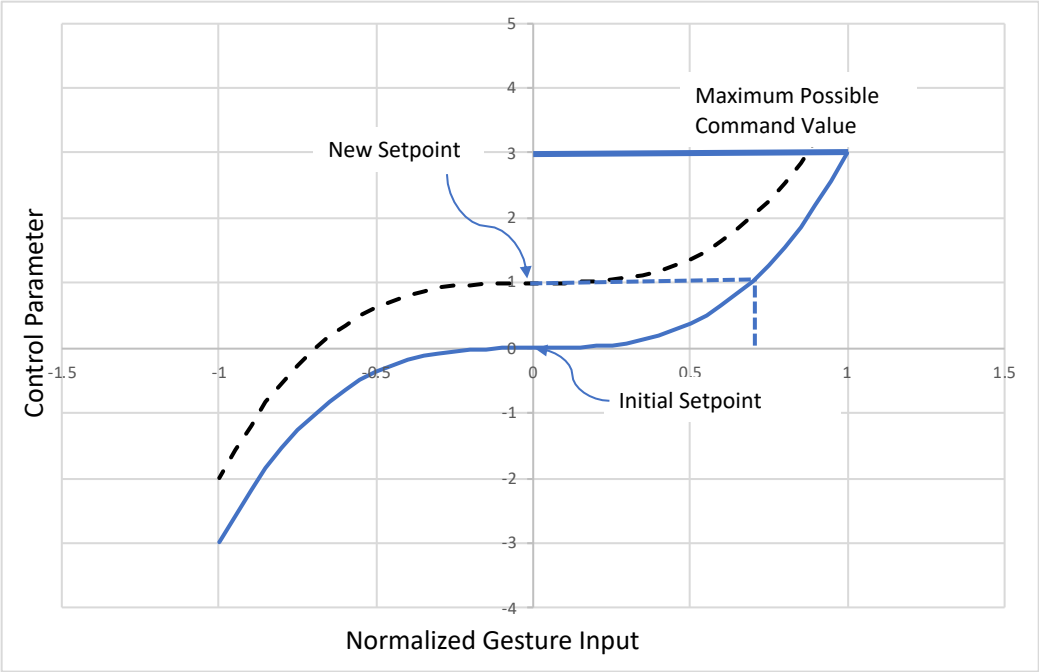

Fig. 31. Schematic of Typical Gesture Input – Setpoint Determination.

illustration the maximum value of the control parameter is set to three. Assume that the vehicle is on the ground waiting for takeoff. This condition then defines the initial setpoint shown in the figure. A change in hand orientation from the reference orientation, such as positive pitch rotation, is then normalized and the new setpoint is determined based on a cubic function. Once this command is set the user can then return their hands to the neutral (e.g. resting) position and the vehicle will continue to follow that last input command by virtue of the PID controller. Note that returning the hands to the reference orientation does not affect the setpoint. Incremental changes to this updated setpoint are made again following the cubic function shown above, so a small change in hand orientation will result in a small change in the setpoint while a larger change will increase it more but not beyond its maximum. This approach provides the operator with a wide range of control anywhere in the flight envelop.

One final note on control commands is the implementation of a data smoothing algorithm. Additional processing of the basic normalized gesture input is based on weighting the previous and current gesture command as follows:

$$Control\ Command = currentCommand * w_1 + previousCommand * w_2 \quad (2)$$

where $w_1 + w_2 = 1$ and command is the normalized gesture shown in Fig. 31. For the current application w_1 was set to 0.8 while w_2 was set to 0.2, so the current command is given four times more weight than the previous command. This helps in smoothing out some of those variations shown in Figs. 16-18.

Two additional points related to this approach are worth mentioning. First, in addition to providing the framework for more a precise control methodology, the cubic function has the benefit of a built-in "near" dead zone and a smooth transition to larger input commands. The maximum hand gesture is set to 30 degrees for this investigation, so this provides a dead zone of approximately ±5º. Second, as mentioned above a change in gesture is based upon a reference hand orientation. A condition is imposed that allows the user to return the hand to its reference orientation without a change in the setpoint. For example, in order for a setpoint to be reduced from a positive setting requires a move beyond the reference position in the negative direction. The reference position is fixed but can be changed based on user input. This point is clarified next. This feature enables a relaxed and low stressful muscle position for the operator since rotated hand positions do not have to be held for the vehicle to continue on its flight trajectory.

One last feature to describe is the addition of a state machine. Three states are defined in the simulation: active control, cruise control, and hover. Switching between states is achieved by pinching the thumb and forefinger together. These are tracked in the GestureListener Class using the SwitchState method. In active control the vehicle will respond to hand gestures in a

manner described above. The last group of setpoints will be maintained in cruise control, so the operator's hands can be totally removed to a relaxed position, or in the future implementation, address another task. Finally, in hover the vehicle will maintain its current position and altitude. At each change in state a new set of reference hand positions is established, which is captured in the GestureListenerClass GetHandReferencePositions method. This gives a user the flexibility to individually determine and update their preferred reference position.

Fig. 32 shows the final class diagram for the UAVLMCController Class. Key changes from that discussed previously are the addition of methods for cruise control and updates to method names to reflect the new control scheme (i.e., VerticalVelocity vs Vertical Force). Fig. 33 shows the final configuration of the flight scene. Text is added below the vehicle data to enable the user to track the control state with green indicating the active state. Also, the sphere that contains the tracking vehicle is moved to a position easily accessible by the right hand. The operator can then use it like a virtual trackball to guide the vehicle in pitch and roll.

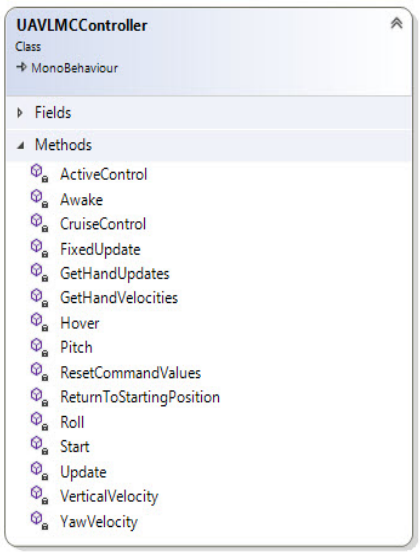

Fig. 32. Updated UAVLMCController Class Diagram.

