Commanding Small Satellites for Simulated Spacecraft Inspections Using Augmented Reality

by

Jessica Eve Todd

B.S., University of Sydney (2016)
B.E., University of Sydney (2016)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2019

Signature redacted

Author .....
Department of Aeronautics and Astronautics
August 22, 2019

Signature redacted

Certified by .....
Leia A. Stirling
Charles Stark Draper Assistant Professor of Aeronautics and Astronautics
Thesis Supervisor

Signature redacted

Accepted by .....
Sertac Karaman
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Commanding Small Satellites for Simulated Spacecraft Inspections Using Augmented Reality

by

Jessica Eve Todd

Submitted to the Department of Aeronautics and Astronautics
on August 22, 2019, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Free-flying teleoperated satellites have been proposed as a replacement for astronaut extravehicular activities (EVA), such as inspection or maintenance, to reduce astronaut risk. A major concern for this type of operation is ensuring the human operator has sufficient spatial and temporal knowledge of the free-flying spacecraft and environment to safely complete the task. This research evaluates Augmented Reality (AR) as a means of providing spatial and temporal information to an operator controlling a free-flying robot to perform an inspection task. Specifically, the research focuses on the effect of command input mode and environmental risk on performance of an EVA inspection task and the strategies adopted by the operator in completing the inspection.

Subjects performed a simulated inspection task of a space station using an inspector small satellite and the Microsoft Hololens platform. Subjects commanded the inspector satellite in three operation modes; satellite body (local) reference frame control, global reference frame control, and global waypointing system (placing markers for the inspector to follow in the global reference frame). Subjects were instructed to inspect the exterior of the station and identify any surface anomalies. Anomalies could occur in areas with low and high risk of contact with the station structure. Subjects performed the inspection task using each of the three fixed command modes (order randomized), with different anomaly configurations. Subjects then performed the inspection task but with free choice of when and how often they utilized the different control modes. Performance was evaluated through primary task measures including percentage of station inspected and accuracy in locating anomaly sites,

number of AR interactions, and number of collisions with the station. Workload was also assessed using the NASA TLX survey.

Operation in both the global and local frame controls was found to maximise the percentage of the station that could be inspected, while the waypoint system was found to minimize the number of collisions between the inspector and the station. When operating with free choice of command mode, subjects preferred to stay in a single mode, typically either the global or local controls, likely due to the high usability of these modes for the selected inspection task. Environmental risk area was not found to have a signficant effect on either detection of anomalies or number of collisions.

Findings of this paper will inform a follow-on study at NASA's HERA facility, with a full analog-crewed mission operating multiple inspection agents over several days.

Thesis Supervisor: Leia A. Stirling
Title: Charles Stark Draper Assistant Professor of Aeronautics and Astronautics

Engineering is the art of modelling materials we do not wholly understand, into shapes we cannot precisely analyse so as to withstand forces we cannot properly assess, in such a way that the public has no reason to suspect the extent of our ignorance.

Dr A. R. Dykes

Acknowledgments

This Masters thesis would not have been possible without the support of some amazing people in my life. Firstly, I have to give a huge thank you to my amazing advisor Leia Stirling. It has been a privilege to be guided and mentored by you over the past two years. Thank you for always listening, and for supporting me through the tough times. You're an amazing role model to me and to young female engineers.

Thank you to Andy Liu, for your continual guidance and insights, for reading my emails and drafts, and always taking the time to answer my questions. To Dave Miller, thank you for financially supporting me when funding situations became uncertain and for being a great source of advice in navigating potential career paths. To my UROPs, Emma and Josh, you were truly amazing. Despite endless debugging and hardware issues, you put in 110%, so thank you for all your hard work.

To the amazing humans of the Humans Systems Lab, I could not have asked for a better grad school family. Two years of Thanksgiving turkey extravaganzas, epic (and unfinished) movie lists, and team Halloween costumes, who could ask for more? The support, friendship, and camaraderie I've found in this lab is unmatched and has helped make MIT a second home.

To the wonderful Liz, Beata, Beth, Joyce, and Pam, I don't think the Department could manage without you. Thank you so much for all you do for us grad students in making this department such a supportive and welcoming place to work.

My phenomenal friends, Ben for being an amazing GA^3 co-president, Cadence for being the world's best roommate, Colin for keeping me sane with sailing and Boston adventures, and all my friends in AeroAstro, you've been my home away from home. The memories I have from the past two years are some of my happiest and I feel so grateful to have met you all. They say grad school takes a village, and you've been

mine.

And finally to my amazing family. The last two years have been tough, stretched across two continents and working on three degrees between us, but through it all, your support, love and encouragement has been a source of strength. To my amazing siblings Isabella and Reilly, you're both a constant source of inspiration to me. Early morning/late night FaceTime chats about everything from movies to the state of the Universe have gotten me through all the stresses and struggles of grad school. To my amazing dad, you've been my sounding board through all the ups and downs, giving career advice, listening when I've had a bad day or yet another existential crisis, and always having a fun science anecdote to keep me amused. I aspire to be an engineer and a leader like you. And to my phenomenal mum, I wouldn't have got here without your strength, love and support. Of all the amazing women I've been privileged to meet here and out in the world, you're still my biggest hero and source of inspiration.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Future human space missions will rely heavily on human-robotic systems for planetary surface and in-space operations, including mobility, exploration, maintenance, and assembly tasks. Human-robotic collaboration in these scenarios will include teleoperation (robots operated remotely by astronauts) and human-robot teaming (humans and robots co-located in the same environment), requiring both the human and robot to react to dynamic and off-nominal situations in real-time. Effective communication and information gathering between the human and robotic components is vital for operational efficiency and risk mitigation in these scenarios.

Exterior inspection tasks are a key component of maintaining safe operations for long-duration crewed spacecraft. The Space Shuttle and International Space Station (ISS) both relied on routine inspections for maintenance, anomaly investigation, mitigation of debris impact risk and reporting on configuration of hardware [31, 49], and inspection tasks will continue to be a vital component of any future crewed space

mission [71] that will require a combined human-robotic effort. Currently, inspection of the ISS exterior primarily occurs through two methods: manual inspections via astronaut extravehicular activities (EVA), and robotic platforms like the Space Station Remote Manipulator System (SSRMS, or Canadarm2). When shifting to a long-duration exploration mission (LDEM) scenario such as the proposed National Aeronautics and Space Administration (NASA) Lunar Gateway mission or future crewed Mars missions, both inspection methods have serious limitations.

Current EVAs are rehearsed extensively on the ground and strictly choreographed on-orbit. During operation they are supported by an extensive ground team, and performed in near constant communication with ground-based Mission Control. This EVA model becomes inviable for LDEMs which will essentially operate independently of Earth. High communication latencies prevent real-time ground support and astronauts may be required to react to off-nominal scenarios without the benefit of rehearsal. Additionally, proposed LDEMs may require up to 24 hours of astronaut EVA per week [21], exposing the astronauts to dangerous space radiation doses as they transit beyond Earth's protective magnetosphere [10].

Fixed robotic platforms like the SSRMS are teleoperated by astronauts inside the space station, using manual interfaces and multiple display screens showing fixed external camera views of the arm. Astronauts can also use a simulated exocentric view of the arm, however all control decisions must be made based on real camera views. The SSRMS has 7 degrees-of-freedoms (DOF) [32], and when combined with the Special Purpose Dextrous Manipulator (SPDM or Dextre) and the Mobile Base System (MBS) a total of 23 DOF, making coordinated control movements very difficult [55]. Manipulation of these systems by a human operator is highly complex due to the restricted camera views, multiple DOF, and high cost of error (possible collisions with the station) and requires excellent coordination, manipulation, and

spatial awareness skills [11]. Errors in robotic manipulation on a LDEM could be catastrophic without possible repair and resupply from Earth.

Free-flying teleoperated satellites represent a possible alternative to EVA and fixed robotic platforms for Earth-based and long duration deep-space missions. Small satellites carrying sensors and cameras could be deployed from the exterior of the station to perform surface inspections, while tele-operated by astronauts inside the space station. Free-flyers operate in six degrees of freedom without limits on motion, offering greater flexibility of motion than their fixed robotic counterparts, and greatly reducing the complexity of control required by the human operator. Free-flying satellites would reduce risk to astronauts by decreasing crewed EVA time for inspection tasks, and could also be used to evaluate trajectories, risks, and procedures prior to required crewed EVA missions, without the need for full ground support. Additionally, free-flying satellites are less massive to carry to orbit, and cheaper than current fixed platform systems.

Despite enormous strides in robotic sensing and control, humans remain superior to robots in building insightful models of our environment and recognising off-nominal situations [3]. Consequently proposed free-flying systems would be semi-autonomous, using a human-in-the-loop for exterior spacecraft operations. Thus a key consideration for the use of semi-autonomous free-flying satellites is the interface type and interaction scheme selected to enable the operator's ability to control and supervise the satellite. Veteran astronauts and EVA operators have highlighted the information-poor environment of space as a key limiting factor of current EVAs. Likewise the fixed camera views used by the SSMRS provide limited viewpoints of the robotic operation. A major concern for operation of free-flyers in space is ensuring the human operator has sufficient spatial and temporal knowledge of the free-flying spacecraft and environment to safely complete the inspection task. Aug-

mented reality (AR) presents a one possible solution, providing both a means of display and control for the free-flying satellite. Augmented reality platforms can render three-dimensional holographic displays of the exterior space station environment and free-flyer motion, and provide a means of operator control through the use of gestures, voice, hand-held controllers, and gaze. NASA's Jet Propulsion Laboratory (JPL) is currently developing a video processing pipeline that could overaly real-time video from the free-flyer satellite onto the 3D model of the station exterior.

In order to properly assess the viability of Augmented Reality for use in spacecraft inspection tasks, a suitable AR interface must be developed, and optimal methods of commanding the free-flyer determined. This thesis details investigations into the use of Augmented Reality as a means of both control and display for free-flying teleoperated satellites. An interface was developed to simulate an on-orbit inspection task, and different modes of commanding a free-flyer were assessed based on operator performance.

1.2 Previous Literature

1.2.1 Free-flying Robots for Space

Several free-flying satellite platforms have been developed for maintenance, assembly, inspection, EVA support, and intravehicular support on the ISS. The Japanese Engineering Test Satellites VII (ETS VII), launched by the Japanese Aerospace Exploration Agency (JAXA), were a pair of free-flyer satellites demonstrated on-orbit in 1997 [27]. The satellites performed several bilateral teleoperation tasks on-orbit, including the autonomous inspection and capture of one satellite by the other. The operator received haptic feedback through a 6-DOF robotic manipulator

on the ground.

Other free-flying satellites, such as the Experimental Small Satellite-10 (XXS-10) developed by the United States Air Force Research Lab [12], the Micro-Satellite Technology Experiment (MiTEx) developed by the Boeing Company [9] and the Demonstration of Autonomous Rendezvous Technology (DART) developed by NASA [52] were each used to demonstrate key technologies for on-orbit servicing, guidance and rendezvous, and autonomous maneuvering and station keeping respectively.

The Autonomous Extravehicular Robotic Camera (AERCam) was specifically designed as an inspection free-flyer for human spaceflight [7]. The satellite featured a stereo-vision camera that could provide on-orbit astronauts and a ground team with visual feedback on the exterior state of the Space Shuttle and ISS. The AERcam was remotely piloted inside the Space Shuttle payload bay on STS-87, demonstrating the viability of free-flyer robots for inspection tasks. The AERcam used a combination rotational and translational hand controller from within the cockpit, with visual feedback on the system through the Space Shuttle payload bay windows [19]. NASA subsequently developed the Miniature Autonomous Extravehicular Robotic Camera (Mini ARECam), a nanosatellite variant of the AERCam [20]. The Mini ARECam was capable of both remote piloted and supervised autonomous flight, and was designed for automatic station-keeping and point-to-point maneuvering.

NASA has previously developed several free-flying systems aimed at aiding astronaut activities within the ISS. The Personal Satellite Assistant (PSA) was an autonomous intravehicular spacecraft developed by NASA Ames to perform sensing activities such as monitoring, diagnosing and calibrating the environmental control and life support systems, and crew support such as visual monitoring, task recording, and scheduling [13]. More recently, NASA Ames has developed the Astrobee platform, to perform mobile sensor and camera tasks within the ISS and provide a

testbed for microgravity robotic research [2]. The system is remotely operated from the ground, and can also interact autonomously with the crew.

The work for this thesis was heavily inspired by the Synchronized Position, Hold, Engage, Reorient Experimental Satellites (SPHERES) testbed, a free-flying satellite developed by the Massachusetts Institute of Technology's (MIT) Space Systems Lab and flown on the ISS. The SPHERES were initially conceived as a testbed for distributed satellite systems, and subsequently expanded their functionality to support the development and test of control, autonomy, and visual navigation algorithms, and human-factors experiments [54]. While SPHERES was primarily used as an autonomous system, Stoll et al. [62] outlined a series of experiments conducted on the ISS to assess the potential benefits of human-controlled inspector satellites. The SPHERES were controlled by a human operator through a series of navigation and avoidance maneuvers using keyboard commands and visual feedback, at varying levels of autonomy and signal delay. One of the key challenges of the human-robot interaction was maintaining operator recognition of the satellite trajectory. Ground tests utilised a virtual reality interface on a computer screen to aid operators, however this system could not be implemented on-orbit. Stoll noted that human operators also struggled with reorienting after relocation of the satellite and suggested virtual reference points may aid in this reorientation. Subsequent ground studies further explored the use of SPHERES as part of a human-robot team in avoiding debris impacts [56].

1.2.2 Teleoperation Interfaces for Free Flying Robots

Teleoperation tasks pose challenges to human operators, due to the difficulty in operators' ability to maintain situational and spatial awareness and build mental models of the remote environment [18]. In teleoperated tasks, the operator's perception is decoupled from the physical environment [5] often due to a lack of sensory feedback, leading to an inability to maintain appropriate situational awareness and thus an inaccurate mental model. While operators do not need a complete mental model of a system's operation to use said system [46], conflicts between the user's mental model and the true system model can lead to degradations in performance. Performance decrements during teleoperations have been consistently identified as the result of key limitations in the user interface [6]. Safe teleoperation tasks, particularly inspection tasks, require the operator to estimate absolute and relative sizes of objects to determine the risk level of a given environment, and a decoupling of perception caused by poorly designed telerobotic interfaces can lead to scale ambiguity during operation [73]. Kanduri et al. found that remote rover operators struggled to estimate the size of geographical features from monoscopic images [29]. Restrictive fields of view, such as those provided by 2D computers screens showing live video feed, have been shown to hinder object identification and spatial orientation [30] as well as degrading depth perception and distance cues [72].

Inappropriate use of viewpoints can detrimentally affect performance in given tasks. Egocentric viewpoints refer to those from the perception of the robot, e.g. placing a camera at the end of a manipulator arm. Exocentric viewpoints are those that are placed on the body of the robot, or suspended behind and above it, providing a wider field of view. A study conducted by Thomas and Wickens found that an egocentric viewpoint induced perceptual narrowing in subjects compared to exocen-

tric displays [66], where subjects would focus on specific areas of the display to the exclusion of other information. An exocentric view of the environment, when compared to two-dimensional first-person viewpoint, can lead to improved navigational performance [53], even when navigating through a restricted or occluded environment [16].

Traditional displays for on-orbit telerobotic tasks like the SSRMS use multiple display screens and manual interfaces. While this type of multi-camera display has been shown to compensate for decreased remote perception [26], this type of display requires an operator to mentally correlate, fuse and rotate data, increasing their mental workload [45]. Studies of remote control of UAV formations have shown the optimal viewpoint changes based on subtask, with egocentric viewpoints supporting improved perception of the immediate environment, but exocentric viewpoints supporting overall task awareness [37]. While the optimal solution would seem to be presenting both viewpoints simultaneously, Olmos et al. found that using dual displays that provided both ego- and exo-centric viewpoints for navigation to a waypoint resulted in a substantial cost to performance, attributed to attentional switching costs [47].

The capability of free-flying robots to maneuver in six DOF during teleoperation tasks adds further complication when compared to terrestrial robots with limited planes of motion. Working in a three-dimensional environment with the unconstrained motion of a free-flyer can make it difficult for observers and operators to project a free-flyer's future state and determine whether or not an action has been executed properly [64]. Stoll found that the SPHERES motion patterns on board the ISS were difficult for humans to recognise [62]. Stoll noted that the lack of reference fames degraded performance due to poor spatial understanding when moving in three dimensions. Tasks may be simplified by changing the command reference

frame, depending on if the task is centered on the free-flyer or on another agent/object. In analyzing human-human interactions on cooperative space tasks, Trafton et al. identified that reference frames often changed, and required communication to coordinate this shift [67]. Thus moving from human EVAs to human-controlled robotic EVAs, use of and communication of appropriate frames is likely to be crucial.

1.2.3 Command Modes in Telerobotics

Modes of control for teleoperated systems vary greatly from robot to robot. Early teleoperated systems relied on joysticks and game controllers to enable low-level command of robots. Later systems used hand-held devices and cell phones, and more recently telerobotic control has utilised augmented and virtual reality. In his overview of teleoperation interfaces, Fong highlights that display and mode of control is highly context dependent [17]. In a review by Sigrist et al. [58], they highlight that efficacy of certain augmented strategies for display are not necessarily transferrable beyond an examined task, and highly task dependent. In order to understand performance as it relates to these variables, operationally relevant tasks are crucial.

Szafir et al. found that implementing interfaces that supported replanning, combined with three-dimensional spatial waypoints, significantly improved users' efficiency in the completion of inventory tasks and data collection [65]. Their work illustrated the need for planning phases in addition to execution phases during a free-flyer task, which a waypoint-based and planner-based (waypoint combined with scheduler) mode afforded.

Previous studies have examined distribution of control modes during robotic operations. Wang and Lewis studied the effect of waypointing issuing (specifying markers to follow), low-level teleoperation (manual, continuous driving of the robots), and

camera control (issuing desired poses to the robot camera) for a single operator and robot team completing a search and rescue task [69]. The task examined both a semi-autonomous robot state, where the human was not required but could intervene, and a completely manual state. Wang and Lewis found that when in either semi-autonomous state or manual state, operators rarely used the teleoperation mode or camera mode.

In implementing different control and command modes, like those available for commanding different segments of a manipulator arm, there will be an associated cognitive cost to switching between modes [39]. Squire et al. examined the impact of interface and interaction scheme on switching costs and overall performance (mission execution time) in controlling teams of simulated robots [60]. A flexible delegation interface enabled users to switch between a lower level *waypoint* control scheme (specifying desired goal states for robots) and higher level *play* control scheme (pre-programmed sets of maneuvers). The study found a marginally significant improvement in mission completion time when subjects could flexibly use either control scheme. While this study supported manual switching of control modes, studies using robotic arms have employed automatic and forced mode switches when completing tasks [25]. Herlant et al. found that while forcing or automating control mode switches did not improve overall performance, user satisfaction with the system greatly increased. Wang and Lewis examined performance associated with cost of switching attention between different robotic agents within a team and found that, despite expectations, switching attention between robotic agents in a manual mode and supervisory modes actually improved performance [69]. It has been posited that to reduce switching costs, interfaces should be designed to maintain the operator's awareness of the robot state, helping the operator recover robot position and pose more quickly after switching.

1.2.4 Virtual and Augmented Reality Interfaces

Virtual and augmented reality interfaces have been suggested through the 1990s and 2000s as a means of assisting in mental model formation [38], and reducing risk in complex telerobotic tasks throughout the late 1990s and early 2000s [1, 28, 35, 63]. Augmented reality addresses some of the concerns regarding viewpoint during teleoperation tasks, providing virtual models of the environment that can be manipulated for improved spatial awareness [61, 38]. While there may be decreased performance when providing an operator with both ego- and exo-centric viewpoints simultaneously [47], gesturally-controlled augmented reality allows an operator to switch between the two camera viewpoints as desired, simply by manipulating the size and position of the environment. The current work at JPL into combining real-time video feed onto virtual holograms of the environment could further help enhance spatial understanding. Previous studies into contextualized videos (combinations of videos with a model of the 3D environment) within displays have shown that users unfamiliar with an environment can achieve comparable performance to users who had worked in the environment [34, 70]. Thus in navigating the exterior of a space station with changing configuraiton (like the ISS), contextualized videos may aid users in developing a mental model of the environment.

In reviewing human-human collaboration as a means to inform human-robotic interaction, Green et al. proposes that human-robotic systems should seek to emulate human-human communication. Humans rarely achieve communication with each other through voice alone, and likewise robotic systems should recognize more than just verbal cues from operators [22]. Humans use speech, gaze, gestures and other non-verbal cues to facilitate meaning. Augmented reality systems like the Microsoft HoloLens platform enable such multimodal control through gaze, gestures, voice and

optionally game controllers [40]. Hall has found that gestures mapped most closely to the natural motions of the human arm seem most promising for gesture-based telerobotics when using gestures to control a robotic manipulator [23].

Augmented reality headset like the Microsoft HoloLens combine interface and display through this use of gestural control to directly manipulate holographic images, without the need for a physical controller. Sita et al. used the 3D visualization and gestural control of the Microsoft HoloLens to control a segmented robotic arm [59]. A 3D model of the arm was generated and users could manipulate this hologram using gestures to control a holographic target at the end-effector of the simulated arm. Kot et al. developed a virtual operator station for a teleoperated mobile robot within the HoloLens framework [33], building an interactive 3D model of a mobile robot along with fixed camera displays within the AR interface. Erat et al. used the HoloLens systems to combine live drone and human vision of an environment, providing an 'x-ray' view point into occluded environments, combined with basic gestural commands as a means of high-level control of the drone [16]. The study found that the exocentric viewpoint afforded by the HoloLens resulted in signficantly better task completion time compared to a standard egocentric drone interface using a joypad. Additionally, egocentric viewpoints in this study provided less spatial awareness compared to exocentric modes, and gestural commands like picking and placing waypoints were more readily adopted in an exocentric HoloLens viewpoint.

1.2.5 Performance measures in telerobotics

Selection of appropriate performance metrics in telerobotic and free-flyer interfaces is vital in determining the effectiveness of an interface in supporting a given task. Szafir et al. [64] used several objective measures to assess the how specific de-

sign elements of interfaces affected operator strategies and performance in free-flyer inventory tasks, including communication bandwidth (number of interface interactions/commands by the user), efficiency in task completion (total task time divided by bandwidth), and planning time used. For controlling free-flyers through various navigation and proximity operations, Stoll et al. [62] evaluated human performance through maneuver completion time and fuel consumption, forcing subjects to find a trade-off between the two factors in order to maximise overall task performance. For proximity operations, relative position of the free-flyer to the target was used as an indication of success in collision avoidance. A study of free-flyer avoidance of space debris measured collision count as an indication of performance [56]. The NASA Task-Load Index (TLX) [15] has repeatedly been used a subjective measure of cognitive load, or workload [64, 56]. The Index includes ratings of mental, physical, and temporal demand as well as performance, frustration and effort.

Previous telerobotic studies have highlighted the need for operators to maintain good awareness of their surroundings and develop accurate mental models [5]. Endsley's decomposition of situational awareness (SA) into three levels (perception, comprehension, and projection) aligns with this development of a mental model [14]. It is important to assess how interfaces support the development of a mental model, so breakdowns in these SA levels should be identified. Schneider used a post-test questionnaire, asking subjects to answer probe questions pertaining to the three SA levels, to assess subjects' SA during the collision avoidance task [56].

1.3 Research Gaps and Thesis Objectives

This thesis seeks to investigate the following research questions:

- Research Question 1: *Can augmented reality (AR) technologies be used as an ef-*

fective means of controlling a free-flying teleoperated satellite for on-orbit tasks?

- Research Question 2: *What are the effects of command reference frame and environmental risk on performance, situational awareness and workload during operation of a on-orbit robotic free-flyer?*
- Research Question 3: *How does availability of command mode affect performance, workload and strategy during operation of a on-orbit robotic free-flyer?*
- Research Question 4: *What strategies do operators adopt when using augmented reality display and interface for on-orbit tasks?*

1.4 Thesis Outline

This thesis is structured as follows

- *Chapter 1:* Motivation of the thesis, critical review of current research in the fields of on-orbit robotics, telerobotics, human-robotic interfaces, and augmented reality, objectives of thesis and addressed research gaps.
- *Chapter 2:* Development of augmented reality interface.
- *Chapter 3:* Experimental design and summary of test protocols, results and data analysis for human-subject tests, and discussion of results.
- *Chapter 4:* Conclusions of thesis, final outcomes of the study, limitations, and recommendations for future development and testing of the system.

Chapter 2

Development of Augmented Reality Interface

To conduct a study investigating the use of Augmented Reality in supporting teleoperation tasks on-orbit, and address Research Question 1, an AR interface that enabled gestural control had to be built. Gesturally-controlled Augmented Reality has not previously been applied to a scenario of commanding free-flying robots on-orbit, and so an appropriate architecture needed to be developed. Initial development focused on the incorporation of robotic hardware, however the subsequent iteration focused solely on a simulated robotic tasks to reduce complexity. This chapter describes the development of the augmented reality interface.

2.1 On-Orbit Task Concept

Following a review of the literature, and discussion with astronauts, EVA operators, and telerobotics engineers, it was determined that the simulated on-orbit task would

take the form of an inspection of a space station using a free-flying robot. The simulated task would require the operator to manuever a free-flying robot (the *inspector* satellite) around a simulated space station, looking for surface anomalies on the exterior of the station. To address the research questions listed in section 1.3, the space station simulation needed to (1) incorporate areas of high and low level risk to the inspector, (2) incorporate different methods of commanding the inspector, and (3) record measures of performance during the simulation.

2.2 Augmented Reality Interface Development

The on-orbit inspection task was originally proposed with a robotic hardware component, inspired by previous studies done by the Human Systems Lab and Space Systems Lab at MIT, which studied human-subject experiments of fatigue for on-orbit operations using the Synchroniszed Position, Hold, Engage, Reorient Experimental Satellites (SPHERES) platform [56]. A full outline of this initial development can be found in Appendix A. Based on the limitations of the SPHERES platform (see Appendix A.4), the researchers made the decision to remove the hardare component of the study, and instead emulate the functionality of the SPHERES within a simulated AR environment. This afforded the researchers greater flexibility in the design of an AR interface and space station environment, and the various command modes that could be investigated.

2.2.1 HoloLens

The Augmented Reality system used in this study is the Microsoft HoloLens Mixed Reality Platform (Figure 2-1). The HoloLens is an Augmented Reality (AR) Head

Mounted Display (HMD), first released as a development edition in 2016, with a second development edition planned for release in 2019. The device consists of a self-contained headset weighing 579g. The HoloLens is a pass-through device, so images (holograms) are projected out in front of the user, whilst being able to view the real world through the device's lenses. The HoloLens optics system is comprised of two HD 16:9 light engines, rendering a resolution of 1268×720 pixels per eye, with a refresh rate of 60 Hz. These light engines provide a total holographic resolution of 2.3M light points and a holographic density of greater than 2500 radiants (light points per radian), projecting images out throgh the HoloLens's tinted holographic lenses. The onboard sensing system is comprised of the following:

- 4 environment understanding cameras
- 1 depth camera (120° × 120°)
- 1 RGB 2MP photo/HD video camera
- 1 Inertial Measurement Unit (IMU) containing accelerometers, gyroscopes, and a magnetometer
- 1 ambient light sensor
- Integrated speakers and 4 microphones for two-way communication

The onboard cameras (depth, RGB, environment understanding) enable the HoloLens to spatially map its environment and thus allow interaction between the virtual and the real world while tracking user movements with the onboard IMU, hence the term *'mixed reality'*. The HoloLens uses an Intel 32-bit (1GHz) CPU and a custom-built Microsoft Holographic Processing Unit (HPU). The HoloLens has 2GB RAM storage and 64 GB Flash Storage, a battery life of 2-3 hours while in active use, and WiFi

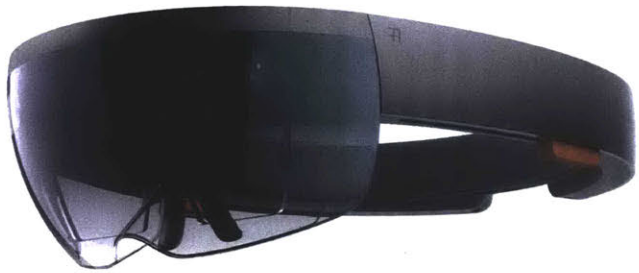


Figure 2-1: The Microsoft HoloLens Mixed Reality headset, generation 1

and Bluetooth connectivity [41]. HoloLens operates using the Windows Holographic Platform under the native Windows 10 operating system, and device's internal storage can be accessed via the Windows Device Portal, a web server on the HoloLens that can be accessed from a web browser on a PC.

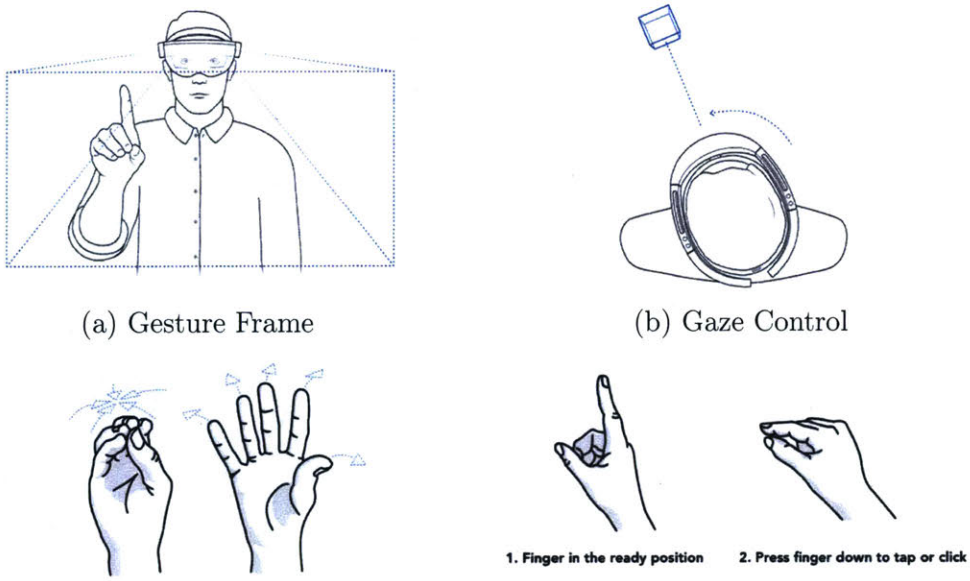
The Microsoft HoloLens has several advantages over earlier AR HMDs. The device is self-contained, not relying on a tether to an external computer in order to operate. It uses gaze tracking, gestural input, voice commands (known as Gaze-Gesture-Voice input) and/or bluetooth-connected clickers for user interaction. Unlike other AR headsets, the HoloLens is capable of running user-designed applications, and is compatible with the Unity 3D gaming engine for application design.

