Responsive Space Environments: New Paradigms in Blending Virtual and Physical Exploration through Human-Robot Operations

by

Don D. Haddad

S.M., Massachusetts Institute of Technology (2018)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2023

Authored by .....
Don D. Haddad
Program in Media Arts and Sciences
August 18, 2023

Certified & Accepted by .....
Joseph A. Paradiso
Alexander W. Dreyfoos (1954) Professor,
Academic Head, Program in Media Arts and Sciences

Responsive Space Environments: New Paradigms in Blending Virtual and Physical Exploration through Human-Robot Operations

by

Don D. Haddad

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on August 18, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Media Arts and Sciences

Abstract

As humanity wanders further into space, the integration of robotics and virtual environments in space systems becomes increasingly pivotal, augmenting our capacity to explore, interact, and study remote environments. This research delves into the use cases, benefits, and challenges associated with Human-robot operations in virtual space analog environments, proposing innovative visualizations that work in tandem with automation to address challenges in mission planning and control, leveraging techniques akin to video game interfaces. In the pursuit of modeling and capturing the essence of distant environments, this research embarks on an investigation of cutting-edge 3D reconstruction techniques, expertly combined with high-definition rendering pipelines, to synthesize virtual environments sourced from real-world information and examine their role in planning and executing planetary exploration missions. This synthesis process becomes especially critical in remote settings, like on the Moon, Mars, and asteroids, where data transmission costs are high and efficiency is paramount in space exploration.

Accordingly, this dissertation also overviews the design and analyzes the performance of AKALL (Azure Kinect à la Luna), a software application for 3D imaging developed during this research that underwent rigorous testing on the SSERVI Lunar regolith testbed at NASA Ames Research Center. This software is designed to operate within an isolated Docker container and was integrated to operate a repurposed commercial imaging payload within Lunar Outpost's Mobile Autonomous Prospecting Platform (MAPP) lunar rover, an integral component of the upcoming Intuitive Machines mission, (IM-2), slated to land on the Moon's south pole early in 2024.

Thesis Supervisor: Joseph A. Paradiso
Title: Alexander W. Dreyfoos (1954) Professor in Media Arts and Sciences
Program in Media Arts and Sciences

This doctoral thesis has been examined by the following committee:

Prof. Joseph A. Paradiso .....
Thesis Supervisor
Alexander W. Dreyfoos (1954) Professor in Media Arts and Sciences
(MIT)

Prof. Dava J. Newman .....
Project Supervisor, Thesis Committee
Apollo Program Professor of Astronautics
(MIT)

Dr. Scott W. Greenwald .....
Reader, Thesis Committee
CEO of Three Space Lab, Inc.
(Founder of MIT Reality Hack)

Acknowledgments

This thesis springs from a shared vision that connects the experience I've gained working with virtual environments to space operations. It represents a cross-disciplinary collaboration between *Prof. Dava Newman* and *Prof. Joe Paradiso*, originally ignited by *Ariel Ekblaw* and the **Space Exploration Initiative (SEI)** at the **MIT Media Lab**.

None of the endeavors detailed within would have been possible without the vital contributions of *Cody Paige*, or the assistance provided by *Ferrous Ward*, *Jessica Todd*, and *Alexandra Forsey-Smerek*. That also extends to members of my thesis committee: *Scott Greenwald*, NASA Ames Research center scientists: Jennifer Heldmann, Amanda Cook, the talented team at Lunar Outpost and Boston Dynamics: Ben Brokaw, Matt Mitchel, David Robert.

To the **MIT RESOURCE** team and the **RESPONSIVE ENVIRONMENTS** group of the MIT Media Lab, starting with the support provided by *Mark Feldmeier*, *Gershon Dublon*, and *Brian Mayton*. Including friends and colleagues that I've met along the way: *Artem Dementyev*, *Irmandy Wicaksono*, *Jie Qi*, *Pragun Goyal*, *Spencer Russell*, *Nan Zhao*, *VRAM*, *Cedric Honnet*, *Fangzheng Liu*, *Evan Lynch*, *Patrick Chwalek*, *Sam Chin*, *Ishwarya Ananthabhotla*, *David Ramsey*, *Juliana Cherston*, *Valentina Sumini*, *Clement Duhart*, *Devora Najjar*, *Nan Wei Gong*, and *Philip Cherner*.

I realize that my own journey through this research has been a living embodiment of these very principles. That includes the process of crafting, refining, and testing ideas at a distance; in many ways, my attempt to reach out across the miles that separated me from my family and loved ones. To my family, *Lida Simonides Haddad*, ***Nina Maria Gentile***, *Bruna*, *Ryan and Bechara Haddad*, *Lisa and Mark Gentile*. To *Lena*, *Rosy*, *Athena Simonides* and their families. To *Georges*, *Catherine, and Alexi*, *Anne-marie*, *Nawall*, *Jean*, *Tony Haddad* and their families. To *Nicolas, Odette, and Rania Tueni* and their families.

To my dear friends from MIT: *Zoz*, *Jack Forman*, *Jifei Ou*, *Dan Oran*, *Ido Calman*, *Alfonso Parra Rubio*, *Gabriela Bila*, *Joao Wilbert*, *Joost Bonsen*, *Arthur Petron*, *Zackery*, *Basheer Tome*, *Ken Nakagaki*, *Dan Pillis*, *Mike Hao Jiang*, *Jimmy Day*, *Carl Yang*, *Brian Tice*, *Tomas Vega*, *Cathy Fang*, *Pat Pataranutaporn*, *Abhinandan Jain*, *Adam Horowitz*, *Sang-won Leigh*.

To Professors and mentors at MIT that inspired me throughout this journey: *Tod Machover*, *Joi Ito*, *Neil Gershefield*, *Glorianna Davenport*, *Hiroshi Ishii*, *Pattie Maes*, *Neri Oxman*, *Hugh Herr*, *Marc Reibert*, *Robert Morris*, and *Edith Ackermann*. To the MIT Dean of Architecture *Hashem Sarkis*, To *Gaby Farhat* and his family, To *Habib Haddad*, and *Calvin Chin*. To *Cynthia Solomon*, *Gloria Rudisch Minsky* and her family. To *Prof. Ken Perlin*. To my Professors and Teachers: *Danielle Azar*,

Munjid Musallam, *Joseph Khalife*, *Chadi Nour*, *Zahi Nakad*, *Bassam Moujabber*, *Thomas Voden*, and *Richard Guglielmino*.

To my dear friends and collaborators: *Stefan Unterhauser*, *Nicolas Saliba*, *Mike Aoun*, *Joe Francis*, *Marc Farra*, *Mario Achkar*, *Maria Saade*, *Peter Rechdan*, *Elie Mahfouz*, *Alan Abi Sleiman*, *Richy Mattar*, *Dante Latessa*, *Austin Iglesias Saragih*, *Elio Massih*, *Joseph Massih*, *Elias Moubarak*, *Georges Kanaan*, *Gerard Feghali*, *Kifah Daher*, *Junior Sfeir*, *Renata Sabela*, *Christopher Safatli*, and *Frederick Daniel B Garzon*.

Throughout the years, to those who helped me navigate the academic and administrative landscapes: *Amna Carreiro*, *Sarra Shubart*, *Maria Auday*, *Keira Horowitz*, and *Linda Peterson*. To the Senior House Community, (R108, R320), to *Joseph Graham, Andrew, Mike, and Rocko*, and to the artificial intelligent agents that helped along the way including *GPT-4*, *Alexa*, *Bard*, and *Lex Fridman*.

To **Prof. Dava J. Newman** It has been a true honor to be part of your team, thank you for including me, and thank you for your trust.

To **Prof. Joseph A. Paradiso** Thank you for having faith in me, especially in those moments when I did not have it in myself.

Thank you all.

To *Elia & Mary & Marcelle Simonides & Antoinette Kamato*

Byblos, & Beirut
Cambridge, Massachusetts

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Prelude

Figure 1-1: *A Pan Am shuttle prepares to dock at Space Station V in this timeless scene from 2001: A Space Odyssey [1]. Image courtesy: MGM/Stanley Kubrick Productions.*

In the grand cosmos of space exploration, a wondrous interplay unfolds between humankind and machines. A symbiotic partnership, ever-evolving and burgeoning in significance, casts its radiant glow upon the horizon of future missions. Behold the awe-inspiring successes of NASA's storied history, a testament to the harmonious dance of human and machine. From distant celestial realms, like the enigmatic Mars, robotic systems emerge as beacons of ingenuity, guiding our ventures into the unknown. In the pages of this thesis, a voyage of concepts related to space operations commences, uncovering the profound connection of Human-robot interactions amidst the cosmic tapestry of space exploration.

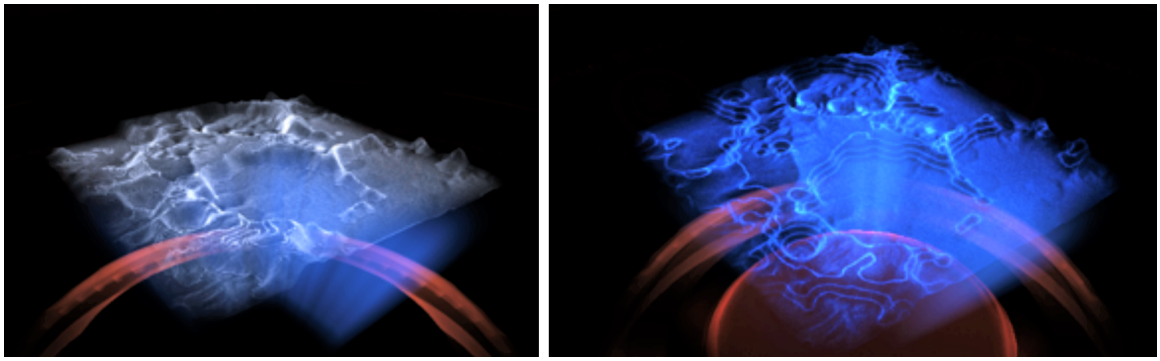

Figure 1-2: *An Interactive Lunar Terrain Visualization I developed in 2019 in collaboration with Cody Paige [2]: This application modifies a holographic shader in the Unity game engine to render lunar terrains from real lunar data gathered from NASA Moon Trek [3], with elevation depicted by rings.*

This thesis and research hope to shed some light into the striking significance of Human-robot interactions and operations in the context of space exploration through novel applications. A central theme in this dissertation entails leveraging Human-robot collaboration to capturing data from an environment, synthesizing the collected information, and crafting immersive virtual landscapes for scientific research. Upon acquiring such a virtual environment, a realm of exciting possibilities emerges, allowing additional robots and humans to inhabit and learn within this simulation, propelled by fragments of reality. This profound synergy of Human-robot and virtual world synthesis, intricately woven within a continuous feedback loop, could potentially evolve into a living model that blends real and virtual scenes, memories, and ultimately discoveries.

The subsequent sections of this chapter delve into the related work pertinent to this thesis, accompanied by a selection of historical backdrop in space exploration and robotic advancements.

1.2 Background and History

Space exploration, a pursuit that has stirred human imagination and ambition for centuries, stands at a critical juncture. The exponential growth in costs, coupled with technological limitations, has prompted the need for innovative approaches to both manned and unmanned missions. In a thought-provoking analysis by Marvin Minsky in 1990 [4], he pointed to the challenges faced by the United States in the design and implementation of space stations, emphasizing the need for a third alternative that goes beyond traditional paradigms. His words still resonate today, as the global community grapples with designing practical, cost-effective, and sustainable space missions. A central consideration in this context is the principle of in situ resource utilization (ISRU) [5], which represents a transformative approach to space exploration. ISRU is identified as a key milestone, as it would enable human

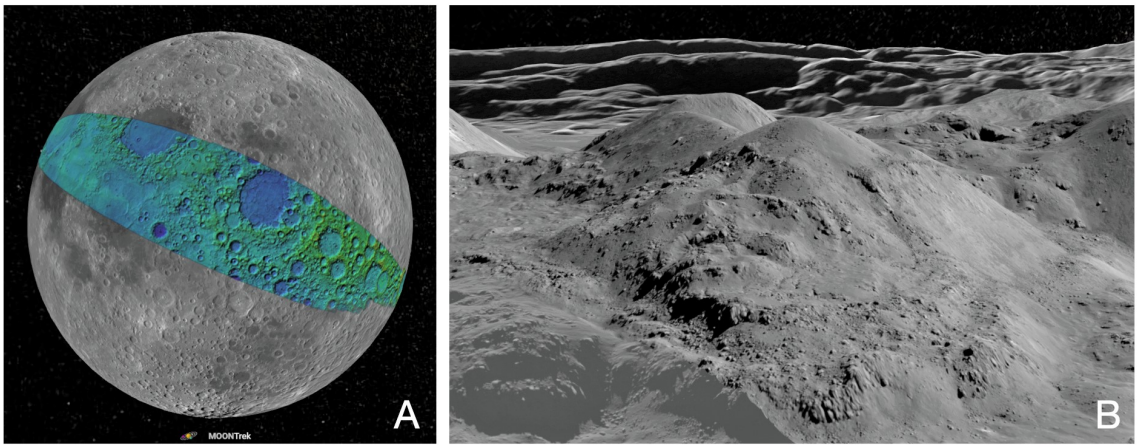

Figure 1-3: *Exploring the Moon with NASA Moon Trek [3]: Image A displays the Apollo 15 Metric Cam DEM, featuring color-enhanced hills shading. Image B reveals a high-resolution rendering of a scanned lunar area, providing detailed lunar exploration capabilities through the user-friendly NASA web portal. Image courtesy: NASA Trek.*

lunar exploration and potentially support a lunar economy, facilitating deep-space exploration. To ensure sustained ISRU missions, Human-computer interaction (HCI) needs to play a central role in mission planning. By treating machines as collaboration entities, cross-discipline communication can be improved, real-time and delayed decision-making processes can be enhanced, task loads can be reduced, and flexibility in spatiotemporal planning can be achieved [6, 7].

The Artemis III Science Definition Team (SDT) report highlights the significant benefits of real-time, and delayed, transmission of data from science instruments, allowing science support teams to provide feedback to the crew and enabling tactical decision-making based on processed data [8]. Virtual Reality (VR) has been suggested as a potential tool to address these requirements. The MIT RESOURCE team. (Section 2.1), focuses on addressing ISRU needs through a structured program that connects science and exploration [9].