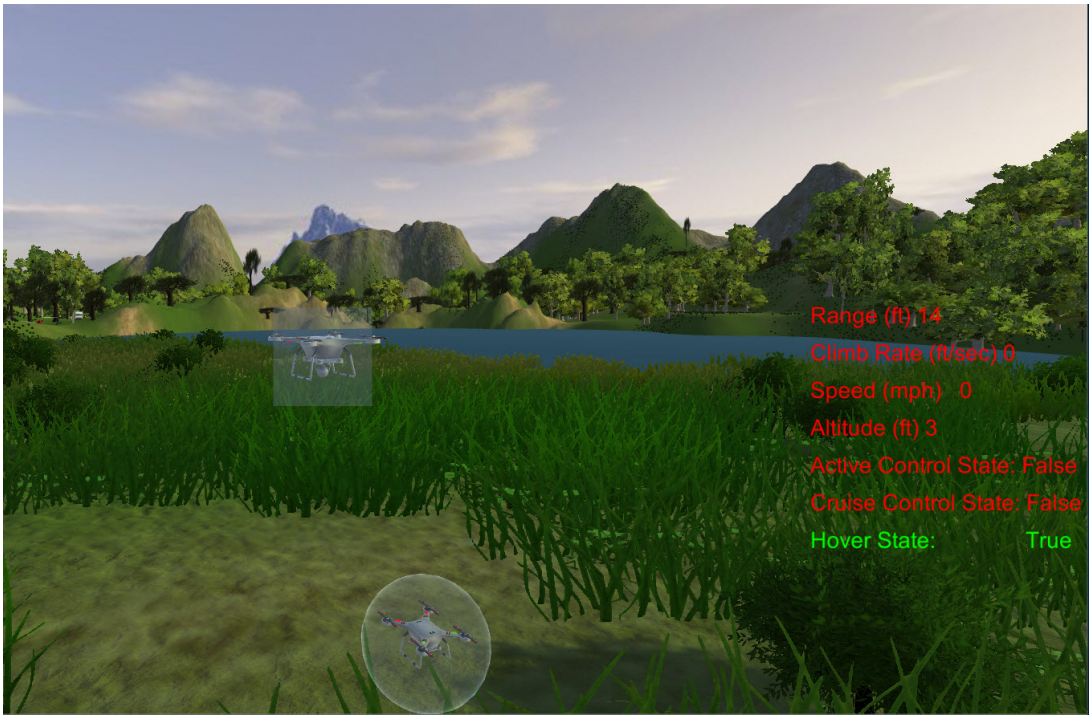

Fig. 33. Final Simulation Configuration

Several tests by the developer and a complete novice demonstrated that these changes resulted in a significantly improved vehicle control system. First, the vehicle is easier to control and is more stable in flight. The control system is also able to stabilize the vehicle when erratic commands are given by the operator. Frequent crashes in earlier versions of the simulation frustrated the participants, so significantly lowering this phenomenon is important in reducing the mental stress on the operator. More precise control of performance parameters and vehicle positioning is enabled too by the new control algorithm. Further, the updated gesture interpretation algorithm combined with the implementation of the state machine resulted in the user being able to keep a lower level of physical stress on the hands and wrist. This is a positive consequence of not requiring the user to maintain off-neutral, fixed hand positions for the vehicle to maintain its current flight trajectory.

4.6 Summary and Key Findings

This chapter described the development and testing of a simulation environment to explore a unique, gesture-based control methodology with application to a recreational quadcopter. Results comparing a traditional joystick/button controller approach and the gesture-based approach were described. Initial testing suggested that the joystick/button configuration is still the preferred approach. Based on observations and post-test interviews this is thought to be rooted in the fact that the gesture-based motion resulted in large hand movements, fatigue, and less reliable vehicle control. Improvements in the simulation to include a training environment, 3D visual aids, and a redesigned control algorithm appears to have closed that gap. As discussed in the next chapter, this knowledge will now be transitioned to the development of a VR environment to control a ground vehicle. It is then immediately followed by the user controlling a vehicle in a real-life environment to determine if the observations in this experiment and lessons learned about basic gestures and training are transitioned from the simulation to the real-life environment.

CHAPTER 5

PHASE 2 - IMPLEMENTATION, RESULTS AND DISCUSSION

This chapter describes the implementation of the control algorithms and lessons learned from the UAV simulation discussed in Chapter 4 to control an actual ground vehicle. The purpose of this demonstration is to validate the observations from Phase 1, which are from a pure simulation environment. Thus, it supports the second research objective described in Chapter 3. A VR environment built to control a model car is first described. This includes some of the more important features of Unity that enabled the kinetic simulation and unique visual features. This is followed by a description of the additions required to enable control of the actual model vehicle. Finally, results from participant demonstrations are discussed.

5.1 Virtual Reality Simulation

A VR simulation was built to provide the training and observation environment for this phase. Fig 34 shows the laboratory, with the model car, where the testing for the physical demonstration was conducted. It is approximately 5m by 10m with 3m ceilings with various obstacles in the room. The VR environment was designed to represent the geometry of this environment and provide the proper scale. Fig. 35 shows the simulated laboratory with the model car. Comparing the two figures shows that scale was maintained. It also contained a few obstacles such as pillars and tables. The tables provided targets when directing test participants to navigate around. Hardware used in this experiment include the Leap Motion Controller (LMC) and Oculus Rift Headset. The LMC generates the operational gesture recognition environment while the Oculus Rift provides an immersive display environment.

Fig. 34. Actual Laboratory Environment with Vehicle.

Fig. 35. VR Simulation with the Model Vehicle.

Fig. 36 shows the virtual trackball concept introduced in the previous chapter and an additional visual aid to support muscle memory training. The virtual trackball and the new visual component, a 2D disk with a crossbar (aka command indicator), work in a coordinated manner. The trackball itself has a diameter of 0.1m, which is about the size of a softball. As discussed in Chapter 5, the intent is to provide the user with an anchor for the hand to rotate around. Erratic readings can result from the LMC if the hand gets too close it. This distance is approximately 2.5 cm above the LMC. An alert range is conservatively set to 5 cm. At this point the trackball turns yellow.