The nature of augmented reality devices means that users' eyes will accomodate to the focal distance of the display, while converging to the distance of the hologram. HoloLens has a fixed optical distance approximately 2 meters from the user, so accomodation occurs at 2 meters [40]. Keeping holographic content around this 2 meter plane reduces eye fatigue caused by conflict between converging and accomodating distances. Holograms should be placed in the optimal zone of 1.25-5 meters from the

user. HoloLens has a clipping plane at 30cm from the user to prevent eye fatigue due to focusing on very close objects. Any hologram closer than 30cm from the user will disappear. The HoloLens has a *Gesture Frame* measuring approximately two feet on either side of the user's head (Figure 2-2a). This region detects hand gestures. The main disadvantage of the HoloLens platform is the limited field of view (FOV) of the device, measuring approximately 35° wide. Users have a restricted view of projected holograms. The 2nd generation HoloLens development model is expected to have a 70° field of view.

2.2.2 Gestural Controls

When not being interacted with, holograms are fixed in the real-world. The HoloLens uses gaze-tracking (i.e. tracking motion of the head) to determine where users are looking in the real world (Figure 2-2b). HoloLens simulations contain a cursor object which acts like a computer mouse and is locked to the position of the user's head. The cursor always appears in the center of the frame. In order to interact with a hologram, the cursor must be on the hologram, i.e. the user head must be pointed at the hologram. The HoloLens utilizes several in-built gestures for hologram interaction. The *bloom* gesture (Figure 2-2c) is the equivalent of a 'home' button and is used for returning to the main menu of the HoloLens. When performed inside an application, it will generate a menu that allows you to exit the app. The *tap* gesture is used to interact with Holograms (Figure 2-2d). The hand is placed out in front of the user in a 'ready' position, which signals to the HoloLens that a tap gesture is about to be performed and the cursor will change shape. A single tap down is used for holographic button presses and single point interactions. If the tap is held, and the hand moved around, the holograms can be dragged around.



(a) Gesture Frame

(b) Gaze Control

(c) Bloom Gesture

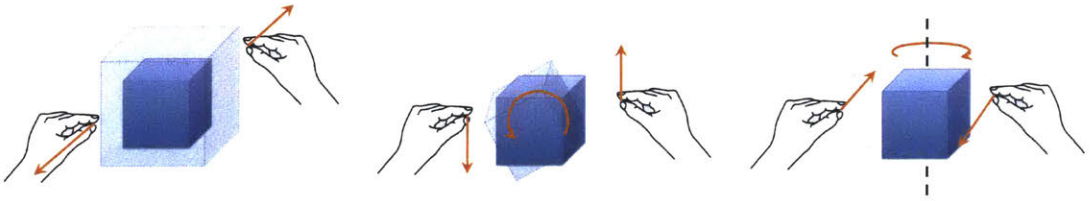
(d) Tap Gesture

Figure 2-2: HoloLens user interactions [43]

The on-orbit inspection task also allowed for scaling, rotating and repositioning of the simulation environment, to allow the user to change their view of the space station throughout the task. Repositioning of the simulation is achieved through a single-handed tap-and-drag gesture. Scaling and rotating requires two-handed manipulation. Both hands are placed in the ready gesture and then tapped and held. Moving closer/further apart scales the hologram. Moving hands in opposite directions rotates the object in the horizontal (Figure 2-3b) and vertical (Figure 2-3c) directions.

2.2.3 Simulation Environment

The simulation environment for the HoloLens was built using the *Unity 3D Gaming Engine* and *Microsoft Mixed Reality Toolkit* and compiled in *Microsoft Visual Studio*



(a) Scaling

(b) Rotating about horizontal axis (roll)

(c) Rotating about vertical axis (yaw)

Figure 2-3: Two-handed gestural controls for HoloLens. Note that rotation works best in the yaw and roll directions. Due to the reduced height of the gesture frame, HoloLens struggles to detect rotation in the pitch direction.

2017. Unity is a cross-platform gaming engine that supports the creation of 2D, 3D, VR, and AR games and simulations. Unity offers a primary scripting API in C#, as well as the importation of 3D models from computer graphics programs like *Blender*. Unity provides the framework for game development, allowing the importing of assets (models, characters, audio, textures), assembly of these assets into scenes, and the ability to add physics models, interactivity, and gameplay logic. The core building blocks of Unity games are *Game Objects*, empty containers into which lighting and audio components, scripts, and objects for the scene are placed.

Unity uses a concept called *Parenting* to organise the Game Objects. Game Objects are organised into a hierarchy which constitute a scene for your game. The top most game object is the *Parent* Game Object, and all objects grouped underneath it are *Child* Objects. Child Objects inherit the movement and rotation of the Parent Game Object. Figure 2-4 shows the hierarchy of the on-orbit inspection simulation developed for this thesis. A number of Game Objects are from the Microsoft Mixed Reality Toolkit, a GitHub repository of scripts and assets developed by Microsoft and other developers specifically for use with the HoloLens. The `MixedRealityCameraParent` is the standard Camera Object used when build-

ing HoloLens simulations, placing a camera within the game at the location of the HoloLens user's head. `InputManager` is the HoloLens Object which organises all gestural, verbal and gaze input that is detected, and `DefaultCursor` provides the cursor users see within the HoloLens that lets them interact with the simulation. `DirectionalLight` is the default Light Object within Unity to illuminate the scene, as holograms would not be lit by ambient light within the room.

The remainder of the Game Objects were custom built for this simulation and will be discussed in the subsequent sections. The primary Parent Objects are the `newISSmodules2`, which contains the simulation environment, and `ControlPad`, which contains the user interface for controlling the inspector.

The simulation environment was designed to simulate an on-orbit inspection task of a space station. The environment consisted of a space station model, a floor surface, an inspector satellite, a control pad, and a waypoint (if active). Figure 2-5 depicts the holographic environment. It should be noted that all distance measurements given regarding the simulation and the results of the simulation are given in Unity unit measurements. In Unity simulations, 1 Unity unit typically corresponds to 1 meter, however HoloLens simulations allow scaling of objects and thus distance units are subject to change. For this study, all distance were measured relative to the world frame of the space station and units scale when objects are scaled, so relative distances between objects were maintained even under scaled conditions.

Space Station

The space station model was constructed in the Blender computer graphics program and imported into Unity. To address aspects of environmental risk outlined in Research Question 2, the design of the space station incorporated various ge-

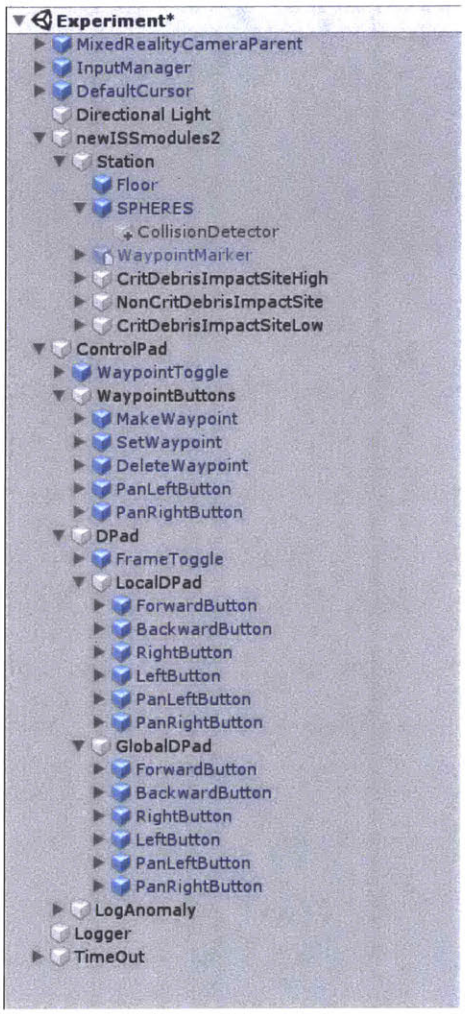
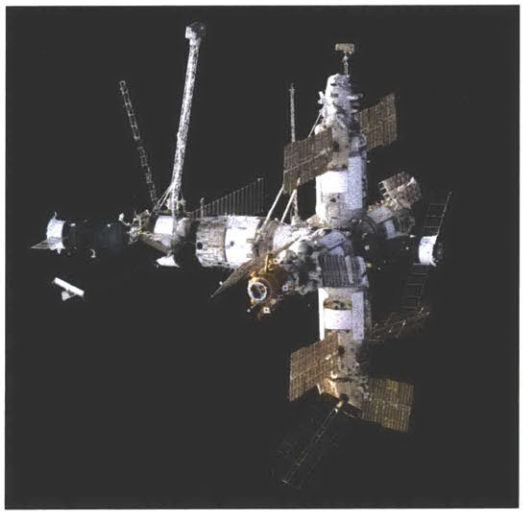


Figure 2-4: Hierarchy of Game Objects for on-orbit inspection simulation. Drop-down arrows have been used to show the key Parent and Child Objects used in the simulation.

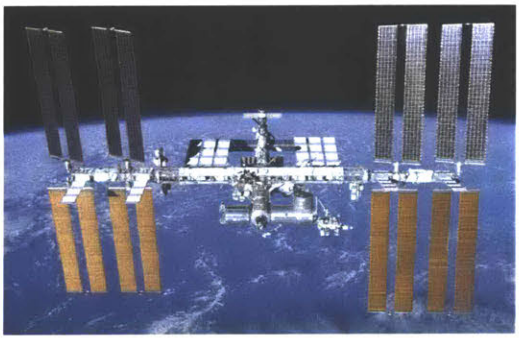
Figure 2-5: Simulation environment within Unity. The inspector is shown in red. The space station, floor, and control pad are shown.

ometries, classified as either high or low risk. Inspiration was drawn from existing and previous space stations designs (Figure 2-6) in the design of the station and these risk areas. High risk areas were areas of limited clearance for the inspector and required increased localization to inspect the region of interest, such as highly concave geometries (e.g. nodes between connecting modules), high vertex objects (e.g. complex attachments on the station like communication dishes), and protrusions (e.g. solar panels). Low risk areas were areas with flat or convex geometries, or limited vertices (e.g. the surface of a module). The final station model is shown in Figure 2-7. The station is comprised of two modules joined parallel, and a third module perpendicularly attached. These modules are modelled after the *Harmony* and *Columbus* modules onboard the ISS. Two solar panels and a communication dish are attached to the exterior, and a cupola is attached to the end of one module. Figure 2-8 highlights the areas of high and low risk around the station model.

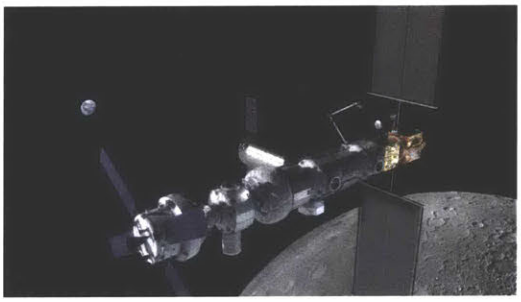
In order to detect when the inspector satellite collided with the station, *colliders* needed to be implemented on the station model. *Colliders* are invisible objects



(a) Mir Space Station

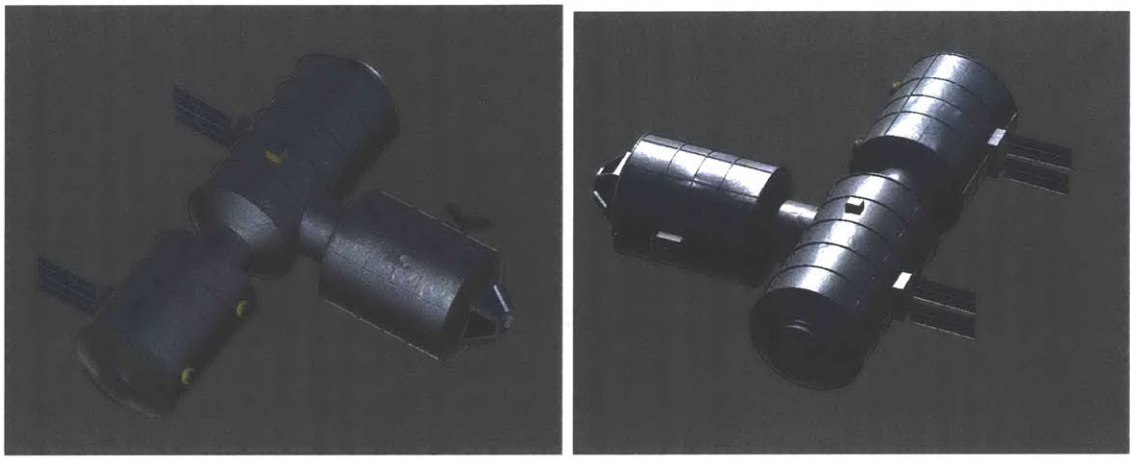


(b) International Space Station



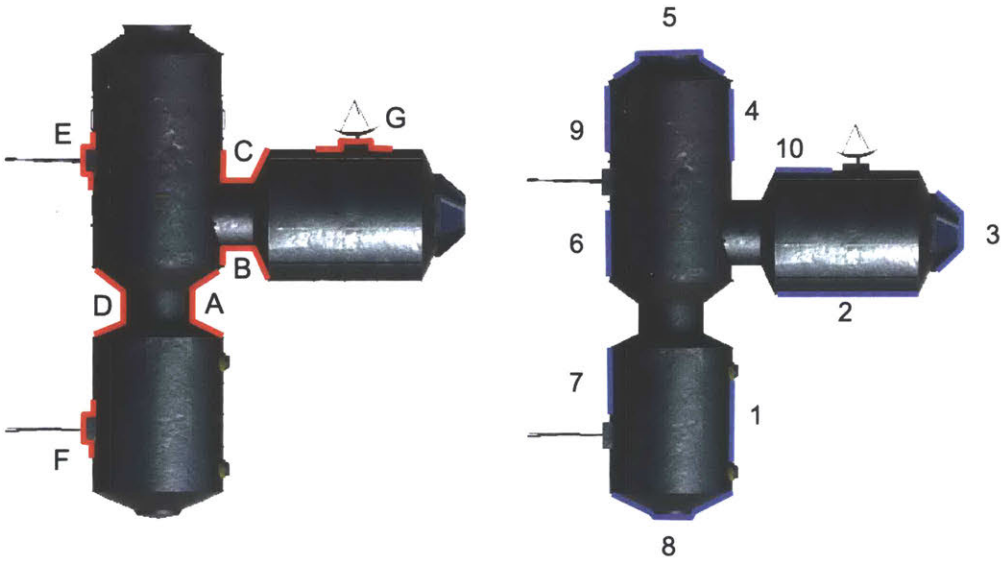
(c) Proposed NASA Lunar Gateway

Figure 2-6: Previous, current and future space stations.



(a) Forward isometric view (b) Aft isometric view

Figure 2-7: Final space station model constructed in Blender.



(a) High risk environments (b) Low risk environments

Figure 2-8: Location of high and low risk environments around the exterior of the space station. High risk areas are labelled with letters, low risk areas are labelled with numbers.

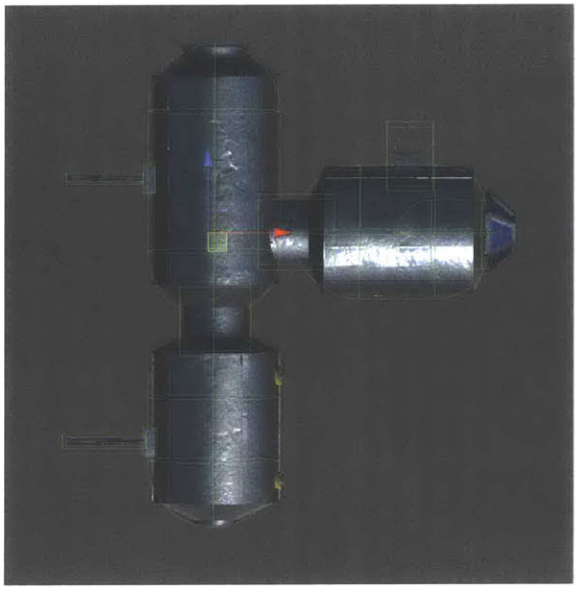
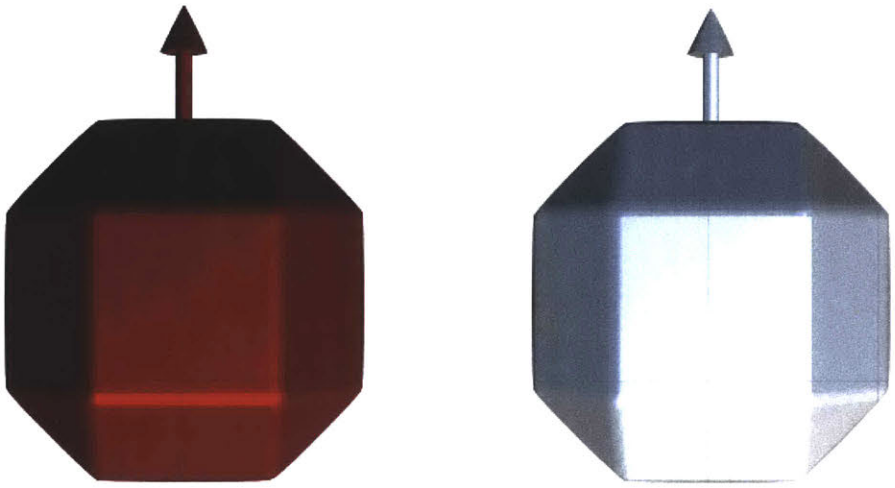


Figure 2-9: Colliders used for the space station model in Unity. Colliders are shown as green transparent shapes.

described by mathematical equations which are used to detect physical collisions between objects within the environment. The least-processor intensive colliders are *primitive* colliders; box (rectangular prisms), sphere, and capsule (two hemispheres and a cylinder). Mesh colliders can be used to exactly match the contours of a given object, however thiey are more computationally intensive and cannot be used to collide with other mesh colliders. For the on-orbit inspection simulation, the space station was approximated by a number of box and capsule colliders (see Figure 2-9). The inspector collision detection will be discussed below.

Users were able to move, scale and rotate the space station (and all Child Objects), using the HoloLens two-handed gesture commands. The HoloLens script `TwoHandManipulatable.cs` (available from the Mixed Reality Toolkit) is attached to the space station Game Object in order to make it interactable with two-handed gestures. This script can restrict two-handed gestures to operate in specific axes,



(a) Default inspector state (b) State of inspector on collision

Figure 2-10: Inspector satellite model in Unity.

and for specific functions (moving, scaling, rotating). For this simulation, two-hand manipulation was restricted to scaling and rotating with no axis restrictions.

Inspector Satellite

The inspector satellite simulation model was originally designed for the hardware experimental tests, and as such is modelled off the SPHERES platform. The inspector model was also built in Blender, based off the polyhedral design of the SPHERES. The forward facing face of the inspector is identified by an arrow representing the normal vector of this face. The default inspector color is red, to match the SPHERES satellite (Figure 2-10a).

As would be the case with the ground-based SPHERES platform, the inspector is restricted to motion in the X-Y plane, and rotations about its vertical Z-axis.

The inspector satellite object (`SPHERES`) is a child of the station object (`Station`), and consequently whenever the station is scaled, rotated, or repositioned by the operator, the inspector moves with it, maintaining its relative position. The inspector is controlled via the control pad, which is discussed in Section 2.2.4. When the inspector collides with the station object, it changes color to white to indicate a collision (Figure 2-10b). Unity uses in-built trigger functions `OnCollisionEnter` and `OnCollisionExit` to determine when an object interacts with a collider. A code containing these functions is attached to the inspector object and whenever the inspector interacts with the station collider, this triggers the above functions, records the collision, and changes the color of the inspector.

2.2.4 Command Modes for Inspector

The inspector is controlled by the user via the `ControlPad` Game Object, which is decomposed into 3 methods of command; Waypoint Mode, Global Mode and Local Mode. The control pad is composed of holographic buttons that register single taps from the user which translate to commands for the inspector. Buttons for a single command mode are visible at one time, and users can switch between command modes by tapping the frame toggles in the uppermost left and right corners of the control pad. While the station object can be repositioned by the user, the control pad remains fixed in the world frame, only rotating about its axis to ensure it always faces the user if they move slightly.

Waypoint Command Mode

Waypointing is a high-level method of commanding robots that uses intermediate target points along a route as goal states for the robot. These waypoint target points

specify a high-level state (usually position coordinates and orientation), rather than dictating the precise control movements a robot must follow. The SPHERES robots use waypoints as a means of navigation. Within the on-orbit inspection simulation, the Waypoint control pad is used for the various functions within the Waypoint Mode (Figure 2-11). Once active, waypoints appear as red flags in the simulation (Figure 2-12). The Waypoint system is based off the SPHERES robot, which can only send data packets containing one waypoint at a time, therefore within the simulation only one waypoint can be active at once. Another waypoint cannot be placed until the inspector has reached the current waypoint, at which point the current waypoint deletes.

The Waypoint Mode has 4 different command functions

- *Generate* - Waypoints are not always active in the simulation, they must be generated. The Generate button on the waypoint control pad spawns a flag marker at the location of the inspector at the beginning of the simulation. Waypoints always spawn in the same position.

- *Move* - Once a waypoint is generated, it can be moved around the simulation to the desire location by tapping and dragging the flag marker.

- *Set* - Once the user is satisfied with the position of the flag marker as a target position for the inspector, the waypoint must be set by tapping on the Set button. This commands the inspector to move towards this waypoint. The waypoint will disappear once the inspector reaches its location.

- *Delete* - A currently active waypoint can be deleted by pressing the Delete button. Delete can be used prior to the Set function if the user loses the waypoint, or whilst the inspector is en route to the waypoint. If the inspector

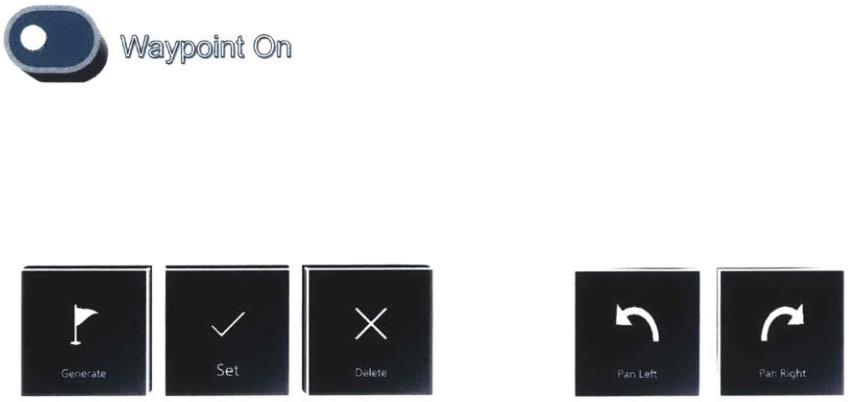


Figure 2-11: Control pad in Waypoint Mode. The toggle in the upper left indicates that Waypoint Mode is switched on.

Figure 2-12: Waypoint marker as it appears in the simulation

is en route when a waypoint is deleted, the inspector will stop at its current position.

The Pan Left and Pan Right buttons rotate the inspector $\pm 15°$ about its vertical axis. The inspector moves in a straight line from waypoint to waypoint at a fixed speed of 1 Unity unit per second. No path planning algorithm has been incorporated into this simulation.

Global Command Mode

The Global Command Mode is a low-level command mode that commands the inspector to move in specific directions and orientations, rather than giving it target locations. Commands in the Global Command Mode are given relative to the Global World Frame of the space station (Figure 2-13). Directional command is achieved by tapping on one of the directional pad buttons, and rotational command is achieved by tapping the panning buttons, as described below:

- *Forward:* +Y
- *Backward:* -Y
- *Right:* +X
- *Left:* -X
- *Pan Right:* +Z rotation
- *Pan Left:* -Z rotation

Holographic buttons could not be scripted to respond to a tap-and-hold, therefore only single taps on buttons could be implemented. Thus the Global Command Mode has discrete, rather than continuous, control. Each directional movement moves the inspector 0.2 Unity units, and each rotation movement rotates the inspector 15°.

Local Command Mode

The Local Command Mode is operated in an identical manner to the Global Command Mode, but moves the inspector relative to its own local body frame (Figure 2-15). Commands are input by pressing buttons on the Local Command directional

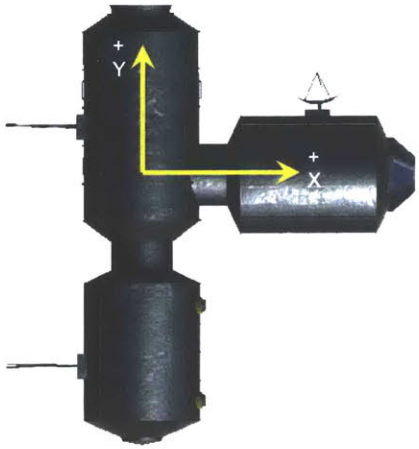


Figure 2-13: Global world frame of the space station, used for Global Command Mode. Note that the + Z axis points up out of the page.

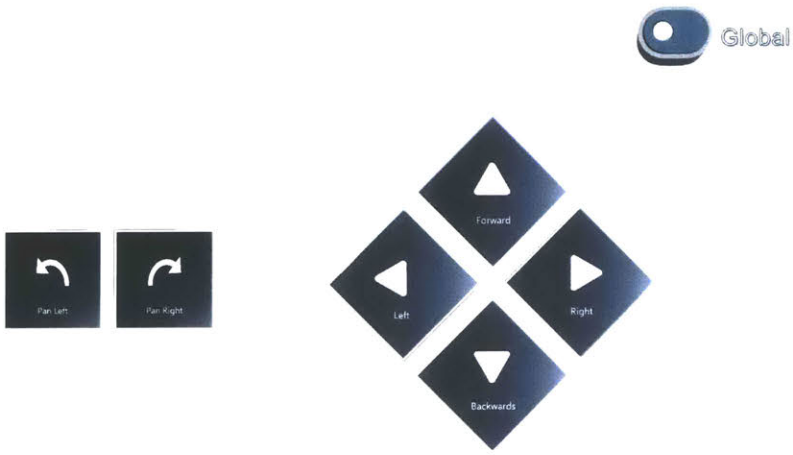


Figure 2-14: Control pad in Global Mode. The toggle in the upper right indicates that Global Mode is switched on.

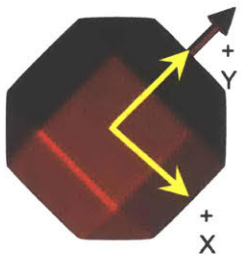


Figure 2-15: Local body frame of the inspector, used for Local Command Mode. Note that the + Z axis points up out of the page.

pad and panning buttons (Figure 2-16). Rotating the inspector about its vertical axis will rotate this local frame. The Local Mode is active when the toggle in the upper right corner of the control pad reads 'Local' and is colored gray.

2.2.5 Anomaly Detection

The goal of this simulation was to detect anomalies on the surface of the space station. Each simulation run contained a number of hidden anomalies, classified as *critcal* and *non-critical* (Figure 2-17).

To simulate 'detection' of anomalies by the inspector, anomalies remained inactive until certain conditions within the simulation were met:

1. *Proximity:* Inspector is within 0.5 units of the anomaly

2. *Orientation:* Inspector is oriented such that the anomaly falls within its camera field of view.

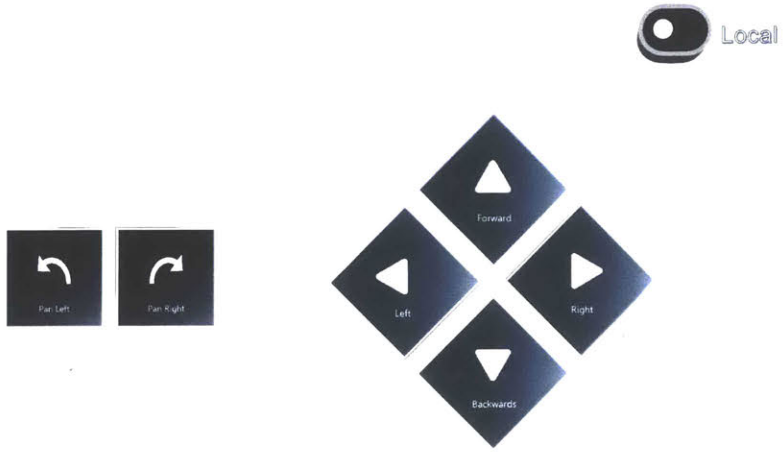
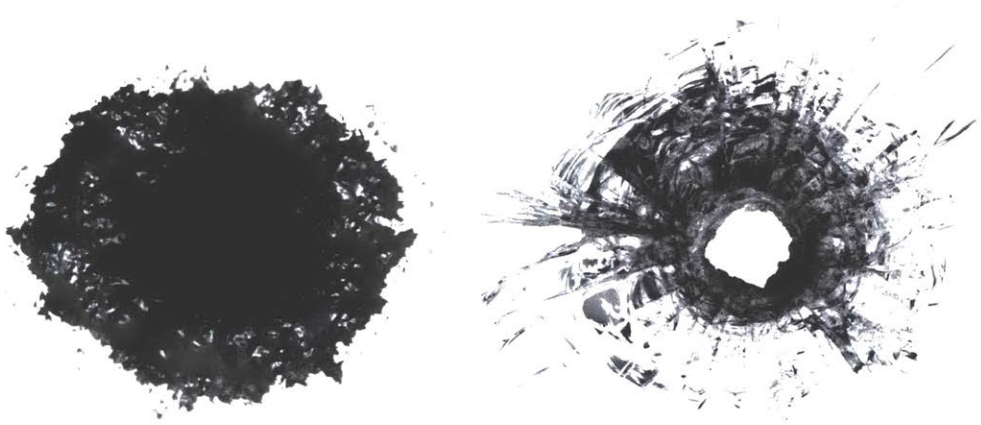


Figure 2-16: Control pad in Local Mode. The toggle in the upper right indicates that Local Mode is switched on.



(a) Critical Anomaly (b) Non-Critical Anomaly

Figure 2-17: Anomaly images used on the surface of the space station model. Anomalies were approximately the same diameter as the inspector.