The MIT RESOURCE specifically concentrates on optimizing Human-robot interaction for resource prospecting missions and lunar ISRU. The team has made initial progress in developing the virtual Mission Simulation System (vMSS) to support these efforts [10]. Creating a three-dimensional map of the lunar surface holds the potential to serve as a basis for analysis tool development and establish an in-situ scale reference system. Existing tools developed by NASA such as Moon Trek [3], (Figure 1-3), and LROC QuickMap allow users to draw traverse paths [11], calculate distances, elevations, sun angles, and overlay orbital data. However, these tools lack the necessary level of detail required for in-situ geological analysis, which demands higher-resolution depth data than what is available through orbital data alone.

Figure 1-4: *"By connecting robots to the metaverse, we will be able to move freely between both real world and virtual realities [14]. Image courtesy: Hyundai.*

1.2.1 Mobile Robotic Telepresence

In his pioneering paper on Telepresence [12], Marvin Minsky talks about the feasibility of a remote controlled economy by the twenty first century [13]. A few decades later now, we are closer than ever. Most recently, Hyundai Motor has branched its interest on advancing robotics and telepresence by acquiring Boston Dynamics, therefore, through a synergistic combination between VR technologies and robotics, reveals a new concept called "Metamobility" [14]. Ultimately, this framework intends to allow people to break through the physical boundaries of movement across space and time, where robots, acting as middleware agents, (Figure 1-4), help bridge the gap between real and virtual experiences. When not under the effects of this "digital possession", the robots can behave in their normal idle state, or by a pre-configured state, where they essentially become Non-player characters (NPC).

The field of mobile robotic telepresence is rapidly expanding, with both commercial systems and research efforts emerging. The related work section throughout this chapter investigates the use of telepresence and robotics in certain applications, particularly focusing on mobile robotic telepresence (MRP) systems [15]. Telepresence, as a concept, revolves around the sense of being present in another environment, and robotic telepresence takes this a step further by enabling remote connection to a distant location while also allowing physical movement and actuation in that location. MRP systems specifically concentrate on enabling social interaction through video conferencing capabilities, coupled with the ability to move and steer the robot to different locations. These systems typically consist of an LCD screen, a web camera, a microphone, and speakers, facilitating communication between two parties. The user operating the MRP system remotely can navigate the robot's environment, effectively

becoming embodied in the system and interacting with individuals on-site. The field of mobile robotic telepresence is rapidly expanding, with both commercial systems and research efforts emerging [16].

1.2.2 History of NASA's Unmanned Rovers on Mars

NASA has successfully deployed a series of rovers, (Figure 1-5), to explore and traverse the enigmatic terrain of the red planet. Rovers play a crucial role in capturing images, conducting experiments, and investigating the mysteries that long shrouded Mars [17]. Unlike landers that remain stationary at their landing sites, rovers possess the ability to move and actively explore diverse areas. Since the inaugural mission in 1996, NASA has impressively landed a total of six rovers, the latest one being the Perseverance rover that landed on February 18, 2021 on Mars [18], with two of them still operational today. Each of these missions serves various objectives, but they share a common goal of unraveling the potential for ancient environments conducive to life [19], including the search for traces of liquid water in Mars' past.

As time progresses, these rovers continue to advance, equipped with an array of cutting-edge instruments that enable comprehensive exploration of the Martian landscape. The progression of NASA's Mars rovers, spanned over the last few decades, marks an intriguing journey filled with remarkable discoveries and invaluable insights. The first foray into Martian exploration with rovers began with the Pathfinder mission, which successfully delivered the Sojourner rover to the red planet's surface in July 1997 [20]. Sojourner, equipped with an x-ray spectrometer and multiple cameras, landed at Ares Vallis using airbags to cushion its descent [21]. The rover's initial images unveiled rounded pebbles and cobbles, providing groundbreaking evidence of the presence of stable liquid water in Mars' history. Sojourner's mission surpassed its intended duration of one week, persisting for several months until September 1997, although it remains the shortest-lived rover mission to date [22].

The subsequent milestone in Martian exploration came with the Mars Exploration Rovers (MERs) Spirit and Opportunity, launched in 2004. These two rovers embarked on a joint mission, each touching down on opposite sides of Mars at Gusev Crater and Meridiani Planum, respectively [23]. Similar to the Pathfinder mission, Spirit and Opportunity relied on airbags to cushion their landings. Equipped with cameras, various spectrometers, rock abrasion tools, and magnet arrays, these rovers aimed to investigate the presence of water and better understand Mars' climate. The discoveries made by Spirit and Opportunity provided compelling evidence suggesting the possibility of past Martian life [24]. The detection of clay minerals, indicative of neutral pH waters, and hematite, a mineral associated with water, offered tantalizing clues. Additionally, the presence of jarosite, a mineral formed in acidic water, hinted at the potential for more extreme forms of life. Spirit operated for over six years, while Opportunity surpassed expectations, continuing its mission until 2018 when a dust storm led to a loss of communication with Earth [25].

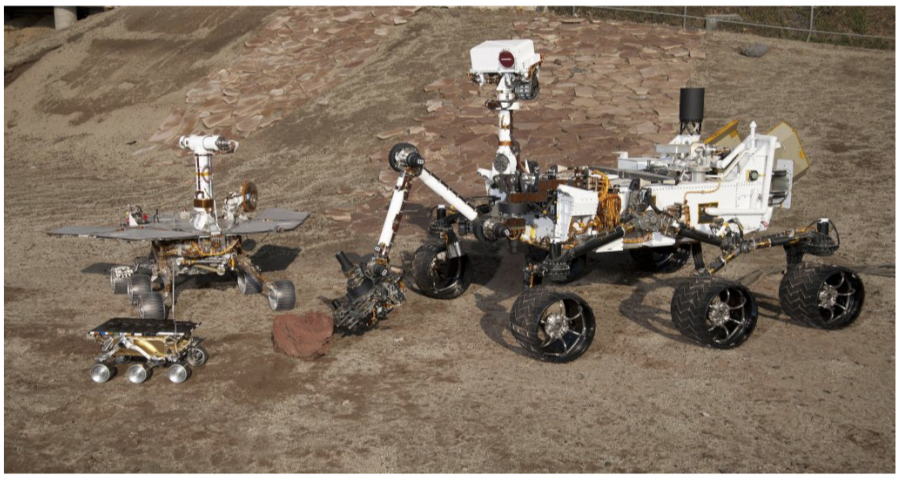

Figure 1-5: *History of Mars Rovers[a]: A Comparison in Size and Evolution. This captivating image showcases three generations of Mars rovers at NASA's Jet Propulsion Laboratory, Pasadena, CA. The flight spare of the pioneering Mars rover, Sojourner, landed on Mars in 1997. On the left is a Mars Exploration Rover Project test rover, a sibling to Spirit and Opportunity, which reached Mars in 2004. Lastly, on the right stands a Mars Science Laboratory test rover, equivalent in size to the remarkable Curiosity rover.*
Image courtesy: NASA/JPL-Caltech.

[a]bmsis.org/history-of-nasa-mars-rovers

The Curiosity rover, which landed in 2012, at the time was the most advanced rover to ever explore Mars. It was equipped with a suite of instruments that allow it to conduct a wide range of scientific investigations, including the search for organic molecules, which are essential for life [26]. Curiosity has so far made some remarkable discoveries, including evidence that Mars once had a much more hospitable environment than it does today. Curiosity's landing in Gale Crater differed from previous rovers, as it utilized a parachute to slow its descent and facilitate Sky Crane landing [27]. Equipped with an impressive suite of instruments, including cameras, spectrometers, radiation detectors, and environmental and atmospheric sensors, Curiosity embarked on a mission to uncover the geological and chemical history of the region. One of its notable discoveries involved the Sample Analysis at Mars (SAM) instrument, which detected organic carbon in rocks from Mount Sharp [28], a prominent mountain within Gale Crater. Moreover, radiation detectors on board highlighted the potential health risks posed by galactic cosmic rays and solar energy particles to future astronauts. Curiosity remains operational, currently traversing from a clay-rich region to a sulfate-rich region on Mount Sharp [25].

The latest addition to NASA's Martian fleet is the Perseverance rover, launched in 2020 and successfully touching down on Mars in February 2021. Perseverance's mission is accompanied by a groundbreaking exploration device, a helicopter named Ingenuity. Additionally, Perseverance initiates the process of collecting and storing

samples of Martian rocks for potential return to Earth—an unprecedented endeavor that could yield invaluable scientific insights. The rover carries a suite of cutting-edge instruments, such as the Mars Environmental Dynamics Analyzer (MEDA) [29] for measuring weather patterns, a ground-penetrating radar called the Radar Imager for the Mars Subsurface Experiment (RIMSE) [30], and the Scanning Habitable Environments with Raman & Luminescence for Organics & Chemicals (SHERLOC) instrument for identifying organics and biosignatures [31]. Notably, Perseverance is equipped with the Mars Oxygen ISRU Experiment, known as MOXIE [32], a unique instrument designed to produce oxygen from the carbon dioxide present in Mars' atmosphere. With an original mission length of one Mars year, equivalent to approximately two Earth years, Perseverance holds great promise for groundbreaking discoveries yet to come [33].

The Mars rovers have made an outstanding contribution to our understanding of the red planet. They have provided us with a wealth of data and images that have helped us to better understand Mars' past, present, and future. The discoveries made by these rovers have inspired and motivated scientists and the public alike, and they have helped to pave the way for future human exploration of Mars. Despite the remarkable strides made in Martian exploration, there is still much to learn about the history of Mars and its potential for supporting life. Rovers continue to be vital technologies, allowing us to gather crucial information about extraterrestrial objects that remain beyond our physical reach [25].

1.2.3 Extended Reality: Pioneering Work & Taxonomies

In 1968, Turing Award recipient Ivan Sutherland, known for inventing the Sketchpad [34] (a computer program built in 1963 at MIT Lincoln Lab that pioneered the field of Human-computer interaction and ran on the Lincoln TX-2 [35]), and his student Bob Sproull, created the ultimate display at Harvard University. This creation was the first Head Mounted Display that could be wired to a computer, not just a camera, marking a significant advancement in the field [36]. Following, In 1969 Myron Krueger, computer graphics artist and pioneer, developed a series of experiences which he termed Artificial Reality in which he developed computer-generated environments that responded to their participants [37]. The projects named Glowflow, Metaplay, and Psychic Space were progressions in this research, which ultimately led to the development of the Videoplace [38] technology. This technology enabled people to communicate with each other in a responsive computer-generated environment despite being miles apart. Even after all of this research and development, there still wasn't an all-encompassing term to describe the field. This all changed in 1987 when Jaron Lanier, founder of the visual programming lab (VPL), coined the term Virtual Reality (VR) [39]. Through his company VPL research, and along with other key players such as Tom Zimmerman, Jaron has developed a range of Virtual Reality add-ons including the Dataglove, and the EyePhone [40] Head Mounted Display, (Figure 1-11). VPL research was the first company to sell VR goggles and gloves, albeit with a nifty price tag. In parallel, NASA, with the help of Crystal River Engineering,

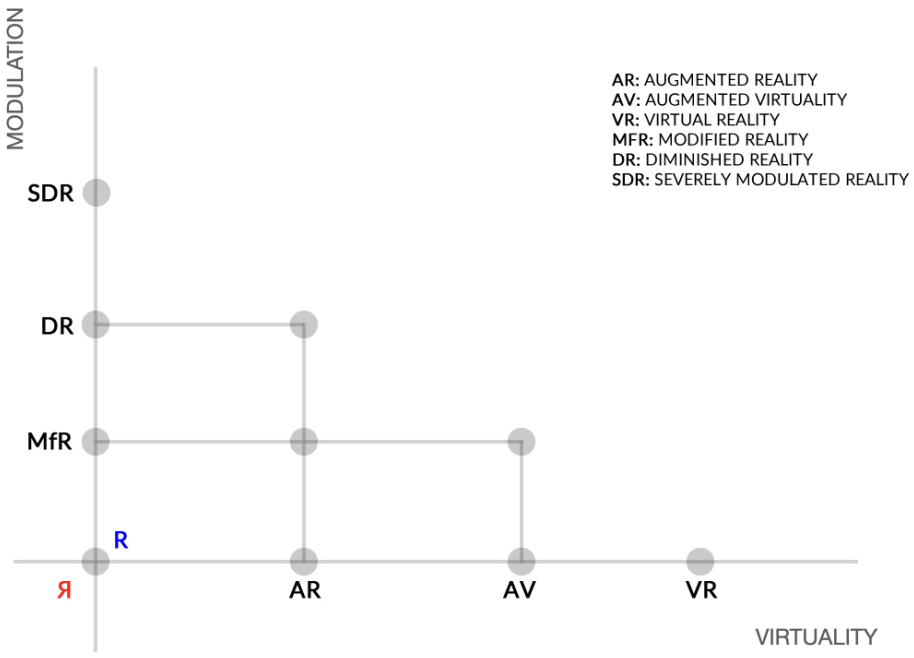

Figure 1-6: *The Axes of Mediated Reality [47].*

created Project View: a VR simulation used to train astronauts [41]. View looks recognizable as a modern example of VR and features gloves for fine simulation of touch interaction. Interestingly, the technology in these gloves lead directly to the creation of the Nintendo Power Glove, originally developed for gaming. In modern academia, Computer Mediated Reality (CMR) highlights the role of the computer in generating, synthesizing or editing the user's reality. Steve Mann was the first to propose the term, as well as a series of taxonomies that he placed on a continuum [42]. Augmented Reality, Mixed Reality and Virtual Reality – presented by the different modern Head Mounted Displays (HMDs) and goggles – provide experiences within the reality-virtuality-continuum, ranging from Augmented Reality (AR) to Augmented Virtuality (AV) [43, 44]. In 1994, Paul Milgram and Fumio Kishino proposed the taxonomy Mixed Reality [45], distinguishing it from both Virtual Reality and Augmented Reality, merging the axes of Reality and Virtuality all together, allowing virtual objects to co-exist within the physical environment [46].

Steven Feiner defines Augmented Reality as displays that add virtual information to a user's sensory perception [48]. In contrast with this nomenclature, Ken Perlin considers anything that mediates perception as just part of our reality. For instance, Ken Perlin and his students have been working with VR headsets to prototype the next generation AR interactions [49]. As these fields are constantly blending into one another, it would be really difficult to distinguish between their "types". In other terms, whether augmented, virtual, extended, diminished or mixed; what we experience remains part of our own reality. Moreover, Diminished Reality introduces the concept of removing elements from any of these given realities presented in the con-

tinuum [50]. Figure 1-6 expands on that topic with the addition of the modulation axis, essentially adding a new dimension to the virtual continuum [45].