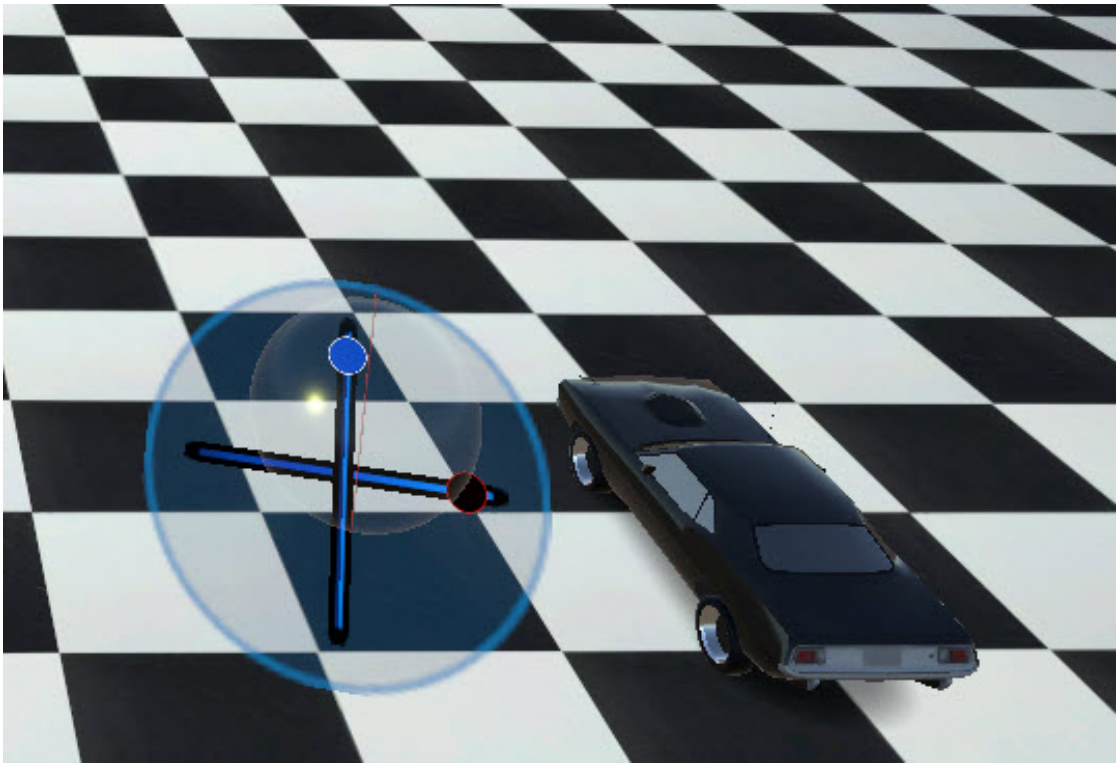

Fig. 36. Trackball and Control Indicator.

The control indicator is composed of a cross bar each with a disk that moves either vertically or horizontally. The Unity slider UI is used to build this. The vertical component is tied to the pitch of the right hand and the forward and backward motion of the vehicle. The horizontal motion is tied to roll of the right hand, which is used to control the steering angle. The maximum motion of these gestures is set to 30 degrees. Control of the vehicle based on these gestures is made through the use of a wheel collider and will be discussed a little later.

To further support muscle memory training users are positioned in a chair with the LMC just below and in front of the right arm rest. This configuration is shown in Fig. 37. While not practical for an actual application it helps to provide an anchor for the arm. This in turn lets the user focus on the small hand motions required for vehicle control.

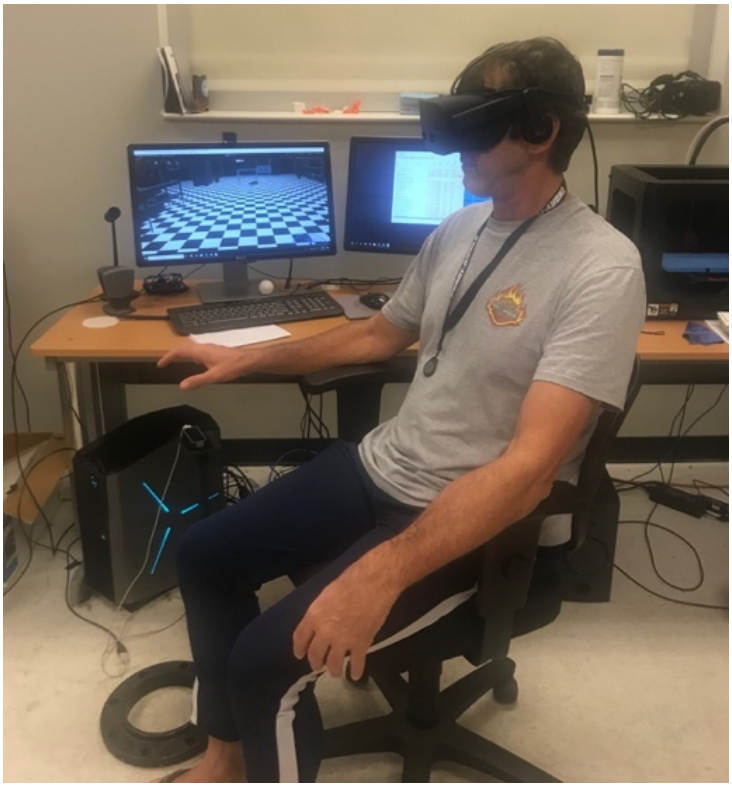

Fig. 37. Phase 2 Participant Setup.

Keeping with the forced based simulation approach the Wheel Collider capability in Unity is applied to each wheel. This is a collider-based capability with application to ground vehicles [31]. It is built upon a collider, wheel physics and a slip-based friction model. Fig. 38 shows a picture of the simulated vehicle in the VR environment with the wheel collider visible. The key parameters are contained in the slip model, suspension system components, and ability to apply torque to the wheels and also control their direction. For the current model steering was applied to the front wheels, while motor torque was applied to the rear wheels.

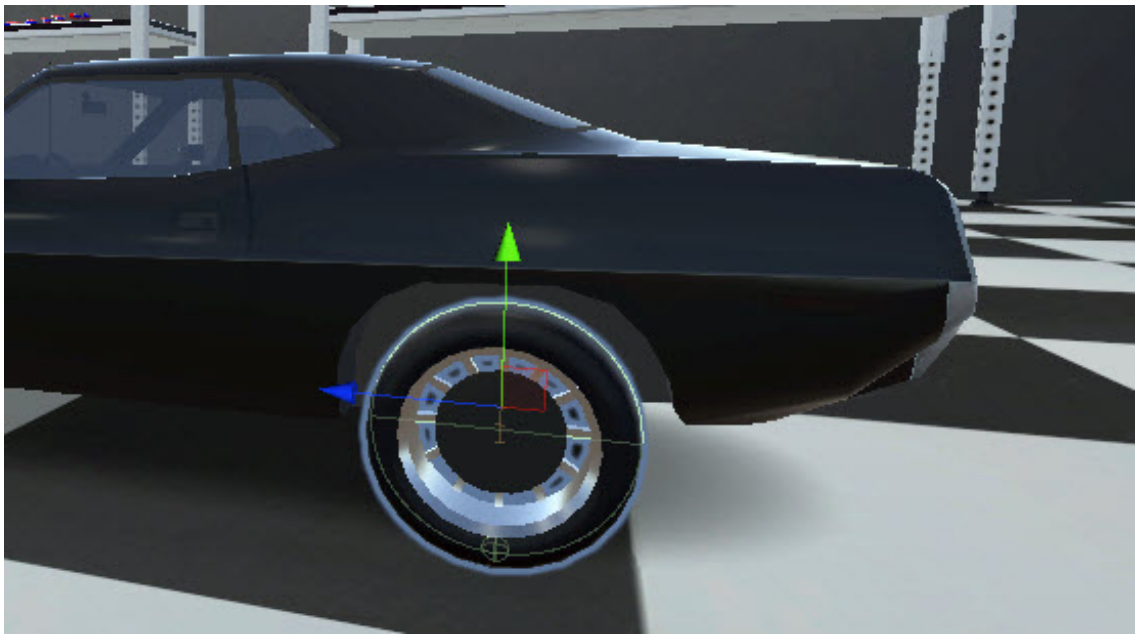

Fig. 38. Model Vehicle Showing Wheel Collider.