The inspector satellite emulated the SPHERES detection capability by having simulated cameras on its faces. There were two Inspector configurations used in the simulation tests. For configuration 1, the inspector was assumed to a have 360° FOV (i.e. cameras on all faces) and so orientation of the inspector was not a consideration in detecting anomalies. For configuration 2, the inspector was assumed to have a more realistic single-front facing camera with a FOV of 45°. For the second configuration, anomalies would only be detected when the relative angle between the normal to the front-face of the inspector, and the vector from the inspector camera to the anomaly must be less than half the camera field of view (i.e. 22.5°)

$$\theta = \cos^{-1}\left(\frac{\vec{n_i} \cdot \vec{v_{i,a}}}{\| \vec{n_i} \| \| \vec{v_{i,a}} \|}\right) \tag{2.1}$$

where n_i is the normal to the inspector, and $v_{i,a}$ is the vector from the inspector to the anomaly (Figure 2-18). Once anomalies are detected by the inspector, they remain active for the remainder of the simulation.

Critical anomalies were required to be logged by the operator, by airtapping on the *Log Anomaly* button (Figure 2-19). Non-critical anomalies were not logged. Logging was incorporated to encourage operators to fully inspect the station and make a determination on whether an anomaly was critical or non-critical, rather than just a cursory glance.

Exporting Simulation Data

A method had to be developed to record and export data during the simulation. The Parent Object `Logger` keeps track of the game time, camera (user head), station and inspector states, and anomalies. The Logger generates a HoloLens empty text file within the HoloLens' local drive, using the following command:

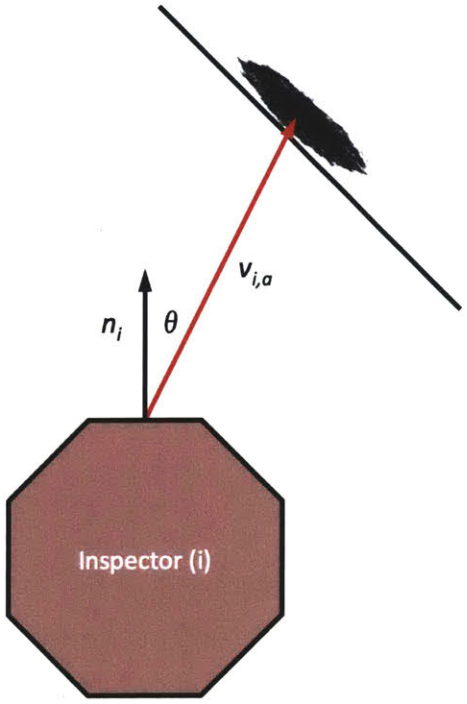


Figure 2-18: Diagram of anomaly detection for inspector configuration 2

Figure 2-19: Log Anomaly Button on the Simulation Control Pad.

```
path = Path.Combine(Application.persistentDataPath, "HoloLogFile.txt");
```

Tracked data is then packaged as a string array and appended to the text file as `File.AppendAllLines(path,<stringarray>)`. This log file can then be accessed via the Windows Device Portal. The log file must be downloaded at the end of each simulation run, as the file is locally stored and overwritten each time a simulation is opened.

2.3 Summary of Augmented Reality Interface Capabilities

An augmented reality simulation environment was developed for the Microsoft HoloLens platform. The environment contains a three-dimensional representation of a space station and inspector satellite, to be used for a simulated on-orbit inspection task. Control of the inspector satellite is enabled via a virtual control pad that operated in three command modes; Waypoint Mode, Global Mode, and Local Mode. The space station could be manipulated gesturally to alter its size, position, and orientation, thus changing the operator's viewpoint of the station environment. The simulation contains anomalies that can be detected by the operator, dependent on the orientation and proximity of the inspector. Simulation data is logged into a local textfile and exported to a computer for processing via the Windows Device Portal. The AR simulation was used in a human-subject study, described in Chapter 3.

Chapter 3

Experimental Evaluation of the Simulation Environment

The chapter describes the experimental design, testing procedures, data collection and analysis, and experimental evaluation for the human study conducted for this thesis. After the successful development of the Augmented Reality Interface, a human-subject study was conducted to examine how augmented reality is used to detect anomalies in a simulated space environment. The research questions addressed were: (2) What are the effects of command reference frame and environmental risk on performance, situational awareness and workload during during operation of a on-orbit robotic free-flyer? (3) How does availability of command mode affect performance, workload and strategy during operation of a on-orbit robotic free-flyer? (4) What strategies do operators adopt when using augmented reality display and interface for on-orbit tasks? Testing also aimed to demonstrate proof-of-concept of the interface as a viable method of commanding inspector satellites for on-orbit inspection tasks. Results will inform subsequent iterations of the interface design.

3.1 Design and Hypotheses

The Augmented Reality simulation was designed to simulate an on-orbit inspection task for a crewed space station. Subjects were required to command an inspector satellite around the exterior of the space station, inspecting the surface for anomalies. Subjects had 390 seconds to complete an inspection.

Research questions (2) and (3) were addressed through analysis of recorded dependent variables during this inspection task. *Performance* in the context of these research questions was assessed based on the initial criteria given to the subjects (in order of priority), through analysis of three dependent variables: collision count, missed anomaly count, and percentage of station inspected. The research questions were formalized with the following hypotheses:

- *Collision Count:* Number of collisions between the inspector satellite and the space station

  H1: Subjects will have fewer collisions between the inspector satellite and the station when operating in a globally oriented frame of reference (i.e. global command mode or waypoint).
  H2: Subjects will have have superior collision avoidance when all command modes are available to the operator.

- *Missed Anomalies Count:* Number of anomalies passed by the inspector satellite but not detected or logged by the operator.

  H3: Operation in the body reference frame of the inspector will enable superior anomaly localisation in high risk areas of the station.
  H4: Operation in the world reference frame will enable superior anomaly

localisation in low risk areas of the station.

H5: Anomaly localization will be higher when operating in the unfixed command mode.

- *Percentage of Station Inspected:* the percentage total of the station that could theoretically be seen from the inspector's path, given the detection distance (0.5 Unity units). *Note:* orientation of the inspector was not taken into account.

  H6: Waypoint command mode will enable a higher percentage of the station to be inspected, compared to other command modes.
  H7: Percentage of station inspected should be maximised when operating with full access to all command modes.

Workload, as described in research questions (2) and (3) was subjectively assessed through the administration of the NASA Task Loading Index (TLX) survey. The following was hypothesized:

H8: Waypoint command mode will require the least cognitive load.

Research questions (3) and (4) were assessed through comparisons of strategy, interactions with the augmented reality system, and button interactions during the task. It was hypothesized (H1, H3, H4, H6) that different command modes support different performance measures overall and within different environmental risk areas. Thus it was hypothesized:

H9: Different command modes will be selected during a trial dependent on perceived level of environmental risk and performance
H10: Subjects will adopt different strategies when command modes are fixed compared to unfixed.

Additional dependent variables considered in the discussion were the *Completion Time* (total time taken for subject to report a full inspection of the station) and *Path length* (total distance the inspector satellite travelled). These variables were used to support analysis of overall strategy.

3.2 Methods

3.2.1 Subject Recruitment

This study was approved by the MIT Committee on the Use of Humans as Experimental Subjects (COUHES) and written informed consent was obtained from all subjects who participated in the study. Subjects were compensated USD$40.00 for participation in both testing and training for the study. Subjects were recruited from the MIT Department of Aeronautics and Astronautics. All subjects were either graduate student and post-doctoral researchers. These groups are considered representative of the astronaut population, being technologically literate in a manner similar to that of astronauts, and falling within the age range of 20 - 50 years old. Potential subjects were contacted via a COUHES-approved recruitment email (Appendix D.2), or via word of mouth from the study researchers. Interested parties were asked to schedule a training and testing time over two consecutive days with the researchers.

Subjects were approved for the study provided they had no self reported auditory or visual impairment that would impede the use of the HoloLens platform. Additional exclusion criteria were an inability to see a computer screen with corrected/uncorrected vision, less than full mobility in hands and arms, or a history

of vestibular/visual/auditory discomfort or motion sickness/nausea when using augmented or virtual reality (see the Screening Matrix in Appendix D.4).

Additional screening took place during the training day. Subjects were excluded from the study if they were unable to complete the three training programs. Subjects were asked to ensure they got 6-8 hours sleep on nights preceeding their training and testing days, to avoid fatigue as a confounding factor.

3.2.2 Experimental Design

The indepedent variable for this experiment was the command mode, which could be either a Fixed Command (FC) or an Unfixed Command (UC) for the duration of the trial. Command mode had four levels (Waypoint - FC, Global - FC, Local - FC, and Unfixed - UC). Each subject performed the inspection task in each mode twice, for a total of eight trials. To address learning effects, order of trials was randomised for the first three FC trials. This order was then mirrored for the subsequent three FC trials, so that the average time (since beginning all tests) for each FC command mode was the same. Each FC trial contained two critical anomalies (one in a high risk location, one in a low risk location) and one non-critical anomaly (low risk location). These locations were randomised across each trial, so every trial had different anomaly locations (Appendix C.1).

After completing the FC trials, subjects then completed two UC trials. In the UC trials, command mode was not dictated by the researcher and could be freely selected and toggled between by the subject during the trial. The order of the UC trials was not randomised, with all subjects completing UC Scenario 1 (UC 1) followed by UC Scenario 2 (UC 2). UC 1 had zero critical anomalies and 4 non-critical anomalies (two high risk, two low risk). UC 2 had one critical anomaly (high risk) and three

non-critical anomalies (all low risk). A full testing matrix is given in Appendix D.1.

After starting the human trials, the researchers made the decision to alter the design of the inspector satellite, to assess participant performance and strategies. The first six partipants (Group A) completed all trials using an inspector with cameras around its circumference, providing 360° field of view. In this scenario anomaly detection was only dependent on the detection distance, and not on the orientation of the inspector. The final six participants (Group B) completed all trials using an inspector with a single camera on one face, providing 45° field of view. Anomaly detection for Group B was dependent on both the detection distance and the orientation of the inspector.

Table 3.1 shows the eight treatment types, with Subject Group (two levels) and Command Mode (four levels).

	Waypoint (FC)	Global (FC)	Local (FC)	Unfixed (UC)
Group A	Group A + Waypoint	Group A + Global	Group A + Local	Group A + Unfixed
Group B	Group B + Waypoint	Group B + Global	Group B + Local	Group B + Unfixed

Table 3.1: Matrix of command test treatment types

3.2.3 Testing Environment and Hardware

The study used the Microsoft HoloLens Mixed Reality platform. The simulated virtual environment was created using the Unity Gaming Engine. Unity built each gaming simulation as a Microsoft Visual Studio solution file, and the simulation was loaded from a connected computer onto the HoloLens from Microsoft Visual Studio.

An application was generated on the HoloLens which was then selected by the study participant.

A consistent light and audio environment needed to be maintained for all tests, as well as sufficient space for subjects to move around and perform gestures. The testing environment also needed to feature minimal distracting objects (such as wall decorations or furniture) that might affect the visibility of the holograms. All human subject testing took place in an office, performing the tests in ambient light conditions between 9 am and 5 pm, with the light level controlled by window blinds. The room was kept quiet, with only the subject and researcher present. Subjects were instructed to stand for the duration of each trial, resting between trials, and were told to select a corner of the room to stand in, facing the open space. The room provided an approximate 2.5 $\times$ 2.5 meter open space for subjects to move around in.

3.2.4 Experimental Procedure

Day 1 (Training)

After providing consent, participants were given a brief overview of the project motivation and scope, and an introduction to the training and testing procedures. To ensure consistent information between participants, the researcher followed the script in Appendix D.5. Participants answered basic demographic questions (see Demographic Matrix in Appendix D.6) and then given the Study Training Document (Appendix D.7) and asked to review the section on the HoloLens. This document gave an overview of the HoloLens platform and introduced the basic gestures of *bloom*, *tap*, and the two-handed gestures required for rotating, scaling, and moving the space station. Participants were trained using three programs. The first training program was a standard HoloLens tutorial to teach the in-built gestural commands of the HoloLens

(*bloom* and *tap*). The second training program was custom-made by the researchers for this study. The program stepped the subject through the required two-handed gestures for positioning, scaling, and rotating a model of the space station. To pass the training program, subjects were required to manipulate the space station model to match the position, scale, and orientation of a target model. Participants were then able to take a break and finish reading the training document, which outlined the various commands modes and button interfaces required for the study. The final training step was a full mock-up of the test simulation, introducing the subject to the three fixed command mode test conditions. Subjects were instructed in the use of each mode and then required to navigate the inspector along a designated path first in the local mode, then the global mode, and finally the waypoint mode. To pass the training, subjects were required to correctly follow these paths in each mode within 10 minutes (per mode). Subjects were guided to the location of two anomalies (one critical, one non-critical) by the researcher and purposely collided with the station to become accustomed to the detection distance, inspector's field of view (Group B only) and collisions with the station. To aid subjects in reaching a steady state level of performance and help mitigate learning effects, subjects were then encouraged to continue to play with the simulation and the various command modes until they were comfortable with the system.

Day 2 (Testing)

Upon successful completion of the training programs, participants completed 8 experimental trials that were up to 390 seconds, with breaks between trials to allow for subject recovery and time for the test administrator to load the next simulation. Each trial required the subject to maneuver the inspector satellite around the space

station and locate anomalies on the surface of the station.

Before beginning testing, subjects reviewed key information from the training day (gestures, command mode functionality). Subjects were told to complete each trial with the follwoing performance goals, in order of priority:

1. Avoid collisions between the inspector and the space station.

2. Locate anomalies, both critical and non-critical, and log critical anomalies using the *Anomaly* log button.

3. Attempt to fully inspect the space station as quickly as possible. The simulation will time out after 6.5 minutes, however subjects may terminate the simluation early if they are satisfied they have completed their inspection.

Subjects were then given an overview of the testing procedure and briefed on the FC and UC tests. They were informed that each trial would have multiple critical and non-critical anomalies. At the conclusion of the six FC trials, subjects were informed they would now be completing the UC trials, with free choice of command mode during the trial. After each FC and UC trial, subjects completed a post-trial Situational Awareness questionnaire (Appendix D.9) and a NASA Task Load Index (TLX) survey to subjectively assess their workload. A qualitative post-test questionnaire was administered at the conclusion of the two UC trials, to get participant feedback on the usability of the AR system (Appendix D.10). All subjects were administed a Vandenberg Mental Rotation Test (MRT) at the conclusion of testing [68].

3.2.5 Data Collection

Trial data was collected on subject performance and interaction with the AR environment. Data was recorded via the HoloLens logger object, which would print data values to a log file stored in the HoloLens Web Portal. After each trial, this log file could be accessed via a laptop and downloaded. These data were used in custom Matlab software.

The following state data was recorded every 10 frames (at approximately 30 frames per second, however framerate can vary slightly due to CPU load), along with a timestep:

- *Camera State:* the 'camera' within the HoloLens environment refers to the location of the HoloLens headset (i.e. the subject's head) and is always aligned to the direction the subject's head is pointing. Camera state refers to both the position and rotation of the subject's head in virtual world coordinates, output as a 3D position vector and quaternion.

- *Station State:* the virtual world position and orientation of the holographic space station, output as a 3D position vector and quaternion.

- *Station Scale:* the scale factor of the space station.

- *Inspector State:* the position and orientation of the inspector satellite relative to the space station, output as a 3D position vector and quaternion.

The following event-based data was recorded whenever the corresponding event occured, along with a timestep:

- *Button Press:* all button presses (waypoint, global, local, anomaly logged) were recorded, along with the button type.

- *Waypoint Position:* when a waypoint is *set*, its position and the corresponding timestep of the set press are recorded.

- *Anomaly Detected:* the first time an anomaly is detected by the inspector, together with the anomaly type and risk region.

- *Collision:* timestep of collision between station and inspector.

The raw data was processed in MATLAB and used to develop the following dependent variables:

- Collision Count: total number of collisions between the inspector and the station.

- Percentage of Station Inspected: gives an indication of how much of the station was actually seen by the inspector. It was calculated by discretizing the station boundary, and checking which discretized points fell within detection distance (0.5 Unity units) of a recorded inspector position. Inspector orientation was not taken into account for the results of either Group.

- Anomalies Missed: an appropriate metric needed to be developed to assess how successful subjects were at detecting anomalies, since the number of anomalies detected was coupled to the percentage of the station inspected. Anomalies missed was calculated as follows

$$A_{missed} = A_{passed} - A_{detected} \tag{3.1}$$

where the number of anomalies passed (A_{passed}) was defined as the number of anomalies located at points on the station that the inspector path passed by,

even if not sufficiently close to detect them. Anomalies detected ($A_{detected}$) was determined from raw data.

- Total Test Time: time taken for the subject to be satisfied that they have completed a full inspection of the station. If the simulation timed out before a full inspection could be completed, this value would be 390 seconds.
- Total Path Length: total distance the inspector traversed within a single trial, determined from inspector state values.
- Button Interactions: count of button presses during each trial, and command mode changes during UC trials.

NASA TLX survey data was recorded using the NASA TLX iPhone app and exported to Microsoft Excel, then parsed to MATLAB for data analysis. Workload was recorded as a dependent variable, using the NASA TLX Composite Score as an indication of overall workload. Component workload scores were also analysed.

3.2.6 Statistical Analysis

All statistical analysis was completed in MATLAB 2017b. Continuous dependent variables (percentage of station seen, path length, workload) were assessed using a n-way Analysis of Variance (ANOVA), with factors subject (random), group (fixed) and command mode (fixed). Subject was a nested factor within group. If a significant main effect was found in ANOVA, appropriate post-hoc pairwise comparison analysis was performed to assess the effect.

Ordinal data (collision count) was assessed using a nonparametric Friedman test, with a fixed factor (command mode). If the Friedman test revealed a significant effect, a post-hoc Wilcoxon Signed-Rank test was performed.

Effects of order were not assessed, as order of the trials was randomized across each subject, and within group A and B each order was only performed once. Effects of learning were not incorporated into the statistical model, given the small sample size.

3.3 Results

The following section reports the results of the human-subject experiments. The results section will be organized as follows:

- Subject demographic information and mental rotation test scores
- Effects of command mode on collisions
- Effects of command mode on anomaly detection - anomalies missed will be examined across high and low risk areas.
- Effects of command mode on percentage of station seen. Path length and total time will also be discussed.
- Effects of command mode on measures of workload
- Effects of commmand mode on button interactions for each trial, and results of command mode changes in UC trials.

Results across all subjects will be presented, and subjects 5 and 12 will be highlighted, as they exhibited different strategies and performance to the rest of the subjects.

3.3.1 Subject Demographics

Twelve subjects completed the full study protocol, six in group A (three men, three women) and six in group B (five men, 1 woman). All subjects were aged between 23 and 29 years (mean age 26.08, SD: 1.93). All twelve subjects reported frequent use of computers and touch screen devices (smartphones, tablets). Eight reported previous limited experience with either virtual or augmented reality. When asked

about previous experience to the current study, seven subjects reported relevant experience: video gaming on either a console or computer, work with 3D modelling through Vicon or SolidWorks, and previous experimental work with telerobotics. In both cases of reported experimental work, the experiment was conducted in the Human Systems Lab at MIT, examining the use of body motion as a means of controlling a teleoperated robotic arm.

Mental Rotation Ability

The mean scores for the Vandenberg MRT were 21.33 for Group A (SD: 6.19) with a minimum score of 15 and maximum score of 32 out of a possible 40. Group B scored a mean of 29 (SD: 6.99) with a minimum of 20 and maximum of 40 (Figure 3-1). A two sample t-test revealed no significant difference in MRT scores between Group A and Group B ($t(df = 10) = -2.0126$, $p = 0.0719$).

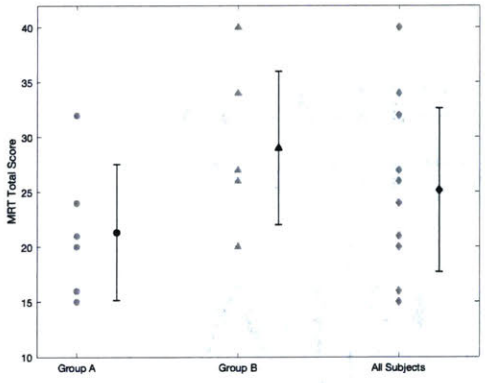


Figure 3-1: MRT total scores. Gray markers depict individual scores, black markers and error bars show the overall mean and standard deviation for the indicated group

3.3.2 Effects of Command Mode on Collisions

Figure 3-2 shows the average collision counts across the command modes. A Wilcoxon Rank Sum Test was performed comparing Group A to B within each command mode, to see if data could be pooled. The Wilcoxon Rank-Sum test revealed no significant difference between Groups A and B in any command mode (Waypoint: $p = 0.731$, Global: $p = 0.584$, Local: $p = 0.290$). As such, results from Groups A and B were pooled for further analysis.

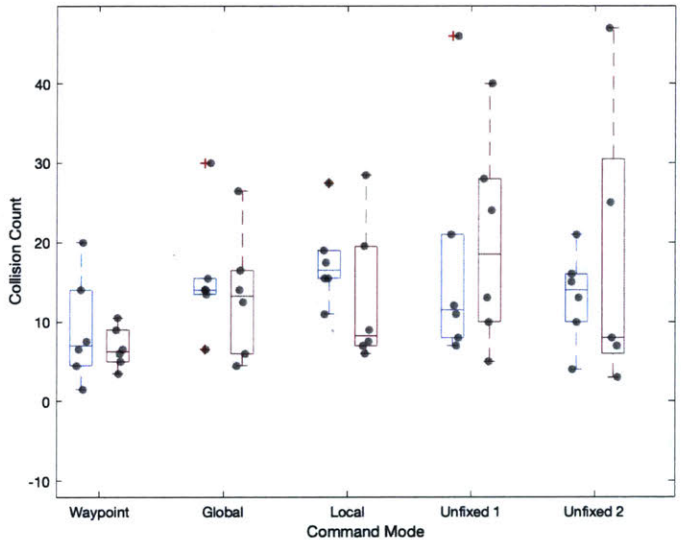


Figure 3-2: Subject average collision count across command modes. Blue boxplots indicate group A, red boxplots group B, and gray markers show the subject average.

A Friedman test (with Groups pooled) supported a significant effect of command mode on collision count ($\chi^2(df = 3) = 12.875$, $p < 0.001$). Post-hoc comparisons using the Wilcoxon Signed-Rank test found that Waypoint Mode had significantly fewer collisions than Global ($p = 0.00488$), Local ($p = 0.00195$) and Unfixed Modes ($p = 0.00244$). There was no significant difference between the Global and Local

Modes in average collision count ($p = 0.5638$). The Unfixed Mode had significantly higher collisions than all three fixed command modes (Waypoint: $p = 0.00488$, Global: $p = 0.00537$, Local: $p = 0.00293$). Significant differences in collision counts can be summarised as Waypoint < Global, Local < Unfixed. Qualitatively the UC trials showed greater variability than the FC trials, however this may have been driven by having a small number of subjects.

Distribution of Collisions Across Risk Areas

The station object was designed with both *high* and *low* risk areas, defined by the concavity and risk of collision of the area (Section 2.2.3). Figure 3-3 shows the distribution of total collision impacts around the station perimeter, across all subjects within each FC mode. The density of collisions in some high risk areas was lower when operating in Waypoint mode compared to both Global and Local, for example when maneuvering through the intersection of all three station modules (Risk Area A and B, see Table 3.2).

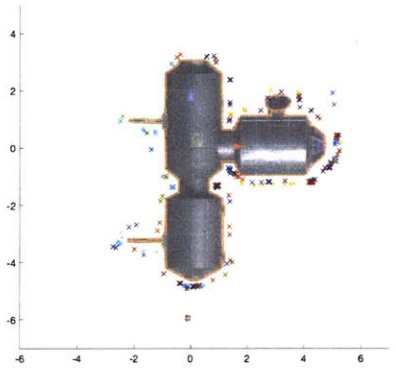
When looking at Global and Local Modes, there did not seem to be a great difference between the density of collisions occuring in high risk versus low risk areas, except when it comes to high convex regions like the cupola (Table 3.2). There was a high density of collisions at both the cupola (low risk area), around the end of Module 3 (low risk area), and at high risk node intersections. The communication dish represented a high-density collision environment.

3.3.3 Effect of Command Mode on Anomaly Detection

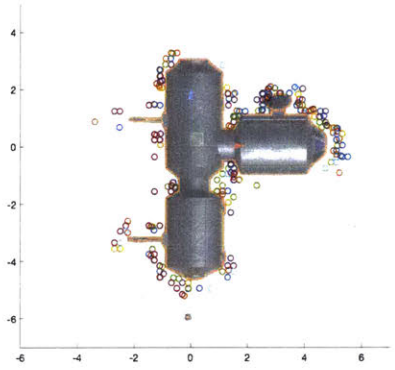
Table C.1 shows the total count of anomalies missed during each trial, where missed anomalies were defined as anomalies that should have been detected by the subject

Area	Collision Count		
	Waypoint	Global	Local
High Risk A and B	14	22	21
High Risk D	8	6	7
Communication dish	10	43	50
Cupola	44	45	28

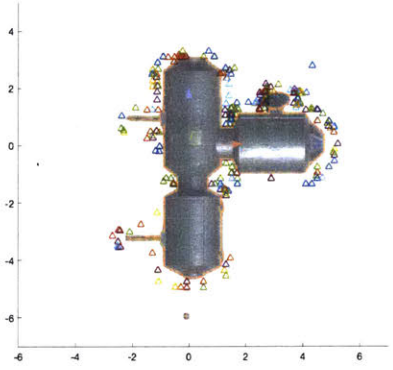
Table 3.2: Select examples of collisions counts from various areas of the Space Station



(a) Waypoint Mode



(b) Global Mode



(c) Local Mode

Figure 3-3: Distribution of collisions around the station object. Subjects are shown in different colors.

Group	Subject	W 1	W 2	G 1	G 2	L 1	L 2	UC 1	UC 2	Total
A	1	1	0	0	1	0	X	1	2	5
	2	1	0	1	0	1	0	0	0	3
	3	X	0	0	0	0	1	0	2	3
	4	1	0	0	0	0	1	0	0	2
	5	2	1	1	2	0	2	3	2	13
	6	1	0	0	0	0	0	0	2	3
	Total	7		5		5		12		29
B	7	0	1	0	1	0	0	0	2	4
	8	0	0	0	1	0	0	1	0	2
	9	0	1	0	2	0	0	1	0	4
	10	0	1	0	0	1	1	0	X	3
	11	0	2	1	1	1	0	0	1	6
	12	0	1	0	0	0	1	0	1	3
	Total	6		6		4		6		22

Table 3.3: Total anomlies missed (passed and not detected). Trials in which no anomalies were passed are indicated by an X. Orange squares indicate that 1 anomaly was missed, red squares indicate that more than 1 anomaly was missed. Note that W = Waypoint Mode, G = Global Mode, L = Local Mode, UC = Unfixed Mode.

but were not due to the inspector's distance to the station in Groups A and B, and the inspector's orientation in Group B. Similar numbers of missed anomalies were observed across all FC modes. Differences in the results become apparent when examining subjects rather than command modes or groups. Overall, it appeared Group B was more successful in detecting anomalies (i.e. fewer misses) than Group A, across all subjects and command modes. However this result seemed to be skewed by Subject 5, who consistently missed anomalies in the majority of trials. Subject 5's missed anomalies account for almost half of the total missed anomalies for Group A.

Across the two trials within each command mode, numbers of missed anomalies were similar. Interestingly, the first Waypoint Mode trial for Group B had zero missed anomalies across all subjects, however there was a increase in missed anomalies when doing the second Waypoint Mode trial in five out of six subjects. In constrast, Group A's first Waypoint Mode trial resulted in five out of six subjects missing at least one anomaly, compared to their second trial where only one subject missed one. The second UC trial resulted in more missed anomalies than the first UC trial in both Group A and Group B.

Table 3.4 shows only the number of high risk anomalies missed in each trial. Recall that Waypoint, Global, Local Mode and UC 2 trials all included 1 high risk anomaly, and UC 1 had two high risk anomalies. Group A performed poorly in detecting high risk anomalies while in Waypoint Mode, with 4 out of 6 subjects missing the anomaly on their first waypoint trial. Group A's performance in detecting high risk anomalies improved by the second Waypoint trial. By comparison, Group B had 4 subjects correctly locate the high risk anomaly on the first and second Waypoint Mode trials. Group B had a number of trials in Waypoint, Global and Local Mode that resulted in no anomalies being passed at all. This suggests that Group B was less likely to

Group	Subject	W 1	W 2	G 1	G 2	L 1	L 2	UC 1	UC 2	Total
A	1	1	X	0	1	0	X	1	1	4
	2	1	0	0	0	X	X	0	0	1
	3	X	X	0	0	0	1	0	0	1
	4	0	0	X	0	0	0	0	0	0
	5	1	1	1	1	0	1	2	1	8
	6	1	0	0	0	X	0	0	0	1
	Total	5		3		2		5		15
B	7	0	X	0	0	X	0	0	1	1
	8	0	0	0	0	0	X	0	0	0
	9	X	0	0	X	0	X	1	0	1
	10	X	1	0	0	0	1	0	X	2
	11	0	0	1	X	1	X	0	1	3
	12	0	0	X	X	X	X	0	1	1
	Total	1		1		2		4		8

Table 3.4: High risk anomlies missed (passed and not detected). Trials in which no high risk anomalies were passed are indicated by an X. Orange squares indicate that 1 high risk anomaly was missed, red squares indicate that more than 1 high risk anomaly was missed. Note that W = Waypoint Mode, G = Global Mode, L = Local Mode, UC = Unfixed Mode.

Group	Subject	W 1	W 2	G 1	G 2	L 1	L 2	UC 1	UC 2	Total
A	1	X	0	X	0	0	X	0	1	1
	2	X	0	1	0	1	0	0	0	2
	3	X	0	0	0	X	0	0	2	2
	4	1	0	0	X	0	1	0	0	2
	5	1	0	0	1	0	1	1	1	5
	6	0	X	0	0	0	0	0	2	2
	Total	2		2		3		7		14
B	7	X	0	0	1	0	X	0	1	2
	8	X	0	0	1	0	0	1	0	2
	9	0	X	0	1	0	0	0	0	1
	10	0	0	X	X	1	0	0	X	1
	11	0	X	X	1	0	0	0	0	1
	12	X	X	0	0	0	1	0	0	1
	Total	0		4		2		2		8

Table 3.5: Low risk anomlies missed (passed and not detected). Trials in which no low risk anomalies were passed are indicated by an X. Orange squares indicate that 1 low risk anomaly was missed, red squares indicate that more than 1 low risk anomaly was missed. Note that W = Waypoint Mode, G = Global Mode, L = Local Mode, UC = Unfixed Mode.

reach a high risk anomaly during their trial using an FC mode.