Modulation is the process of varying one or more properties of an environment, or a scene, presented to the perceiver, such as subtracting, adding, or altering elements of their reality [51, 47].

- R: In the mediated reality framework, unaltered reality is referred to as the 'real environment'. It is the baseline from which the horizontal virtuality axis, and vertical modulation axis extend.
- AR: First along the virtuality axis is Augmented Reality (AR). AR can be understood as primarily the real environment with some virtual information or computer generated imagery overlaid over the user's view. This can be done either through a screen interface or see-through Head Mounted Display.
- AV: Next on the virtuality axis is Augmented Virtuality (AV). As opposed to AR where computer generated graphics are superimposed onto a mostly real environment, AV enhances a mainly virtual environment with some aspects or elements of the real world.
- VR: Virtual reality (VR), last on the virtuality axis, is a completely artificial, computer-generated environment. This emulation typically requires the use of a Head Mounted Display, such as an Oculus Rift or HTC Vive, to fully immerse the user.
- MfR: First along the modulation axis is modified reality (MfR), where the user's perception is altered through the filtering and modification of real elements.
- DR: Next along the modulation axis is diminished reality (DR), where technology is used to hide or remove real elements from the users' perception. Similar to removing an object from a photo in Photoshop but with a real time-Head Mounted Display.
- SDR: At the extreme of the modulation axis is severely diminished reality (SDR), where the entirety of the real environment is removed. This type of reality, though confusing, essentially results in a form of sensory deprivation.

1.3 Related Work

1.3.1 Digital Twin

The concept of a Digital Twin (DT) was first conceived in the 1960s at NASA as a "living model" of the Apollo mission. As a countermeasure taken to prevent future disasters such as the Apollo 13's oxygen tank explosion [52] and subsequent damage to the main engine, NASA employed multiple simulators to evaluate the failure and extended a physical model of the vehicle to include digital components [53]. By definition a digital twin is a virtual replica of a physical object or system that is updated in real time with data from the real-world object. It is used to monitor and improve the performance of the physical object or system. This is the well-grounded definition of the digital twin amongst engineering systems, nevertheless, this concept has wider ramifications that extends to multiple fields such as healthcare [54], aerospace [55], architecture [56], monitoring and art [57]. Moreover, a digital twin is a virtual representation of a physical object or system, (Figure 1-7). It encompasses several disciplines, such as mathematical modeling and statistics, real-time sensing and actuation, and 3D modeling and data visualization. In other words, digital twins abstract the properties of real-world objects or systems and map them to a virtual space, enabling monitoring [58], predicting [59], and decision-making [60], thus offering valuable capabilities in various domains. Digital twins can be utilized to enhance the performance of physical objects or systems by providing valuable insights into their behavior [61]. Through creating a virtual representation of their real-world counterparts, they enable continuous monitoring and analysis, leading to informed optimizations and refinements.

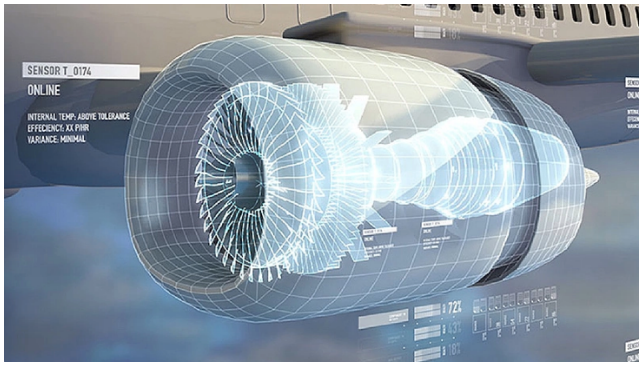

Figure 1-7: *Digital Twin technologies gained popularity in 2017 within aerospace engineering [62]. Companies and educators create a digital model, in this case a model of an aircraft engine, to monitor and troubleshoot problems with the real engine.*

Image courtesy: GE.

Additionally, digital twins can be programmed to have predictive capabilities, enabling the anticipation of future behavior in physical objects or systems. Through data analysis and modeling, potential issues or inefficiencies can be identified, enabling proactive measures to be taken [63]. Lastly, digital twins can be leveraged to facilitate decision-making processes concerning physical objects or systems. By simulating different scenarios and evaluating their outcomes, stakeholders can make informed choices based on accurate and up-to-date information provided by the digital twin [64]. Overall, digital twins serve as powerful tools for improving performance,

predicting behavior, and supporting decision-making in a wide range of applications. In 2012, NASA revisited the concept of digital twins, defining it as a multiphysics, multiscale, probabilistic, ultra-fidelity simulation that reflects the state of a corresponding twin in a timely manner based on historical data, real-time sensor data, and physical models [65]. The digital twin is a virtual representation of a physical object or system. It can be used to monitor and control the physical object, as well as to simulate different scenarios.

In the Fourth Industrial Revolution, also known as Industry 4.0, digital twins will play an increasingly important role in the development and deployment of new technologies. They will contribute in improving the efficiency and productivity of manufacturing processes, and to create new products and services [66]. Digital twins of high-fidelity simulations of actual Unmanned Vehicles (UVs) such as rovers and robots are rapidly becoming indispensable tools for Lunar and Martian surface exploration [67]. They offer a secure, efficient, and economically viable platform for testing various mission scenarios, assessing equipment performance, and strategizing responses to potential challenges. This is achieved without incurring the risks inherent in physical space travel, significantly increasing the breadth of scenarios that can be examined. Beyond their value for mission planning, digital twin environments also play a seminal role in astronaut training [68].

These complex simulations provide a powerful platform for visualizing the complexities of extraterrestrial environments and familiarizing trainees with the systems they will be operating. By reproducing the challenges that await in these uncharted landscapes, digital twins contribute significantly to the development of practical skills and preparedness for real-life space missions [61].

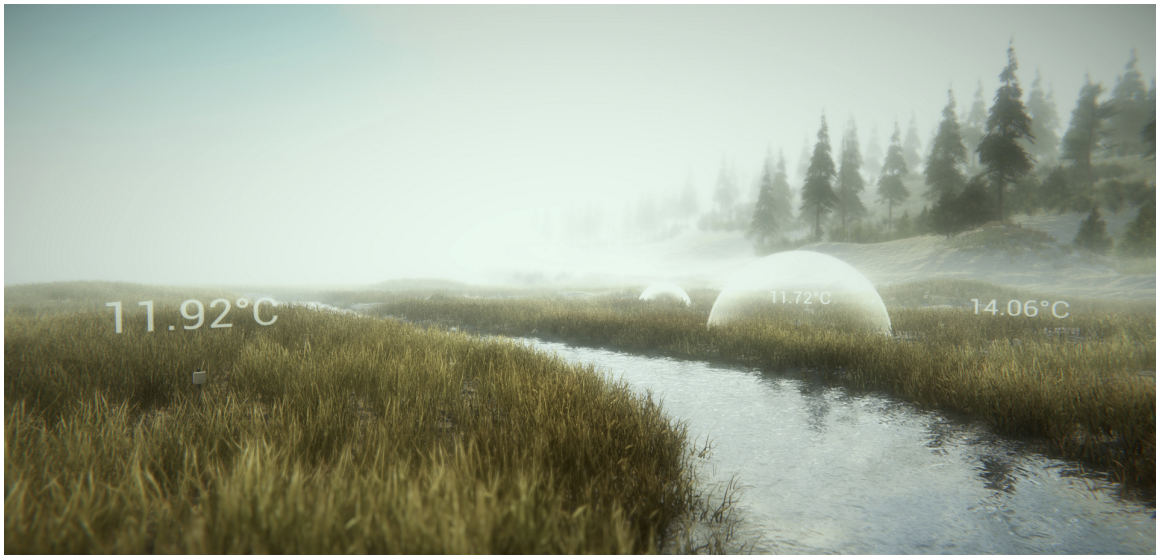

Figure 1-8: *Doppelmarsh, a virtual counterpart of a real marsh in Plymouth, Massachusetts. Representing sensor network data by constructing rich virtual environments to explore vibrant landscapes with elements driven from real-time and historical data [69].*

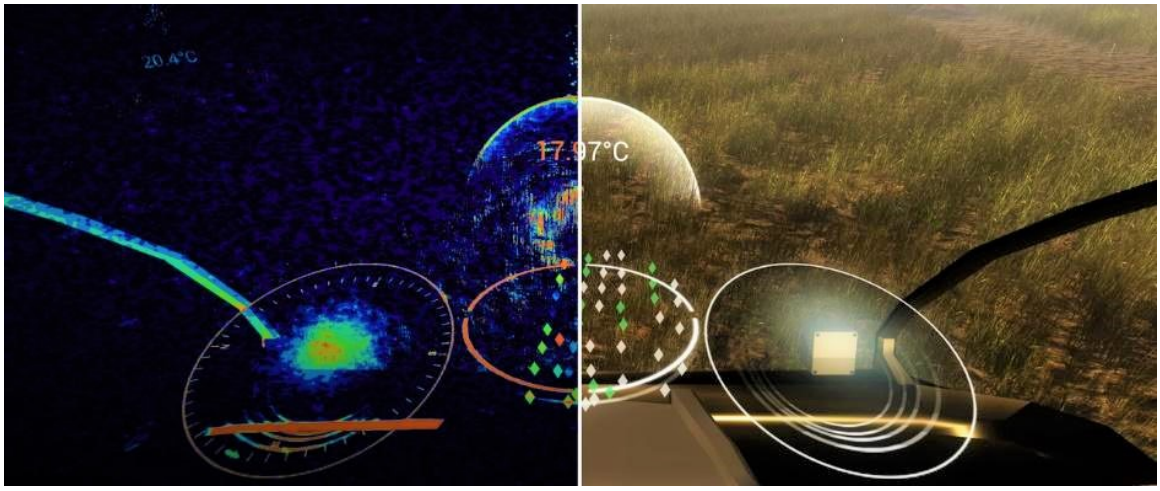

Figure 1-9: *Doppelmarsh's cockpit VR browser serves as a remote monitoring vehicle for sensor network data. By toggling on the "virtual lens", known as SensorVision, it subtracts the texture from the environment, allowing a focused view on the data visualization [69].*

1.3.2 Data Visualization and Manifestation

The convergence of ubiquitous sensing and the Internet of Things, through abstraction has seamlessly infused these technologies into the fabric of everyday life [70]. This transformative trend is not limited to terrestrial applications alone; a promising direction of research indicates that ubiquitous sensing is also converging into space operations and shaping the future of space habitats and everyday life in orbit [71]. Through pairing these digital twin models with their physical counterparts mediated by sensors and actuators, in real-time, a whole paradigm of Cross-Reality (XR) interactions was born [72], polarizing fields such as architecture [73], monitoring [74], healthcare [75], and presence [76] to name a few.

The representation of sensor networks data has evolved beyond traditional information visualizations and monitoring platforms through the integration of modern video-game engines, cutting-edge computer graphics techniques, combined with low power sensing systems, to drive virtual environments with information harvested from the real-world [77]. Since 2009, the Responsive Environments group of the MIT Media Lab has been widely exploring, and pioneering, the interconnection of human perception with blending both real and virtual environments [78]. Coupling the human senses with sensors, all around us, enables a state of superposition amongst one another in both local and remote environments that we can physically or virtually inhabit [79]. These newly acquired “cybersensations“ augmented by the physical affordances of these systems, extending human perception beyond its spatiotemporal limitations, and therefore introduce new sensory modalities [80].

Internet of Things (IOT) devices are the fundamental building blocks that serve to empower the underlying infrastructure where sub-systems come into play, overseeing remote sensing and actuation on a large scale in both indoor and outdoor environ-

ments, (as presented in a sensory landscape [81]). For instance, the project HearThere by the group embodies this concept of distributed perception. This technology is designed to enhance users' auditory abilities within a given environment [82]. It achieves this by leveraging a network of microphones and utilizing a pair of bone conduction headphones that includes an inertial measurement unit (IMU) to also measure head orientation. Through this innovative setup, HearThere can unravel several modalities, effectively expanding the users' hearing perception and immersing them in a rich landscape of sound. This is achieved through a careful mixing of real-time or cached microphone audio signals, providing options for focused remote hearing, noise cancellation, and even granting access to "chronosonic" bubbles, which enable the user to experience sonic journeys through both space and time [83].

In the realm of virtual environments, there are three main categories: fictional, data-driven, and a hybrid of both. Doppellab [84], the Media Lab's Digital Twin, and Doppelmarsh, (Figures 1-8 and 1-9), located at Tidmarsh Wildlife Sanctuary, exemplify this concept. Both projects seamlessly blend elements of the physical environment, utilizing the powerful Unity game engine to render realistic representations. Moreover, they offer users access to a diverse range of sensor data through immersive sonifications and visualizations, comprising a combination of real-time and cached information. This synthesis of real-world data and virtual elements allows for a multi-modal and immersive exploration through an enriching experience within dynamic digital landscapes [85]. Virtual environments serve as a prosthetic for the user's imagination, offering an array of visualizations that leverage humans' natural ability to interpret sensory cues in the physical world. A striking example of this can be observed in Doppelmarsh, where the virtual marsh accurately mirrors the real-time weather conditions on the physical site. Just as prosthesis enhance and extend physical capabilities, virtual environments enrich and amplify the imaginative capacities of users, providing them with a captivating realm to explore and interact with [69].