As with the previous simulation the gesture control script and the vehicle control script are the primary mechanisms to make it functional. The gesture capture script was used as is, so a good example of code reuse. A class diagram of the vehicle script, CarController, is shown below in Fig. 39. There are 13 methods within this class. While the names are different from the

UAVLMCController class described in the previous chapter, the basic command and control concepts, such as the cubic gesture interpretation algorithm and application of a PID controller, are similar. The main departures are made as a result of using the wheel collider. In this case the hand gestures were transformed to steering commands on the front wheels and motor torque commands for the rear wheels.

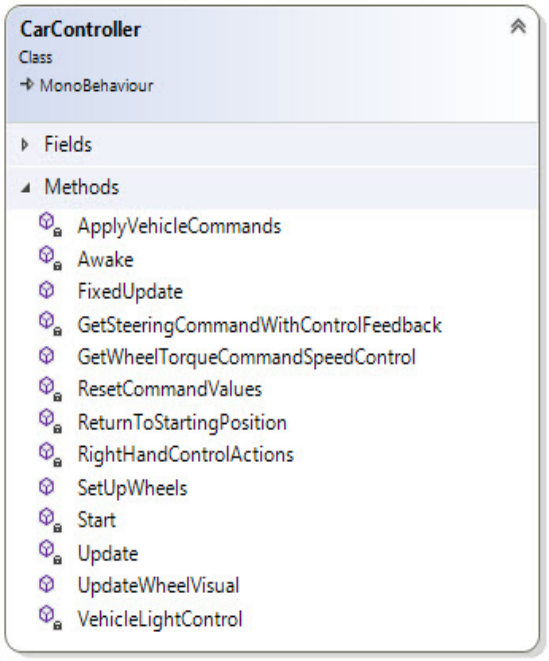

Fig. 39. CarController Class Diagram.

5.2 Remote Control Car Control

Basically, all the code used in the VR simulation was applied to control the actual model vehicle. The exception comes in the form of an additional script to transmit the Unity Actions to the vehicle control system. The components of this are shown schematically in Fig. 40. It is broken down into two parts: those that occur internal to Unity and those that occur with the vehicle communication system. Within Unity an ArdunioInterface class, or script, was added. It

contained three methods to control port operations such as initialization and send commands. Port initialization involved opening the port and setting the baud rate, which was set to 4800 bps for the current application. The Send method sent a command string via the USB port to the Ardunio Nano located on the controller shown in Fig. 12. The command string contained three components. A motor command to instruct the forward and aft motion of the case, a break command to indicate that this part of the command ended, and a steering command. After that the communication system took over and sent the command string to the car or processing and execution.

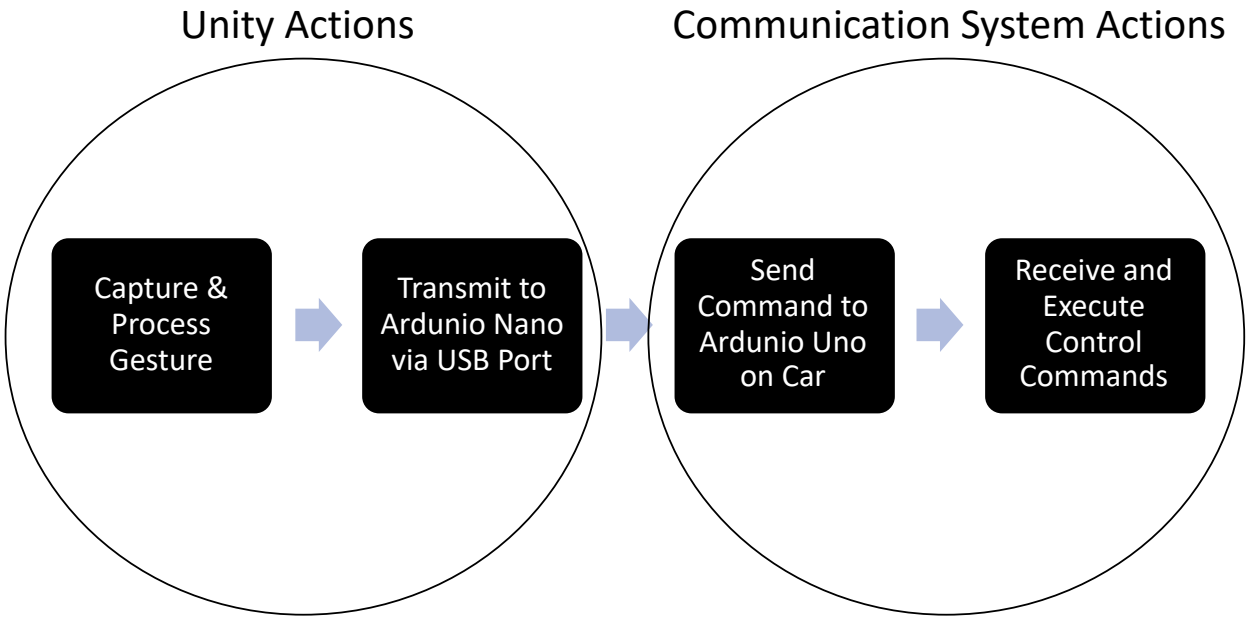

Fig. 40. Schematic of Unity-Communication System Interactions.

The microprocessor on the controller is an Ardunio Nano while that on the car is a an Ardunio Uno. There controlling code is written in C++ and came with the vehicle kit. Therefore only a few commands on the Ardunio Nano had to be altered to achieve the desired response.

For example, the delivered system has three primary control modes: joystick, self-navigation with ultrasonic sensor, controller orientation (gesture like). For the current application the controller orientation was the most similar to gesture commands so that part of the code was modified to accept the Unity originated commands.