In examining the Unfixed Mode, performance in reaching high risk anomalies was considerably better than in previous trials in both Group A and B. Recall that UC 1 has 2 high risk anomalies, and therefore in all but one trial at least one high risk anomaly was detected. By comparison, in UC 2, five out of 12 trials missed the high risk anomaly.

Overall, Group A missed a greater number of high risk anomalies, but again this appears to have been driven by Subject 5, who accounts for over half of the missed High Risk anomalies in group A, and Subject 1.

Table 3.5 shows the number of missed low risk anomalies. Within Group A, the

number of missed anomalies across the three FC modes was similar. Performance in detecting anomalies seemed to improve in Waypoint Mode for Group A when moving from the first waypoint trial to the second, with fewer misses and fewer subjects where no anomaly is passed. For Group B, Global Mode resulted in the highest missed low risk anomalies, with four subjects missing a low risk anomaly in their second trial.

UC mode resulted in few trials where the anomalies were not passed. Group A performed poorly in UC when compared to Group B. Overall Group A had a greater number of missed low risk anomalies. This was partially driven by Subject 5, although misses were seen across 4 of the 6 subjects in Group A. Across both the high and low risk anomalies, results showed that the UC modes made it more likely for subjects to pass an anomaly, possibly the result of the increased number of anomalies in the UC trials.

3.3.4 Effect of Command Mode on Efficiency of Station Inspection

Percentage of the space station seen by the inspector (percentage inspected) was analyzed using an n-way ANOVA (Factor 1: Subject (random), Factor 2: Command Mode (fixed), Factor 3: Group (fixed), with Subject nested within Group) with two replicates. Analysis of Variance revealed a signficant main effect of command mode on percentage of station seen ($F(3, 94) = 28.51, p << 0.001$).

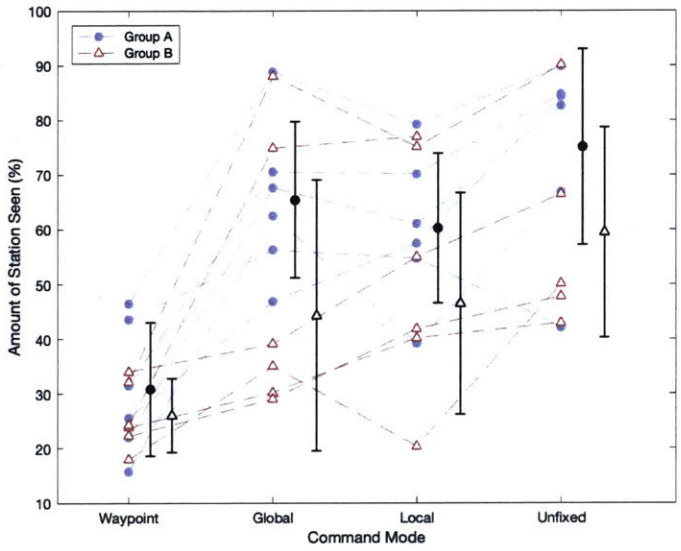


Figure 3-4: Subject average percentage of station seen across command modes. Blue markers and red markers indicate subject means in Group A and B respectively. Black markers and error bars indicate the standard deviation from the overall mean.

Group had no main effect ($F(1, 94) = 1.42$, $p = 0.260$) or interaction effect($F(3, 94) = 0.49$, $p = 0.670$). The lack of an interaction effect between Mode and Group suggests that both Group A and Group B have a similar behaviour, with both groups having higher average percentage inspections for Unfixed and lowest average percentage inspections for Waypoint. A Tukey post-hoc comparison found Waypoint Mode resulted in a signficantly smaller percentage of the station being inspected ($p << 0.001$ in all cases). Overall the average percentage of the station inspected in Waypoint Mode was 28.2766% (SD: 10.6904%), approximately half of either Global Mode (Mean: 57.40% , SD: 23.41%), Local Mode (Mean: 55.93%, SD: 19.35%) or Unfixed Mode (Mean: 69.03%, SD: 20.28%).

There was no signficant difference between the amount of station inspected in either Global or Local Mode ($p = 0.967$), however the Unfixed Mode resulted in a

significantly higher percentage of the station being inspected than either Global or Local ($p < 0.001$ for all cases).

There was an interaction effect of Subject and Mode ($F(30, 94) = 2.12$, $p = 0.0103$). However Figure 3-4 reveals a fairly consistent trend in the subject data across modes, with an increasing trend from Waypoint to Global, and Local to Unfixed, and a decrease from Global to Local, in terms of percentage of station seen. The interaction effect may arise from the magnitude of this decrease/increase across subjects. The n-way ANOVA also revealed a significant effect of subject ($F(10, 94) = 6.54$, $p < 0.001$), suggesting that there was significant difference in the amount of station inspected across the different subjects.

Figure 3-5 shows the data from each subject's trials for percentage of station seen. There appears to be an upward trend across subjects in Group A when examining the raw trial data between repetitions of the same mode. Learning was not explicitly investigated, but could be examined in longer follow-on studies with a higher number of repetitions within modes.

Duration of Inspection

Figure 3-6 shows the amount of time each subject spent completing their inspection. Seven of the twelve subjects used the maximum allowable time (390 seconds) to complete their inspection for all trials, meaning the simulation timed out before they could terminate the test themselves. One subject (Subject 1, Group A) demonstrated a consistently faster time across their trials, achieving a total test time of approximately 5 minutes in their final trial. Of particular note is Subject 5 (Group A) who had the lowest inspection duration time across all trials except Waypoint 1, exiting the trial after approximately two minutes in their second Global and Local

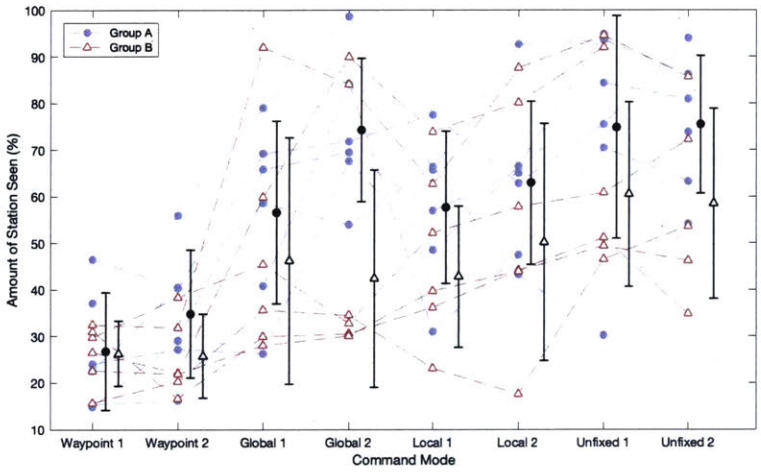


Figure 3-5: Subject raw percentage of station seen across command modes. Blue markers and red markers indicate subject means within each command mode in Group A and B respectively. Black markers and error bars indicate the standard deviation from the overall mean.

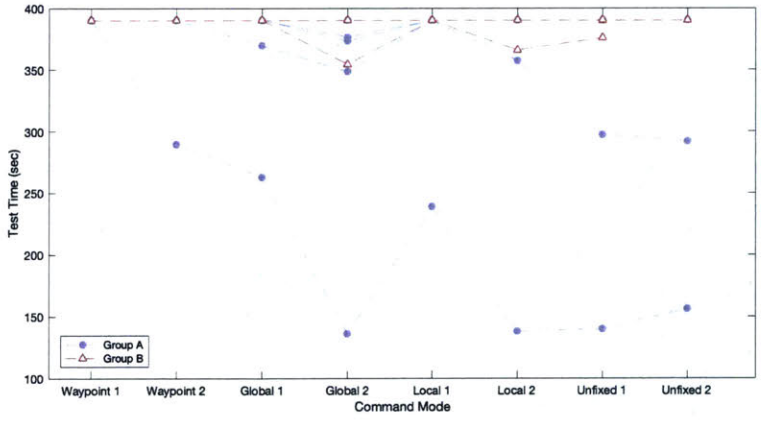


Figure 3-6: Subject total test time across all trials. Blue markers and red markers indicate subject times in each trial within Group A and B respectively.

trial, and both Unfixed trials. Overall Group B had a higher total test time than Group A, driven primarily by Subjects 1 and 5.

Path Length of Inspection

An n-way ANOVA was fit to examine the average path covered by the inspector across command modes. The ANOVA revealed a main effect of group on the distance covered ($F(1,94) = 5.28$, $p = 0.0444$). Post-hoc independent t-tests showed that Group B covered significantly less distance overall than Group A ($t(df = 45) = 2.88$, $p = 0.006$). No interaction effect of group was observed ($F(3,94) = 1.19$, $p = 0.3302$).

Command mode was found to have a significant effect on path length ($F(3,94) = 42.89$, $p << 0.001$). Post-hoc Tukey comparisons revealed that Waypoint had signficantly lower path length than any other mode for both Group A and Group B ($p << 0.001$ in all cases). There was no statistical difference of Global mode with Local mode for either Group A ($p = 0.633$) or Group B ($p = 0.563$). In Group A, there was no signficant difference between the Unfixed Mode and either Global ($p = 0.696$) or Local ($p = 0.128$) whereas Group B showed a signficantly greater path length for Unfixed when compared to Global ($p = 0.0018$) and Local ($p << 0.001$).

Subject was found to have a main effect on path length achieved ($F(10,94) = 5.62$, $p << 0.001$), as some subjects were less successful than others at moving the inspector large distances. An interaction effect of subject and mode was also observed ($F(30,94) = 3.1$, $p << 0.001$) however from Figure 3-7 it appears that subjects generally trended in the same direction.

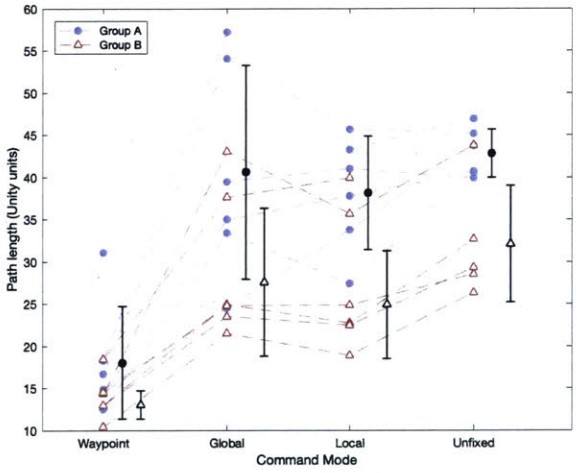
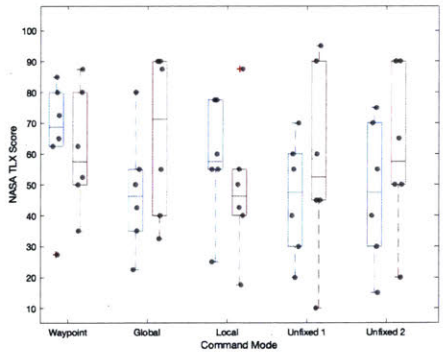


Figure 3-7: Average subject path length across each trial. Blue markers and red markers indicate subject values in each trial within Group A and B respectively. Black markers and error bars indicate the standard deviation from the overall mean.

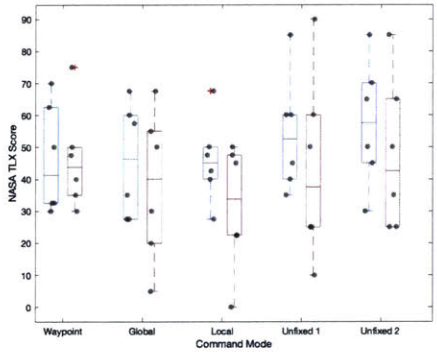
3.3.5 Effect of Command Mode on Workload

An n-factor ANOVA supported a significant main effect of command mode on NASA TLX composite scores ($F(3, 94) = 7.86$, $p << 0.001$). Group was found to have no significant main effect ($F(1, 94) = 0.97$, $p = 0.348$) or interaction effect ($F(3, 94) = 1.33$, $p = 0.282$) on Workload composite score. Figure 3-9 reveals both Group A and B trending downward from Waypoint through to the Unfixed Mode. A post-hoc Tukey comparison found that Waypoint had a significantly higher composite score than either Global ($p << 0.001$), Local ($p = 0.009$), or Unfixed ($p << 0.001$). Global and Local Mode were not signficantly different ($p = 0.991$), however the Unfixed Mode had the lowest composite score, signficantly lower than Waypoint, Global ($p = 0.0072$) or Local ($p = 0.0049$).

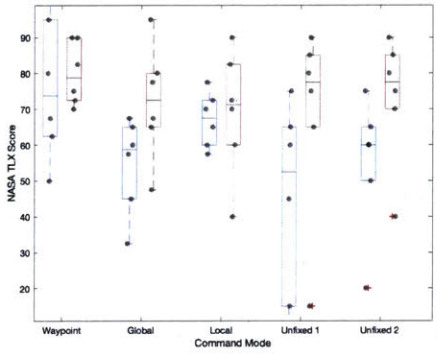
The ANOVA revealed both a significant effect of subject on composite workload score ($F(10, 94) = 7.97$, $p << 0.001$) and interaction effect of subject with command



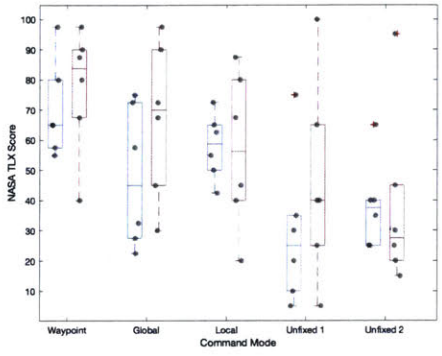
(a) Mental Demand



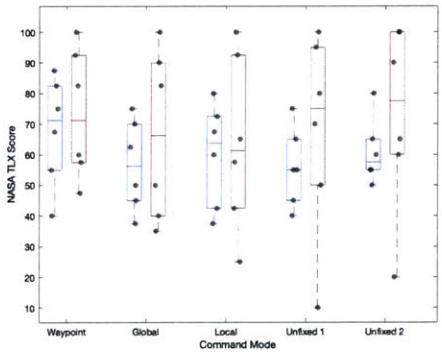
(b) Physical Demand



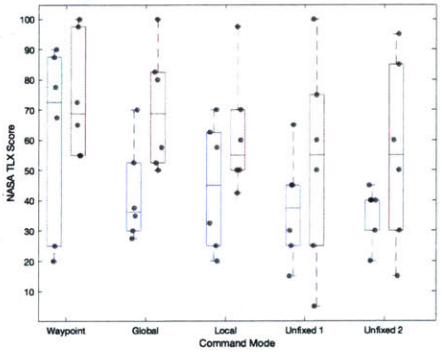
(c) Temporal Demand



(d) Performance



(e) Effort



(f) Frustration

Figure 3-8: NASA TLX component scores across command mode. Grey markers indicate subject means for Waypoint, Global and Local and subject raw scores for Unfixed 1 and 2. Blue boxplots indicate group A, red boxplots indicate group B and outliers are indicated by a red cross

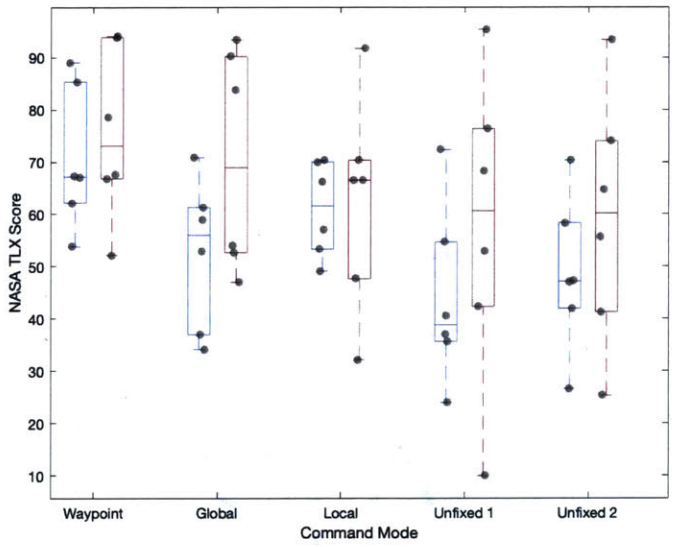


Figure 3-9: NASA TLX average composite workload across command modes. Blue and red boxplots indicate Group A and B respectively, gray markers show the subject average score.

mode ($F(30, 94) = 2.45$, $p = 0.0029$). There were significant differences in the way that individual subjects rated overall workload in their various trials, and significant differences in how each mode was rated between subjects. Looking at the Unfixed Mode ratings, subjects ratings range from approximately 10 to 95 out of a possible 100. By comparison, the range of workload scores for Waypoint Mode is smaller (between 50 and 95). Some subjects consistently awarded a high workload rating across all modes.

The TLX composite score is generated from six weighted components; mental demand, physical demand, temporal demand, performance, effort and frustration (Figure 3-8). These components are weighted based on an initial scoring of the subject from pairwise comparisons between components at the beginning of the experiment. ANOVAs were fit to each workload component score, and Group was found to have no significant effect in any case ($p >> 0.05$). In removing group from the model, ANOVA revealed a significant effect of mode on rating for *Physical Demand* ($F(3, 94) = 3.11$, $p = 0.0393$), *Temporal Demand* ($F(3, 94) = 3.5$, $p = 0.026$), *Performance* ($F(3, 94) = 11.49$, $p << 0.001$), and *Frustration* ($F(3, 94) = 5.16$, $p = 0.049$), with these components contributing to the shape of the TLX data for overall composite workload score. For both the *temporal* and *frustration* scores, Waypoint was given a significantly high workload rating than any other mode ($p < 0.05$ in all cases), while there was no statistically significant difference between the ratings of any other mode. Waypoint mode was also given a signficantly high workload score for *Performance* ($p < 0.01$ in all cases) whereas the Unfixed Mode was rated signficantly lower in terms of workload than all other modes ($p << 0.001$ in all cases). Local was rated significantly lower than Unfixed for *physical demand* ($p = 0.0052$) however not signficantly different to either Waypoint or Global.

3.3.6 Effect of Command Mode on Button Interactions

An n-way ANOVA showed a statistically significant effect of command mode on the number of times a subject interacted with the virtual buttons within the augmented reality system ($f(3, 94) = 106.9, p << 0.001$). Group was found to have no significant main effect ($F(1, 94) = 2.3$, $p = 0.1603$) or interaction effect ($F(3, 94) = 0.88$, $p = 0.461$) on button presses. Tukey post-hoc tests revealed a signficantly lower button press count in Waypoint Mode than any other mode ($p << 0.001$ in all cases). These Waypoint button presses were required to place an average of 11.92 (SD 2.78) waypoint markers in Group A, and an average of 12 (SD 3.52) waypoint markers in Group B.

There was no signficant difference in button presses found between Global and Local modes ($p = 0.994$) however Unfixed Mode utilised signficantly more button presses than either Global ($p = 0.006$) or Local ($p = 0.0032$). There was both primary and interaction effects of subject on the number of button presses (Subject: $F(10, 94) = 6.59$, $p << 0.001$, Interaction: $F(30, 94) = 2.57$, $p = 0.0018$). There appears to be a visual trend in the number of butotn presses across subjects, increasing across modes with Waypoint $<$ Global, Local $<$ Unfixed, however two subjects in Group B do not appear to fit this trend. Further study, with a larger pool of subjects, may provide additional insight on subject button strategy.

Unfixed Command Mode Effects

Of particular interest is how command mode was utilised when subjects had control over which mode they could use and when. In analysing the composition of button presses in Group A, it initially appeared that both the Global and Local Modes were being utilised for the Unfixed Modes. Closer inspection revealed that for 5 out of the

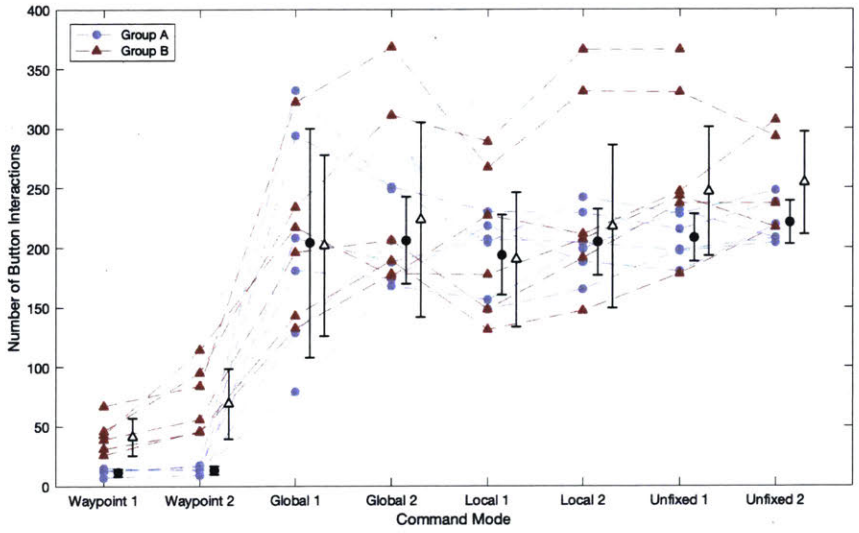
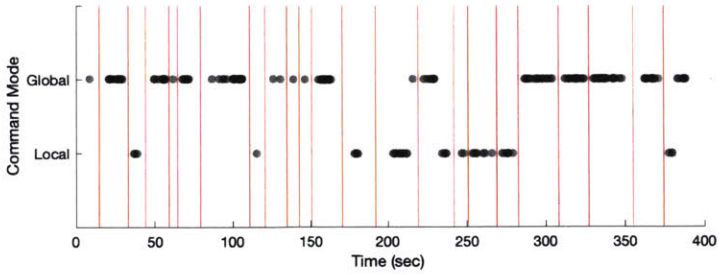


Figure 3-10: Count of button interactions across all trials. Blue and red markers indicate subject counts within Group A and B respectively. Black markers and error bars indicate the standard deviation from the overall mean

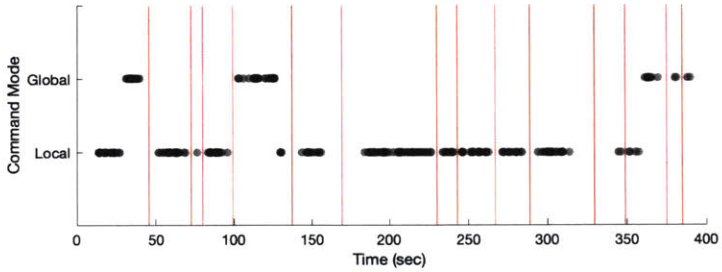
6 subjects in Group A, there was no rotation of the Local Frame during the trial. The frame remained aligned to the station (i.e. the world frame) acting as a globally-oriented reference frame. The final subject of Group A employed a single rotation of the local frame in UC 2, and performed the entire trial in the Local Command Mode, however no other rotations were recorded. Only one subject opted to utilise the Waypoint Mode at any point in the UC trials, placing a single waypoint in UC 1.

Conversely, four out of six subjects in Group B utilised the Local Mode exclusively for both Unfixed Trials. One subject exclusively used the Global command mode for both UC trials. Subject 12 is the only subject across both groups to interchange between Global and Local Modes of command. In Unfixed 1, subject 12 changed command modes 10 times, and in Unfixed 2 the subject changed command modes

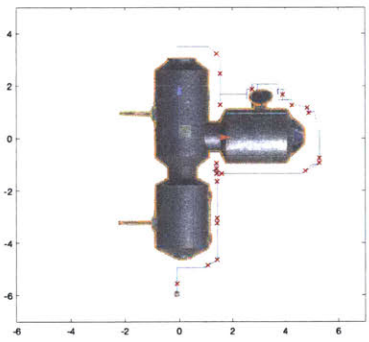
6 times. Comparing the command modes selected (Figures 3-11a and 3-11b) to the path taken by the inspector (Figure 3-11c and Figure 3-11d), it appears that the Local Command Mode was utilised when the subject desired to move in a direction not orthogonal to the axes of the space station.



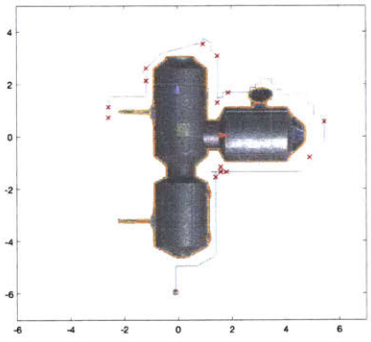
(a) Command Mode selection over test time, Subject 12, Unfixed Trial A.



(b) Command Mode selection over test time, Subject 12, Unfixed Trial B.

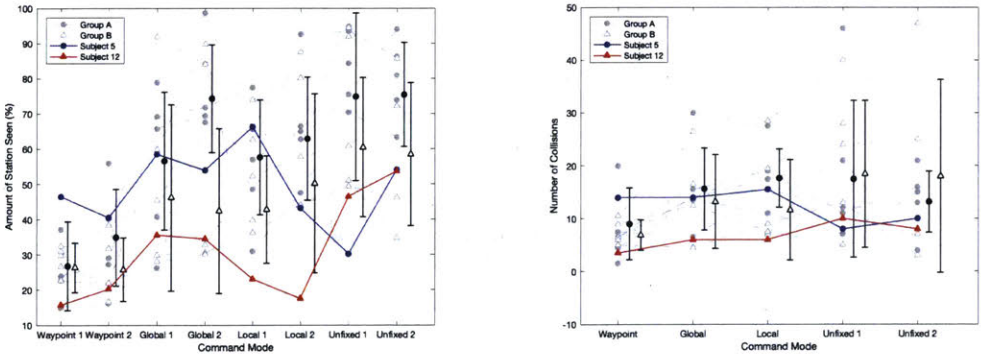


(c) Inspector path around station, Subject 12, Unfixed Trial A.



(d) Inspector path around station, Subject 12, Unfixed Trial B.

Figure 3-11: Subject 12 Unfixed trial command mode selections and path around the station. For *(a)* and *(b)*, grey markers show button presses, vertical red lines indicate an idle time of greater than 5 seconds between button presses. For *(c)* and *(d)*, lue markers show inspector position. The red crosses position when inspector idle for more than 5 seconds between button presses.



(a) Subject 5 and 12 results for percentage of station seen highlighted against study data.

(b) Subject 5 and 12 result for collision count highlighted against study data.

Figure 3-12: Subject 5 and 12 data against study data

3.3.7 Additional Subject Results

Both Subject 5 and 12 exhibit unique behaviour in this study. This behaviour will be discussed in section 3.4. Figure 3-12 highlights Subject 5 and Subject 12 results against the study results. Subject 5 took the shortest amount of time in each trial, remaining at distance greater than the detection distance for much of the trial. Subject 5 inspected between 30 and 70 % of the station across their 8 trials, performing worst in UC 1. Subject 12 adopted a switching strategy for the unfixed trials, resulting in this subject seeing a higher percentage of the station in the two UC trials compared with the FC trials.

3.4 Discussion

This study examined ten hypotheses relating to interface command mode for the inspector satellite, and the level of environmental risk of the area being inspected.

Hypotheses one, three, four and six relate specifically to the effect of command mode on measures of performance across different command modes, anticipating that command mode would have a significant effect on performance. Hypotheses three and four also suggest a relationship between performance, interface and the level of environmental risk. Hypothesis two, five and seven predict an improvement in performance measures when command modes are unfixed and subjects have free choice over which to employ. Hypothesis eight examines the effect of command mode on cognitive load. Hypotheses nine and ten pertain to strategies adopted by the subjects in order to maximise performance.

Subjects were instructed to prioritise avoidance of collisions over other aspects of performance. While it was hypothesized that subjects would accrue fewer collisions when operating in a globally oriented frame of reference (Hypothesis 1), this was found to be only partially supported. Both Waypoint and Global Mode are considered 'globally-oriented' frames, however each mode require the subject to interact with the system differently. It was hypothesized that these globally-oriented command modes would support a greater spatial awareness of the station boundaries, and thus result in fewer collisions. Additionally, the Waypoint Mode forces the subject to plan and place waypoints in a global environment, which may have supported a heightened awareness of the station extremeties. There was a significant decrease in collision count when operating in Waypoint Mode across all subjects. It is difficult to assess if this low collision count in Waypoint Mode is purely a result of a globally-oriented command mode and not also a shift in strategy motivated by the Waypoint Command Mode. There was also no significant difference in collisions between the Local and Global Modes. If globally-oriented reference frames were found to support a greater ability to avoid collisions, we should see a significant improvement in collision avoidance when comparing Global to Local. Pooled results in Table C.1

show that Waypoint mode resulted in the fewest correctly detected anomalies, and Figure 3-4 shows Waypoint Mode resulting in the lowest percentage of station seen. The low percentage and poor anomaly detection suggests a less agressive strategy when subjects were operating in Waypoint Mode, moving around the station at a much greater distance than in Global and Local Modes. These results indicate that Waypoint Command Mode has potential as a command interface for scenarios where a detailed inspection of all areas is not required, for example, navigating around known exteriors of the ISS to get to a suspected anomaly location.

Of particular interest was how collision count was affected by the designated risk level of the environment. The lower collision density in high risk areas when operating in Waypoint Mode is likely a result of subjects opting for a more cautious strategy when operating in this mode, as described above. However there does not seem to be a noticeable trend in collisions across risk areas (Figure 3-3). Some convex areas exhibit lower collision density in Local Mode (e.g. the cupola and ends of station modules) whereas highly concave areas (e.g. node intersections) exhibit high density collisions across all command modes. A larger sample size, with an increase test time to allow subjects to perform multiple inspections of the station may shed light on any patterns in collision distribution.

Hypothesis 2 (subjects will have have superior collision avoidance when all command modes are available to the operator) was not supported. When comparing the results of UC trials to the FC trials, UC trials were found to have a statistically higher collision count than all three FC modes. The increase in number of collisions may be indicative of a more agressive strategy in searching for anomalies, taking greater risks when inspecting close to the station in an effort to increase anomaly localisation. A contributing factor to this result may also be that subjects became more comfortable with the AR system by trial 7 and 8 (UC trials) and were therefore

less cautious in operating the system. In examining the mode usage in the UC mode (see Section 3.3.6), it should be noted that even when permitted to alter command mode during a trial, few subjects opted to switch. In Group A they operated almost exclusively in the Global Command Mode, and in Group B the Local Command Mode was favoured. Hypothesis 2 predicted that subjects would alter their command mode at different points during the trial, to optimize their performance. In the case of collision count, it would seem that the Waypoint Mode offers the superior choice in avoiding collisions, however this option was never selected.