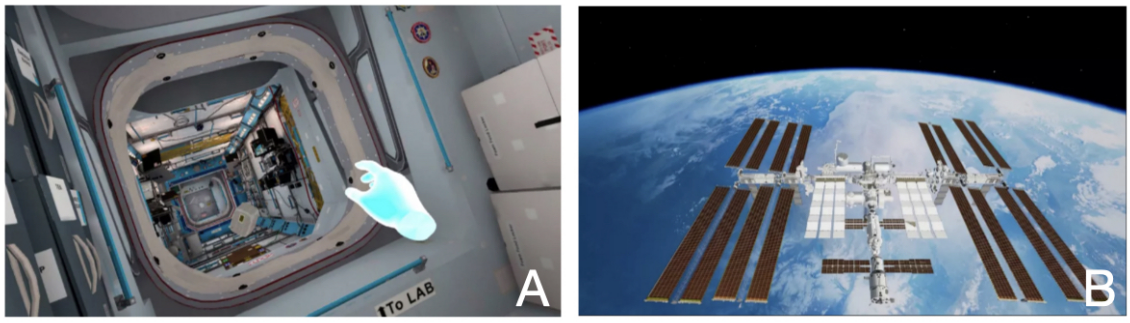

Figure 1-10: *Mission ISS [86] - An immersive VR exploration of the International Space Station (ISS) running on modern VR headsets.*

1.3.3 Virtual Operations in Space Exploration

Figure 1-11: *Pioneering the realm of Virtual Reality, NASA Ames, in collaboration with VPL Research, Inc., unveiled the Virtual Interface Environment Workstation (VIEW) in 1990. The stereoscopic head-mounted display system allowed users to immerse themselves in computer-generated or real environments captured from remote video cameras. The DataGlove, adorned with fiber optic cables and sensors [40], faithfully translated finger movements into the virtual world, enabling users to interact with computer-generated objects.*

Image courtesy: NASA / VPL.

Virtual environments and digital presence bring about a radical transformation in the space exploration landscape. They allow for the convergence of physical and digital worlds [87], akin to digital twins, enabling detailed and dynamic replication of actual environments and scenarios. These digital replicas, driven by real-world information, have long provided an immersive, adaptive, and realistic platform for testing, training, and planning, augmenting old practices in space exploration simulators. The video game industry has long been influenced by space themes and science fiction. In particular, the Real-Time Strategy (RTS) and space simulator genres that emphasize on a set of unique user interactions and interfaces that inspire Human-robot Interaction (HRI). Chapter 3 in this thesis, *Doppelbots*, introduces several of these mechanics and modalities as a concept to be integrated into rover control systems to allow for more engaging navigation in simulations. This development, alongside the use of leading game development platforms such as the Unity and Unreal game engines, opens up new avenues for enhancing the realism while preserving the immersive aspect of virtual space exploration.

Virtual environments are starting to play a crucial role in the realm of space exploration, offering immense potential for remote space exploration and other space-related activities. These simulated digital realms possess distinctive characteristics that enable immersive experiences and interactions. Currently, there exist a variety of virtual worlds, encompassing both commercial and open source platforms. Some notable examples include No Man's Sky, Surviving Mars, Endless Space, among others, comprising numerous islands managed by individuals, organizations, and government entities, and offering downloadable client software for public access.

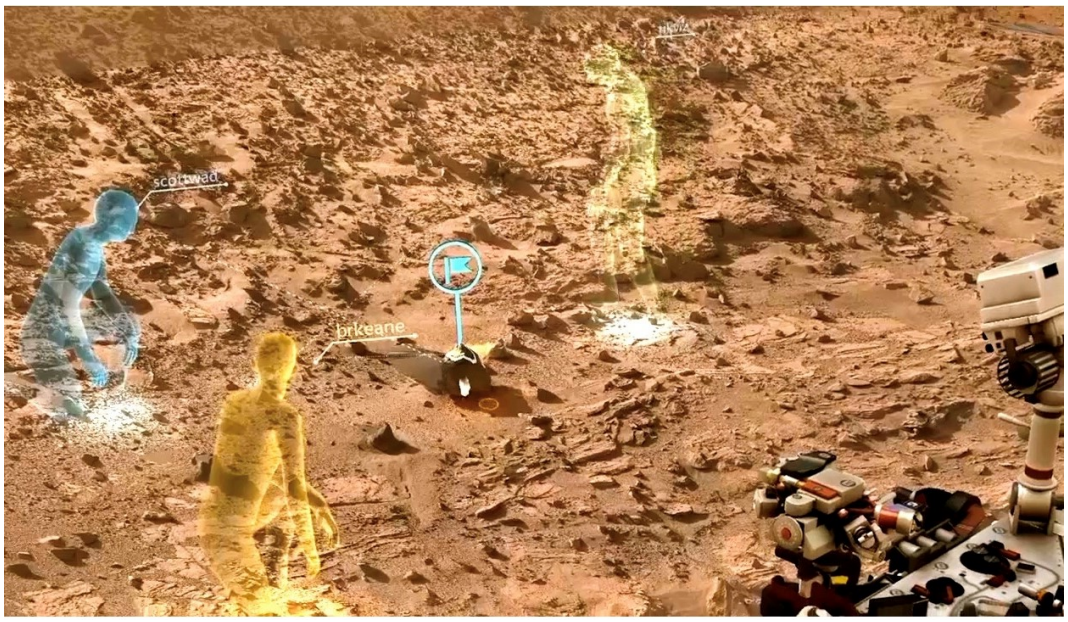

Figure 1-12: *OnSight Project: NASA and Microsoft join forces to explore "holographic computing" in space exploration. In this image, scientists utilize the HoloLens device to virtually work on Mars, thanks to the collaborative software development of OnSight. Image courtesy: NASA JPL.*

The utilization of virtual worlds in space exploration extends beyond gaming and socialization. NASA, aerospace industry, and other space organizations have recognized their value and have embarked on programs to develop and employ immersive synthetic environments [88]. These virtual spaces find application in various domains, including mission support, astronaut training [89], distributed collaboration [90], and public engagement [91]. Augmented Reality already plays a critical role in extreme piloting, as seen, for example, in the heads-up displays in the pilot's helmet for the F-35 Lightning II joint strike fighter [92, 93].

In the context of space exploration, virtual worlds hold immense potential for visualizing and simulating future space missions. They provide high-fidelity visual simulations, enabling the public to experience a sense of presence and immersion in these futuristic endeavors. By harnessing the transformative power of virtual worlds, numerous facets of future space missions can be impacted. Additionally, they foster the concept of collaborative participatory virtual space exploration, facilitating international cooperation and knowledge sharing among space-faring nations. The continuous advancement and adoption of virtual world technologies and facilities are essential for transforming the two-dimensional flat internet into a 360° multi-sensory immersive virtual world experience. This vision aligns with the goal of creating a remote space exploration ecosystem (RSEE), incorporating massively multi-participant synthetic environments and advanced facilities. By offering engaging and inspiring experiences, the RSEE aims to motivate and involve the international community in space activities [94].

In contrast, space analog environments always served a critical role in the progression of space exploration [95]. For instance, environments such as the Mars Desert Research Station (MDRS) [96], or NASA's Extreme Environment Mission Operations (NEEMO) [97], happen to resemble extraterrestrial conditions on Earth, allowing scientists to test and prepare for potential challenges that might be encountered during actual space missions. They offer a preliminary glimpse into the difficulties of remote environments in space exploration, thus contributing to the development of effective strategies for future missions. However, creating such physical analog environments can be challenging and resource-intensive. This is where the concept of the Virtual Space Analog Environments (VSAE) comes into play. Chapter 5 in this thesis describes in details the processes of constructing virtual renditions of physical space analog environments, generated and driven by real-world data, thus providing an immersive and realistic testing ground for geologists [98].

In 2015, NASA's JPL, in collaboration with Microsoft, developed project OnSight [100], (Figure A-1), a Mixed Reality application running on the HoloLens that enables engineers and scientists to fully immerse themselves in a re-creation of the area surrounding the Curiosity rover [101]. Previously, scientists mostly used 2D and panorama photos captured by the rover for their work, making it very challenging to establish a three-dimensional relationship with the environment in a manner similar to how one would explore a terrestrial environment [102]. Using head-worn interactive displays for assisted/augmented maintenance applications has a long history in aerospace, going back at least to pioneering work by researchers at CMU on aircraft inspection and repair in the late 1990s [103]. In recent years, Lockheed Martin, a major American aerospace corporation, lead contractor for NASA's Orion spacecraft, used Mixed Reality (MR) to increase efficiency in building the spacecraft for Artemis II, (Figure 1-14), the first crewed mission aboard Orion. Equipped with a HoloLense2 Head Mounted Display (HMD), the employees started constructing the crew seats for the spacecraft without checking any physical manual or instructions pages, through an MR application that showed them everything they needed to view, including animations demonstrating, (Figure 1-15), how parts fit together, engineering drawings, and torque numbers for tightening bolts [104].

1.3.4 Virtual Human-Robot Interaction

Virtual Human-robot Interaction (HRI) represents an emerging field that seamlessly blends the latest advancements in Mixed Reality headsets with robotic systems [105]. By fusing virtual environments and physical robots, this multidisciplinary area aims to augment how humans and robots interact and collaborate [106]. The integration of Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR) technologies with robotic systems [107] opens up a vast array of possibilities for space exploration, assistive robotics and remote surgeries [108], Human-robot collaboration [109], and more.

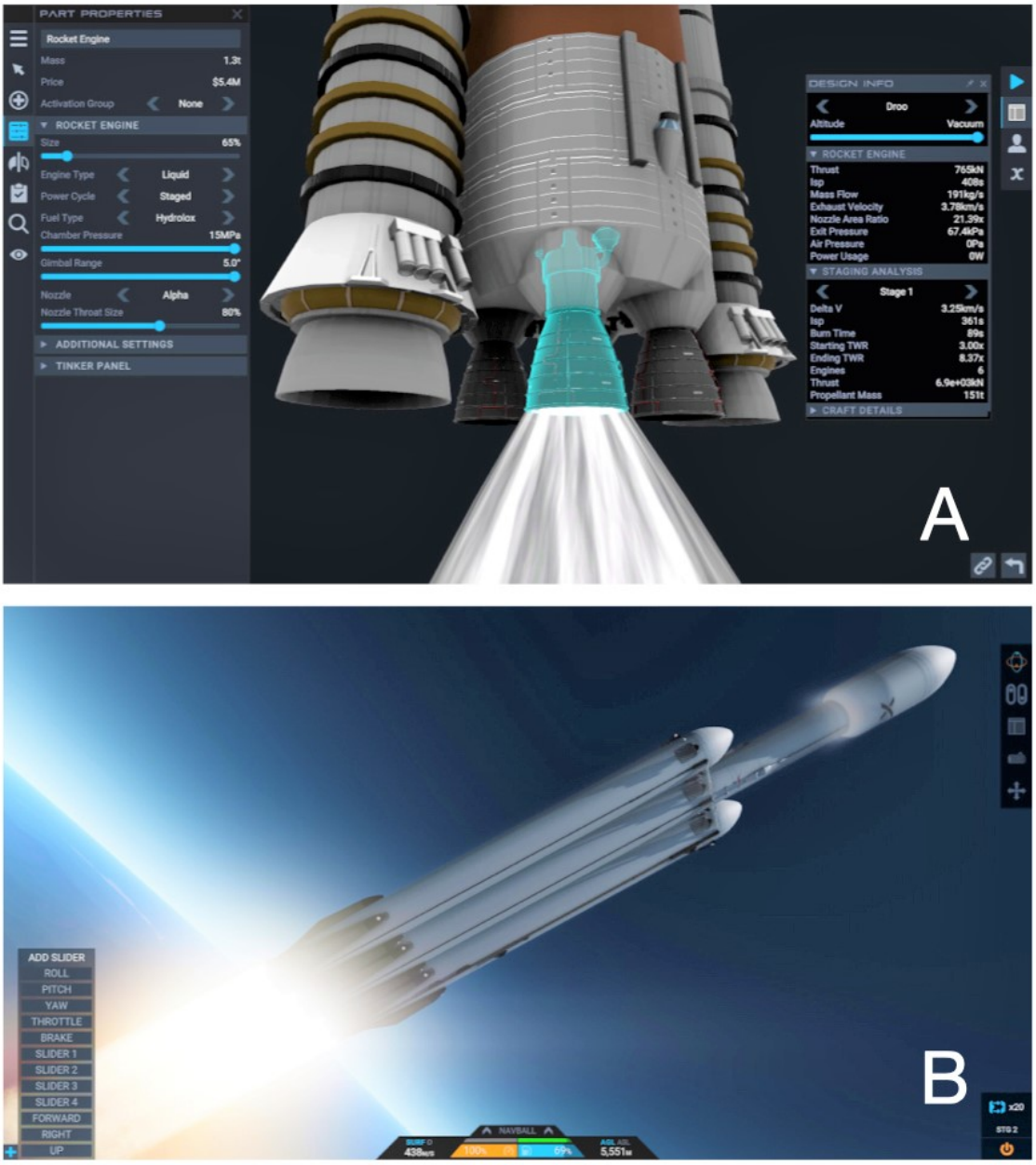

Figure 1-13: *Juno: New Origins [99] is a 3D aerospace sandbox where players can use customizable parts to construct and test rockets, planes, cars, or anything they can imagine in an environment with realistic physics across land, sea, air, and space.*
Image courtesy: Juno: New Origins on Steam.

Figure 1-14: *HoloLens 2 devices assisted technicians during the manufacturing process for the Artemis II heat shield. One of the most critical elements of Orion, it protects the capsule and crew during reentry through Earth's atmosphere.*

Image courtesy of NASA.

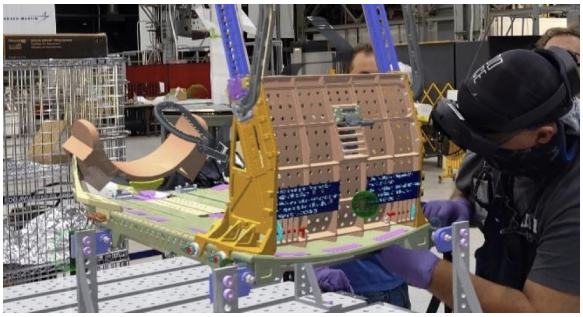

Figure 1-15: *Lockheed Martin technicians at the NASA Michoud Assembly Facility use HoloLens to assemble Orion's crew seats for the Artemis II mission. This image shows the holographic instructions overlaid on the crew seats.*

Image courtesy of Lockheed Martin.

The following sections will explore the multidimensional aspects of this exciting and transformative field, highlighting modern and historical research conducted on blending virtual environments with Mechatronics.

Researchers in this domain are exploring novel applications where robots can interact with humans in Mixed Reality environments. These environments can be either fictional [110] or based on real-world data [111], providing immersive and dynamic experiences for both humans and robots. From remotely controlling robots through VR systems [112] to designing Augmented Reality interfaces intertwines human and robotic activities [113], the affordances provided by blending such systems could impact a long envisioned teleoperated economy, transforming individuals that excelled in interacting within virtual environments and video games into a driving force of action.