5.3 Virtual and Real-Life Demonstration Tests

The purpose of these tests was to see if the human performance observations from the simulation environment transfer to a real-life demonstration. Adding validity to the lessons learned and observations from Phase 1 and supporting the realism of the simulation. There were two rounds of testing conducted. In the first round the initial control algorithm described in Chapter 4 was implemented, so this was a direct control of the vehicle motor torque and steering using a linear gesture interpretation algorithm. The second round of testing implements most but not all of the modified control algorithm. This is due to the face that feedback parameters (i.e., vehicle speed) are not always available. For example, vehicle speed is available in the VR simulation but not in the actual vehicle. The following features are included: cubic interpretation of gestures, incremental command inputs, commands based on a neutral reference position. So, this still captures some of the foundational elements of the control approach related to lowering the physical stress on user.

5.3.1. First Round of Testing

Two sessions were conducted in the first round. One a high participation count (around 15 participants) but informal activity and one a more formal but lower participation count (2 participants). In either case there was a training period followed by a play time in the VR

environment, followed by an event where the user controlled the model vehicle in the lab. The initial training mode involved no vehicle movement, but the indicator was free to move. This enabled the user to become familiar with the hand position and the small range of motion required for vehicle control. After that forward and backward motion was enabled to allow the user to become familiar with the visual effect of the moving car. Finally, the car steering was enabled. This step-by-step training process was inspired from the findings of Phase 1.

Informal observations of approximately 15 people took place as a result of an Employee Open House at the National Institute of Aerospace in Hampton, Virginia. During these engagements it was observed that within about 20 minutes the majority of the participants were able to reasonably control the vehicle in both virtual and real-life scenarios (10 minutes in each environment). Further, the large gestures from previous testing were not observed and the participants used small, relaxed gestures. So, it appears that the combination of the virtual visual aids and anchoring of the arm produced the desired results. One issue that was observed is that the turning performance was a little unstable. More like seeing someone ride a wobbly bicycle rather than the smooth, consistent motion.

Two additional but more formal tests were conducted. In this case each participant was processed through the same rigorous training process before enabling the play mode. Similar observations were made to the informal test described above about had motions and vehicle control. It was further observed that users became more confident in their ability to control the car in around 10 minutes. As described above in the informal test session, speed control was smooth but the turning was still a bit unstable. This is something that the latest control algorithm corrected.

One last point to mention related to this first round of testing is the initial hand position participants preferred. This position is described as that which is natural (i.e., not forced) and easy to achieve the range to control the vehicle. Each participant identified a palm rotation of approximately 25° in roll (right hand palm pointing inward) and 10° down in pitch. From these positions they were able to easily achieve a range of motion of approximately ±30°. These positions were natural and enabled a sufficient range of motion while maintaining minimal stress on the wrist. The main drawback of this first generation of testing is that this is a set position once the simulation started. Implementing the last control methodology described at the end of the previous chapter made setting this position dynamic, and thus, less restrictive.

5.3.1. Second Round of Testing

Participation in the second round of tests was limited to the developer. In this round the developer implemented and exercised the new control scheme in both the VR environment and with the remote-control car. The main differences in implementation between car-based scenarios and the UAV simulation is in selection and implementation of setpoints. For the VR simulations vehicle speed and wheel angle were the setpoints used with PID controller scheme. These setpoints are not available with the remote-control car. That would require additional vehicle sensors to provide feedback, so the active PID controller is not implemented. Other features, such as the cubic gesture interpretations and data smoothing, are integrated into the control methods.

After some initial testing it was decided to slightly modify the control algorithm to more smoothly control the car. In the UAV control a performance parameter is set such as climb rate or desired roll angle to achieve the desired speed. Then the PID controller maintains that

condition. In the case of the car controller this worked well for the speed control. In the VR simulation the user could adjust desired speed and then the PID controller would determine the required torque to apply to the wheel. In the remote-control car case the user torque is directly linked to the cubic gesture function. In each of these scenarios the user can still return their hand to the neutral position and the car will continue at that speed.

To stabilize the steering required returning to a more rudimentary approach. This is due to the fact that steering, especially in confined venues, is a more dynamic event requiring constant adjustments. It was found that the best way to steer the car was to maintain the hand in a rotated position while turning but release it once the target direction was achieved. The wheel position would in turn then return to a zero angle. If it was desired to drive in a circle then the cruise control concept described in Chapter 4 could be implemented. It was also decided to implement a three-to-one steering ratio. For example, the maximum recognized hand rotation angle was set to ±30° while the maximum steering angle of the car was set to ±10°. This is another feature that translates the less accurate human performance to more precise control of the car. These adjustments made executing basic maneuvering such as ovals and figure eight's more manageable. This final exercise illustrates the complexity of the control process and several of the features that need to be considered in the design of such a system.

5.4 Summary

This chapter described the development and testing of a VR simulation and demonstration of a remote-control model vehicle. This is the first step in validating findings on human performance and concepts to develop realistic simulations described in Chapter 4. A VR simulation was built around a model ground vehicle that used the leap motion controller as a

control source. Unique features in this simulation included visual aids to help the user anchor their hands and familiarize themselves with the small ranges of motion required to control the vehicle. This was then followed by a demonstration where the participant was asked to navigate a model vehicle in a real environment. Initial tests showed promise and validated observations and implementation of lessons learned from the pure simulation environment of Phase 1. Further improvement in control and human performance resulted by implementing a PID controller methodology. This process also demonstrated the evolutionary nature of a control system to address the range of variables and scenarios encountered with different systems.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Ground and air vehicles are emerging with capabilities to operate at various levels of autonomy. This has been enabled in part by micro sensors and navigation systems combined with control algorithms and powerful microprocessors that can use the data to process the required command and control functions. While most of the operations can be accomplished autonomously there will most certainly always be a human in the loop providing some level of command and control. This can range from high level functions such as directing a vehicle to a point in space (i.e., on a map) to more immediate and direct control of the vehicle operations.

Interaction with these systems has been designed based on legacy approaches such as keyboards and joysticks. New technology is emerging that can result in "hands-free" multi-modal command and control systems based on hand gestures and voice commands. Being new interfaces, it will require multiple investigations into how well they can work, how people will use them, and what are the environments to familiarize operators with these new interfaces. The gesture-based control component was the focus of this investigation.

6.1 Summary

There were two research objectives for this project. One addressed the human factors aspect of gesture-based control while the second addressed the suitability of a simulated environment as an assessment and training tool. The human factors aspect focused on the application of gesture-based control of semi-autonomous systems to identify capability, challenges, and limitations to assess the feasibility (can you do it) and viability (does it add

value) of the approach. While the second assessed the suitability of a simulation environment to (1) support assessment of human performance and interface preferences for vehicle control and (2) provide a training environment for transition to a real-world system. These objectives were achieved through the use of simulations developed in Unity and then subsequently verified using a real-life model.