The large variance in the UC collision counts, particularly in Group B, is possibly a result of learning and fatigue confounds. Subjects were trained on collision during the training day, however there is no visual indicator of the true collider boundaries, which are not always perfectly aligned with the station rendering (see Section 3.5). Thus there is a degree of learning that occurs on the testing day, with the UC trials always conducted last, which may lead to some subjects exhibiting improved collision avoidance. Increased training time, to bring all subjects to the same steady state prior to testing, may alleviate some of these confounds. Further study of the learning curve, and the shifts in strategy and performance optimisation that occur across a series of trials, could aid in optimizing training. Additionally, while it is difficult to refine the collision boudaries without a signficant increase in computing power onboard the HoloLens, future studies could employ predictive warnings to alert operators to impending collisions, or visually depict colliders with a rendered boundary.

Physical fatigue may be a confounding factor in later tests. The simulation required subjects to stand upright with arms outstretched for 6.5 minutes at a time, and repeated use of AR has been known to cause eye fatigue. As a result, subjects may have reduced motor control and visual perception of the station, leading to an

increase in collisions. Both Groups opted to use either Global or Local Modes when in the UC trials. The Global and Local Command Modes require considerably more interactions to move the inspector, and Figure 3-10 shows a high number of button interactions in both groups for the unfixed modes, which likely added to fatigue levels. While physical demand was consistently scored lower than other workload components in the NASA TLX survey (Figure 3-8b), the effect of fatigue should not be neglected, particularly when shifting from a simulated to real environment. Greater rest periods between trials, or a redesigned control interface could aid in reducing physical fatigue.

Hypothesis 3 posited that operation in the body reference frame of the inspector would enable superior anomaly localization in high risk areas, while Hypothesis 4 posited that operation in the world reference frame would enable superior anomaly localization in low risk areas of the station. Waypoint Mode provides a much coarser degree of control over the inspector than either Global or Local, and so it was hypothesized that the fine control provided by both the Global and Local Mode was superior in enabling navigation close to the surface of the station to detect anomalies. When operating in high risk areas, station design may occlude or restrict access to anomalies, and subject's attention is likely to be highly focused on the high risk region, rather than the station as a whole. Consequently it was hypothesized that the Local Mode of control, which enables navigation without reference to the station itself, would enable superior navigation in this scenario. In low risk areas, where less attentional narrowing is likely to occur, the Global Mode aligned to the station was hypothesized to enable better anomaly localizaiton.

While Hypothesis 3 and 4 could not be statistically verified, given the small sample size and small range of results, a review of the results can provide insight

into the effects of command mode on the success of anomaly detection. Anomaly detection was the second priority for subjects, and they were directed to locate both critical and non-critical anomalies, which were distributed across both high and low risk areas. There were similar numbers of missed anomalies across all FC modes, which does not support the above hypotheses. Within each command mode, results from trial 1 to trial 2 were also similar. Breaking anomaly detection down by risk level, results were again relatively similar across command modes with each risk group. It is difficult to make an assessment on which mode supports anomaly detection in different areas of risk due to the small sample size, when trials with no anomalies passed are removed. A future study should increase the number of replicates subjects perform, allow greater time for the inspection, and increase the number of anomalies in each trial to aid in identifying any signficant trends across risk area or command mode. Future study designs should also consider research questions related to the detection of anomalies compared to subject vigilance in finding anomalies when few anomalies are present.

When operating in the UC modes with free choice of command mode, it was hypothesized that anomaly localization would be higher compared to the FC modes (Hypothesis 5), as subjects could select the command mode they felt gave them the best performance in anomaly detection. Hypothesis 5 could also not be statistically verified. Group B's missed anomalies for UC are higher than of its FC modes, and higher than Group A's results, however this appears to be driven primarily by Subject 5. Subject 5 consistently missed a large number of anomalies in each command mode, and thus with the high number of anomalies present in the UC trials, this drove the Unfixed Mode results for Group A. Subject 5 consistently passed and missed anomalies, suggesting they were moving around a large portion of the station, but not at sufficiently close range to detect anomalies. Looking at Figure 3-12, subject

5 had above average percentage of station seen in Waypoint Mode compared to the group mean, and below average percentage of station seen in Local 2 and Unfixed when compared to the group mean. Figure 3-6 also highlighted that Subject 5 completed their station inspections the fastest of all subjects, suggesting Subject 5 reprioritized the goals for the simulation, attempting to complete a full inspection of the station as quickly as possible at a greater distance from the surface, rather than moving in closer to detect anomalies.

Anomalies missed in Group B were similar across FC and UC. Further investigation is required to determine if permitting subjects to select their own command mode does in fact affect their anomaly localization. Studies should aim to isolate the Unfixed and Fixed modes, possibly testing across multiple days with adequate rest. While missed anomaly count was higher in the Unfixed Mode, the Unfixed Mode consistently demonstrated better performance in actually passing anomalies with all subjects passing anomalies in each trial, aligning with the results below regarding percentage of station inspection.

Hypothesis 6 was that waypoint command would enable a higher percentage of the station to be inspected. This hypothesis was not supported by the study results. Participants were instructed to attempt a full inspection of the space station in the alotted time as their third performance priority, with the ability to exit early if they deemed their inspection complete. It was initially hypothesized that the Waypoint Command Mode would enable a greater percentage of the station to be inspected as it was predicted that the design of the Waypoint Mode provided more idle time for the subject and thus facilitated greater planning, allowing a subject to maximise their inspection performance in the given test time. In a study examining different telerobotic free-flyer interfaces for inventory tasks, Szafir et al. [64] found that waypoint methods of navigation significantly improved user efficiency in completing

tasks as compared to a manual game controller. Contrary to these expectations, Waypoint Mode resulted in the lowest percentage of station being inspected (less than 30% on average), significantly less than either the Global or Local Modes. In practice, the Waypoint Mode had poorer usability than either of the Local or Global Modes. The poor usability can be seen in both the time taken for inspection (Figure 3-6) and total path length (Figure 3-7). In all but one case, subjects operating in Waypoint Mode used the full 390 seconds and had a shorter path length compared to other modes. The method of generating, moving and setting a waypoint proved cumbersome and subjects were often required to delete and re-generate waypoints when they became stuck inside the space station model or were lost in the field of view. Additionally, the generation of waypoints at the same location introduced an inherent delay in the trial, as subjects had to return to the starting inspector position to acquire a waypoint. Increased training, and a more refined waypoint interface, that spawns waypoints at the current inspector location, and has finer motor control, may address some of these concerns. Global and Local Modes did not result in significantly different percentages of station inspected.

No effect of group on percentage of station inspected was detected. Whilst percentage of station inspection did not take inspector orientation into account, Group B was required to rotate their inspector in order to detect anomalies. The added rotation command takes up time, thus leaving less time for translation. Group B subjects routinely used the maximum allowable inspection time (Figure 3-6). Group B also demonstrated a significantly lower total path length than Group A (Figure 3-7), however not significantly less percentage inspected. This suggests that Group B was moving their inspector more efficiently in terms of station inspected per unit path length. They were moving the inspector around the station close enough to detect anomalies, but without moving the inspector more than was needed. In reality,

robotic systems are likely to adhere more closely to the Group B scenario, and so future studies should take this additional rotation time into account when designing experiments.

The UC trials demonstrated a signficantly improved performance in percentage of station inspected when compared to the FC trials, however 11 out of 12 subjects did not continuously switch modes during the UC trials, so Hypothesis 7 was not supported. Qualitative feedback from the subjects highlighted their preference for the Unfixed Mode in providing the switching capability, however only one subject actually chose to switch modes during the UC trials. Some subjects achieved near 95% station inspected in the UC trials. Upon starting the UC trials, each subject has conducted two trials in each command mode and is very familiar with their performance and limitations within each. The higher percentage of station inspected in these unfixed trials may suggest that subjects were selecting the command mode which would maximise this performance criteria, however improvements in percentage inspection may also be the result of additional time with the system. Considering fatigue is likely a confounding factor in results for the UC trials, even with fatigue subjects still achieved a high percentage inspection in the final two trials. These results indicate that training is an important component in maximising the percentage of station seen.

Hypothesis 8 postulated that the increased subject idle time provided by the Waypoint Mode would result in a lower cognitive load. This hypothesis was not supported. Waypoint Mode was found to have the highest composite workload score, driven primarily by its significantly poor (high) performance rating, physical demand rating, temporal demand rating, and frustration rating. While this is a subjective analysis, these workload measure results are supported by the poor path length performance, long test time, poor percentage inspection results seen previously, and

the usability issues discussed above. In contrast to Waypoint Mode, Unfixed Mode had a signficantly lower rating than all other modes in performance workload, despite self-reported fatigue in subject evaluations.

Hypothesis 9 and 10 both address the UC state. Hypothesis 9, that command mode selected would be dependent on perceived level of risk and performance, was not supported. It was predicted that based on their baseline performance in each of the three command modes, subjects would appropriately select and switch between command modes during the UC trials, to maximise different performance aspects. In Group A, four out of six trials remained in a globally oriented command mode (either Global, or globally-aligned Local). Of the two remaining subjects, both remained in a globally-oriented frame with the exception of two waypoints placed (Subject 4, UC 2) to maneuver from the starting point to the first solar panel, and a single local rotation (Subject 6, UC 2). In Group B, four out of six subjects remained exclusively in the Local Mode (recall that inspector rotation is required for detection in Group B) and one subject remained in Global mode exclusively. The final subject did routinely swap between modes (Subject 12) for both UC trials. These results suggest that generally subjects preferred an exclusive command mode, as opposed to toggling between them. Likewise, in a study conducted by Wang and Lewis [69], they found that operators tasked with controlling a robot team for search and rescue using an unfixed command mode, operators rarely opted to switch modes. Previous studies have found an associated cost with switching modes of control [60, 39] which may be contributing to the choice of an exclusive command mode. In Wang and Lewis' study, a waypoint mode was favoured more heavily than a low-level teleoperating mode. The negilible use of the waypoint mode in the UC trials for this study is understandable considering the poor performance it yielded in both percentage of station inspected and anomalies detected.

While subjects were not switching command mode based on risk level, the choice of using Global or Local Mode over Waypoint Mode is indicative of selecting a command mode that will yield superior performance for percentage of station inspected. There is possibly a disconnect between subject perception and reality driving this choice. Qualitative feedback from subjects shows that most had a preference for either Global or Local, however this did not always align with the mode that gave them optimal performance. Selection of mode may have been driven by greater trust in one mode over another. Follow-on studies should examine the effect of trust in dictating command mode selection, and how training can affect this trust.

It was hypothesized that subjects would adopt different strategies when command modes were fixed compared to unfixed (Hypothesis 10). Hypothesis 10 was examined by looking at trends in the data across different performance metrics. A possible explanation for the increased number of collisions for some subjects in the Unfixed Mode is that subjects had reprioritised their performance metrics, rating collisions lower and thus taking more risk with their navigation around the station in an effort to detect more anomalies. Figure 3-6 clearly shows some subjects prioritising achieving a 'full' inspection of the station as quickly as possible, with reduced test durations in the UC trials. There is an obvious tradeoff between speed and effectiveness of inspection, and as expected subjects that prioritised speed had a higher count of missed anomalies.

Subject 12 is an interesting case study in strategy, as they were the only subject to routinely swap between modes (Global and Local). From the path generated by this subject (Figure 3-11), they utilised the Local Mode to perform maneuvers around irregularly shaped portions of the station (e.g. the cuppola and the ends of modules) before returning to the Global Mode. Looking at the performance metrics for Subject 12, in switching between command modes, the subject increased their

percentage of station seen when compared to the FC trials. In their post-experiment evaluation, Subject 12 highlighted their preference for the Unfixed Mode.

Overall, with the exception of Subject 12, subjects were self-selecting a single mode of command. Subjects were not trained on risk areas, or on using different command modes at particular times within the simulation. Thus a determination cannot be made if toggling mode improves performance in certain risk areas. Future studies could have subjects explicitly trained on when to switch modes during the simulation, and determine if there is an improvement in performance across the criteria decribed above, and within difference areas of risk.

3.5 Additional Limitations

The nature of augmented reality interfaces introduces inherent errors into the results, particularly with regards to collision count. The Unity 3D Gaming Engine used to render the station and build the AR interface has limited ways of building colliders for irregularly shaped objects. A mesh collider perfectly sculpted to the surface of an object results in degraded game performance and latency due to the high vertex count of the object, thus introducing lag and severely reducing frame count. Instead, colliders are typically constructed from simple mathematical models (ellipsoids, cubes etc) which are easier to render, but do not perfectly match object surfaces. Consequently there are sections of the space station where the perceived boundary (rendered surface) and the true boundary (collider surface) are misaligned. This is particularly true in High Risk Area A, and around protrusions. The high collision count around the communication disk is likely a result of a poorly fitting collider. This can be addressed in future studies through the incorporation of a warning system, that alerts subjects to an impending collision.

The gestural controls were highly sensitive to hand motions within the gesture frame, and sometimes had difficulty distinguishing between a ready gesture and a tap gesture. Subjects would accidentally 'drop' the station when trying to move their hand out of the frame of reference, because the HoloLens mistook the gesture for a repositioning of the station. The near clipping plane at 30 cm also also required a small learning curve, as subjects would try to move the station closer in order to improve their view, and consequently lose sight of the station. Additionally the two-handed rotation gestures were limited to yaw and roll motions, with limited success in the pitching direction due to the height of the gesture frame. Pitching of the station necessitated a roll and yaw motion. These limitations of the HoloLens interface may have contributed to a slight degradation in performance over the course of the trials. The increased FOV and gesture frame of the HoloLens generation 2 may address some of these limitations in subsequent studies.

Demographic data was gathered prior to testing, in an effort to identify confounding factors such as a subject's prior experience with augmented/virtual reality or telerobotics. These parameters were not included in the statistical model. The above analysis is by no means definitive due to the limited subject pool, and a larger follow-on study with more targeted subject recruitment may investigate the effect of prior AR/VR experience on performance. Given that the target population for applications of this study is on-orbit servicing, future studies should recruit subjects with astronaut, or astronaut-equivalent training.

Chapter 4

Conclusions, Limitations, and Recommendations

This chapter summarizes the conclusions of the human-subject protocol and development of the AR interface. Limitations of the current study are highlighted, with implications for future study iterations. Further areas of research are highlighted.

4.1 Conclusions of Study

Telerobotic systems and human-robotic teams will form a core component of future Earth-orbit and deep-space crewed missions. In future space exploration, tasks such as inspection, maintenance, and assembly can no longer be carried out solely by astronaut EVA or fixed robotic platforms. Free-flying telerobotic spacecraft present an appealing alternative due to their greater maneuverability and flexibility, lower cost and weight, and wider applications for EVA planning and astronaut support. Advances in visualization and augmented reality technologies can now address the

shortfalls of previous teleoperation and free-flyer interfaces, providing greater spatial awareness and more intuitive control during free-flyer inspection tasks.

Through the development and testing of a custom AR interface for on-orbit inspection, this work has demonstrated that augmented reality technologies can be an effective means of controlling a free-flying satellite for an on-orbit inspection task (Research Question 1). A simulated on-orbit environment was constructed for the Microsoft HoloLens using the Unity 3D Gaming Engine and Mixed Reality Toolkit. The AR interface developed incorporated gestural control to command a small inspector satellite in three different command modes and allowed users to manipulate their viewpoints of the station environment. The HoloLens AR headset appears to be a suitable technology for controlling small satellites, and subjects gave positive feedback on the experience of using the simulator. Additional features of the interface are required to improve performance, such as indication of predicted collisions, and improved waypointing functions. A hardware-in-the-loop architecture was also designed using the SPHERES platform, HTTP servers, and the HoloLens headset (Appendix A). Future iterations of this study should implement components of this architecture in building a more realistic on-orbit simulation with hardware.

This study investigated how command modes within the AR interface might affect measures of human performance in inspecting a space station (Research Question 2). It was initially hypothesised that different command modes would supported different performance criteria. Waypoint Mode was found to support better collision avoidance as users appeared to adopt less agressive inspection strategies when in Waypoint Mode, though collision impacts were not noticeably higher in high risk regions compared to low risk regions, as would be expected given the resticted motion. There were no significant differences in anomaly detection between the command modes, however follow-on studies involving a larger number of anomalies within a

longer trial may reveal more distinction between command modes. Contrary to expectations, the Global and Local Modes supported the highest percentage of station inspected in a fixed mode. Waypoint placement took longer than simply driving the inspector in Global or Local Mode, and subjects operating in Waypoint Mode often used the full game time to attempt an inspection whilst moving the least distance of any mode. By comparison, the better usability and finer control of the Global and Local Modes resulted in several subjects inspecting over 90% of the station surface. Contrary to expectations, the complexity of waypoint movements led to higher cognitive load. In selecting modes to optimise performance in future AR interfaces, the lower collision count and less agressive strategies seen when using waypoints suggests that Waypoint Mode could be used as the command mode when detailed inspections are not required. When the amount of ground covered is of high priority, interfaces should support lower levels of control that operate using directional pads, however results suggest the frame of reference of these controls is not important for the selected inspection task.

A key area of interest in this study was how the Unfixed Command Modes were utilized by the subjects (Research Question 3). Previous studies [60] have observed improved performance when operators have flexibility of command mode. Subjects were expected to switch between command modes in order to prioritize aspects of performance, or when inspecting different areas of the station. Waypoint Mode was expected to support more of a navigational function, and thus allow subjects to maximise station inspection percentage (i.e. see more of the station). The Local and Global Modes were hypothesized to support the inspection aspect of the task, the finer control enabling greater anomaly localisation. Based on these initial hypotheses, it was predicted that when permitted to use any or all of these command modes, subjects would select certain command modes at different times/subtasks

during the simulation to maximise their performance. However, while Waypointing may facilitate planning, as suggested by Szafir et. al. [64], its poor usability and idle time generated frustration from the user and so its use was limited in the UC mode. Discrete waypointing may not be well-aligned with a continuous inspection task. Based on the linear algorithm for the waypoint motion, additional waypoints would be needed to track the station geometry and perform the inspection whereas the manual control of the Global and Local Modes permitted continuous inspection with with the inspector's motion. Subjects who used the inspector with 360° FOV never rotated their local frame, operating in either Global or Local Mode in an exclusively global reference frame. Subjects with a restricted FOV exclusively used either Global or Local modes, with the exception of one subject who opted to switch modes frequently throughout the UC trials. In this study subjects were not forced to switch modes at any point, however previous studies have highlighted this technique as a means of increasing user satisfaction with the system [25]. Training subjects on when to use different modes, or forcing mode changes may yield interesting results in subsequent studies and should be further explored. This thesis suggests that providing flexibility in command mode does not affect performance as this flexibility was not utilised.

Across both FC and UC trials, subjects adopted different strategies and prioritized different aspects of performance (Research Question 4). Over the course of 8 trials, results suggest that performance metrics were re-prioritized in an order other than that given to subjects. More collisions with a greater percentage of station inspected suggest that subjects were performing riskier maneuvers in an effort to find anomalies, at the cost of a higher collision count. While strategies in AR interactions were not specifically analysed within this study, subject feedback reveals differing preferences in viewpoint, display position, and movement around the room

while in simulation. Some subjects preferred a top-down isometric viewpoint to enable better navigation and collision avoidance, whereas others opted for zoomed-in front-facing views that they felt improved anomaly localization. Several subjects moved around the virtual space station, moving themselves rather than the station to change viewpoint. These qualitative results highlight the flexibility and suitability of AR interfaces for this type of task. Further investigation is required to determine if this led to improved performance, and thus an advantage of AR interfaces over more conventional displays.

Overall AR has been demonstrated as a viable method for inspection in simulation, and so this must now be verified using hardware. There is a clear effect of command mode on performance of an inspection task, however further studies are required to clarify which modes can optimally support different aspects of performances. During a true on-orbit inspection of a spacecraft, there will be multiple subtasks that need to be completed, such as navigation to the suspected anomalous site, inspection of the site, proximity maneuvers around delicate hardware, and autonomous docking and undocking of the inspector. Each of these subtasks prioritises different performance metrics, e.g. proximity maneuvers require vigilance in avoiding collisions, rather than in performing the maneuver quickly. While flexible command modes were not used by subjects in this study, different subtasks may require the use of different command modes.

The lessons learned from this research study have direct applications to future telerobotic research. A follow-on study conducted at the Human Exploration Research Analog (HERA) facility at NASA Johnson Space Center will perform multi-day studies to investigate the use of AR interfaces in commanding multiple semi-autonomous free-flying agents. These HERA tests will simulate a long-duration exploration mission and examine the effects of fatigue, learning, and trust on perfor-

mance, as well as clarify results of this study. The HERA tests will also incorporate work done by JPL in scene reconstruction, combining live video from robotic agents with the 3D holographic model of the station to give a high fidelity real-time visualization of the environment.

4.2 Limitations and Implications for Further Study

The investigation outlined in this study had several limitations which should be addressed in future iterations of the interface and architecture design. Hardware was not incorporated into the study and thus results and recommendations from simulated inspection tasks may not directly migrate across to applications involving real robots. Humans have been shown to perform differently in simulated environments as compared to real-life scenarios with robotic agents [4]. Based on the researchers' experience with SPHERES, future studies should use unmanned aerial vehicles (UAVs) as an alternative analog for on-orbit free-flyers, given their greater flexibility and faster control loop (SPHERES uses 1Hz). Additionally, a major drawback of the current test architecture is the communication framework required. The framework requires several laptops, incorporating the HoloLens Web Portal and an HTTP server, and is not easily compatible with external hardware. The Robotic Operating System (ROS) presents a viable alternative to the SPHERES communication architecture. UAVs can be configured to work with the Robotic Operating System (ROS) [51], an open-source operating system that provides a structured communication layer to interface different robotic components. Recent advances in Unity 3D mean ROS is compatible with the HoloLens architecture and HoloLens-Robotic systems using ROS have been demonstrated in studies of Unity-ROS integration for simulated robots [8], hand-gesture-controlled robots [50], and robots controlled

through hologram manipulation [59].

As the study was initially designed as a hardware ground test, the AR interface restricted the plane of motion of the inspection to the horizontal plane only. Free-flying robots (either UAVs or satellites) have 6 degrees of freedom (DOF). While the majority of subjects expressed or demonstrated a preference for either the Global or Local Mode when operating the inspector, the restricted plane of motion meant that simplistic directional pad was sufficient to provide control for these limited degrees of freedom. Controlling a robot in six degrees of freedom, as would be the case on the ISS, would necessitate a redesign of the control interface, and consequently may alter subject performance and preference in the various modes. Szafir et al. found that when controlling a UAV in 6 DOF inside a warehouse to perform inventory tasks, task completion performance in Waypoint Mode was signifcantly better than when using a low-level game controller [65], in contrast to the results of this study. Future studies should expand the architecture to a more realistic on-orbit scenario, incorporating a free-flyer with 6 DOF, as this may reveal different preferred modes of command.

As described previously, the HoloLens Generation 1 interface had several limiting factors. The HoloLens Generation 1 has a very small 35° FOV and restricted gesture frame. The gestural controls employed in the simulation were restricted to the existing controls inbuilt in the HoloLens, limiting the possible control inputs that could be tested, as holograms interactions were limited to taps, drags and scaling/rotating. Research has been done into customized hand gestures such as finger tracking and continuous button input, however the researchers in this study found the custom gestures suffered from lag and could not match the HoloLens gestures in terms of tracking speed and accuracy. Continuous low-level control in the Global and Local Modes was not possible with the tap gesture and instead all commands

had to be discrete. In the post-experiment questionnaires, several subjects expressed frustration and reported fatigue with having to continually tap the global and local buttons. Continuous controls for directional pads are far more intuitive, particularly to people with experience in gaming or telerobotics. Additionally, the gaze control meant that users had to move their head in order to focus on a new object. Some subjects reported fatigue due to this feature, particularly when switching from moving the station to interacting with the buttons. The HoloLens Generation 2 reportedly uses eye-tracking and will employ a *navigation* gesture that operates as a virtual joystick, moving your hand around a virtual 3D cube for velocity-based continuous control [42], and thus may be able to address the fatigue and control limitations of this study's interface.

4.3 Recommendations for Future Work

Over the course of this thesis study, several areas of future work were identified. Previous studies have highlighted interface design for teleoperations tasks as a key factor in contributing to a poor mental model [6] and consequently degradations in performance. Endsley's decomposition of situational awareness[14] aligns with the development of a mental model, informed by what the operator perceives and understands from their experience, and is used to project future states. Breakdowns can occur at any of the three levels of situational awareness (perception, comprehension, projection) and thus influence the fidelity of the mental model. Situational awareness was not explicitly measured in this study, and instead implicitly evaluated through performance metrics like collision count and anomaly localization. The post-trial questionnaire also incorporated probe questions aligned with the levels of situational awareness to reveal the underlying mental model of the participants. A

full investigation of these results was not included in this study but merits further investigation to assess if the proposed AR interface contributed to breakdowns in SA and consequently the fidelity of the mental model. Additionally, secondary tasks that align with the natural tasks could be incorporated into future studies to gauge breakdowns in SA levels during the task, such as required call-outs of inspector clearance distances, or responding to a secondary task using a required gesture.

Traditional space teleoperation displays and interfaces use fixed 2D camera views, 3D visualization software to show perspective views, and manual control interfaces (joysticks, keyboards etc), compared to the stereoscopic 3D environment representation of Augmented Reality. Previous studies have shown that use of a head-mounted display for search tasks decreases subject performance time for search tasks [48]. While literature hypothesizes that synthesizing the visual information in this manner would increase situational awareness and decrease cognitive workload, there may be a trade-off with physical workload due to the nature of gestural control. Several subjects reported fatigue after completing the 8 trials and commented that a game controller might be easier to use than the gestural controls. Future studies should incorporate a control study that uses current telerobotic controls and displays, and compare to a gesturally-controlled AR HMD interface to assess the tradeoff between physical and cognitive workload. The physical demands of the AR system should also be assessed in the proposed operational environment of the system, on-orbit inspection. Physical fatigue may be less of a concern in microgravity, however given the complexity and delicacy of the tasks required by astronauts, this should be investigated. A possible platform for this investigation would be NASA's new Astrobee platform on the ISS, a free-flying robot which can be used by guest scientists for human factors experimentation.

Further, the use of various AR viewpoints was not analysed in this study. Pre-

liminary data analysis of head position and virtual station position/scale/orientation during the tasks indicates a variety of strategies in the interactions with the AR interface among participants, with some subjects remaining in a primarily zoomed-out top-down to isometric viewpoint and others favouring a close-up, horizontal viewpoint. Several subjects expressed a desire for preset viewpoints to be available, in the manner used in modelling software such as SolidWorks. Fixed viewpoints or adaptive displays that shift viewpoint depending on the task/location may lessen the physical requirements by decreasing the need for hologram interaction while still maintaining the benefits of a 3D projected model.

Human trust in a robotic system is a crucial element of effective human-robotic collaboration as it directly effects how and when the system is used, and thus the benefit of integrating the system into operational tasks. The goal is to achieve calibrated trust, where the operator has the appropriate trust level aligned with the robotic system's capability and thus the system is not misused or unused. Issues of mistrust arose during the study as a result of misaligned colliders and missed registering of gestures. While this study maintained a fixed low level of automation, future studies may examine varying automation levels and thus the issue of trust will be of greater concern. Many factors have previously been found to influnce trust, including factors relating to the operator's performance, the capability of the robot, and the operational environment [24]. Measures such as the amount of human and robot idle time, and probe questions on the perceived reliability of the interface, the inspector capability, and human characteristics (self-esteem, trust in technology, attitude towards robots) may provide insight into levels of trust in future studies. Calibrated trust may be inferred by quantitative surrogate measures such as the human operator using fewer waypoints in navigation, or by reductions in human/robot idle time with the human performing concurrent tasks to robotic operations. Trust

can also be enhanced in future studies through improvements to the display such as collision indications, increased training on the system and its reliability, and by showing the process by which inspectors make decisions at higher levels of autonomy [36]. Trust in the robotic system will be even more crucial when dealing with off-nominal scenarios which may occur during a real on-orbit inspection. Operators need to have calibrated trust to ensure appropriate actions are taken when unexpected events occur. Szafir highlighted the importance of interface design in dealing with off-nominal scenarios [64], showing interfaces that supported planning performed better when dealing with off-nominal scenarios. Future work should incorporate off-nominal scenarios during inspection to assess if displays are supportive of better performance and appropriate trust.

With further study, hardware integration, and practical applications of this work to long-duration analogue missions, AR interfaces for teleoperation can be refined, with designs that maximise SA and performance, enable calibrated trust, support human decision-making and strategy, and improve human-robot collaboration for space operations.

Bibliography

[1] R. T. Azuma. A survey of augmented reality. *Teleoperators and Virtual Environments*, 4(August):355–385, 1997.

[2] M. Bualat, J. Barlow, T. Fong, C. Provencher, T. Smith, and a. Zuniga. Astrobee: Developing a free-flying robot for the International Space Station. In *AIAA SPACE Forum*, Pasadena, CA, 2015. AIAA.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[4] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. Bridging the gap between simulation and reality in urban search and rescue. In *RoboCup 2006: Robot Soccer World Cup X*, pages 1–12, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[5] J. Y. C. Chen, M. J. Barnes, and M. Harper-Sciarini. Supervisory control of multiple robots: Human performance issues and user-interface design. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(4):435–454, 2011.

[6] J. Y. C. Chen, E. C. Haas, and M. J. Barnes. Human performance issues and user interface design for teleoperated robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37:1231–1245, 2007.

[7] H. Choset, R. Knepper, J. Flasher, S. Walker, A. Alford, D. Jackson, D. Kortenkamp, R. Burridge, and J. Fernandez. Path planning and control for arecam, a free-flying inspection robot in space. In *IEEE International Conference on Robotics and Automation (ICRA), Detroit*, volume 2, pages 1396–1403, May 1999.

[8] R. Codd-Downey, P. M. Forooshani, A. Speers, H. Wang, and M. Jenkin. From ROS to Unity: leveraging robot and virtual environment middleware for immersive teleoperation. In *Proceedings of the IEEE International Conference on Information and Automation*, pages 932–936, 2014.

[9] Boeing Company. Microsatellite launch. *Crosslink - The Aerospace Corporation Magazine of Advances in Aerospace Technology*, 7:1–2, 2006.

[10] F.A. Cucinotta, S. Hu, N.A. Schwadron, K. Kozarev, L.W. Townsend, and M.Y. Kim. Space radiation risk limits and earth-moon-mars environmental models. *Space Weather*, 8(12):517–527, 2010.