An essential aspect of Virtual HRI is the ability to bridge the gap between simulated and real-world interactions. To achieve this, researchers are creating sophisticated VR scenarios where humans can teach robots about objects, tasks, and language while collecting simulated perceptual data. This "Sim2Real" [114] approach allows for the transfer of knowledge gained in virtual settings to real robotic systems, enabling efficient learning and adaptability in diverse environments. Additionally, Virtual Reality laboratories with fully simulated robotic arms, like ArmSym [115], have emerged as testbeds, or sandboxes, for conducting comprehensive experiments on human control of robotic arms in realistic environments. Such environments cater to a range of applications, from upper limb prosthetics to wheelchair-mounted robotic manipulators, significantly advancing assistive robotics research.

For instance, the Kinetic AR framework addresses some of the challenges faced by current robotic systems, aiming to provide an intuitive and deployable solution for path planning, flexible programming, and enhanced visualization of robotic motion. Through interviews with robotics experts, the framework's goals were established, including consistent path planning across robots with different degrees of freedom, improved flexibility in programming to reduce costs and training times, and real-time visualization tools for motion analysis. The framework enables users to perform path planning by anchoring points in the environment using a mobile device, allowing for adjustments in landing position, rotation, speed, and height. It also facilitates motion programming by linking checkpoints to trigger actions in other hardware or software systems. Real-time steering enables robots to follow the motion of a human, and spatial synchronization with the visual tracking system allows for comprehensive motion visualization. The Kinetic AR framework offers a simplified approach to complex spatial hardware programming, addressing the needs of various applications and experts in the field [116].

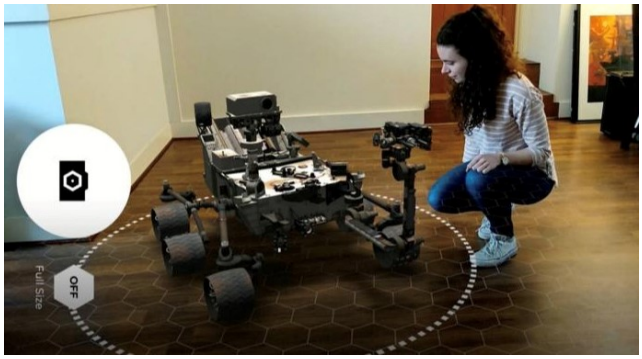

Figure 1-16: *The free Spacecraft AR app uses Google ARCore technology to depict virtual 3-D models of NASA robotic spacecraft, such as the Curiosity Mars rover seen here.*

Image courtesy: JPL / Caltech.

Moreover, Virtual Reality displays have proven to be powerful tools for mediating Human-robot interactions. They offer users unique perceptual benefits [117], especially in situations where perceiving the robot is challenging. By exploring how Virtual Reality can enhance collaborative tasks and improve performance in simulated robotics applications. Furthermore, researchers are delving into the realm of shared virtual experiences, unearthing valuable insights into their potential to enhance Human-robot interactions across multiple agents, particularly in the context of swarm robotics [118]. Both Chapter 3, and 4 delves into the diverse facets of Virtual HRI, addressing novel contributions through a language of interactions and visualisation techniques that has been long present in virtual experiences and video games, yet adapted to physical and virtual space analog environments.

In a related context, noteworthy projects like FlightGoggles [119], (Figure 1-17), involve the adaptation of virtual robotic systems to learn and operate within simulated environments. Rather than directly linking these simulations to the real world in real-time, the primary aim of this research is to harness the potential of the virtual world to design and optimize advanced robotics systems for practical deployment. It centers on providing a versatile development environment, facilitating the design, im-

plementation, testing, and validation of autonomous super-vehicles [119]. The robust tool encompasses exteroceptive sensor simulation deeply integrated into the Unity engine, complemented by vehicle dynamics and inertial sensor simulation capabilities.

Expanding on these concepts, Chapter 5 of this thesis delves further into similar methodologies, utilizing photogrammetry, sensors, and robots to synthesize space analog environments. To enable this interconnection between robots and virtual environments, the Unity game engine, a leading platform in the game development industry [120], has emerged with recent modules to standardize the integration with robotics platforms. This was made possible through the addition of the Unity Robotics Visualizations Package, a toolkit for visualizing and debugging the internal state of robotics simulations, allows Unity to be used as an all-in-one in combination with the Robot Operating System (ROS) [121] as simulation and visualization tool. With these robust features and capabilities, Unity provides an ideal environment for developing realistic, and immersive simulations. Potentially, leveraging Unity's capabilities for scenarios presented in remote exploration, such as space exploration, to allow for the creation of simulations with detailed and interactive 3D environments, responsive mechanics, and advanced physics, all crucial for replicating the conditions and challenges of space operations.

Figure 1-17: *FlightGoggles [119] is a cutting-edge simulator developed at MIT with environments generated from high-quality photogrammetry of indoor environments. In this scenario, the environment was adapted to serve as a testing simulation for nine teams participating in the AlphaPilot program [122], a popular autonomous drone racing challenge.*

Utilizing user interfaces from video games, particularly those from Real-Time Strategy (RTS) genre, provide intuitive mechanisms to navigate virtual environments from a bird's eye view. The nature of these games requires players to control multiple units, strategize, and adapt in real-time collaborative problem solving. Large scale

user studies could reveal if the skills acquired by the video gamer are transferable to operating rovers in space scenarios, raising several questions about video game interfaces and rover operations in providing operators with a broader and more precise control range while facilitating better decision-making during exploration missions.

Figure 1-18: *A 220-pound, 13-foot-long snake-like robot designed to search for life on Saturn's icy moon Enceladus designed by NASA JPL. Named Exobiology Extant Life Surveyor, or EELS, the robot is engineered to be both self-propelled and autonomous. As it undergoes testing and development, EELS aims to navigate diverse planetary and lunar terrains, including undulating sand, icy surfaces, steep cliffs, craters, lava tubes, and narrow spaces, such as crevices or fissures, within glaciers. Image courtesy: NASA JPL.*

1.3.5 Human-Robot Systems in Space Exploration

The realm of space exploration has seen a growing interplay between humans and robots, an interaction that is expected to intensify with future missions. NASA's history of successful missions is testament to the effectiveness of this synergy, with robotic systems in facilitating the exploration of Mars [123]. The first remote robot deployed by NASA in space was the Robotic Arm on the Space Shuttle.

The Space Shuttle program, which began in the early 1980s, included a Remote Manipulator System (RMS) commonly known as the "Canadarm." The Canadarm was a robotic arm developed by the Canadian Space Agency, and it was used to perform various tasks during space shuttle missions, such as capturing and deploying satellites, assisting with spacewalks, and moving cargo in and out of the shuttle's payload bay. The Canadarm was a groundbreaking development in space robotics and played a significant role in space exploration during the Space Shuttle era [124]. Upcoming

missions, including the Artemis missions, plan to extend human presence beyond low Earth orbit, and the achievement of these objectives relies heavily on the effective integration of humans and robotic technology [125].

As the reliance on robotic systems for space exploration grows, so does the diversity of Human-robot interactions. Previously, interaction was largely limited to remote communication between humans on Earth and robots in Space. These telerobotic operations ranged from direct manual control to intermittent, supervisory control. However, recent research has started to explore a wider range of Human-robot arrangements. These include co-located teams, remote teams, one-to-one setups, and groups. The exploration of Human-robot teaming theory and system design, efficient interaction methods, and Human-robot communication is underway and evolving [126]. The criticality of these aspects is emphasized in the unique context of space, where several factors, including the space environment itself, play a significant role in determining the effectiveness of Human-robot interaction.

Despite the promise of Human-robot synergy in space exploration, this interaction is fraught with challenges, primarily due to the unique circumstances of deep space missions. High communication latencies and limited bandwidth between non-collocated humans and robots, operations in reduced or zero gravity environments, and operations on other planetary bodies with the associated issues due to radiation, temperature, illumination, and dust present significant difficulties [127]. Additionally, the growing complexity of space missions has necessitated the development of new paradigms for Human-robot interaction.

As the number of astronauts on missions is limited and their schedules are constrained, ground-control personnel will likely need to remotely supervise and assist hundreds of robots [128]. NASA's concept of Human-robot interaction encompasses not only co-located teams, but also Human-robot teams distributed across both time and space. These Human-robot teams are expected to perform a wide range of tasks, from ISRU based activities to assembly of large structures or systems, reconnaissance, and sample deposition/collection [129]. Such teams will prove crucial in missions that aim to establish a sustainable human presence on celestial bodies like the Moon and Mars.

The future of space exploration is likely to see an increased reliance on teams composed of humans and robots with complementary capabilities. However, this will also necessitate unique methods for humans and robotic systems to interact. Robots may be tasked with duties that are dull, dirty, or dangerous, thereby allowing human crew members to perform more complex tasks or those requiring real-time modifications due to contingencies.

The coordination of these distributed teams of humans and robots is complex, and the development of tools and techniques to facilitate effective interaction is critical. This includes enabling clear communication about capabilities, intent, state, and accomplishments, which will increase the likelihood of mission success and improve the

capability of Human-robot team members to coordinate and solve problems. Given the diversity of Human-robot teaming scenarios in space, it is essential that a variety of HRI tools and techniques be developed to cater to the varying types, frequency, and criticality of interactions [130].

NASA's Vision for Space Exploration emphasizes the collaboration between humans and robots as partners, leveraging the unique capabilities of each entity. Close collaboration between humans and robots will be essential for mission tasks both in-space and on planetary surfaces. However, due to cost constraints and the need to minimize risks, astronaut teams will be small, highlighting the significance of effective Human-robot interaction (HRI) for future missions. The objective of the "Peer-to-Peer Human-robot Interaction" (P2P-HRI) project is to advance the state-of-the-art in HRI to facilitate sustained and affordable space exploration [131]. The project focuses on developing HRI techniques that enable humans and robots to work as partners in various team configurations, including side-by-side, line-of-sight remote, and far remote interactions.

The approach includes a novel interaction framework called the "Human-robot Interaction Operating System" (HRI OS), computational cognitive architectures to model human behavior and enhance Human-robot understanding, and a series of evaluations using research robots, analog environments, and exploration-relevant tasks. The project primarily focuses on supporting essential operational tasks, such as construction, assembly, inspection, and resource collection and transport. The goal is to enable efficient teamwork between humans and robots by specifying high-level operations and utilizing interaction to address issues during task execution. The approach aims to create a bidirectional communication model, where robots can ask questions and seek assistance from humans when necessary, fostering coordination and quick issue resolution. Challenges in the project include enabling autonomous robot task performance while allowing robots to seek human expertise when needed and ensuring robots understand task-oriented commands as a human teammates [132].

In alignment with the goals of enhanced HRI and the growing importance of integrating data analysis capabilities into robotic interfaces, there is a clear need for increased collaboration between robotics and data visualization experts. This collaboration would leverage the unique strengths of both fields, combining HRI's focus on effective human control and supervision of robots while the visualizations emphasize designing interfaces for data exploration.

By bridging the gap between HRI and data visualization, it becomes possible to develop interfaces that not only facilitate Human-robot interaction but also consider how users engage with the data provided by robots. This collaborative effort holds significant potential to address challenges related to dynamic, uncertain, and spatiotemporal data, which are particularly relevant in the context of robotics and space exploration. It is through successful collaboration and co-innovation that the field of HRI can benefit from the expertise of data visualization experts and vice versa,

leading to advancements in both fields and enabling more efficient and meaningful Human-robot collaboration in various team configurations, including in-space missions and planetary surface exploration [133].

Early notable advancements have been made in the field of robotic systems, particularly with the Mars Exploration Rovers (MER) missions involving "Spirit" and "Opportunity." These rovers have successfully demonstrated the effectiveness of concepts like visual odometry and autonomous path selection using passive stereo vision as their primary sensor suite for terrain assessment. Their utilization of stereo imagery for this has significantly contributed to our understanding of Mars' surface. This alternative sensing modality holds promise for enhancing path planning and navigation strategies in planetary exploration [134]. The autonomous navigation of rovers on the surface of Mars, the Moon, or other celestial bodies, holds great potential for significantly enhancing daily traverses, particularly in unexplored regions away from the lander.

The CNES (Centre National d'Études Spatiales) has developed a robust autonomous navigation process that utilizes stereo cameras for perception, enabling the creation of an environment model and the generation of trajectories. This approach incorporates multiple perception merging techniques while propagating locomotion and localization errors. The abstract discusses the algorithms developed for Mars exploration programs, the vision hardware employed, validation tools utilized, experimental platforms employed, and the evaluation of results. Additionally, the abstract addresses the portability of the system and the assessment of computing resources for potential implementation on a Mars rover. The findings highlight the energy and computing efficiency of the implemented autonomy, indicating minimal resource consumption while maximizing rover capabilities. As a result, the autonomous navigation system enables significantly longer daily traverses compared to strategies solely based on ground-planned approaches [135].

During the Apollo program, astronauts faced specific challenges to navigation in the lunar environment during their Extravehicular Activities (EVA). These difficulties were exemplified in the second EVA of the Apollo 14 mission, where Commander Alan Shepard and Lunar Module Pilot Ed Mitchell had to rely on a paper map developed from lunar surface photographs to navigate to Cone Crater, approximately 1.5 km away. However, they encountered difficulty in identifying specific craters visually, leading to poor situational awareness and confusion about their exact position. Distortions from the helmet visor and the lack of atmosphere on the lunar surface further complicated their ability to judge distances and travel times. In the end, the astronauts turned back, unknowingly just 40 meters from the crater's rim.

The Apollo 14 mission's challenges highlighted the importance of addressing navigation difficulties, which will become even more significant during future missions to the Martian surface, with increased frequency, length, and duration of traverses. Overcoming these obstacles is crucial for the success of surface explorations [136].

Eventually, the moon may be equipped with radio location systems, [137], but until they are widely available and ubiquitous, visual navigation remains crucial for near-term missions. To tackle these challenges and improve planetary surface explorations, the Surface Exploration Traverse Analysis and Navigation Tool (SEXTANT), serves as a decision support aid for planning and optimizing paths for both suited astronauts on foot and transportation rovers. It presents crucial information to assist users in making decisions rather than making the decisions itself. SEXTANT features an interface with a 3D terrain elevation map where users can specify Activity Points, which represent points of interest. Terrain obstacles, defined by a user-inputted maximum slope, are taken into account when determining the most-efficient path between Activity Points based on traverse distance, time, or explorer energy consumption [136].