There were two simulations built in Unity with hand gestures being captured using a small optical based sensor (the leap motion controller or LMC). The first simulation was of a recreational quadcopter. It was a kinetic based simulation where a unidirectional force and a 3D torque vector were applied to the vehicle based on hand gestures captured by the LMC. Not unlike what would happen as propeller thrust was adjusted on the real vehicle. Two rounds of test were conducted. In the first participants evaluated the vehicle control capability between a traditional joystick/button device (an Xbox 360 controller) and the gesture-based system. The simulation was that of a campground by a lake. Participants were first exposed to a play environment. This gave them an opportunity to learn how to control the vehicle and familiarize themselves with the vehicle information displayed on the screen. After that they were tasks to search for and navigate to specific targets in the virtual environment, such as the RV parking area or a group of tents. Data was recorded based on observations by the test leader and a post-test interviews with the participants. Results from this set of tests indicated that users preferred the Xbox controller over the gesture-based system. This was founded in the participants ability to easily and precisely control the vehicle and familiarity with the traditional approach.

Based on feedback from participants the quadcopter simulation was modified to include some additional features and a significant change in the gesture interpretation and vehicle control algorithm. One feature was the inclusion of a highly controlled simulated training environment.

In this environment the vehicle motion was restricted to limited degrees of freedom. This allowed the user to become familiar with the small range of hand motion required and built confidence in their ability to control the vehicle. Two unique features that came out after the first test was use of a 3D attitude indicator showing the vehicle orientation and more initiative camera view selection interface. The latter changed from a dynamic menu to simply touching the vehicle when a change was desired. Both of these resulted in lower load on working memory. Finally, a significant change in the vehicle control algorithm was implemented. In the first version of the simulation vehicle forces and torques were directly linked to hand gestures. In the modified control algorithm hand gestures were interpreted to set vehicle performance attributes (or setpoints), such as vertical velocity. These setpoints can be incrementally changed. Then a proportional-integral-derivative, or PID, control scheme adjusts the forces and torques to maintain the desire vehicle performance based on these setpoints. The final virtual environment resulted in better control of the vehicle and its onboard functions. It resulted in a much more relaxed user too. This allowed for the observation of more natural desired hand movements and neutral positions.

To validate findings from the quadcopter simulation a second configuration of testing was developed. In this configuration participants engaged in a VR simulation followed by real-life scenario. It each case the user controlled a remote control, electric car, about the size of a standard textbook, based on gestures captured using the LMC. Lessons learned from the quadcopter simulation, such as providing a well-controlled training environment and supporting visual aids were included to aid the participant in developing the muscle memory for gesture control. Feedback and observations showed the virtual environment transitioned to good control

of actual vehicle, thus, supporting the observations on human performance acquired from the virtual environments.

6.2 Conclusions

The ability to control vehicles via gesture-based control is achievable. Additionally, with the emergence of head mounted, augmented reality technology it may make it preferable. However, at this point there is still a strong preference for the joystick approach. This maybe the result of a combination of familiarization and maturity of the technology. The joystick-based controller has been around for a number of years, its basic design is well tailored, and its functions are well developed. The gesture-based system is still new and can be intuitive, but it is not so yet. While care was taken in this effort to implement natural gestures, they were still new ideas. However, participants learned the new system quickly (on the order of fractions of an hour) to achieve a moderate level of vehicle control and stated they felt more connected to the vehicle using this approach.

Another conclusion is that the available gesture capture systems are highly accurate and capable of detecting a wide range of hand motions. These hand motions can subsequently be transformed into control commands. However, the human is not as precise. To make these systems more useable data processing and control systems need to be implemented that smooth out the variations in human performance and thus stabilize the vehicle control. Also, being a new interface will require training environments to familiarize the user with the range of motion required since it is basically unlimited by any mechanical constraints. For example, a joystick is a mechanical based system and it has motion limits. A hands-free gesture system is wide open and limited only by the physical makeup of the operator. Establishing proper training

environments showed that this motion can be easily learned. Further, the research has revealed other subtilties on motion that were not originally considered. For example, the original gesture concept was to simply rotate flat hands via a pitch and roll motion of the wrist. Observations showed the preferred neutral position of a hand was slightly offset and semi-rounded making it more suitable for a virtual track ball concept. Testing on a larger scale is required to further investigate human performance and preferences for this control idea. The system developed in this research is now setup to conduct these larger scale tests.

6.3 Future Work

This project uncovered some fundamental principles to guide the development of gesture-based command and control systems. A big challenge is that the ability for the technology to capture a gesture is more accurate than the human's ability to consistently make the gesture. Thus, some additional processing techniques need to be implemented to smooth out the flight performance and make vehicle control more precise. Future systems should also contain calibration routines to allow the user to personalize the range of hand motions. A benefit of this research is that it can assist in designing applications for AR products coming on the market. A possible next step in that direction is to use existing, lower cost, VR headsets [13] in conjunction with the LMC to design and test more complex systems that revisit the idea of dynamic menus and overall multi-modal command and control systems.

REFERENCES

[1] B. Terwilliger, D. Ison, D. Vincenzi and D. Liu, "Advancement and Application of Unmanned Aerial System Human-Machine-Interface (HMI) Technology," in *HCI International* , Crete, Greece, 2014.

[2] U.S. Department of Defense, "Unammned Systems Integrated Roadmap FY2013 (Report No. 14-S.0553)," 2013.

[3] Volpe, John A., "Unmanned Aircraft System (UAS) Service Demand 2015-2035. Literature Review and Projections (Report No. DOT-VNTSC-DoD-13-01)," U.S. Department of Transportation, 2015.

[4] C. Balog, B. Terwillinger, D. Vincenzi and D. Ison, "Examining Human Factors Challenges of Sustainable Small Unmanned Aircras Systems (sUAS)," in *Advances in Human Factors in Robots and Unmanned Systems. Vol 499 of the series Advances in Intelligent Systems and Computing*, New York, NY, Springer International Publishing, 2016, pp. 61-73.

[5] Army Field Manual FM 21-60, 1987.

[6] M. S. Hamilton, P. Mead, M. Kozub and A. Field, "Gesture Recongition Model for Robotic Systems of Military Squad Commands," in *Interservice/Industry, Training, Simulation and Education Confernce*, Orlando, FL.

[7] A. Sarkar, R. K. Ganesh Ram, K. A. Patel and G. K. Capoor, "Gesture control of drone using a motion controller," in *International Conference on Industrial Informatics and Computer Systems (CIICS), pp. 1-5*, Sharjah, 2016.

[8] S. International, "Touch Interactive Display Systems: Human Factors Considerations, System Design and Performance Guidelines ARP 60494," SAE International, 2019.

[9] A. P. Tvaryanas, "Human Systems Integration in Remotely Piloted Aircraft Operations," *Aviation, Space, and Environmental Medicine,* vol. 77, no. 12, pp. 1278-1282, 2006.