[11] N.J. Currie and B. Peacock. International space station robotic systems and operations - a human factors perspective. In *Proceedings of the Human Factors and Ergonomics Socity 46th Annual Meeting*, pages 26–30, 2002.

[12] T. M. Davis and D. Melanson. Xss-10 microsatellite flight demonstration program results. *Spacecraft platforms and infrastructure*, 5419:16–25, 2004.

[13] G. A. Dorais and Y. Gawdiak. The personal satellite assistant: an internal spacecraft autonomous mobile monitor. In *2003 IEEE Aerospace Conference Proceedings*, volume 1, pages 1–348, March 2003.

[14] M. Endsley. Towards a theory of situational awareness in dynamic systems. *Human Factors*, 37(1):32–64, 1995.

[15] E. E. Entin and D. Serfaty. Development of the NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Advanced Psychology*, 52:139–183, 1988.

[16] O. Erat, W. A. Isop, D. Kalkofen, and D. Schmalstieg. Drone-augmented human vision: Exocentric control for drones exploring hidden areas. *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1437–1446, 2018.

[17] T. Fong and C. Thorpe. Vehicle teleoperation interface. *Autonomous Robots*, 11(1):9–18, 2001.

[18] T. Fong, C. Thorpe, and C. Baur. Multi-robot remote driving with collaborative control. *IEEE Transactions on Industrial Electronics*, 50(4):699–704, 2003.

[19] S. Fredrickson, S. Duran, and J. Mitchell. Mini AERCam inspection robot for human space missions. In *Space 2004 Conference and Exhibit, San Diego, CA*, September 2004.

[20] S. E. Fredrickson, L. W. Abbott, S. Duran, J. D. Jochim, J. W. Studak, J. D. Wagenknecht, and N. M. Williams. Mini AERCam: development of a free-flying nanosatellite inspection robot. In *Proc. SPIE 5088, Space Systems Technology and Operations*, volume 5088, 2003.

[21] M.L. Gernhardt, J.A. Jones, R. A. Scheuring, A.F. Abercromby, J.A. Tuxhorn, and J.R. Norcross. Evidence report: Risk of compromised EVA performance and crew health due to inadequate EVA suit systems. Technical report, National Aeronautics and Space Administration, NASA Lyndon B. Johnson Space Center, Houston, TX, 2017.

[22] S. A. Green, M. Billinghurst, X. Chen, and J. G. Chase. Human-robot collaboration: A literature review and augmented reality approach in design. *International Journal of Advanced Robotic Systems*, 4(1), 2008.

[23] S. Hall. *Effect of Control Interface Implementation on Operation of a Multi Degree of Freedom Telerobotic Arm.* PhD thesis, Massachusetts Institute of Technology, 2012.

[24] P. A. Hancock, D. R. Billings, K. E. Schaefer, J. Y. C. Chen, E. J. De Visser, and R. Parasuraman. A meta-analysis of factors affecting trust in human-robot interaction. *Human Factors*, 53(5):517–527, 2011.

[25] L. V. Herlant, R. M. Holladay, and S. S. Srinivasa. Assistive teleoperation of robot arms via automatic time-optimal mode switching. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 35–42, March 2016.

[26] S. B. Hughes and M. Lewis. Task-driven camera operations for robotic exploration. *IEEE Trans. Syst. Man, Cybern. A, Syst., Humans*, 35(4):513–522, 2005.

[27] T. Imaida, Y. Yokokohji, T. Doi, M. Oda, and T. Yoshikwa. Ground-space bilateral teleoperation experiment using ETS-VII robot arm with direct kinesthetic coupling. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, volume 1, pages 1031–1038, May 2001.

[28] C. A. Johnson, J. Adams, and K. Kawamura. Evaluation of an enhanced human-robot interface. *IEEE International Conference on Systems, Man and Cybernetics*, pages 900–905, 2003.

[29] A. K. Kanduri, G. Thomas, N. Cabrol, E. Grin, and R. C. Anderson. The (in)accuracy of novice rover operators' perception of obstacle height from monoscopic images. *Trans. Sys. Man Cyber. Part A*, 35(4):505–512, July 2005.

[30] K. A. Kempster, R. P. Darken, and B. Peterson. Effects of streaming video quality of service on spatial comprehension in a reconnaissance task. In *Proceedings of I/ITSEC 2001*, Orlando, FL, 2001.

[31] S. Kimura, T. Okyuama, N. Yoshioka, and Y. Wakabayashi. Robot-aided remote inspection experiment on STS-85. *IEEE Transactions on Aerospace and Electronic Systems*, 36(4):1290–1297, 2000.

[32] D. King. Space servicing: Past, present and future. In *Proceeding of the 6th International Symposium on Artificial Intelligence and Robotics and Automation in Space: i-SAIRAS 2001*, St-Hubert, Quebec, Canada, June 2001.

[33] T. Kot, P. Novák, and J. Bajak. Using hololens to create a virtual operator station for mobile robots. In *2018 19th International Carpathian Control Conference (ICCC)*, pages 422–427, May 2018.

[34] D. Krum, D. Bowman, T. Smith-Jackson, E. Coalho, S. Peck, Y. Abdrazakov, D. Bailey, T. Kennedy, Y. Wang, and S. Anand. Effects of video placement and spatial context presentation on path reconstruction tasks with contextualized videos. *IEEE Transactions on Visualization and Computer Graphics*, 14(06):1755–1762, nov 2008.

[35] J. Lapointe. COSMOS: A VR-based proof-of-concept interface for advanced space robot control. *International Conference on Augmented Tele-Existence*, pages 215–219, 2005.

[36] J.D. Lee and K. A. See. Trust in automation: designing for appropriate reliance. *Human Factors*, 46(1):793–801, 2013.

[37] J. Luck, P. L. McDermott, L. Allender, and A. Fisher. Advantages of co-location for effective human to human communication of information provided by an unmanned vehicle,. In *Proc. Hum. Factors Ergonom. Soc. 50th Annu. Meet.*, 2006.

[38] J. Marquez. Spacecraft in miniature: A tool for the aquisition of mental representations of large, complex, 3-d environments. Master's thesis, Massachusetts Institute of Technology, Massachusetts Institute of Technology, 2002.

[39] N. Meiran, Z. Chorev, and A. Sapir. Component processes in task switching. *Cognitive Psychology*, 41(4):211–253, 2000.

[40] Microsoft. Hololens - get started with design - comfort. `https://docs.microsoft.com/en-us/windows/mixed-reality/comfort`. April 2019.

[41] Microsoft. Hololens (1st gen) hardware details - mixed reality. `https://docs.microsoft.com/en-us/windows/mixed-reality/hololens-hardware-details`. March 2018.

[42] Microsoft. Hololens 2 - core building blocks - gestures. `https://docs.microsoft.com/en-us/windows/mixed-reality/gestures#composite-gestures`. February 2019.

[43] Microsoft. Hololens support - use gestures. `https://support.microsoft.com/en-gb/help/12644/hololens-use-gestures`. Accessed 21 August 2019.

[44] D Miller, A Saenz-Otero, J Wertz, Allen Chen, G Berkowski, C Brodel, S Carlson, D Carpenter, S Chen, S Cheng, D Feller, S Jackson, B Pitts, F Perez, J Szuminski, and S Sell. SPHERES: a testbed for long duration satellite formation flying in micro-gravity conditions. *Advances in the Astronautical Sciences*, 105, 2000.

[45] C. W. Nielsen, M. A. Goodrich, and R. W. Ricks. Ecological interfaces for improving mobile robot teleoperation. *IEEE Trans. Robot.*, 23(5):927–941, 2011.

[46] D. Norman. Some observations on mental models. *Mental Models*, pages 7–14, 1983. D. Genter and A. Stevens, Eds. Psychology Press.

[47] O. Olmos, C.D. Wickens, and A. Chudy. Tactical displays for combat awareness: An examination of dimensionality and frame of reference concepts and the application of cognitive engineering. *The International Journal of Aviation Psychology*, 10(43):247–271, 2000.

[48] R. Pausch, M. A. Shackelford, , and D. Proffitt. A user study comparing head-mounted and stationary displays. In *IEEE Symposium on Research Frontiers in Virtual Reality*, San Jose, CA, 1993.

[49] L. Pedersen, D. Kortenkamp, D. Wettergreen, and I. Nourbakhsh. A survey of space robotics. Technical report, National Aeronautics and Space Administration, NASA Ames Research Center, Mottfett Field, CA, March 2003.

[50] D. Puljiz, E. Stöhr, K. S. Riesterer, B. Hein, and T. Kröger. Sensorless hand guidance using Microsoft Hololens. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 632–633, 2019.

[51] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, volume 3, Jan. 2009.

[52] T. Rumford. Demonstration of autonomous rendezvous technology DART project summary. In *Proceedings of Space Systems Technology and Operations*, volume 5088, Orlando, FL, April 2003.

[53] D. Saakes, V. Choudhary, D. Sakamoto, M. Inami, and T. Lgarashi. A teleoperating interface for ground vehicles using autonomous flying cameras. In *23rd International Conference on Artificial Reality and Telexistence (ICAT)*, 2013.

[54] A. Saenz-Otero and D.W. Miller. SPHERES: a platform for formation-flight research. In *Proceedings of SPIE 5899, UV/Optical/IR Space Telescopes: Innovative Technologies and Concepts II*, San Diego, CA, 2005.

[55] J. Z. Sasiadek. Space robotics - present and past challenges. In *2014 19th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 926–929, Sep. 2014.

[56] J. Schneider. A pilot study of fatigue and situational awareness during simulated small satellite operations. Master's thesis, Massachusetts Institute of Technology, Massachusetts Institute of Technology, 2016.

[57] J. Schneider, A. Saenz-Otero, and L. Stirling. A pilot study of fatigue and situation awareness during simulated small satellite operations. In *Annual Scientific Meeting of the Aerospace Medical Association*, 2016.

[58] R. Sigrist, G. Rautner, R. Riener, and P. Wolf. Augmented visual, auditory, haptic, and multimodal feedback in motor learning: a review. *Psychon. Bull. Rev.*, 20(1):21–53, 2013.

[59] E. Sita, M. Studley, F. Dailami, A. Pipe, and T. Thomessen. Towards multimodal interactions: Robot jogging in mixed reality. In *Proceedings of VRST âĂŹ17, Gothenburg, Sweden*, November 8-10 2017.

[60] P. Squire, J.G. Trafton, and R. Parasuraman. Human control of multiple unmanned vehicles: effects of interface type on execution and task switching times. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 26–32, Salt Lake City, Utah, March 2006. ACM.

[61] Richard Stoakley, Matthew J. Conway, and Randy Pausch. Virtual reality on a wim: Interactive worlds in miniature. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 265–272, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[62] E. Stoll, S. Jaekel, J. Katz, A. Saenz-Otero, and R. Varatharajoo. SPHERES Interact – Human-machine interaction aboard the International Space Station. *Journal of Field Robotics*, 29(4):554–575, 2012.

[63] E. Stoll, M. Wilde, and C. Pong. Using virtual reality for human-assisted in-space robotic assembly. In *Proceedings of World Congress on Engineering an Computer Science*, volume 2, San Francisco, CA, 2009.

[64] D. Szafir, B. Mutlu, and T. Fong. Communication of intent in assistive free flyers. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction*, pages 358–365, 2014.

[65] D. Szafir, B. Mutlu, and T. Fong. Designing planning and control interfaces to support user collaboration with flying robots. *The International Journal of Robotics Research*, 36(5-7):514–542, 2017.

[66] Lisa C. Thomas and Christopher D. Wickens. Effects of battlefield display frames of reference on navigation tasks, spatial judgements, and change detection. *Ergonomics*, 49(12-13):1154–1173, 2006. PMID: 17008251.

[67] J. G. Trafton, N. L. Cassimatis, M. D. Bugajska, D. P. Brock, F. E. Mintz, and A. C. Schultz. Enabling effective human-robot interaction using perspective-taking in robots. *Trans. Sys. Man Cyber. Part A*, 35(4):460–470, July 2005.

[68] S. G. Vandenberg and A. R. Kuse. Mental rotations, a group test of three-dimensional spatial visualization. *Perceptual and Motor Skills*, 47(2):599–604, 1978. PMID: 724398.

[69] J. Wang and M. Lewis. Human control for cooperating robot teams. In *2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 9–16, March 2007.

[70] Y. Wang, D. M. Krum, M. David, E. M. Coelho, and D. A. Bowman. Contextualized videos: Combining videos with environment models to support situational understanding. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1568–1575, November 2007.

[71] W. L. Whittaker, C. Urmson, P. Staritz, B. Kennedy, and R. Ambrose. Robotics for assembly, inspection, and maintenance of space macrofacilities. In *AIAA Space 2000 Conference and Exposition*, 1801 Alexander Bell Drive, Suite 500, Reston, VA 20191, 2000. AIAA.

[72] B. G. Witmer and W. J. Jr. Sadowski. Nonvisually guided locomotion to a previously viewed target in real and virtual environments. *Human Factors*, 40(3):4478–488, 1998.

[73] D. D. Woods, J. Tittle, M. Feil, and A. Roesler. Envisioning human-robot coordination in future operations. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 34(2):210–218, 5 2004.

[74] T. Ziemke. On the role of robot simulation sin embodied cognitive science. *AISB Journal*, 1(4):389–399, 2003.

Appendix A

Initial Development with SPHERES Hardware

The study described in this thesis was initially conceived as a robotic hardware task, using the HoloLens headset to command a real free-flying satellite around a model space station while presenting the operator with a virtual model of this station and inspector via the headset. This initial concept was inspired by previous studies done by the Human Systems Lab and Space Systems Lab at the Massachusetts Institute of Technology, which had incorporated robotic hardware into human-subject experiments of fatigue for on-orbit operations [56]. Testing with hardware offers several advantages over testing purely in simulation. Simulations often cannot capture the full complexity of the real system [74], and a human operator may behave differently when working with a real robot versus a simulated robot.

A.1 SPHERES Testbed

The proposed robotic hardware was the Synchronized Position, Hold, Engage, Reorient Experimental Satellites (SPHERES) testbed (Figure A-1). SPHERES are small satellites originally borne out of a senior capstone project in the Department of Aeronautics and Astronautics at the Massachusetts Institute of Technology (MIT), under Professor David Miller. The project migrated to the Space Systems Laboratory (SSL) at MIT, and three SPHERES were launched to the ISS in 2006. The satellites were in operation onboard the ISS from 2006 to 2019, used primarily as a testbed for guidance, navigation, and control algorithms; autonomous docking; formation flying; and vision-based navigation [44]. In addition to the microgravity test facility onboard the ISS, there were two ground facilities for SPHERES; the SSL flat floor facility, and the SSL micro-friction glass testbed. Both MIT facilities could be used to test the SPHERES for planar translation (X and Y axes translation) and single axis rotation (rotation about the Z axis). The SSL flat floor facility is an octagonal poured epoxy surface, measuring 5 meter diameter. The SSL micro-friction glass testbed is a 1.2 $\times$ 1.2 meter glass surface mounted horizontally on a lab bench. For both the flat floor and micro-friction glass surfaces, the SPHERES are mounted to an air carriage system, which stands on three porous carbon pucks. Compressed carbon dioxide is forced through these pucks, allowing the SPHERES to float on the surface with micro-friction. SPHERES are actuated by twelve cold-gas carbon dioxide thrusters that enable them to move in six degrees of freedom in microgravity, and three degrees of freedom on the ground when levitating. SPHERES house onboard power, propulsion, communications, sensing, and computer subsystems, and are capable of operating semi-autonomously (requiring human operators to change the propulsion tanks and battery power packs). SPHERES localize their position and orientation

Figure A-1: SPHERES onboard the JEM Module of the International Space Station, performing a formation flight maneuver

within the test volume by means of the Position and Attitude Determination System (PADS), an ultrasonic beacon system mounted around the test volume. They also carry inertial measurement units (IMUs) for state estimation.

A.2 Experimental Setup

Figure A-2 shows the proposed experimental set-up incorporating the SPHERES testbed and SSL flat floor facility. A model space station would be constructed out of cardboard boxes and placed on the flat floor for the SPHERES to inspect. The HoloLens headset would project a scaled hologram of the model space station and current SPHERES position. The operator could then interact with this hologram gesturally and/or verbally to send commands to the SPHERES. Figure A-3 shows a screenshot from the initial HoloLens simulation. The operator would be visually isolated from the flat floor. A screen in front of the operator would project a live

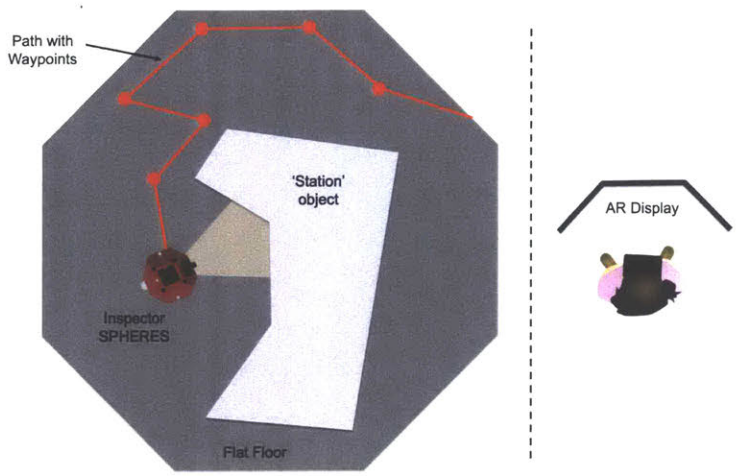


Figure A-2: Proposed experimental set-up for SPHERES hardware tests with Augmented Reality. Subject is visually isolated from the SPHERES.

camera feed from a camera mounted on the front face of the SPHERES, to allow the operator to inspect the station surface. Anomalies on the surface would take the form of visual markers that the operator had to detect and correctly identify.

A.3 System Architecture

A system architecture needed to be developed which would enable data to be parsed back and forth between the HoloLens platform and the SPHERES. The SPHERES uses a Flight GUI program for ground and on-orbit tests, which allows tests to be loaded onto the SPHERES. This GUI incorporates a MATLAB plugin, which allows users to do real-time data processing and visualization of data parsed from the SPHERES. The SPHERES GUI Gateway program is a backend program that acts as the interface between the MATLAB plugin and the Flight GUI.

SPHERES completes maneuvers by periodically executing a closed-loop control function to make estimates on thrust requirements to maneuver the SPHERES to

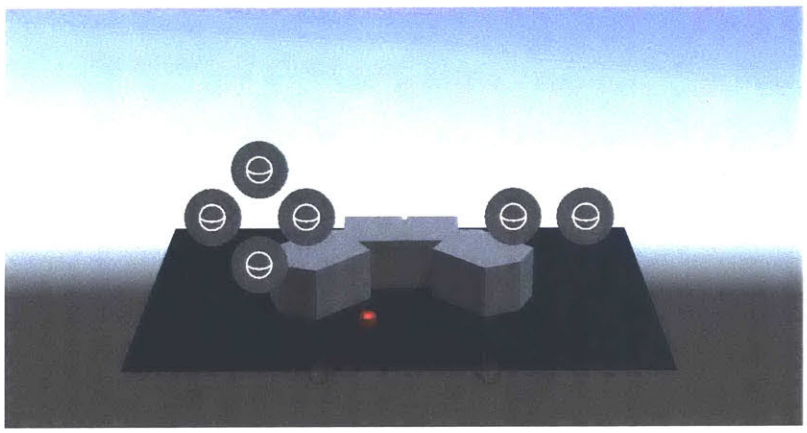


Figure A-3: Initial HoloLens simulation. The red SPHERES hologram corresponds to the current state of the true SPHERES, and can be commanded through direct manipulation of the hologram or by interacting with the collection of grey buttons. The station object is shown in grey and corresponds to a model station constructed out of cardboard boxes on the flat floor. The faint hand icons indicate that gestural control is engaged, and are not visible when simulation is running on the HoloLens

the desired goal state. Within this control loop, SPHERES parses data packets containing state information, telemetry, and state-of-health to the GUI using a radio transceiver. The Gateway program imports this data and organises it into structure arrays which can be accessed via the MATLAB plugin. SPHERES has the capability of receiving data packets from the MATLAB plugin via the GUI during this periodic call (every time the control cycle of the SPHERE executes). The initial concept for the on-orbit inspection simulation was to utilise this capability to pass target locations for the SPHERES to move to, from the HoloLens headset into the MATLAB plugin, and then onto the SPHERES. A template code for SPHERES is included in Appendix B.1.

SPHERES and the MATLAB plugin have no way of directly interfacing with the HoloLens, so an HTTP server was developed in Python to act as the go-between between the AR Headset and the SPHERES (See server script in Appendix B.2).

The `send(request,uri)` inbuilt function in MATLAB allows requests to be sent to HTTP servers via the `MATLAB.net.http.RequestMessage` class. HoloLens apps are built using the Unity 3D Gaming Engine, and coded in C#. Unity can use the `UnityWebRequest` class of functions to send POST and GET requests to HTTP servers (See Appendix B.3 for an example).

When the HTTP server receives a GET request, it will return to plain text file. When the HTTP server receives a POST request, the server will write the incoming data to a plain text file. Figure A-4 gives an overview of the system architecture. SPHERES parses its state information to the SPHERES Flight GUI using the Gateway Program and MATLAB plugin. The MATLAB program would then restructure this state data, and send it as a POST request to the server, which prints this state data to a text file. When HoloLens makes a GET request (which would occur every few frames), the server pushes this text file to the HoloLens. HoloLens can then use this state data to update its simulation of the SPHERES and station. When the HoloLens user moves virtual SPHERES in the AR simulation, this is converted to a commanded position (or waypoint), which is sent to the server via a POST request, and printed to a plain text file. The MATLAB plugin within the Flight GUI would be configured to periodically make GET requests to the server and the server would return the plain text file containing the new waypoints. These waypoints can then be parsed to the SPHERES in a data packet from the MATLAB plugin via the radio transceiver and added to the waypoint queue onboard.

An independent camera system would also be mounted to the front of the SPHERES and be transmitting a live camera feed via Bluetooth to a laptop.

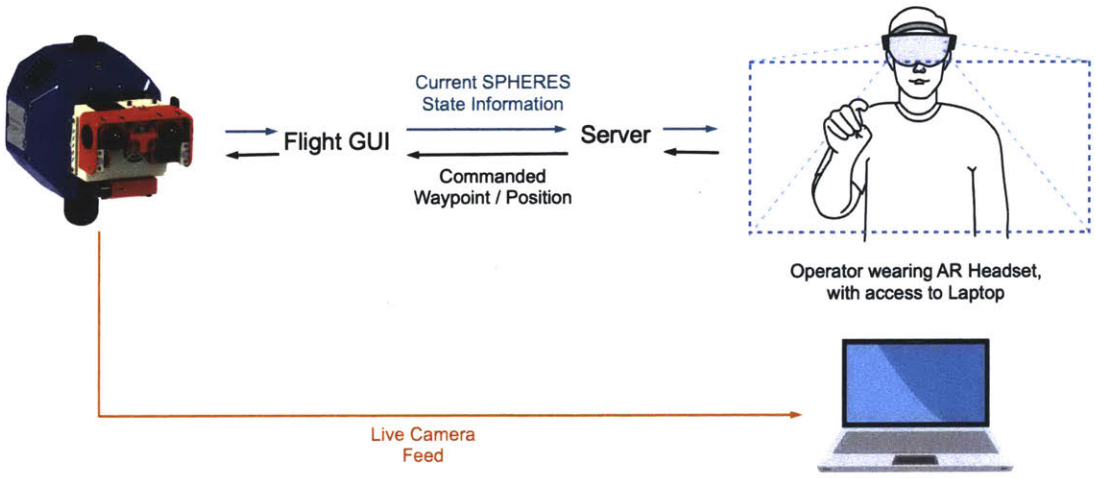


Figure A-4: Proposed architecture for SPHERES hardware tests with Augmented Reality. SPHERES state information is fed to the SPHERES Flight GUI to the Hololens via the HTTP server. Commands made by the human operator in AR are translated into waypoint commands which are uploaded to the server from the HoloLens. The SPHERES Flight GUI pulls any new waypoint commands from the Server and incorporates these waypoints into the SPHERES running waypoint queue. The SPHERES also carries a camera which provides a live feed to a laptop, which can be seen by the operator while in AR

A.4 Limitations and Recommendations

Working with hardware can introduce uncertainty into an experiment, such as time lags or sensor noise. In the case of the SPHERES testbed, imperfections and contaminants on the glass surface and flat floor perturb the motion of the SPHERES during a simulation. SPHERES also had several communication limitations. There was a significant time delay in receiving and transmitting data. The state data would be parsed at a frequency of 1Hz, making precise time measurements of SPHERES state difficult. This would also made updating the HoloLens simulation with current state information very slow. The radio transmitter the SPHERES use to send and receive data would randomly lose connection, meaning data packets to and from the

SPHERES could be dropped. The Flight GUI MATLAB plugin was not configured to support the needed MATLAB functions to access the HTTP server, and difficulties arose in trying to connect the SPHERES GUI to an outside server. A possible solution would be to have the MATLAB GUI write and read directly to the plain text file, and have the HoloLens access this local text file via the HTTP server.

The main limitation of the SPHERES was the manner in which the SPHERES are commanded. The current code architecture for SPHERES is constructed to support either designated waypoints (commanded position, orientation, velocity), or specific thrust times and directions. Previous experiments have incorporated user input during tests [57], however this capability is limited to either keyboard presses or a joystick, and a means could not be found to utilise this capability via the MATLAB plugin. Any commands the user made in HoloLens, (designating a waypoint, commanding the SPHERES in a particular direction) had to be converted to waypoints for the SPHERES in order for MATLAB to parse the data. This method was clumsy and did not accurately represent non-waypoint modes of command.

Appendix B

Initial development code

B.1 SPHERES Template Code

This code was intended for use on the SPHERES platform. The `gsp.c` file is the Guest Scientists Program file, providing to engineers and scientists who wish to use the SPHERES platform. The code is a customisable front-end script, with maneuvers for the SPHERES to follow placed into the `gspControl` function. The function `gspProcessRXData` is used o process incoming data packets from the Flight GUI.