Most recently, robotic systems in the field of planetary exploration face a myriad of challenges, including constraints related to weight, size, and the harsh conditions of extraterrestrial environments. These limitations significantly impact the selection of suitable sensors and actuators for space missions. Moreover, the vast distances between planets introduce substantial communication delays, highlighting the need for high levels of autonomy to ensure efficient robot operation. Addressing these challenges, the Lightweight Rover Unit (LRU), designed by the European Space Agency (ESA) emerges as a compact and agile rover prototype specifically tailored for planetary exploration. The LRU integrates an individually steered wheel locomotion system, enabling exceptional maneuverability in rough terrains. Its sensor suite predominantly features stereo cameras, rendering it well-suited for space missions. Complementing these hardware capabilities, a comprehensive range of software components has been developed by the research team to enhance the LRU's performance. These components encompass self-localization in GPS-denied environments, autonomous exploration, mapping, computer vision, planning, and control modules. Notably, autonomous localization, object pickup, and assembly tasks employing a manipulator are among the LRU's capabilities.

Additionally, the LRU incorporates high-level mission control components to facilitate autonomous behavior and remote monitoring of system status over delayed communication links. The LRU's impressive autonomous capabilities were demonstrated during the SpaceBotCamp challenge—an esteemed national robotics contest focused on autonomous planetary exploration. During the challenge, the LRU autonomously navigated through an unfamiliar Moon-like rough terrain, successfully located and collected two objects, and executed their assembly after transportation to a third object. Remarkably, the LRU accomplished these tasks flawlessly on its first attempt, exhibiting remarkable efficiency while operating in a fully autonomous mode [138].

Moreover, recent research has demonstrated innovative approaches to automate rover operations using machine learning techniques. One such study focused on enhancing the surface navigation software, Enhanced AutoNav (ENav), employed by NASA's Perseverance rover Reference to the study. The researchers addressed the computational challenges associated with the Approximate Clearance Evaluation (ACE)

algorithm, a crucial component for ensuring rover safety. To optimize path selection, two heuristics were introduced. The first heuristic integrated Sobel operators and convolution to incorporate the cost of traversing high-gradient terrain. The second heuristic utilized a machine learning (ML) model trained with physics simulations to predict areas deemed untraversable by ACE. Through extensive simulations and Monte Carlo trials, the researchers demonstrated that the integration of these heuristics significantly reduced ACE evaluations and computation time per planning cycle, while improving path efficiency and maintaining or even enhancing the success rates of rover traverses. This research showcases the potential of machine learning in enhancing the automation and efficiency of rover missions, particularly in addressing critical safety considerations in real-time rover navigation [139].

The use of telepresence has also been explored in various analog environments to enhance the control of remote vehicles and facilitate scientific exploration. One notable experiment focused on the utilization of a Telepresence-controlled Remotely Operated underwater Vehicle (TROV) in the marine environment beneath the sea ice near McMurdo Station, Antarctica [140]. The mission aimed to operate the TROV through telepresence and VR technology, both locally and remotely. Local control involved using a control box with joysticks and switches, while the operator viewed stereo video camera images on a display monitor. Remote control, facilitated by a satellite communications link, utilized a similar display monitor or a head-mounted display. Remote operators could also access a computer-generated representation of the underwater terrain based on the vehicle's sensors. Through the satellite link, stereo video from the TROV was transmitted to NASA Ames, and bi-directional Internet communication enabled remote control of the vehicle.

The experiments conducted in Antarctica demonstrated the feasibility of surface rovers with real-time telepresence control, potentially expanding the range of human exploration from a base on the Moon or Mars [141]. In another study focused on telepresence in the exploration of Mars, the NASA-funded Biologic Analog Science Associated with Lava Terrains (BASALT) program [142], aimed at advancing Mars mission operations and capabilities, conducted its third field test in November 2017. The test involved ten simulated extravehicular activities (EVAs) in the Kilauea Caldera and Kilauea Iki regions of Hawaii. Real-time geo-biochemical science objectives were accomplished by a team of two extravehicular crewmembers, while two intra-vehicular crew members provided support. Remote scientists and operators in the Mission Support Center contributed scientific expertise through Mars relevant communication latencies. The field test incorporated new capabilities such as high-resolution panoramic imagery, mobile automated light detection and ranging data, immersive mixed-reality terrain models, and augmented-reality field systems [143].

These studies highlight the significance of telepresence and teleoperations in space exploration, showcasing their potential to enable remote control and scientific investigation in challenging and distant environments. The utilization of telepresence technology, coupled with VR/AR interfaces and the control of remotely operated

vehicles, presents promising opportunities for human exploration and enhances our comprehension of extraterrestrial environments [144]. However, it is important to acknowledge that communication latency in space exploration poses real challenges for such systems to achieve real-time responsiveness. In many scenarios, delays of milliseconds can render certain video games unplayable. Nevertheless, real-time VR navigation and robotics in space exploration can still be instrumental in aiding astronauts within orbit to control these entities.

With advancements in sensing, actuation, and machine learning, autonomy has been a traditional approach for robotics systems in space. This approach allows robots to operate with relative independence, reducing the need for constant communication with Earth. However, for more intricate tasks and in environments where human expertise is crucial, telepresence and teleoperation technologies remain vital tools. They offer a means for humans on Earth to actively participate in space exploration, remotely guiding robots and making critical decisions in real-time, despite the communication challenges posed by vast distances. As space exploration continues to evolve, the combination of autonomy and telepresence will likely play a complementary role in maximizing the potential of robotic systems beyond our planet's boundaries.

Figure 1-19: *Project NeBula (Networked Belief-aware Perceptual Autonomy. NeBula enables autonomous robots to navigate uncertain terrain, overcome challenges, and gather vital data. This innovative technology has been successfully demonstrated in terrestrial analog missions, including cave exploration on Earth, paving the way for future Mars exploration [145, 146].*

Image courtesy: NASA JPL.

Most recently, the Spot Nebula project, (Figure 1-19), developed by NASA's Jet Propulsion Laboratory (JPL) and Caltech in collaboration with Boston Dynamics, represents a significant advancement in autonomous cave exploration and mapping in analog space environments. The four-legged [147] explorer prototype combines the intelligence and autonomy provided by JPL. The development was motivated by the Subterranean Challenge, a contest sponsored by the Defense Advanced Research Projects Agency (DARPA). One of the primary objectives of the Spot Nebula project is to demonstrate the autonomy required for navigating extreme environments without human guidance or access to GPS. legged robots have the potential to better explore complex terrain that is inaccessible to standard wheeled rovers.

The autonomous cave exploration and mapping capabilities of this electronic canine have direct implications for future space missions. Identifying suitable shelters for astronauts on the Moon or Mars could be achieved by mapping caves. Additionally, caves offer a better chance of preserving microbial life, if it exists, as they provide protection from cosmic radiation and extreme temperature fluctuations. Planetary moons like Europa, Enceladus, and Titan, which may have icy seas, could also harbor microbial life beneath their surfaces [146]. NASA recognizes the potential significance of exploring caves in the search for extraterrestrial life [148]. The Spot Nebula project aligns with NASA's broader initiative known as BRAILLE, which focuses on exploring Mars-like caves that exist on Earth to refine key technologies for future missions [149]. These caves closely resemble the challenging cave environments on other planets, including Mars. By undertaking the first-ever fully autonomous robotic exploration of such caves, the project has achieved a significant milestone.

These caves, extending several hundred meters, pose communication limitations with the surface, simulating conditions that scientists may encounter on Mars. During the exploration of the caves, the robots operate without any prior information about the environment. Meanwhile, a team of researchers stationed outside the cave conducts actions that mirror what scientists on Earth would execute during a real Martian mission. The integration of Boston Dynamics' Spot robot with NeBula has proven highly effective, demonstrating Spot's capabilities in traversing rough and extreme terrains [150].

Chapter 2

Thesis Overview

2.1 Prologue

As emphasized in the previous chapter, space exploration enters a new era with increased collaboration between humans and robots in physical, virtual, and analog space environments. Robotic systems have played a pivotal role in successful NASA missions, notably in Mars exploration. The future Artemis program aims to extend human presence beyond low Earth orbit, making Human-robot integration even more vital. This thesis explores the ever evolving paradigm of space exploration, blending multiple fields like computer graphics, virtual environment synthesis, simulation development, three-dimensional reconstruction, photogrammetry and photorealism, Human-computer interaction, Human-robot interaction, Human-robot operations, systems engineering, telepresence, communication and networking, space systems, and Artificial Intelligence (AI). Three distinct robotics systems with varying complexities were investigated: commercial, industrial, and space robotics. Throughout this research these were explained through the integration, design and development of custom payloads and modules.

These efforts were conducted in collaboration with fellow students and researchers, including Cody Paige, Ferrous Ward, Jessica Todd, and Alexandra Forsey-Smerek from the MIT Resource Exploration and Science of our Cosmic Environment (MIT RESOURCE). This team is funded by NASA's Solar System Exploration Research Virtual Institute (SSERVI), under the supervision of Prof. Dava J. Newman, the Apollo Program Professor of Astronautics, and director of the MIT Media Lab.

I am deeply honored to have been part of both the MIT RESOURCE team and the Responsive Environments group, with my advisor, Prof. Joseph A. Paradiso, the Alexander W. Dreyfoos (1954) Professor in Media Arts and Sciences, and Academic Head of the program in Media Arts and Sciences at the MIT Media Lab.

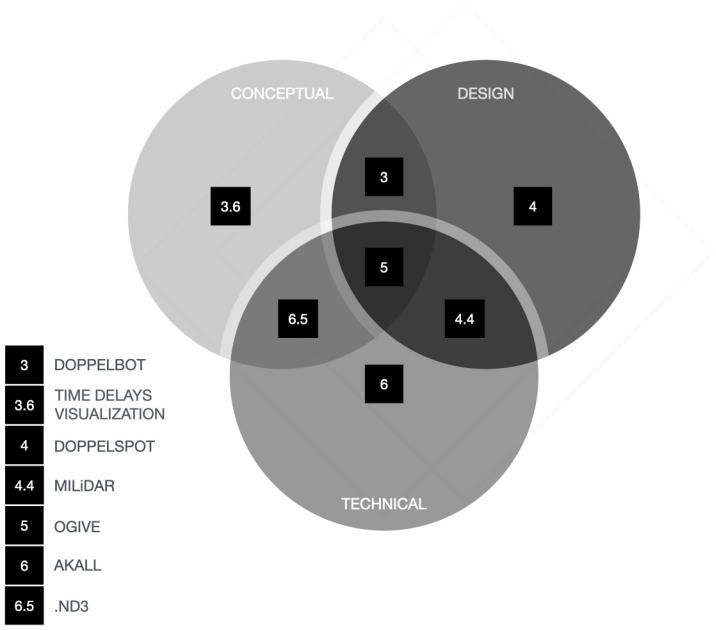

Figure 2-1: *Intellectualized framework represented in a Venn Diagram to illustrate this dissertation's chapters. Chapter 3 -* ***Design & Conceptual*** *introduces the foundational concept of the Doppelbot. Building upon this concept, Chapter 4 -* ***Design & Technical*** *presents the introduction of Doppelspot in the Mobile Immersive LiDAR (MILiDAR), (Section 4.4), of Chapter 4. This section explores the integration of robotic telepresence with 360 video & LiDAR technology. In Chapter 5 -* ***Design, Technical & Conceptual****, the focus shifts to the synthesis of virtual analog environments using real-world data through Human-robot collaboration. Chapter 6 -* ***Technical*** *delves into the software implementation of a commercial camera into a lunar rover module, facilitating lightweight 3D reconstruction for in-situ planetary exploration. Additionally, (Section 6.4.3), of Chapter 6 elaborates on the ".ND3" file format, providing essential insights into this specific data format.*

2.2 Research Roadmap

The research conducted in this thesis is derived from a legacy of prior work developed by the Responsive Environments group, and brought to space operations with the MIT RESOURCE team, (Figure 2-2), The contributions in this thesis are organized into four chapters, (Figure 2-1), each addressing a distinct aspect of Human-robot synergy in analog space environments.

Chapter 3 explores innovative concepts and applications of "Doppelbots," revealing some emergent abilities and interactions resulting from the fusion of virtual and physical space analog environments. By introducing versatile mechanics for navigating and exploring distant environments, this research addresses a wide range of aspects, including training remote operators and testing new technologies in simulations that are synchronized with real-world events, with a futuristic vision to studying the syn-

ergistic effects of different environments on both humans and machines. The novel visualizations and interfaces employed in this research draw inspiration from the perpetual interplay between video game mechanics and Human-robot interactions.

Chapter 4, titled "Doppelspot," represents an advancement in the concept of Doppelbots, utilizing a sophisticated robotic system designed and developed by Boston Dynamics. Equipped with cutting-edge payloads, including a LiDAR (Light Detection and Ranging) unit, an advanced camera capable of capturing 360° video, and stereo audio, this chapter introduces an extraordinary telepresence encounter. Spot, the quadruped robot, serves as the embodiment of this transformative experience, enabling the tele-explorer to navigate rugged and challenging outdoor terrains. The integration of these sensory inputs is seamlessly streamed into a Virtual Reality (VR) headset, resulting in an immersive encounter with a space analog environment.

Chapter 5 builds upon the foundational knowledge presented earlier and delves into the realms of 3D reconstruction and the synthesis of virtual space analog environments sourced from real-world data. This chapter takes the reader on a compelling journey that led a team of graduate students from MIT to face a series of challenges and test their abilities in capturing and collecting meaningful data from a hostile and remote environment. The expedition began with the team embarking on their first analog mission to Svalbard, Norway, organized by the Space Exploration Initiative (SEI) of the Media Lab. The operation was divided into two groups: one group dedicated countless hours to leveraging robotics systems, sensor nodes, and various cameras, while the second group honed their skills in scientific data analysis, deploying and operating an array of scientific instruments, both terrestrial and aerial, throughout the course of the mission. Advanced 3D reconstruction techniques, including photogrammetry, were employed to capture and process high-resolution textures and meshes, resulting in immersive virtual environments closely resembling the selected sites in Svalbard. To validate the effectiveness of these environments for virtual geological studies and research, virtual explorers underwent an in-depth user study in several synthesized space analog environments. In summary, this chapter represents a significant step forward in understanding the potential of 3D reconstruction and virtual analog environments as a whole, demonstrating their value in documenting scientific research and expeditions.