[10] P. Milgram and F. Kishino, "A Taxonomy of Mixed Reality Visual Displays," *IEICE Transactions of Information Systems,* Vols. E77-D, no. 12, 1994.

[11] MicroSoft, "HoloLens 2 Mixed Reality is Ready for Business," [Online]. Available: https://www.microsoft.com/en-us/hololens. [Accessed 13 September 2019].

[12] D. R. Lampton, B. Knerr, B. R. Clark, G. Martin and D. A. Washburn, "ARI Research Note 2306-6 - Gesture Recognition System for Hand and Arm Signals," United States Army Research Institute for Behavioral Sciences, Alexandria Va, 2002.

[13] J. R. Cauchard, L. E. Jane, K. Y. Zhai and J. A. Landay, "Drone & me: An exploration into natural human-drone interaction," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, Osaka, Japan, 2015.

[14] Leap Motion, "Leap Motion," [Online]. Available: https://www.leapmotion.com/. [Accessed 16 09 2019].

[15] F. Weichert, D. Bachmann, B. Rudak and D. Fissler, "Analysis of the accuracy and robustness of the leap motion controller," *Sensors,* vol. 13, no. 5, pp. 6380-6393, 2013.

[16] J. Guna, G. Jakus, M. Pogacnik, S. Tomazic and J. Sodnik, "An Analysis of the Precision and Reilabiity of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking," *Sensors,* vol. 14, no. 2, pp. 3702-3720, 2014.

[17] A. H. Smeragliuilo, N. J. Hill, L. Disla and D. Putrino, "Validation of the Leap Motion Controller using markered motion capture technology," *Journal of Biomechanics,* vol. 49, no. 9, pp. 1742-1750, 2016.

[18] D. Bachmann, F. Weichert and G. Rinkenauer, "Evaluation of the Leap Motion Controller as a New Contact-Free Pointing Device," *Sensors,* vol. 15, no. 1, pp. 214-233, 2015.

[19] D. Wigdor and D. Wixon, Brave NUI World Designing Natural User Interfaces For Touch and Gesture, Burlington, MA: Morgan Kaufmann, 2011.

[20] A. Scicali and H. Bischof, "Useability Study of Leap Motion Controller," in *Proceedings of the International Conference on Modeling, Simulation and Visualization Methods (MSV)*, Athens, Greece, 2015.

[21] I. Staretu and C. Moldovan, "Leap Motion Device Used to Control a Real Anthropomorphic Device," *International Journal of Advanced Robotic Systems,* vol. 13, no. 113, 2016.

[22] M. Chandarana, E. L. Meszaros, A. Trujillo and B. D. Allen, "Natural Language Based Multimodal Interface for UAV Mission Planning," in *Proceedings of the Human Factors and Ergonomics Society 2017 Annual Meeting*, Los Angeles, CA, 2017.

[23] R. A. Fernandez, J. L. Sanchez-Lopez, C. Sampedro, H. Bavle, M. Molina and P. Compoy, "Natural Under Interfaces for Human-Drone Multi-Modal Interaction," in *International Conference on Unmanned Aircraft Systems*, Arlington, VA, 2016.

[24] S. Zollman, C. Hoppe, T. Langlotz and G. Reitmayr, "FlyAR: Augmented Reality Supported Micro Aerial Vehicle Navigation," *IEEE Transactions on Visualization and Computer Graphics,* vol. 20, no. 4, pp. 560-568, 2014.

[25] A. Givens, M. E. Smith, H. Leong, L. McEvoy, S. Whitefield, R. Du and G. Rush, "Monitoring Working Memory Load during Computer-Based Tasks with EEG Pattern Recognition Methods," *Human Factors: The Journal of the Human Factors and Ergonomics Society,* vol. 40, no. 1, pp. 79-91, 1998.

[26] C. Rorie and L. Fern, "UAS Measured Response: The Effect of GCS Control Model Interfaces on Pilot Ability to Comply with ATC Clearances," in *Proceedings of the Human Factors Ergonomics Society 58th Annual Meeting*, 2014.

[27] S. Dodd, J. Lancaster, A. Miranda, S. Grothe, B. DeMers and B. Rogers, "Touch Screens on the Flight Deck: The Impact of Touch Target Size, Spacing, Touch Technology and Turbulence on Pilot Performance," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Chicago, Ill, 2014.

[28] P. Ravassard, A. Kees, B. Willers, D. Ho, D. Aharoni, J. Cushman, Z. M. Aghajan and M. R. Mehta, "Multisensory Control of Hippocampal Spatiotemporal Selectivity," *Science,* vol. 340, no. 6138, pp. 1342-1346, 2013.

[29] P. Antonenko, F. Paas, R. Grabner and T. Van Gog, "Using Electroencephalography to Measure Cognitive Load," *Educational Psychology Review, ,* vol. 22, no. 4, pp. 425-438, 2010.

[30] B. Sanders, D. Vincenzi and Y. Shen, "Scale and Spatial Resolution Guidelines for the Design of Virtual Engineering Laboratories," in *In: Kantola J., Barath T., Nazir S., Andre T. (eds) Advances in Human Factors, Business Management, Training and Education. Advances in Intelligent Systems and Computing, vol 498. Springer, Cham*, Orlando, FL, 2016.

[31] Unity, "Unity Documentation," [Online]. Available: https://docs.unity3d.com/Manual/. [Accessed 19 September 2019].

[32] Adeept, "Adeept," [Online]. Available: https://www.adeept.com/. [Accessed 18 September 2019].

[33] Elecrow Bazaar, "Elecrow Bazaar 2.4G Wireless nRF24L01," [Online]. Available: https://www.elecrow.com/wiki/index.php?title=2.4G_Wireless_nRF24L01. [Accessed 19 September 2019].

APPENDICIES

APPENDIX A
PHASE 1 POST-TEST QUESTIONNARIE

1. Which system did you prefer (1 handheld, 4 gesture-based)?

   1 2 3 4

Rationale for selection:

2. Which system was easier operate (1 handheld, 4 gesture-based)?

   1 2 3 4

Rationale for selection:

3. How much confidence did you have in the ability to control the vehicle (1 no confidence, 4 complete confidence)?

   Handheld System 1 2 3 4

   Gesture Based System 1 2 3 4

Rationale for selection:

4. Which system is better for quick response (1 handheld, 4 gesture-based)?

   1 2 3 4

Rationale for selection:

5. Which is better for small flight adjustments (1 handheld, 4 gesture-based)?

   1 2 3 4

Rationale for selection:

6. Which view did you prefer (1 operator view, 5 drone view)?

1 2 3 4 5

Rationale for selection:

General Questions

7. Suggestions for control gestures other than the options provided

8. Improvements to the environment and training

APPENDIX B
PHASE 2 POST-TEST QUESTIONNARIE

Part 1 – Simulated Vehicle Control

1) How Confident Were You in Your Ability to Control the simulated Vehicle?
   a) 1 - no confidence – could not make it do what I wanted
   b) 3 – somewhat confident – could make it do what I wanted but it was slow going
   c) 5 - confident – could make it do exactly what I wanted, when I wanted

   1 2 3 4 5

2) Did the simulated vehicle respond to you hand gestures (1 – not at all, 5 completely responsive)?