Listing B.1: gsp.c

```
1 /*
2  * gsp.c
3  *
4  * SPHERES Guest Scientist Program.
5  *
6  * MIT Space Systems Laboratory
7  * SPHERES Guest Scientist Program
8  * http://ssl.mit.edu/spheres/
```

```
 *
 * Copyright 2018 Massachusetts Institute of Technology
 *
 ******************************************************
 *                   Input Waypoint Testing           *
 *                   By: Jessica Todd                 *
 *                                                    *
 *  Building on the SPHERES Training Program 2014 and *
 *  the Matlab Demo code                              *
 ******************************************************
 *              Last modified: 28 Apr 2019            *
 ******************************************************
 */


/*-----------------------------------------------------------------------*/
/*                  Do not modify this section.                          */
/*-----------------------------------------------------------------------*/

#include "comm.h"
#include "commands.h"
#include "control.h"
#include "gsp.h"
#include "gsp_task.h"
#include "pads.h"
#include "prop.h"
#include "spheres_constants.h"
#include "spheres_physical_parameters.h"
#include "spheres_types.h"
#include "std_includes.h"
#include "system.h"
```

```
40 #include "util_memory.h"
41
42 /*----------------------------------------------------------------------*/
43 /*              Modify as desired below this point.                     */
44 /*----------------------------------------------------------------------*/
45
46 // C includes
47 #include <string.h>
48 #include <math.h>
49 #include <stdio.h>
50
51 // Controller and Mixer
52 #include "ctrl_mix.h"        // Standard mixer functions
53 #include "ctrl_attitude.h"   // Standard 'non-linear PID'-type
54                              // attitude controllers
55 #include "ctrl_position.h"   // Standard 'PID'-type position controllers
56
57 // Additional Includes
58 #include "comm_datacomm.h"
59 #include "comm_internal.h"
60 #include "comm_process_rx_packet.h"
61 #include "pads_internal.h"
62 #include "SMT335Async.h"
63 #include "find_state_error.h"
64 #include "gsutil_thr_times.h"
65 #include "gsutil_checkout.h"
66 #include "housekeeping.h"
67 #include "housekeeping_internal.h"
68 #include "control_internal.h"
69 #include "pads_convert.h"
70 #include "fpga.h"
```

```
71
72 // My algorithms
73 //#include "gsp_Waypoint.h"
74 //#include "gsp_WaypointCopy.h"
75 //#include "gsp_WaypointLarge.h"
76 //#include "gsutil_checkout.h"
77
78 // Standard maneuvers
79 //#include "gspJoyStick.h"
80
81 #ifdef SPH_MATLAB_SIM
82 #include "mex.h"
83 #define DEBUG(arg) mexPrintf arg
84 #else
85 #define DEBUG(arg)
86 #endif
87
88
89 // Initialize variables
90 // Incoming comms
91 default_rfm_payload data_payload;  // 37 byte (32?) data packet
92                                    //to signal request for commands
93
94 // Flags
95 unsigned int loops_without_answer = 0; // Number of loops with no
96                                        // MPC packets received
97 //unsigned int storedPacket = 0;       // 1: new packet stored
98 //unsigned int askPacket = 0;          // 1: new packet requested
99 //unsigned int sentPacket = 0;         // # packets sent to Matlab
100 unsigned int packetError = 0;         // # errors in packets
101
```

```
102 // Tolerance
103 /*state_vector prev_ctrlStateError = {0.0f,0.0f,0.0f, ...
104  0.0f,0.0f,0.0f,  0.0f,0.0f,0.0f,0.0f,  0.0f,0.0f,0.0f};
105 // State Vector Error (target --> state) from previous control loop
106 int counter_man;
107 // Count for number of control movements in each maneuver
108 float tol = 0.02f;
109  // Tolerance value for terminating maneuver
110 unsigned int waypoint_time_max = 75000;
111  // Maximum allowable maneuver time
112 float dist_to_target = 99.0f; // Current distance to target
113 */
114
115 // Vectors
116 //float inputStateVector[13] = { 0 }; // Empty input vector for incoming data packet
117
118 float shortPacket[3];            // Empty input vector [pos_X, pos_Y, quat_1]
119 state_vector ctrlStateInput = { 0 };        // Inputed target vector from Matlab
120
121 short DebugShort[16];                    // For output (debug vectors)
122 float DebugFloat[16];                    // short DebugVector_short[16];
123
124 // Sets the Satellite identity (Called when sat powered on/reset or powered on)
125 void gspIdentitySet()
126 {
127     // Set the logical identifier (SPHERE#) for this vehicle
128     sysIdentitySet(SPHERE_ID);
129 }
130
131
132 // Intitializes Communications and PADS subsystems
```

```
133 // (Called when sat powered on/off/reset)
134 void gspInitProgram()
135 {
136     // Set the unique program identifier (to be assigned by MIT)
137     sysProgramIDSet(PROG_ID);
138
139     // Set up communications TDMA frames
140     commTdmaStandardInit(COMM_CHANNEL_STL, sysIdentityGet(), ...
141                          NUM_SPHERES);
142
143     // Enable communications channels
144     commTdmaEnable(COMM_CHANNEL_STL);
145     //commTdmaEnable(COMM_CHANNEL_STS);
146
147     // Allocate storage space for IMU samples
148     padsInertialAllocateBuffers(50);
149
150     // Inform system of highest beacon number in use
151     padsInitializeFPGA(NUM_BEACONS);
152
153 } // end of gspInitProgram function
154
155
156 // Specifies task trigger masks. Also called when the satellite is
157 // reset or powered on
158 void gspInitTask()
159 {
160     // NOT USED
161
162     //taskTriggerMaskSet(PADS_GLOBAL_BEACON_TRIG|PADS_INERTIAL_TRIG);
163
```

```
164 } // end of gspInitTask
165
166
167 // Performs test-specific configurations.
168 // Called at the start of each test
169 void gspInitTest(unsigned int test_number)
170 {
171
172     // Set SPHERE initial position in estimator
173     // NB: (does not necessarily need to match where SPHERE is initially
174     // placed but will aid in convergence)
175     // vector = {x, y, z, vx, vy, vz, q1, q2, q3, q4, omegax, omegay, omegaz}
176
177 #ifdef GROUND_TEST
178     state_vector initState = {0.0f,0.0f,0.8f,  0.0f,0.0f,0.0f,  ...
179                                 1.0f,0.0f,0.0f,0.0f,  0.0f,0.0f,0.0f};
180 #else
181     state_vector initState = {0.0f,0.0f,0.0f,  0.0f,0.0f,0.0f,  ...
182                                 1.0f,0.0f,0.0f,0.0f,  0.0f,0.0f,0.0f};
183 #endif
184
185
186     // Turn on background telemetry
187     commBackgroundTelemetryPeriodSet(200);
188
189     // Tell background comm to send this state vector
190     commBackgroundPointerDefault();
191
192     // Set the control period (How often gspControl is executed in ms)
193     ctrlPeriodSet(1000);
194
```

```
195     // Initialize estimator by providing initial condition, setting the
196     // inertial period to 50ms, the global period to 200ms, the wait time
197     // to 105ms, modeling/propogating thruster forces & torques, and by
198     // setting the beacon numbers from 1 to 9
199     padsEstimatorInitWaitAndSet(initState, ...
200                     padsInertialBufferCapacity(), 200, 105,...
201                     PADS_INIT_THRUST_INT_ENABLE,PADS_BEACONS_SET_1TO9);
202
203     // Reset relevant values for each test
204
205     // Calculate any initial values that need to be calculated
206
207 } // end of gspInitiTest
208
209
210 // Perform state estimation based on inertial data. Called periodically.
211 void gspPadsInertial(IMU_sample *accel, IMU_sample *gyro,
212                      unsigned int num_samples)
213 {
214     // NOT USED
215 } // end of gspPadsInertial
216
217
218 // Records global data. Called at the end of each beacon's
219 // transmission period.
220 void gspPadsGlobal(unsigned int beacon,
221                    beacon_measurement_matrix measurements)
222 {
223     // NOT USED
224 } // end of gspPadsGlobal
225
```

```
226
227 // Event-driven task for estimation, control and communications.
228 // Called whenever a masked event occurs.
229 void gspTaskRun(unsigned int gsp_task_trigger, unsigned int extra_data)
230 {
231     // NOT USED
232 } // end of gspTaskRun
233
234
235 // Apply control laws and set thruster on-times. Called periodically.
236 // This is the control interrupt
237 void gspControl(unsigned int test_number, unsigned int test_time,
238                 unsigned int maneuver_number, unsigned int maneuver_time)
239 // test/man number = current number, test/man time = elapsed test time
240 {
241
242     // Initialise state vectors
243     state_vector ctrlState = { 0 };// Current estimated state of SPHERE
244     state_vector ctrlStateTarget = { 0 }; // Current target state of SPHERE
245     state_vector ctrlStateError = { 0 };  // Error b/wn target & current
246
247     // Declare control variables declaration
248     float ctrlControl[6];      // 6 element vector (3 forces, 3 torques)
249     prop_time firing_times;    // Prop vector for thruster firing times
250     float duty_cycle = 40.0f;
251     const int min_pulse = 10;
252
253     // Control loop default gains from position-derivative controller
254     extern const float KPpositionPD; // Prop. gain for position PD controller
255     extern const float KDpositionPD; // Deriv. gain for position PD controller
256     extern const float KPattitudePD; // Prop. gain for attitude PD controller
```

```
257     extern const float KDattitudePD; // Deriv. gain for attitude PD controller
258
259     // Clear memory of control variables
260     memset(ctrlControl, 0, sizeof(float) * 6);
261     memset(&firing_times, 0, sizeof(prop_time));
262     memset(ctrlState, 0, sizeof(state_vector));
263     memset(ctrlStateError, 0, sizeof(state_vector));
264     memset(ctrlStateTarget, 0, sizeof(state_vector));
265
266     memset(DebugShort, 0, sizeof(short) * 16);
267     memset(DebugFloat,0,sizeof(float) * 16);
268
269     memset(data_payload,0,sizeof(data_payload));
270     // default_rfm_payload => 32 unsign char (uint8)
271
272     // Get current state estimate, load SPHERES state into 'ctrlState'
273     padsStateGet(ctrlState);
274
275     switch (test_number) {
276         case 1: // Receive commands from Matlab
277
278             switch (maneuver_number)
279             {
280
281             case 1: //Maneuver 1: Estimator Convergence
282                     // (move on after 7 sec)
283
284                 // At end of estimator convergence, terminate maneuver
285                 if (maneuver_time > 7000)
286                 {
287                     ctrlManeuverTerminate();
```

```
288
289                 // At end of estimator convergence, ask for
290                 // first data packet
291                 //askPacket = 1;
292             }
293             break; // End Maneuver 1
294
295             DEBUG(("Estimator Convergence"));
296
297         case 2: //Maneuver 2: Command SPHERES to
298                 // incoming packet State Info
299
300             DEBUG(("Receiving inputs"));
301
302                 // Set current state target to stored data packet state
303                 ctrlStateTarget[POS_X] = ctrlStateInput[POS_X];
304                 ctrlStateTarget[POS_Y] = ctrlStateInput[POS_Y];
305                 ctrlStateTarget[POS_Z] = 0.0f;
306                 ctrlStateTarget[QUAT_1] = ctrlStateInput[QUAT_1];
307
308             // Keep track of how many control inputs per maneuver
309             /*if (maneuver_time == 0) {
310                 counter_man = 0;
311             } else {
312                 counter_man  = counter_man + 1;
313             } */
314
315             // Calculate current absolute distance to target
316             dist_to_target = sqrt(prev_ctrlStateError[POS_X]*
317                                   prev_ctrlStateError[POS_X] +
318                                   prev_ctrlStateError[POS_Y]*
```

```
319                                 prev_ctrlStateError[POS_Y]);
320
321             // Terminate current data packet if sufficiently close to
322                 // target, or we've done too many loop iterations
323                 //(i.e. counter is too high)
324             if ((counter_man > 0 && dist_to_target < tol) || ...
325                                     counter_man == 100 )
326             {
327
328                 // Reset data packet and counter
329                 counter_man = 0;
330                 storedPacket = 0;
331                 askPacket = 1; // i.e. we've reached the current
332                                // target waypoint, ask for next
333
334
335
336             // Terminate navigation maneuver after 4 minutes
337
338             if (maneuver_time > 4*60*1000)
339             {
340                 ctrlManeuverTerminate();
341             }
342             break;
343
344         case 3: // Maneuver 3: Position hold at end of test
345             if (maneuver_time > 10000){
346                 ctrlTestTerminate(TEST_RESULT_NORMAL);
347             }
348             break; // end of Maneuver 3
349
```

```
350             // If the test number does not match a test,
351                 //end the test and return error code
352         default:
353             ctrlTestTerminate(TEST_RESULT_ERROR);
354             break;
355
356     } // end of switch(maneuver_number)
357
358         // Execute control loop after estimator convergence
359         if (maneuver_number > 1)
360         {
361             // Copy input vector from matlab to target state
362             ctrlStateInput[POS_X] = shortPacket[0];
363             ctrlStateInput[POS_Y] = shortPacket[1];
364             ctrlStateInput[POS_Z] = 0.0f;
365             ctrlStateTarget[QUAT_1] = shortPacket[2];
366
367             // Clear shortPacket vector
368             memset(shortPacket,0,sizeof(shortPacket));
369
370
371             // Find error between target state and current state
372             findStateError(ctrlStateError, ctrlState, ctrlStateTarget);
373
374             // Position Control Law
375             ctrlPositionPDgains(KPpositionPD, KDpositionPD,
376                                 KPpositionPD, KDpositionPD, KPpositionPD,
377                                 KDpositionPD, ctrlStateError, ctrlControl);
378
379             // Attitude Control Law
380             ctrlAttitudeNLPDwie(KPattitudePD, KDattitudePD,
```

```
381                                 KPattitudePD, KDattitudePD, KPattitudePD,
382                                 KDattitudePD, ctrlStateError, ctrlControl);
383
384             // For ground test, ensure no thrusting in z-direction
385             #ifdef GROUND_TEST // only 3 DOF on the table
386             //(NB: sometimes 'wrong' thruster will fire due to mixing
387             // --> Look at REDDI_Session_1\MIT_P4100\...\gsp.c
388             // Cancel forces and torques in unwanted axis because
389             // 3 DOF ground test
390                 ctrlControl[FORCE_Z] = 0.0f;
391                 ctrlControl[TORQUE_X] = 0.0f;
392                 ctrlControl[TORQUE_Y] = 0.0f;
393             #endif
394
395             // Mixer
396             ctrlMixWLoc(&firing_times, ctrlControl, ctrlState,
397                         min_pulse, duty_cycle, FORCE_FRAME_INERTIAL);
398
399             // Disable Estimator
400             padsGlobalPeriodSet(SYS_FOREVER); //padsGlobalPeriodSetAndWait(
401
402             // Set Firing Times
403             propSetThrusterTimes(&firing_times);
404
405             // Reset Estimator
406             padsGlobalPeriodSetAndWait(200, 205); //(200,400)
407
408
409         } // End of Control If Statement
410
411         // Send packet info to Flight Laptop
```

```
412             DebugFloat[0] = shortPacket[0];
413             DebugFloat[1] = shortPacket[1];
414             DebugFloat[2] = shortPacket[2];
415
416             commSendPacket(COMM_CHANNEL_STL, GROUND, sysIdentityGet(),
417                            COMM_CMD_DBG_FLOAT, (unsigned char *)DebugFloat, 0);
418
419             // Download commanded data in Unsigned Short Debug Vector
420             DebugShort[0] = (float)ctrlStateTarget[POS_X] * 1000;
421             DebugShort[1] = (float)ctrlStateTarget[POS_Y] * 1000;
422             DebugShort[2] = (float)ctrlStateTarget[POS_Z] * 1000;
423             DebugShort[3] = (float)ctrlStateTarget[VEL_X] * 1000;
424             DebugShort[4] = (float)ctrlStateTarget[VEL_Y] * 1000;
425             DebugShort[5] = (float)ctrlStateTarget[VEL_Z] * 1000;
426             DebugShort[6] = (float)ctrlStateTarget[QUAT_1] * 1000;
427             DebugShort[7] = (float)ctrlStateTarget[QUAT_2] * 1000;
428             DebugShort[8] = (float)ctrlStateTarget[QUAT_3] * 1000;
429             DebugShort[9] = (float)ctrlStateTarget[QUAT_4] * 1000;
430             DebugShort[10] = (float)ctrlStateTarget[RATE_X] * 1000;
431             DebugShort[11] = (float)ctrlStateTarget[RATE_Y] * 1000;
432             DebugShort[12] = (float)ctrlStateTarget[RATE_Z] * 1000;
433             DebugShort[13] = (short)(test_time / 100.0f);
434                     // send back test time
435                     // in 10ths of a second
436
437             commSendPacket(COMM_CHANNEL_STL, GROUND, sysIdentityGet(),
438                            COMM_CMD_DBG_SHORT_SIGNED,
439                            (unsigned char *)DebugShort, 0);
440
441         default:
442             ctrlTestTerminate(TEST_RESULT_UNKNOWN_TEST);
```

```
443             break;
444
445     } // end of switch(test_number)
446
447 } // end of gspControl function
448
449
450 // This function is used to process RX data
451     // (data that the SPHERES satellite receives)
452 void gspProcessRXData(default_rfm_packet packet)
453 {
454
455     // Should only be receiving data here when SPHERES
456     // has told Matlab it needs it
457     // i.e. askPacket = 1;
458
459     // Put new state info into shortened vector (NB: start at
460     // 6th byte, first 5 are metadata)
461     memcpy(&shortPacket, &packet[5], sizeof(shortPacket));
462
463     length = sizeof(shortPacket)/sizeof(shortPacket[0]);
464
465     for (i = 0; i < length; i++){
466
467         // NB: shortPacket = [pos_X, pos_Y, quat_1]
468         shortPacket[i] = shortPacket[i] - 10;
469         }
470
471     // Reset flag values
472     //storedPacket = 1;
473     //askPacket = 0;
```

```
474
475 }
476
477 // Use this function if you want to do something right
478 // at the end of a test
479 #ifdef TEST_END_TRIG
480 void gspEndTest(unsigned int test_number, unsigned char ctrl_result)
481 {
482 }
483 #endif
```

B.2 Python HTTP Server

B.2.1 Python Server Script

This is the basic HTTP server code, developed by Andrew Liu in the Human Systems Lab, MIT.

Listing B.2: server.py

```
1 # Basic http server for the AR-SPHERES project
2 # Andrew Liu - 20 March 2019
3 # When a GET request is sent to this server, it should return a
4 # plain text file
5 # When a POST request is sent, this server should write the
6 # incoming data to a file
7
8
9 from http.server import BaseHTTPRequestHandler
10 from pathlib import Path
11 import time # Just for time stamps
```

```
12
13 import cgi
14 import os
15
16 class Server(BaseHTTPRequestHandler):
17   def _set_headers(self):
18       self.send_response(200)
19       self.send_header('Content-type', 'text/plain')
20       self.end_headers()
21
22   def do_HEAD(self):
23       self._set_headers()
24
25   def do_POST(self):
26     # open the data file, log file, and a lock file
27     # All of these are in the same filder as the python scripts.
28     fdlock = open("datafile.lock","w")
29     fddata = open("datafile.txt","w")
30     fdlog = open("datafile.log","a")
31
32     # Another from https://www.codexpedia.com/python/python-
33     # web-server-for-get-and-post-requests/
34     # need to 'import cgi'
35     # Data will come in as one keyword with a string that needs
36     # to be parsed for individual values
37
38     self._set_headers()
39     form = cgi.FieldStorage(
40       fp=self.rfile,
41       headers=self.headers,
42       environ={'REQUEST_METHOD': 'POST'}
```

```
43      )
44  ##      print(form.getvalue("state"))
45      dataline = "State(" + form.getvalue('state') + ")\n"
46      fddata.write(dataline)
47      datalog = time.asctime() + "␣" + dataline
48      fdlog.write(datalog)
49
50  ##      self.wfile.write("<html><body><h1>POST
51  ##      Request Received!</h1></body></html>")
52
53      # Close up files and remove lock file?
54      fdlock.close()
55      os.remove(Path("datafile.lock"))
56      fddata.close()
57      fdlog.close()
58
59  ##      return
60
61    # The code for serving a GET request based on the lesson from this URL
62    # https://medium.com/@andrewklatzke/creating-a-python3-webserver-
63    # from-the-ground-up-4ff8933ecb96
64
65    def do_GET(self):
66      self.respond()
67
68    def handle_http(self):
69      status = 200
70      content_type = "text/plain"
71      response_content = ""
72
73      response_content = open("datafile.txt", encoding=('utf-8'))
```

```
74      response_content = response_content.read()
75
76      self.send_response(status)
77      self.send_header('Content-type', content_type)
78      self.end_headers()
79      return bytes(response_content,  "UTF-8")
80
81
82    def respond(self):
83      content = self.handle_http()
84      self.wfile.write(content)
```

B.2.2 Python Script to run Server

This python script is run from the command line, and starts the HTTP server.

Listing B.3: startServer.py

```
1 # Main python script to start an HTTP server for the AR-SPHERES project
2 # Make sure that the file server.py is also in the same directory.
3 # Code is from the following URL
4 # https://medium.com/@andrewklatzke/creating-a-python3-webserver-from-the
5 # -ground-up-4ff8933ecb96
6
7
8 import time      # Just use this to get time stamps
9 from http.server import HTTPServer
10 from server import Server
11
12 HOST_NAME = 'localhost'      # May need to change this to ipaddress
13 PORT_NUMBER = 8080
14
```

```
15 if __name__ == '__main__':
16     httpd = HTTPServer((HOST_NAME, PORT_NUMBER), Server)
17     print(time.asctime(), 'Server UP - %s:%s' % (HOST_NAME, PORT_NUMBER))
18     try:
19         httpd.serve_forever()
20     except KeyboardInterrupt:
21         pass
22     httpd.server_close()
23     print(time.asctime(), 'Server DOWN - %s:%s' % (HOST_NAME, PORT_NUMBER))
```

B.3 Unity Web Request Code

B.3.1 Unity C# POST Server Script

This C# script uses the inbuilt Unity class UnityWebRequest to push data to the server via the POST request. Note that this script is simplified to show the basic functionality of the UnityWebRequest class. It simply verifies that a connection to the Server has been made and POSTS a simple message to the Server text file.

Listing B.4: PostData.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.Networking;
5
6 public class PostData : MonoBehaviour
7 {
8
9     // Start is called before the first frame update
```

```
10      void Start()
11      {
12          StartCoroutine(Upload());
13      }
14
15      IEnumerator Upload()
16      {
17          // First create data in the form of formData
18          WWWForm form = new WWWForm();
19
20          // Add field to the form
21          string state = "3.14159";
22          form.AddField("state", state);
23
24
25          // UnityWebRequest.Post requires URL and postData
26          //(i.e. Form body data)
27          // Sets the url = uri string argument
28          // Sets method to POST
29          // "localhost:3000/api/post-diag"
30
31          // Create object for UnityWebRequest class from
32          // unityengine.networking
33          // using (UnityWebRequest www = UnityWebRequest.Post(...
34          // "http://localhost:8080", form))
35          using (UnityWebRequest www = UnityWebRequest.Post(...
36                          "http://18.20.234.155:8080/", form))
37          {
38
39              // We should return in ienumerator
40              yield return www.SendWebRequest();
```

```
41
42              // Check API is valid
43              if (www.isNetworkError || www.isHttpError)
44              {
45                  Debug.Log(www.error);
46              }
47
48              // If completed
49              else
50              {
51                  Debug.Log("Form upload complete!");
52              }
53          }
54      }
55
56      // Update is called once per frame
57      void Update()
58      {
59
60      }
61  }
```

B.3.2 Unity C# GET Server Script

This C# script uses the inbuilt Unity class UnityWebRequest to request data from the server via the GET request. Note that this script is simplified to show the basic functionality of the UnityWebRequest class. It simply verifies that a connection to the Server has been made and GETS a simple message from the Server text file.

Listing B.5: GetData.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.Networking;
5
6  // Using UnityWebRequest.Get to download a page and test a
7  // non-existent page
8
9  public class GetData : MonoBehaviour
10 {
11
12     // public GameObject
13
14
15     // Start is called before the first frame update
16     void Start()
17     {
18         // A correct website page
19         //StartCoroutine(GetRequest("http://www.google.com"));
20         //StartCoroutine(GetRequest("http://localhost:8080"));
21         StartCoroutine(GetRequest("http://18.20.234.155:8080/"));
22         Debug.Log("We're in");
23         // A non-existent page
```

```
24          //StartCoroutine(GetRequest("https://error.html"));
25      }
26
27      IEnumerator GetRequest(string uri)
28      {
29          using (UnityWebRequest webRequest = UnityWebRequest.Get(uri))
30          {
31              // Request and wait for the desired page
32              yield return webRequest.SendWebRequest();
33
34              string[] pages = uri.Split('/');
35              int page = pages.Length - 1;
36
37              if (webRequest.isNetworkError)
38              {
39                  Debug.Log(pages[page] + ": Error " + webRequest.error);
40              }
41              else
42              {
43                  Debug.Log(pages[page] + ":\nReceived: " + ...
44                                          webRequest.downloadHandler.text);
45              }
46          }
47      }
48      // Update is called once per frame
49      void Update()
50      {
51      }
52  }
```

Appendix C

Final Simulation

C.1 Anomaly Locations

A total of eight simulations were built, two for each command mode. Each simulation had a different anomaly configuration, as described by Table C.1.

Simulation	Command Mode	Critical Anomalies	Non-Critical Anomalies
Mal	Waypoint	A , 5	7
Zoe	Waypoint	E , 1	3
Wash	Global	D , 3	9
Inara	Global	C , 7	1
Jayne	Local	B , 4	6
Simon	Local	F , 2	5
River	UF	-	B, E, 4, 1
Kaylee	UF	C	2, 5, 7

Table C.1: Distribution of anomlies at high/low risk areas for each simulation. Letters and numbers correspond to high and low risk station areas respectively, as per Figure 2-8

Appendix D

Human Study Protocol Documents

D.1 Subject Testing Matrix

Each experimental scenario was named after a different *Firefly* TV character, corresponding to the appropriately configured HoloLens simulation.

Group	Subject	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8
A	1	Mal	Wash	Jayne	Simon	Inara	Zoe	River	Kaylee
	2	Mal	Jayne	Wash	Inara	Simon	Zoe	River	Kaylee
	3	Wash	Mal	Jayne	Simon	Zoe	Inara	River	Kaylee
	4	Wash	Jayne	Mal	Zoe	Simon	Inara	River	Kaylee
	5	Jayne	Wash	Mal	Zoe	Inara	Simon	River	Kaylee
	6	Jayne	Mal	Wash	Inara	Zoe	Simon	River	Kaylee
B	7	Mal	Wash	Jayne	Simon	Inara	Zoe	River	Kaylee
	8	Mal	Jayne	Wash	Inara	Simon	Zoe	River	Kaylee
	9	Wash	Mal	Jayne	Simon	Zoe	Inara	River	Kaylee
	10	Wash	Jayne	Mal	Zoe	Simon	Inara	River	Kaylee
	11	Jayne	Wash	Mal	Zoe	Inara	Simon	River	Kaylee
	12	Jayne	Mal	Wash	Inara	Zoe	Simon	River	Kaylee

D.2 Subject Recruitment Email

Recruitment Email

Subject Line: Subjects needed for Astronaut Study on Robotic Operations in Space

Email:

Have you ever wanted to experience what's its like to be an astronaut? Or get to work with an augmented reality system to control robots remotely?

MIT AeroAstro Department's Human Systems Lab is looking for subjects for a study on Astronaut operations. The study is examining how astronauts could communicate with and control small robotic satellites for future spacewalks, making it safer and more efficient for astronauts on deep space missions to the Moon and Mars.

The study will be conducted over 2 days (2-3 hours each day) in which you'll get to play with the Microsoft HoloLens, control actual robots designed for outer space, and get to pretend to be an astronaut.

All subjects will be compensated $20 for their time and participation in the study.

If you're interested, please email Jessica Todd (Office 41-206, PH: 857-998-1463, Email: jetodd@mit.edu) or fill in this google form:
https://goo.gl/forms/s33uEDGVAKAC48kX2

D.3 Subject Recruitment Flyer

D.4 Screening Matrix

Screening Matrix

DATE: _____

Please tick in the appropriate box for the following conditions.

	Yes	No
Do you have any auditory impairment that might prevent you from following auditory commands spoken to you by the investigator?		
Can you see a computer screen with corrected/uncorrected vision?		
Do you have any visual impairment that will affect your ability to use the HoloLens augmented reality system? (Note, the system can be used with glasses)		
Do you have full mobility in your hands and arms to perform gestural motions?		
Have you previously experienced any significant visual/auditory/vestibular discomfort when using an augmented reality/virtual reality system that you believe will affect your use of the system now?		
Have you previously experienced any significant motion sickness/nausea when using an augmented reality/virtual reality system that you believe will affect your use of the system now?		
Are you comfortable using the HoloLens augmented reality system?		

D.5 Study Protocol Script

D.5.1 Training Day

1. Subjects are asked screening matrix questions.

2. Following screening form, given consent form to review and sign.

3. Test administer gives overview of experiment and the training and test procedures.

Script: Thank you for agreeing to be a test subject for my Masters Thesis work. Let me give you some context for the experiment you're about to take part in. Current maintenance and inspection of the space station and other crewed spacecraft relies heavily on EVAs ('spacewalks'), which are strictly choreographed and rehearsed, and rely on constant communication and support from the ground. NASA's plans to push crewed missions further into deep space, meaning a harsher radiation environment outside the spacecraft, and communication delays with Earth. So we need alternatives to sending humans out to perform EVA. One option is to use small, free-flying inspector spacecraft that can be operated from inside the space station by astronauts. The purpose of this study is to examine the use of Augmented Reality in controlling such a free-flying inspector robot for performing an on-orbit inspection tasks. You will be using the Microsoft HoloLens headset to perform these experiments. The experiment requires you to navigate around the exterior of the space station to look for surface anomalies.

Today will be a training day, where you will learn the basics of the Augmented Reality headset, the Microsoft HoloLens, and how you will be interacting with the experimental simulation. This training is a screening process, if you fail to successfully

complete all training to a satisfactory level, you will not be permitted to perform the experimental stage tomorrow. Tomorrow, you will undergo a quick review of what you learned today, and then proceed to perform 8 tests using the HoloLens to inspect a virtual space station. I'll give you more details on that tomorrow.

If you need to take breaks at any point during this training, please ask.

5. Demographic matrix completed.

6. Subjects given reference handout to read over for a few minutes and informed about what data will be recorded during training.

7. Subject instructed to read HoloLens section of training document.

Script: HoloLens is a mixed reality platform, meaning it allows you to see both the real world, and a virtually generated world, simultaneously. This kind of system is being considered for on-orbit use, as it would allow astronauts to maintain awareness of their surroundings while performing inspection tasks outside the space station.

8. Subject puts on Hololens and is introduced on how to perform basic Hololens gestures (bloom and click) and the cursor, with demos from the test administer.

Script: Hololens utilises two basic gestures, the bloom gesture and the tap gesture. The bloom gesture is used exclusively for exiting an app and returning to the main menu. Pinch all your fingers together in front of you, pointing up, and then spread your fingers. You should see the main menu of the HoloLens appear, and hear a small tone in your ear.

While we're here, lets introduce the cursor. The cursor is essentially your mouse, and allows you to indicate what you want to interact with. It always appears in the center of the frame, in line with your head. Move your head around a bit to see how

the cursor follows your head movements. To click on something in HoloLens, the cursor must be hovering over that hologram.

The tap gesture is used for interacting with holograms. To perform the tap gesture, raise your dominant hand in front of you, pinch your thumb and middle, ring and pinky fingers, and stick your index up to form an 'L' shape. This is the ready gesture, and it signals to the HoloLens that you are about to perform a tap gesture. Focus your cursor on the menu item that says âĂIJAll AppsâĂİ off to the right. Now click your index and thumb together to perform the tap gesture and select this menu item. Practice the bloom (return to menu) and tap gesture a few times.

9. Subject is introduced to the restricted Hololens field of view, and then instructed to complete introductory HoloLens tutorial.

Script: The Hololens has a field of view of approximately 35 degrees. The Hololens will not register any gestures outside of this field of view. Place your hand in the open click position (the 'L') and move your hand in and out of your field of view one at a time to get a sense of the boundaries of the Hololens sensors. When the hand is in the field of view, the cursor should expand into an open circle. Now try your other hand.

To make sure you understand all the HoloLens components, we'll do a quick run through of a HoloLens training app. Tap on the icon called âĂIJLearn GesturesâĂİ, and complete the tutorial. Let me know when you've finished.

10. Subject instructed to select first training app. Subject works through scaling, rotating and moving objects with demos from the test administer.

Script: Now we will start your astronaut training. We will be learning how to move, rotate and scale holograms. In our experiment, you will be able to move, scale and rotate the space station to change your viewpoint.

Please navigate to the main menu, and select the app called "Alpha TrainingâĂİ by moving the cursor over the app and tapping. I'll walk you through the training.

The space station can be moved by placing the cursor over the object, tapping and holding, and then dragging your hand around in 3D space. Try moving the station object around. Then click the 'next' arrow.

To change the scale of the space station, you need to perform a two-hand tap. Focus your cursor on the station, tap and hold with both hands, and pull your hands apart and back together to watch the station expand and shrink. Try and scale the station. Then click the 'next' arrow.

Finally, rotating. Once again, we need to use the two-hand tap. Tap and hold with both hands, then move your hands around a central point in between them, as if rotating a real object. You can rotate in all three directions as shown by the demonstrator. It can sometimes be difficult to rotate around the horizontal axes due to the limited field of view. In this case you'll need to perform two rotations around the other two axes to get the station where you want. Try and rotate the station in each of the three directions, then click next.

Now we can combine moving, scaling and rotating. Manipulate the station to match the position, scale and orientation of the red station. The red station should turn green when you match up. During your experiment, your virtual station can be rotated, moved and scaled to provide you with different viewpoints during the test. When the station moves, any tethered features, such as waypoints you place, or the inspector satellite, will move and scale with it.

To exit the app, perform the bloom gesture, then use the cursor and tap on the

'Home' button.

11. Subject removes HoloLens and takes a break.

12. Subject instructed to read the Experimental Set-Up section of the document.

13. Subject then goes through the second training app to learn to navigate using waypointing, and d-padding. If completed to satisfactory level, subject is cleared to continue with study.

Script: You've completed basic hologram training, congratulations! Now its time to practice the different types of control you will use for the experiment. During the experiment, you will be controlling the inspector satellite, to maneuver it around the exterior of the space station, looking for anomalies on the surface that may pose a risk to you and your crew. You will be asked to locate both critical and non-critical anomalies. There will be multiple anomalies in each test. These anomalies will only appear when you are within a certain proximity.