In Chapter 6, we delve into the cutting-edge development of "AKALL" (Azure Kinect à la Luna), the software that drives a module that showcases the adaptation of a commercial RGBD camera, the Microsoft Azure Kinect, for use as a custom payload mounted on a lunar rover. NASA employed rigorous hardening techniques to ensure the camera's resilience in the harsh vacuum of space through sever modifications to withstand the unique challenges posed by lunar environments, including extreme temperature fluctuations, radiation exposure, and mechanical stresses. Furthermore, to optimize its functionality and meet the strict weight limitations of space missions, this modified Azure Kinect underwent a careful engineering process that entailed selective feature removal. Non-essential components were deliberately excluded to

reduce the camera's overall weight while retaining its essential capabilities for lunar exploration. Additionally, the camera was thoroughly shielded to protect it from the damaging effects of cosmic radiation and micrometeoroid impacts, safeguarding its sensitive electronics and ensuring its reliability during the span of the mission. This entire process involved a deep investigation that spanned across various entities, and is briefly referenced and documented in this dissertation. While NASA primarily concentrated on the hardware aspect of the project, the MIT RESOURCE team played a significant role in software development, testing, and refining the Concepts of Operations (ConOps) throughout the research's evolution.

The Azure Kinect DK depth camera implements the Amplitude Modulated Continuous Wave (AMCW) Time-of-Flight (ToF) principle. The Azure Kinect camera emits modulated illumination within the near-IR (NIR) spectrum, illuminating the scene. It subsequently captures an indirect measurement of the round-trip time taken by the light to travel from the camera to the scene and back [151]. These recorded measurements are then processed to create a depth map (depth.b16g), which contains Z-coordinate values for each pixel in the image, measured in millimeters. Additionally, alongside the depth map, a so called "clean IR reading" (ir.b16g) is recorded. The pixel values in this reading correspond to the quantity of light reflected back from the scene, providing valuable information about the scene's characteristics. The second sensor included in the Azure Kinect DK allows for capturing colored images with native support for JPEG compression.

The default software packages provided with the Azure Kinect DK, specifically the k4a-tools, capture and store data in the (.mkv) format. This format is known for its capabilities in storing video, audio, and subtitle data in a single file. In contrast, the optimization pursued in this research necessitated the development of a distinct approach. This approach was due to the single-capture nature of the ConOps and expensive data rates in space exploration that required further compression of the generated data. One of significant contributions highlighted in this research was done on optimizing this process, through AKALL, as well as through the implementation of a novel file format called (**.nd3**). "ND" stands for "Natural Depth": as it entails that the (".nd3") file extension is tailored for processing, capturing and storing "3D images", using 3 main components, a color image for example (**color.jpeg**) depth image (**depth.b16g**) and calibration data (**calibration.json**). Optionally, the (**ir.b16g**) image could be utilized to further focus on preserving "natural depth" perception, through advanced light mapping techniques to achieve more immersive and realistic 3D textures. As part of the research, integration sessions were conducted with engineers from Lunar Outpost, who worked to incorporate AKALL into their Mobile Autonomous Prospecting Platform (MAPP) lunar rover. To evaluate AKALL's effectiveness and capabilities, comprehensive testing was performed at the NASA Ames Research Center within the SSERVI simulated Lunar Testbeds. For reader orientation, a list of relevant topics explored in each Chapter is included as a short preface.

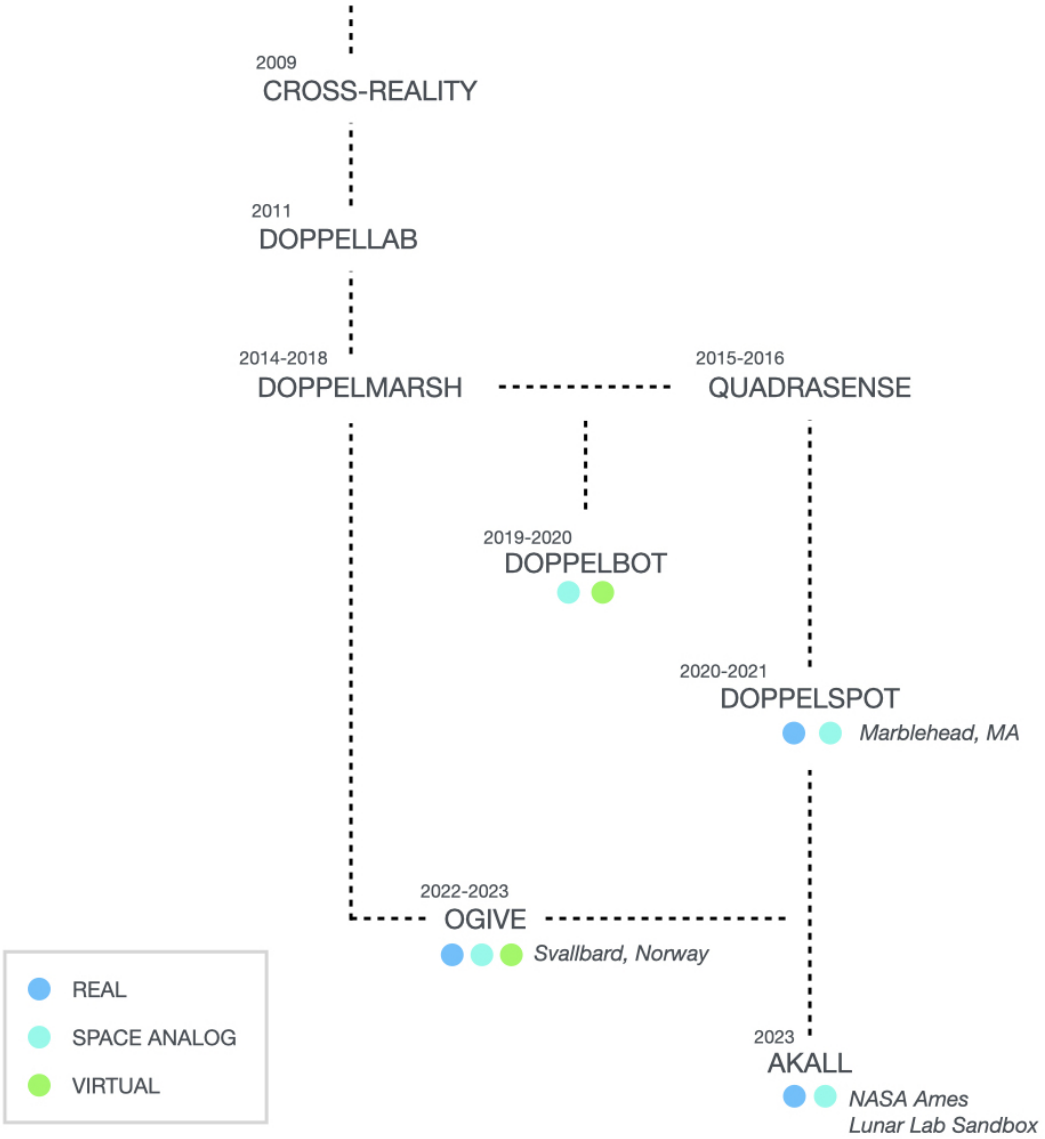

Figure 2-2: *A chronological progression of projects derived from predecessor initiatives involved in my research including Doppelmarsh, (Section 1.3.2), and Quadrasense [152]. Each project builds upon the previous one, advancing technology and concepts applied for scientific exploration and Human-robot operations.*

2.3 Selected Research Publications

This dissertation builds upon a rich body of work that has laid the foundation for the research presented here. The following list of co-authored publications includes both essential references that have informed the study and original contributions that have documented the research process. Together, these works represent the intellectual context within which this dissertation was conceived and executed.

- D. D. Haddad, C. A. Paige, B. Brokaw, F. S. Ward, J. A. Paradiso, J. Heldmann, and D. J. Newman, "**Azure Kinect à la Luna (AKALL): Leveraging Low-Cost RGB and Depth-Camera in Lunar Exploration**," in Proceedings of IEEE Aeroastro, 2024.

- C. A. Paige, D. D. Haddad, F. S. Ward, A. Cook, V. Jhac, A. Deutsch, A. Shimadac, A. Colaprete, J. Heldmann, and D. Newman, "**Development and Testing of the Concept of Operations for a Low-Cost RGB and Depth-Camera for a Lunar South Pole Mission**," Nature Microgravity, Waiting review, 2024.

- D. D. Haddad, S. Unterhauser, C. A. Paige, B. Brokaw, F. S. Ward, J. A. Paradiso, J. Heldmann, and D. J. Newman, "**Leveraging Docker Containers for Azure Kinect Integration in Space Robotics: An Overview of the Azure Kinect à la Luna (AKALL) Framework**," in Conference on Human-Computer Interaction for Space Exploration (SpaceCHI 3.0), June 22-23, 2023.

- C. A. Paige, D. D. Haddad, F. S. Ward, T. J. Piercy, J. E. Todd, J. A. Paradiso, and D. J. Newman, "**Operational Geology In a Virtual Environment (OGIVE) Novel Approaches to Virtualizing Geological Expeditions for Planetary Exploration**," in Proceedings of Conference on Human-Computer Interaction for Space Exploration (SpaceCHI 3.0), 2023.

- F. S. Ward, C. A. Paige, D. D. Haddad, J. E. Todd, J. Heldmann, D. Lim, A. Ekblaw, and D. J. Newman, "**Multi-Sensor 3D Data Visualization in Virtual Reality for Planetary Science and Mission Operations**," in The International Conference on Environmental Systems ICES, Under revision, 2023.

- C. A. Paige, D. D. Haddad, F. S. Ward, J. Todd, A. Ekblaw, and D. Newman, "**Data collection in Svalbard, Norway to test the use of virtual reality for Lunar and planetary surface**," in The International Conference on Environmental Systems ICES, 2023.

- A. Ekblaw, J. Cherston, F. Z. Liu, I. Wicaksono, D. D. Haddad, V. Sumini, and J. A. Paradiso, "**From UbiComp to Universe - Moving Pervasive Computing Research Into Space Applications**," IEEE Pervasive Computing, 2023.

- C. A. Paige, F. S. Ward, D. D. Haddad, J. MacNeil, P. McGaffigan, A. Ekblaw, and D. Newman, "**MIT Zero-G Outreach Initiative: using experiment design and virtual reality to inspire the next generation of space scientists and engineers**," Acta Astronautica, 2023.

- A. Forsey-Smerek, C. A. Paige, F. S. Ward, D. D. Haddad, L. M. Sanneman, J. Todd, J. Heldmann, D. Lim, and D. Newman, "**Assessment of Depth Data Acquisition Methods for Virtual Reality Mission Operations Support Tools**," in 2022 IEEE Aerospace Conference (AERO), pp. 1-14, IEEE, 2022.

- C. A. Paige, A. Forsey-Smerek, D. D. Haddad, F. S. Ward, T. Piercy, J. Heldmann, D. Lim, A. Colaprete, A. Cook, and D. Newman, "**A virtual reality platform for lunar rover missions to reduce decision-making time and improve situational awareness**," ASCEND, p. 4203, 2021.

- C. A. Paige, F. S. Ward, D. D. Haddad, A. Forsey-Smerek, L. M. Sanneman, J. Todd, A. Colaprete, D. S. S. Lim, J. Heldmann, and D. Newman, "**Towards the Development of 3D Lunar Surface Depth-Data Collection for Geology in Virtual Reality**," AGU Fall Meeting, 2021.

- C. A. Paige, D. Newman, D. D. Haddad, F. Ward, and T. Piercy, "**Lunar Instrument Data Integration into the Virtual Reality Mission Simulation System for Decision Making and Situational Awareness**," 50th International Conference on Environmental Systems (ICES-2021-227), 12-15 July 2021.

- D. D. Haddad, "**Resynthesizing Reality: Driving Vivid Virtual Environments from Sensor Networks**," S.M. Thesis, Massachusetts Institute of Technology, 2018.

- B. Mayton, G. Dublon, S. Russell, E. F. Lynch, D. D. Haddad, V. Ramasubramanian, C. Duhart, G. Davenport, and J. A. Paradiso, "**The Networked Sensory Landscape: Capturing and Experiencing Ecological Change Across Scales**," Presence, 26(2), 182–209, MIT Press, 2017.

Chapter 3

Doppelbots

Tunneling Through Realities - Applications in Human-Robot Operations Adapting Mixed Environments into Space Analogs.

3.1 Introduction

3.1.1 A Brief Overview of this Chapter

The concept of the "Doppelbot" intersects with the digital twin paradigm, (as introduced in Section 1.3.1), by sharing physical properties such as the geometry, material, and behavior of a given object, in this case, robotic systems. However, the Doppelbot has distinct core constraints that set it apart from typical digital twin applications. It is permanently attached to a virtual space called the "Doppelspace," where the Doppelbot and its environment are considered as a whole [153]. These environments can be generated from real-world data, as demonstrated in both Doppelmarsh [85] and Doppellab [84] by the Responsive Environments group, or they can be fictional environments.

The bidirectional communication between these virtual and physical worlds was initially established through Cross-Reality (XR), (as described in Section 1.3.2), which involved tunneling real-world information from sensor networks into rich virtual environments and introducing various interaction modalities to allow users to influence physical environments through actuators. Similarly, users can control a Doppelbot from within the virtual environment or switch it to autonomous behavior, essentially turning it into a Non-Player Character (NPC). By setting predefined autonomy, the Doppelbot performs a given activity until interrupted or taken over by the user.

Doppelbots offer a versatile tool with a wide range of potential applications. One prominent application involves training remote operators, akin to flight simulators, but with the added advantage of enabling real-world actions in both simulated and physical environments. This unique capability allows astronauts to prepare for space exploration missions through immersive experiences that bridge the gap between virtual and real-world scenarios. By interacting directly with physical environments,

such as space analog environments, astronauts can familiarize themselves with the challenges they may encounter during teleoperation missions. Moreover, Doppelbots serve as a valuable platform for testing and validating new technologies in a controlled and simulated environment, ensuring their reliability and functionality before actual deployment. Additionally, concepts like Doppelbots and digital twins facilitate the study of the synergistic effects of different environments on both humans and machines.