   1 2 3 4 5

3) Where you able to navigate to a designated location in the simulated environment?
   a) 1 – no
   b) 3 - yes, but with some difficulty
   c) 5 - yes, with ease

   1 2 3 4 5

4) Where you able to navigate around obstacles location in the simulated environment?
   a) 1 – no
   b) 3 - yes, but with some difficulty
   c) 5 - yes, with ease

   1 2 3 4 5

5) Where you able to start and stop as desired location in the simulated environment?

   No Yes

6) Where you able to turn as desired location in the simulated environment?
   a) 1 - no ability
   b) 3 – yes but required to start and stop frequently and slowing down to stay on track
   c) 5 - yes, able to do just what I wanted to with small adjustments

   1 2 3 4 5

Part 2 Model Vehicle Control

7) How Confident Were You in Your Ability to Control the Model Vehicle?
   a) 1 - no confidence – could not make it do what I wanted
   b) 3 – somewhat confident – could make it do what I wanted but it was slow going
   c) 5 - confident – could make it do exactly what I wanted, when I wanted

1 2 3 4 5

8) Was the vehicle responsive to your gestures (1 – not at all, 5 completely responsive)?

1 2 3 4 5

9) Where you able to navigate to a designated location?
   a) 1 – no, only able to wander around
   b) 3 - yes, but with some difficulty and lots of big course corrections
   c) 5 – yes, picked the target and able to get there with small corrections

1 2 3 4 5

10) Where you able to navigate around obstacles?
   a) 1 – no
   b) 3 - yes, but with some difficulty
   c) 5 - yes, with ease

1 2 3 4 5

11) Where you able to start and stop the vehicle as desired?

No Yes

12) Where you able to turn as desired?
   a) 1 - no ability
   b) 3 – yes but required to start and stop frequently and slowing down to stay on track
   c) 5 - yes, able to do just what I wanted to with small adjustments

1 2 3 4 5

Part 3 Comparison

13) Was the similarity of the simulation environment compared to the actual environment helpful (1 not at all, 5 made transition easy to actual model seamless)?

1 2 3 4 5

14) Did the Simulation Environment Help Prepare You for Controlling the Actual Vehicle (1 not at all, 5 completely)?

1 2 3 4 5

15) Did the Vehicle Response in the Simulation Environment Reflect the Actual Response of the Model Vehicle?
   a) 1 - completely different
   b) 3 – some features behaved the same but some responses were different
   c) 5 - behaved the same

1 2 3 4 5

16) Comments about how to modify the simulation environment to improve learning how to control the car.
   a) Useful aspects?

   b) Distracting/detrimental?

OBSERVER GATHERED DATA

1) What was the most comfortable neutral position of the hand to achieve the desired range of motion for the participant:
   a) Palm pitch angle: _____
   b) Palm roll angle: _____

2) Did the participant use minimum energy hand gestures or broad sweeping motions (1-broad motions, 4 minimum energy motions)?

   1 2 3 4 5

3) Did the participant demonstrate an ability to conduct Basic Maneuvers (start/stop, drive straight forward/backward, turn, control speed) (1 no control, 5 excellent control):
   a) Simulated Environment

      1 2 3 4 5

      i) Comments:

   b) Actual Environment

      1 2 3 4 5

      i) Comments:

4) Did the participant demonstrate an ability to Navigate to a target (i.e., hit the avatar) (1 is lowest, 5 is highest)?
   a) Simple (out in the open)

      1 2 3 4 5

      i) Rational for Rating:

   b) Complex (navigation around an obstacle required)

      1 2 3 4 5

      i) Rational Rating:

GENERAL SUGGESTIONS/COMMENTS

1) Suggestions for natural (physically and cognitively) control gestures other than the options provided

2) General Comments

VITA

Brian Sanders

Education

1993 Ph.D., Aerospace Engineering, Air Force Institute of Technology, WPAFB

1987 M.S., Aerospace Engineering, University of Dayton, Ohio

1985 B.S., Aerospace Engineering, University of Southern California

Professional Experience

2017-Present *Associate Professor, Department Chair, Engineering and Technology Department, Embry Riddle Aeronautical University*

2013-2016 *Assistant Professor, Engineering Sciences Department, Embry Riddle Aeronautical University*

2008-2013 *Assistant Chief Scientist, Air Combat Command, USAF*

2000-2007 *Adjunct Professor, University of Dayton*

1999-2008 *Senior Research Scientist, Air Force Research Laboratory, Air Vehicles Directorate*

1995-1999 *Program Manager, Structural Mechanics Program, Air Force Office of Scientific Research*

1992 -1995 *Composites Materials Research Engineer, Air Force Research Laboratory, Materials Directorate*

1985 -1989 *Logistics Engineer, Short Range Attack Missile Program Office, Wright Patterson Air Force Base*

1980-1983 *Weapons Crew Team Chief, Holloman Air Force Base, NM*

Publications

25 Refereed journal articles
59 Conference publications and presentations
1 Book Chapter

Related Publications

1. Sanders, B, Vincenzi, D, Holley, S. and Shen, Y, "Traditional vs Gesture Based UAV Control", Proceeding from the 8th International Conference on Applied Human Factors and Ergonomics, July 17-21, 2018, Los Angeles, part of Springer Nature 2019 J. Chen (Ed.): AHFE 2018, AISC 784, pp. 15-23, 2019. https://doi.org/10.1007/978-3-319-94346-6_2

2. Sanders B., Vincenzi D., Shen Y. (2018) Investigation of Gesture Based UAV Control. In: Chen J. (eds) Advances in Human Factors in Robots and Unmanned Systems. AHFE 2017. Advances in Intelligent Systems and Computing, vol 595. Springer, Cham https://doi.org/10.1007/978-3-319-60384-1_20

3. Sanders B., Vincenzi D., Shen Y. (2017) Scale and Spatial Resolution Guidelines for the Design of Virtual Engineering Laboratories. In: Kantola J., Barath T., Nazir S., Andre T. (eds) Advances in Human Factors, Business Management, Training and Education. Advances in Intelligent Systems and Computing, vol 498. Springer, Cham. https://doi.org/10.1007/978-3-319-42070-7_34