For the experiments tomorrow. I will be assessing your performance while undergoing these tasks. You should prioritise your performance based on the following, in order of priority:

1. Avoid collisions with the station or with any delicate protrusions such as solar panels or communication disks.

2. Locate anomalies. Prioritise critical anomalies and log them using the anomaly log button.

3. Complete a full inspection of the station and locate anomalies in a timely manner. The simulation will time out after 6.5 minutes.

You may manipulate the station to achieve different viewpoints however you like to help you best complete the inspection task.

You will perform the test using three different control modes for the inspector, repeating each mode twice. The modes are:

- Navigating using a directional pad, in the world frame. This axis are locked to the station itself, as you can see in
- Navigating using a directional pad, in the body frame of the satellite. If you look at the tutorial document you can see how the axis of the inspector are oriented.
- Waypointing. This will require you to place markers for the inspector to move to.

Navigate to the training called âĂIJBeta TrainingâĂİ. As you saw in your training packet, movement of the inspector is restricted to the x-z plane. Lets first try toggling between these different control modes, by clicking the toggles in the upper left and right hand corners. You'll notice that different buttons will appear and disappear.

Lets first try navigating using the global directional pad. Click the global/local toggle in the upper right of the control pad so that it read global and glows blue. Try following the white path as closely as you can, using the global directional pad buttons. It will turn green when you're on the right path. Feel free to manipulate the station view to assist you in navigating.

Its also important to remember that the directional pad does not change position when the station does. So even if the station is rotated in a different direction, the forward area will still move the inspector along the +Z axis of the station.

Now try navigating in the local frame back along the same path. Toggle back to a local directional pad, and once again move along the white path. Now you are

moving relative to the body frame of the inspector. A small arrow jutting out from the satellite indicates the forward direction. Note the difference between the world and local coordinate frames when you rotate and then move the inspector. You are now following its body coordinate frame, as opposed to the world frame.

Finally, lets try navigating using waypoints. Waypoints act as markers for your inspector, giving it a target location to move to. If you toggle the waypoint button in the upper left, you'll notice that the d-pads disappear and three waypoint buttons appear. These allow you to generate, set and delete a waypoint. When clicking the generate button, you'll see a small flag appear in front of you, near to the starting position of the inspector. This is the Waypoint marker. This flag will always generate at the same spot on the station, and then you are free to tap and drag this marker to your desired location. If you click the delete button, this waypoint will disappear. This is also useful if you lose track of a waypoint. To make the inspector move towards a waypoint, you must 'set' it by clicking the set button. This commands the inspector to move in a straight-line path to the x-z coordinate of the waypoint. If you click delete while the inspector is en-route, it will stop at its current location. Try placing some waypoints along the white path.

Ok, lets talk anomalies. You should also notice, as you near the end of the white path closest to the station, you will see a critical anomaly appear. If it doesn't appear immediately, navigate closer to the surface and it should appear. This is a critical anomaly, and poses risk to your crew. Click on the exclamation icon-ned button, to log the anomaly and notify your crew. There is another non-critical anomaly hidden on this part of the station, where the two modules join together. See if you can navigate over and find it. You do not need to log this kind of anomaly.

14. Subject given free time to play with the Hololens app to familiarise themselves

with the modes of control and interaction. May continue until they are satisfied that they can use the system sufficiently.

Script: If successful: Congratulations, you've completed your astronaut training! Please take some time now to play around with the HoloLens Beta Training app, familiarising yourself with the gestures and control modes until you're confident you can repeat this tomorrow in test conditions.
If unsuccessful: Unfortunately, you have not been able to complete the necessary astronaut training to proceed with this experiment. Thank you for your time in performing the training procedure.

D.5.2 Testing Day

1. Subjects are quickly reviewed on training from the previous day to ensure they can continue with the study.

Script: Thank you for agreeing to continue the testing today. Today is the fun stuff! As I said yesterday, you will be performing several different tests to study your performance using different control modes. You will perform 8 tests in total, 2 in each control mode with the control mode fixed, and then 2 tests in which the control mode is unfixed and you have free choice over which control mode you use.

Before we begin, lets review the training from yesterday very quickly. (If they don't know the answer, walk them through it).

- *What are the goals in order of priority of the simulation?*
    1. *Avoid collisions with the station.*
    2. *Local critical and non-critical anomalies and log appropriately.*
    3. *Complete inspection in a timely manner.*

- *Demonstrate how you would move the station or a waypoint, scale the station, and rotate the station.*
- *What is the difference between the global and local d-pads and how do you change between them?*
  - *Global is in the world frame of the station , Local is in the body frame of the satellite. They are toggled.*
- *How do you place a waypoint?*
  - *Generate, move to position, Set, Delete.*
- *Can you identify which anomaly is critical and which is non-critcial?*

3. Subjects are given the cheat sheet and post-test questionnaire to review before starting the tests.

4. Subjects move through each fixed mode trial as specified by test matrix.

Script: Lets begin the tests. The goals of each test, in order of priority, are as follows:

1. Inspect the exterior of the space station while avoiding collisions.
2. Locate critical and non-critical anomalies.
3. Complete inspection of the station in a timely manner. The simulation will time out after 6.5 minutes.

In each test there are multiple anomalies to detect, both critical and non-critical.

Between each test you will be given some time to rest and remove the HoloLens. At the end of each test, you will fill out this post-test questionnaire. Questions differ slightly depending on the control mode you are using. You will also complete a NASA-TLX survey to gauge your workload during the test.

During the test I will be recording your HoloLens viewpoint and hand movements.

Your first test will be (*trial name*) so please select Simulation (*trial name*). When you are satisfied that you have completed your inspection, perform the bloom gesture to return to the main menu.

Please verbally confirm that you see the application window, followed by the Unity logo. These may take a few seconds to appear. Good luck!

5. At the conclusion of each test, subject is permitted a break while test personnel save the video recording of the test, and download the log file from the Web Portal.

6. At the conclusion of each test, subject fills out the SA questionnaire and NASA TLX assessment.

7. After completing 6 structured tests, subjects are instructed to perform 2 unstructured tests.

Script: You have completed your first 6 tests. The final two tests are called 'unstructured' tests. This means that the control mode you perform the test in is not fixed. You may use any of the modes (waypoints, global d-pad, local d-pad) and may switch between them as you prefer during the test. The simulation will open on the default local dpaf setting, but you may change it immediately if you wish. The conditions of the test are the same. There are multiple anomalies to detect, and the test will time out after 6.5 minutes. At the conclusion of each unstructured tests you will fill out the post-test questionnaire.

First, select Simulation River and complete. When you are satisfied you have finished your inspection, perform the bloom gesture to exit.

Your second Simulation Kaylee . Please select it and complete. When you are satisfied you have finished your inspection, perform the bloom gesture to exit.

8. At the conclusion of all testing, subjects are asked to provide qualitative feedback on their experience of the station inspection task.

9. Subject completes an MRT.

D.6 Demographic Data Matrix

Demographic Information Matrix

DATE: _____

Please fill in the following matrix.

Gender	
Age	
Are you an MIT student/staff member?	
Do you use a computer regularly?	
Do you use a touch-screen device regularly?	
Have you previously used a virtual reality system? (e.g. Oculus rift, HTC Vive etc). If yes, please specify.	
Have you previously used an augmented reality system? (e.g. Microsoft HoloLens, Magic Leap One etc). If yes, please specify.	
Do you have any relevant prior experience to the current study? If yes, please specify.	
Have you previously worked with the SPHERES platform? If yes, please specify in what capacity.	
Have you previously worked with a tele-operated robotic system? If yes, please specify in what capacity.	

D.7 Subject Training Document

Station Inspection Training Document

Experiment Overview

During this experiment, you will be using the Microsoft HoloLens. You will be acting as an IVA (intra-vehicular) astronaut, tasked with performing an inspection of the exterior of your space station to look for anomalies on the surface. The Microsoft HoloLens enables you to interact with an Augmented Reality display depicting your space station, and the small inspector satellite you can control to detect the anomalies.

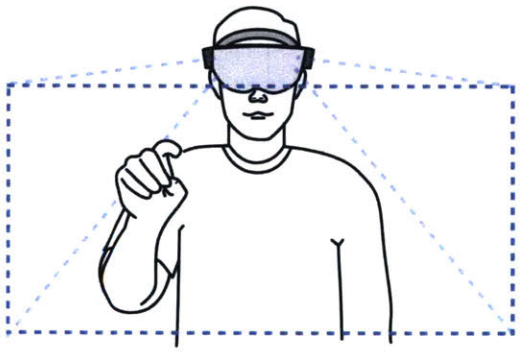
HoloLens Overview

The Microsoft HoloLens platform is a mixed reality head-mounted display. The visor projects a virtual reality image or 'hologram' out in front of the user, and uses sensors mounted into the front of the unit to track hand movements. The HoloLens uses a Gaze-Gesture-Voice input model to interact. For this experiment we will only be utilising the Gaze and Gesture functionality.

Gesture Frame and Field of View

The *field of view* of the Hololens is approximately 35°. This view can be quite restrictive and objects will disappear outside of this view, or if you move them to close to you.

The Hololens will not register any hand gestures outside of its *Gesture Frame*. This is the boundary of where the HoloLens can detect hand gestures, approximately a foot on either side of the user. In order to perform a two-handed gesture, both hands must be within this field of view.

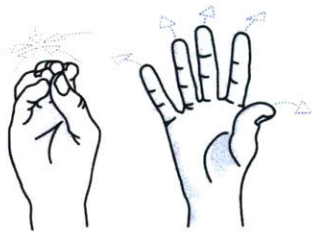


Gestures

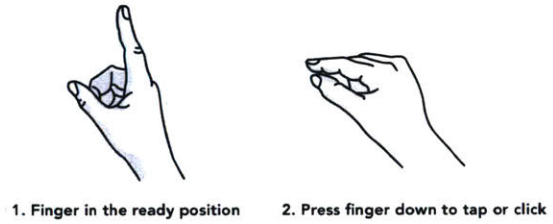
HoloLens utilises two basic gestures:

Bloom: this is the "home button" and is exclusively used to exit an app and return to the main menu.

Tap: this is used to interact with holograms. Place fingers into an 'L' shape to signal to HoloLens you are in the ready position (cursor should become an open circle). A single tap down can be used for button presses. If you tap and hold, you can drag holograms around.



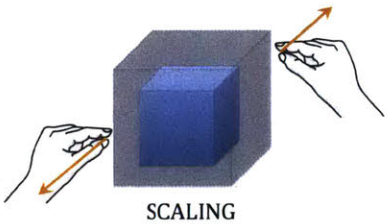
BLOOM



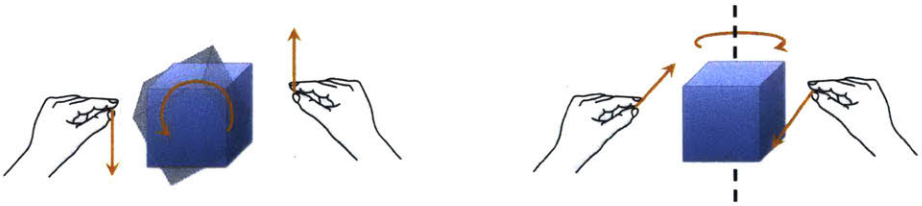
TAP

Scaling and Rotating

Two hands can be used to scale and rotate holograms. Place both hands in the 'L' shape, tap and hold. To scale, move hands closer together/further apart. To rotate, rotate hands around a central point



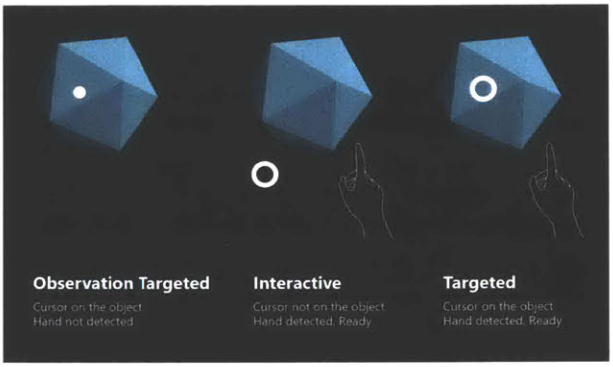
SCALING



ROTATING

Gaze and Cursor

Holograms are fixed in the real world. The HoloLens uses head-tracking to determine where the user is looking in the world, not eye-tracking, with the cursor always appearing in the centre of the frame as a small white dot. Think of your head like a computer mouse, used to move the cursor around.



To interact with holograms, your cursor must be on the hologram, so your head must be pointed at the object. If your hand is in the frame when you perform a gesture, the cursor will expand from a dot into an open circle, meaning the HoloLens has detected the gesture. If the cursor is hovering over an object you can interact with, the cursor will morph to the shape of the object.

When you tap on an object, the cursor will shrink back to a white dot, until you release the tap.

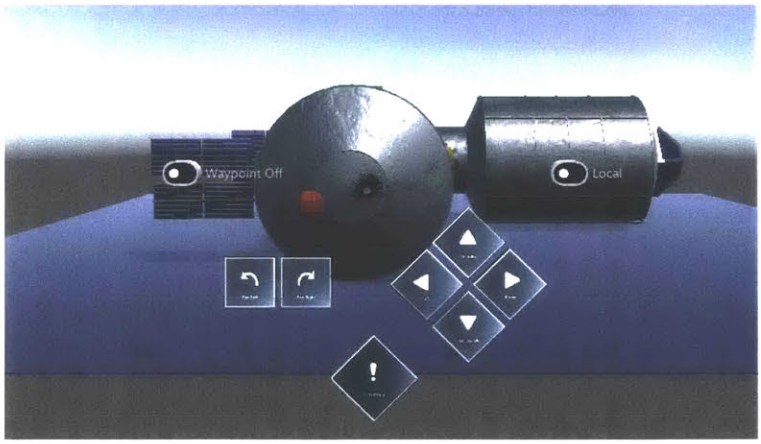
Experimental Set-Up

During the experiment, you will be moving the inspector satellite around the virtual station. The virtual station scene can be scaled, rotated and moved to give you different viewpoints at any time throughout the test. You are encouraged to use this feature as much or as little as you would like to, to help you best complete the inspection task. When the station moves, rotates and scales, all tethered features (i.e. the inspector satellite, any waypoints you've placed) will move and scale with it.

During the experiment, you will be controlling the inspector satellite. For this experiment, its movement will be restricted to two dimensions (x and z in world coordinates), and all anomalies will be located at the same height as the inspector. You will control the inspector in 3 different control modes.

Control Pad

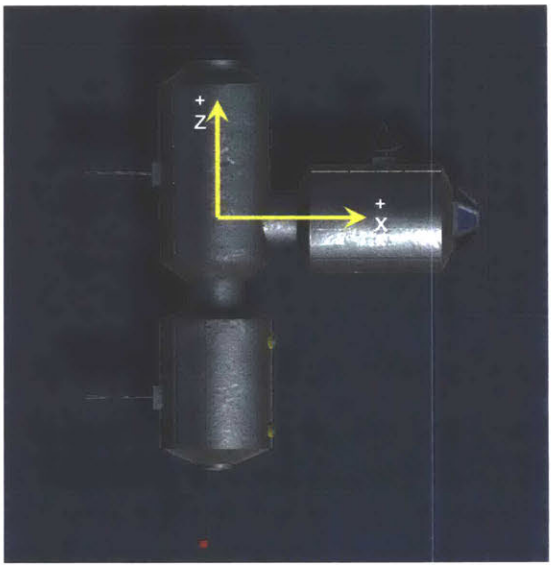
Control of the inspector is achieved through the holographic control pad that will appear in front of your eyes. This control pad cannot be moved during the simulation. You can toggle between waypointing and directional navigation pads, as well as toggle the directional navigation pad between a global and a local mode, as explained below.



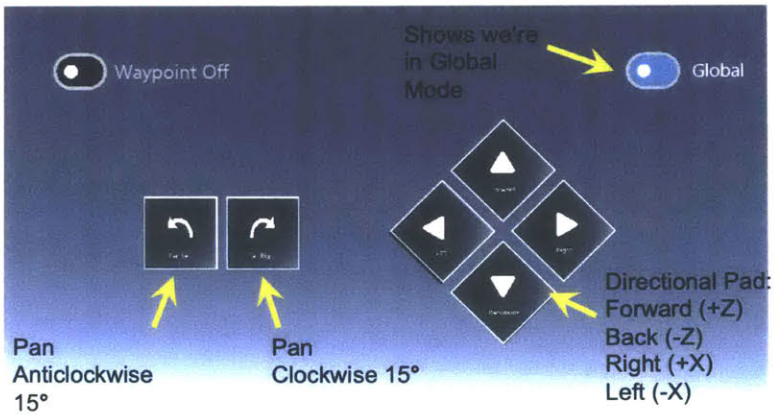
Global Directional Pad

The global directional pad allows the inspector to be moved incrementally relative to the world frame of the space station. The global directional pad acts like a d-pad on a game controller, allowing you to move the inspector forward/back/left/right and to pan left/right relative to the world coordinate frame (see diagram).

WORLD FRAME



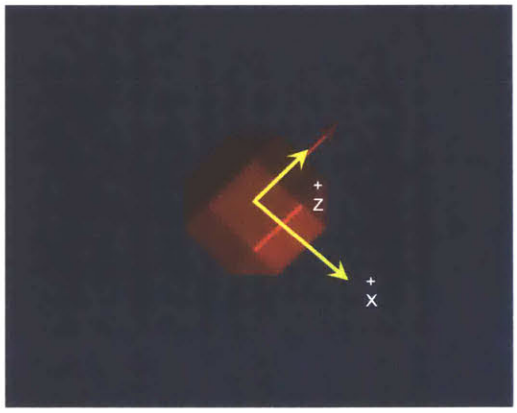
Note: the directional pad buttons cannot be held down for continuous movement. To move the inspector repeatedly in one direction, the button must be pressed multiple times.



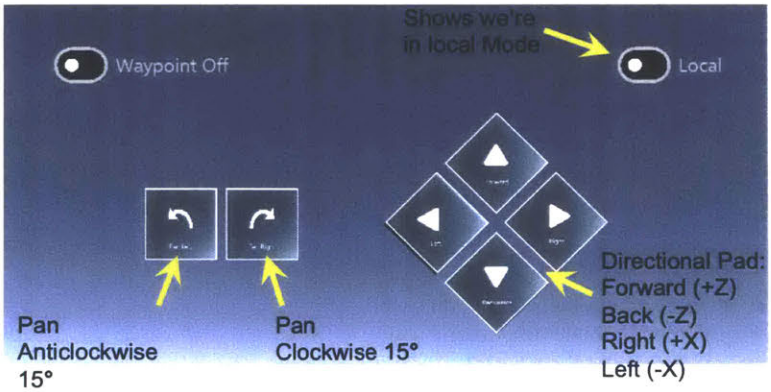
Local D-Pad

The local directional pad allows the inspector to be moved incrementally relative to its own body frame. The local directional pad allows you to move the inspector forward/back/left/right and to pan left/right relative to its own body coordinate frame (see diagram).

LOCAL BODY FRAME



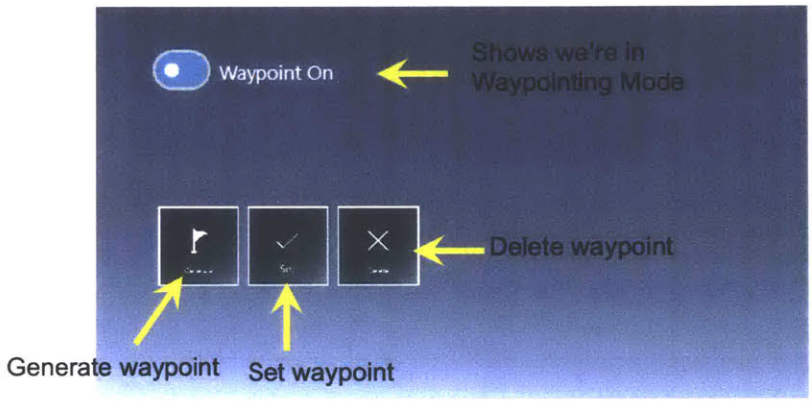
Note: the directional pad buttons cannot be held down for continuous movement. To move the inspector repeatedly in one direction, the button must be pressed multiple times.



Waypointing

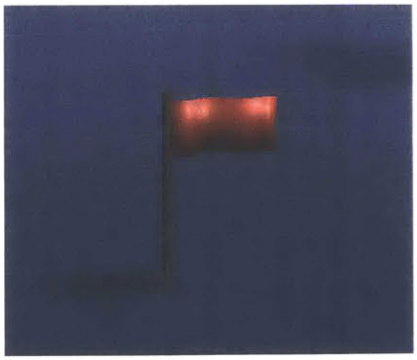
Waypointing is used by placing a marker in the virtual world, as a target for the inspector to move to. The waypoint command mode can be operated by first toggling on the waypoint buttons (see diagram). The three buttons have the following function:

- **Generate:** this generates a new waypoint object in front of the user. The user can then move this waypoint around on the station and position it as desired.
- **Set:** this sets the waypoint as the current target for the inspector. When this button is pressed, the inspector will begin to move towards the waypoint, and the waypoint will disappear when the inspector reaches it.
- **Delete:** this deletes the current waypoint. If the sphere is moving towards the waypoint when you delete it, it will stop at its current position. Delete can also be used prior to set, to remove the active waypoint object.



Note: only one waypoint can be placed at a time. You cannot place the next waypoint while the sphere is moving to the first waypoint. When you scale/rotate/move the space station, any existing waypoints will also scale/rotate/move with it.

Also it is important to realise that the inspector will only move in straight lines between waypoints at a fixed speed. To avoid collisions with the station, you must place your waypoints accordingly.



WAYPOINT MARKER

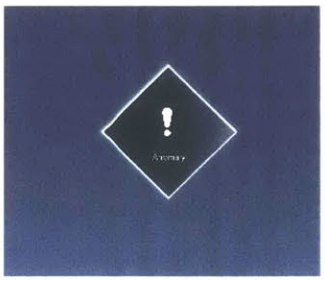
Anomalies

There are multiple anomalies in each test, and between 0 and 3 of these will be critical anomalies. To simulate navigation around an unknown space station, anomalies will only appear when 'detected' by the inspector, i.e. when the inspector gets within certain **proximity** of the anomaly. The direction the inspector is pointing does not matter for the detection of anomalies, only its position relative to the space station. At the conclusion of the test, you will be given a post-test questionnaire that will ask you to identify the locations at which you found critical and non-critical anomalies.

Critical Anomaly

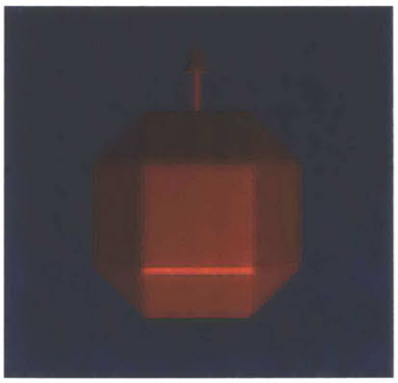
Non-critical Anomaly

Anomalies can be located anywhere on the surface of the space station, at the same y ('vertical') coordinates as the inspector satellite, i.e. at inspector eye level. When you detect an anomaly and determine it to be critical, you must notify your crew by logging the anomaly. You do this by tapping on this button, located in the upper center of your control panel.

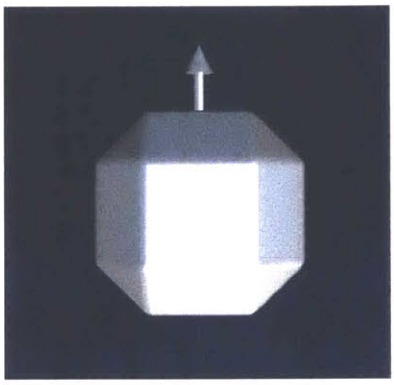


Collision

Your inspector satellite is a delicate piece of equipment, and as such, you want to avoid collisions with the space station which could damage the inspector. When your inspector satellite collides with the space station, the inspector will change colour, from red to white, to indicate you have collided. When the inspector moves away from the station, it will return to its red colour.



Nominal Inspector

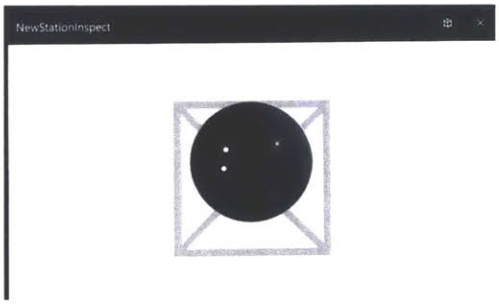


Inspector on Collision

Running the HoloLens App

When you initially turn on the HoloLens, and whenever you return to the main menu. You should see the image below. If you have exited an app and do not see this image, perform the 'bloom' gesture to return here.

When you initially start one of the HoloLens apps, you should see an application window like the one below. Please verbally confirm you can see this window to your test administer. If you do not see this window before the app opens, inform the administer.



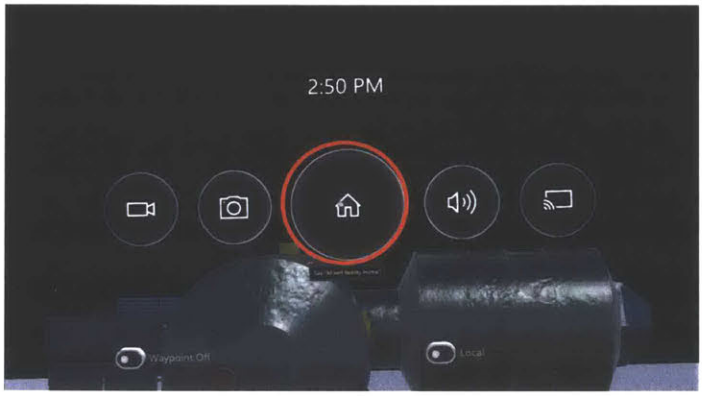
Following the application window, your screen should go black, and then the UNITY Gaming Engine logo should appear. As before, please verbally confirm you see this window, and if you do not, please tell the test administer.

Exiting the HoloLens App

You may exit the app in one of two cases:

- when you are satisfied that you have completed your inspection of the space station, OR
- when the game timer runs out and a pop-up appears telling you its game-over.

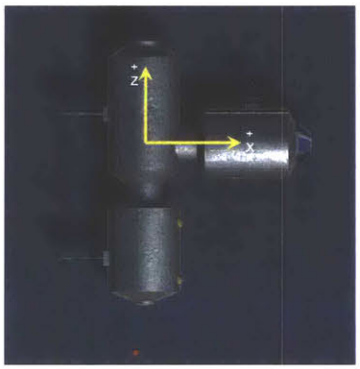
To exit the app, perform the 'bloom' gesture. When the following menu appears, select the centre 'Home' button with your cursor and 'tap'.



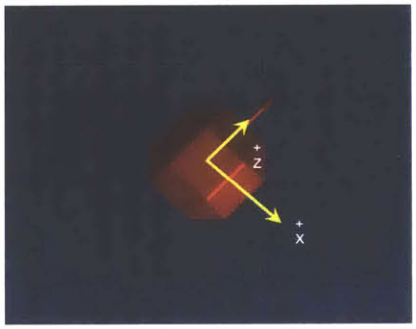
D.8 Subject Cheat Sheet

Station Inspection CHEAT SHEET

Frames of Reference



Global D-Pad:
WORLD FRAME



Local D-Pad:
LOCAL BODY FRAME

Anomalies

CRITICAL Anomaly

NON-critical Anomaly

D.9 SA Questionnaire

Post-Test Questionnaire:

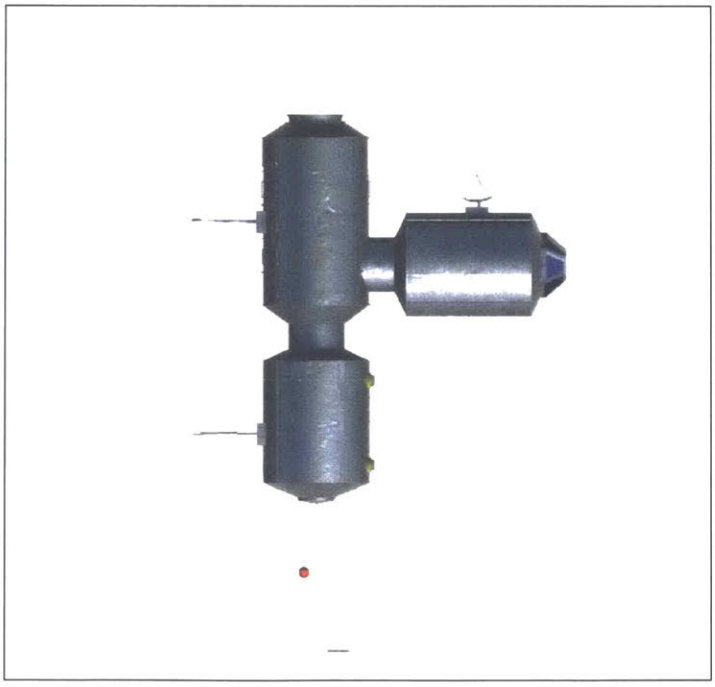
Please fill out the following questions at the end of each trial:

Subject ID: _____
Trial Name: _____

1. On the diagram of the station below, please draw the path you believe the inspector satellite took on its inspection of the space station using a red marker.
2. How many critical anomalies did you detect? _____

   Please place a blue X on the diagram at the location of the critical anomalies.
3. How many non-critical anomalies did you detect? _____

   Please place a green circle on the diagram at the location of the non-critical anomalies.



4. How many collisions do you think you made with the space station over the course of the test? _____
5. ONLY IF WAYPOINTING:
    a. How many waypoints did you place? _____
    b. Please indicate with a purple X on the diagram above where you placed waypoints.
6. Approximately how many times do you think you
    a. Moved the station? ____
    b. Rotated the station? ____
    c. Scaled the station? ____

D.10 Post-Test Qualitative Questionnaire

Post-Experiment Questionnaire:

Please fill out these questions at the end of the unstructured tests

1. Were there any areas you took longer to inspect than others?

2. What areas of the station did you consider high risk? Did your strategy change based on the control mode you were using?

3. What areas of the station did you consider low risk? Did your strategy change based on the control mode you were using?

4. Did you have a preference of control mode? Did this preference change across the task?

5. Which viewpoints of the station did you prefer and why? Did this preference change across the task? (Consider scale, position and orientation in your answer)

6. How does your performance in the unstructured test compare to that of the structured test?

7. Would you prefer to perform this task in an unstructured or structured way? If structured, which of the three modes would you prefer?

8. Overall, how did you find the experience of using this simulation? Would you use this system again?

9. Other comments/feedback?