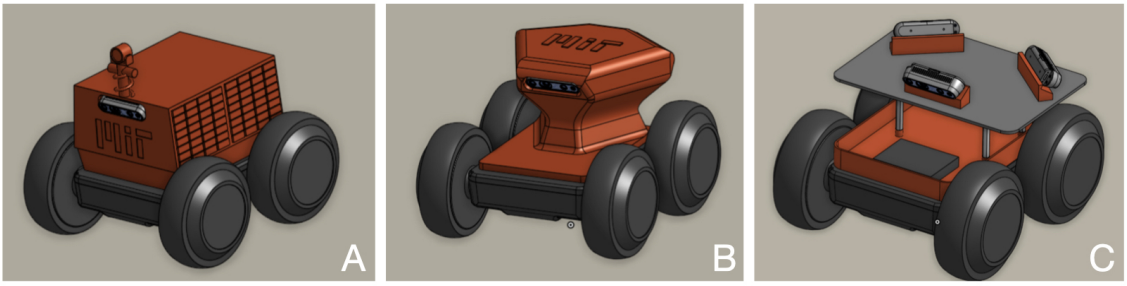

Figure 3-1: *Concept Designs for Rover Mini Custom Payload: These images showcase three design concepts developed in collaboration with Rover Robotics. Image A draws inspiration from NASA's VIPER rover, designed for lunar exploration. Image B presents a conceptual design featuring one Intel RealSense camera. Ultimately, the selected design, as depicted in image C, combines key elements for optimal functionality. Image courtesy: Rover Robotics.*

3.1.2 Rover Mini By Rover Robotics

The concept introduced in this chapter comprises two different implementations. The first one, introduced in this chapter, is a 4-wheel rover, (Figure 3-1), named the Rover Mini designed by Rover Robotics[1] and adapted in collaboration with the MIT RESOURCE team. Similar efforts by the team were conducted with Boston Dynamics[2] on their quadruped robots series known as Spot, (as detailed in Chapter 4).

The Rover Mini is a compact and versatile rover specifically designed to operate on even surfaces, making it suitable for a wide range of applications. Its small size and maneuverability allow it to navigate various flat terrains effectively. To augment the Rover Mini's capabilities, Rover Robotics incorporated a custom modular payload system. This design permits the integration of different components and sensors based on specific requirements. In the case of this Doppelbot, a unique payload configuration has been developed, with several modifications (Appendix C.2), on the core functionalities of the Rover Mini, adapting 3x Intel RealSense modules for precise navigation and spatial localization, or a Microsoft Azure Kinect camera for environment scanning and 3D reconstruction.

[1] roverrobotics.com
[2] bostondynamics.com

A list of stock and custom features has been implemented on the Rover Mini, adapting it for the Doppelbot projects, including:

- Rover Mini Robot Operating System (ROS) [121] and Simultaneous Localization and Mapping (SLAM) [154] method through the SLAMCORE[3] third party framework: This integration enables a communication and coordination between the rover's hardware, such as the Intel RealSense cameras, and software components. It allows for efficient data processing, mapping, and localization capabilities.

- Slam Core Navigation: The Rover Mini leverages Slam Core navigation, enabling the rover to autonomously navigate and map its surroundings. This feature is essential for accurate localization and path planning, enhancing the rover's overall autonomy and efficiency.

- ROS Integration on Nvidia Xavier Board: The Rover Mini leverages the powerful Nvidia Xavier board, a powerful single-board computer (SBC) specifically designed for artificial intelligence (AI) and robotics application. Combined with ROS to provide robust computing capabilities for advanced navigation and control algorithms in a UNIX environment.

- ROS Navigation via WiFi: The Rover Mini supports ROS navigation through WiFi connectivity. This feature allows for wireless communication and control, providing flexibility and ease of use during rover operations.

- Ubuntu 18.04 Integration with ROS Melodic: The Rover Mini is integrated with Ubuntu 18.04 and ROS Melodic, proving to be a stable and reliable platform rovers operations and control.

- Custom Payload System: The Rover Mini is designed for easy assembly, and modularity, allowing users to quickly set up and configure the rover for operation featured in swapping 3D printed payloads modules.

- Intel RealSense Cameras: The Rover Mini incorporates several Intel RealSense cameras, which are tested to operate on the Nvidia Xavier running Ubuntu 18.04. The Intel RealSense[4] D435i stereo and RGB integrated depth camera enables depth sensing and perception capabilities, contributing to accurate mapping and obstacle avoidance, among other things.

- Server Integration with Unity 3D via Node.js: The Rover Mini's server is integrated with Unity 3D, a popular game engine, through Node.js, a server side scripting language. This integration facilitates the visualization and simulation of the rover's movements and interactions within a virtual environment.

- PS 4 Bluetooth controller: Allowing the control of the Rover Mini, also serving as a debugging tool.

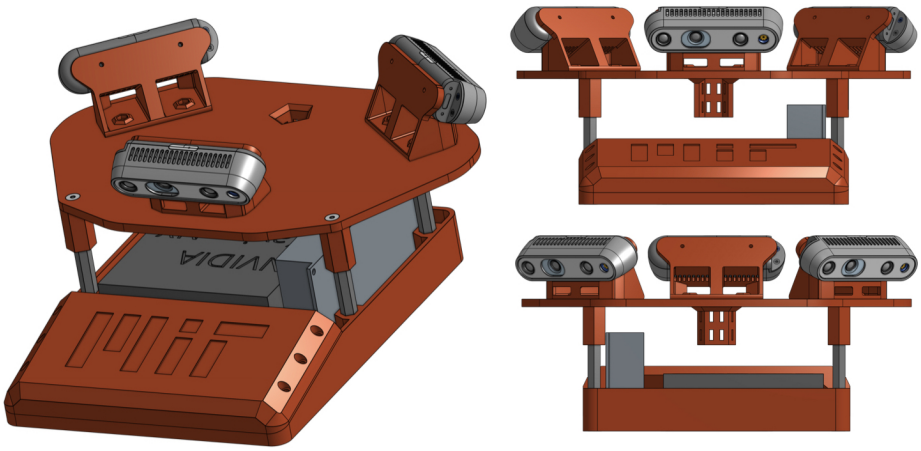

Figure 3-2: *Selected Rover Mini 3D Printed Payload: This image highlights the 3D printed payload for the Rover Mini, equipped with three Intel RealSense D435 cameras. This advanced configuration empowers the Rover Mini with enhanced perception and computing capabilities for autonomous navigation and exploration. Image courtesy: Rover Robotics.*

The combination of these features makes the modified version of the Rover Mini a compact and powerful robotic system, capable of autonomous navigation, mapping, and interaction within both physical and virtual environments, (Figure 3-2). Its integration with ROS, SLAMCORE, and various software and hardware components enables advanced functionalities and a smooth user experience.

3.2 Implementation and Design

The virtual robot model provided by Rover Robotics was adapted to integrate into a virtual lunar environment constructed using the Unity game engine. The recreation of the Rover Mini also includes the custom payload and Intel RealSense D435 cameras. By implementing independent motor control for each wheel in the virtual environment, the Doppelbot successfully mimics the behavior of the real rover, allowing it to execute movements such as forward/backward motions and rotation.

The subsequent section explores the incorporation of video game metaphors and user interface elements in the design of the Doppelbot interface. Notably, elements inspired by video games, such as a minimap and the use of a computer mouse to highlight navigation points, have been also implemented into the interface. These game-inspired elements enrich the user's interaction and enhance their navigation experience within the simulated environment of Doppelbot, (Figure 3.2.3).

[3]slamcore.com
[4]intelrealsense.com/depth-camera-d435i

3.2.1 User Interface (UI) and User Experience (UX)

The Doppelbot navigation interface (Figure 3-4) draws particular inspiration from the real-time strategy (RTS) game genre, featuring classic titles such as the "Command and Conquer" series, originally produced by Westwood Studios and currently developed by Electronic Arts. In this timeless genre, users experience the virtual environment in a "bird's-eye" perspective, enabling control over various scene objects like vehicles and humanoids, among others, in either single target or group formation. These controls are often hotkeyed on a computer keyboard for a higher level of abstraction and control, ultimately leading players to consider the "bigger picture" and act on it, while tackling challenges against other players or the environment. An essential and widely adopted user interaction in RTS games involves the use of a computer mouse to select and set waypoints, allowing for daisy-chained commands and creating an automation path for the vehicle to follow. The virtual navigation of Doppelbot builds upon these foundational concepts while introducing a set of unique "features," setting apart the overall look-and-feel of the interface and interaction from traditional RTS games.

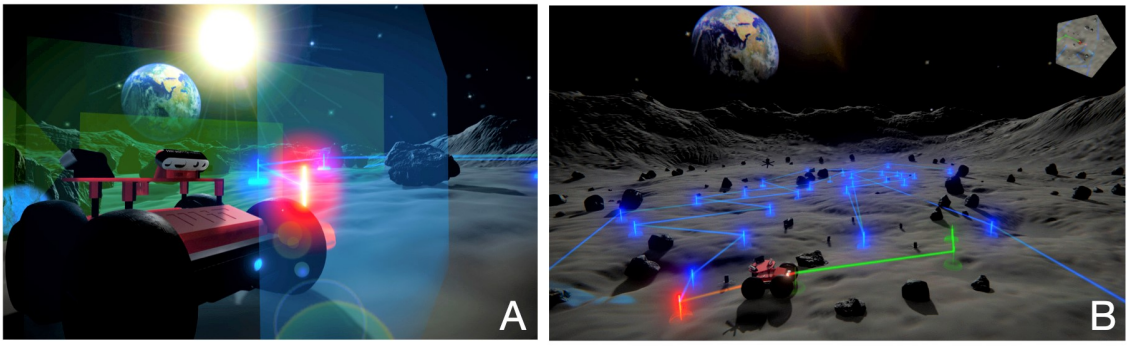

Figure 3-3: *Exploring Perspectives: Image A showcases the Doppelbot from a unique perspective, offering users the option to switch between various cameras within the environment, including a First-person view as if from the robot's cameras, to ultimately enhance their situational awareness and aids them in planning navigation driven missions within the simulated environment. Image B captivates the Doppelbot in action as it traverses a mission-planned route. Users can witness the rover's movement and interact with the environment. Additionally, users have the flexibility to disable the visualization of the cameras' field of view, allowing for a clear view of the rover's surroundings.*

For instance, to ensure intuitive rover control, a color-coded time-based visualization approach was implemented, (Figure 3-6). When the user designs a path, it appears as green lines on the screen. As the rover starts moving, these lines dynamically transition to blue, visually representing the actual path being traversed. Once the movement is completed, the lines turn red, indicating past actions. This color scheme allows users to easily distinguish between the rover's past, present, and future paths within the virtual environment. Other elements reminiscent of the RTS genre in the user interface include the minimap, (as show in the top right panel of Figure 3-3-B), which provides users with a comprehensive overview of the virtual environment, the

rover, as well as objectives. The minimap displays the rover's position and overlays its path on a 2D representation of the terrain, enhancing situational awareness and aiding in effective navigation. At any time, the user can interrupt the autonomous path and manually control the rover using the arrow keys of a computer keyboard. In addition to path planning and navigation, users have the flexibility to interrupt the automated path navigation at any time and switch to manual control. By utilizing the arrow keys on the keyboard, users can take direct control of the rover's movement.

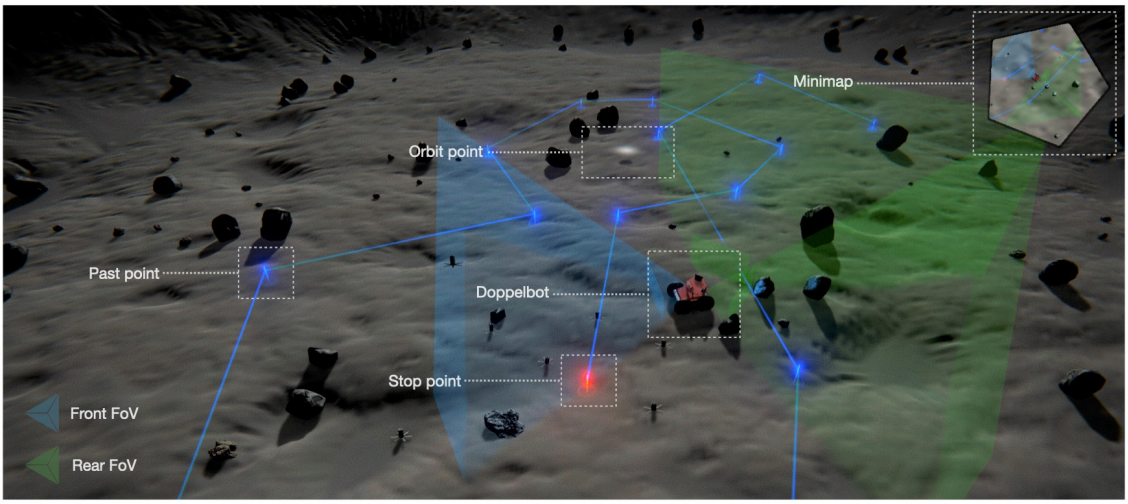

Figure 3-4: *Real-Time Strategy Inspired User Interface: A Comprehensive Overview. This image provides an in-depth look at the user interface and interaction of the rover system. At the top right of the screen, a minimap is displayed, centered at the rover's position. The traverse path is highlighted, with the final point marked in red and past points visible in blue. The user has the ability to change the orbit point, allowing for terrain exploration. The FoV of the real-sense cameras is visualized by blue and green tetrahedra. Users can set waypoints in the terrain or manually control the rover using a keyboard or joysticks, providing flexibility and control during the mission.*

Moreover, users can further explore the virtual environment through a range of interactive functionalities. They can orbit around and anchor in the virtual scene, scroll to zoom in and out for a closer look at specific areas of interest, and utilize hotkeys to swiftly change the camera perspective within the scene. The visual representation of the Intel RealSense D435i camera played a crucial role in providing users with insights into its field of view (FoV). To achieve this, the front-facing camera's FoV was visualized using a transparent blue tetrahedron positioned in front of the Rover Mini. This tetrahedron acted as a visual indicator of the area captured by the front camera, aiding users in understanding the camera's perspective. Similarly, the two rear cameras were represented using green tetrahedra of the same size, clearly displaying their respective FoVs. By visualizing the FoV of each camera with a distinct tetrahedron, users were empowered with a comprehensive understanding of the camera setup and its coverage, facilitating effective decision-making during navigation and exploration. These additional capabilities not only provide users with enhanced control over their

interaction with the virtual environment but also contribute to an immersive and engaging experience. It is worth noting that giving the users the freedom to toggle between manual and automated control modes through waypoints is a unique feature distinguishing the overall interaction from classic RTS games.

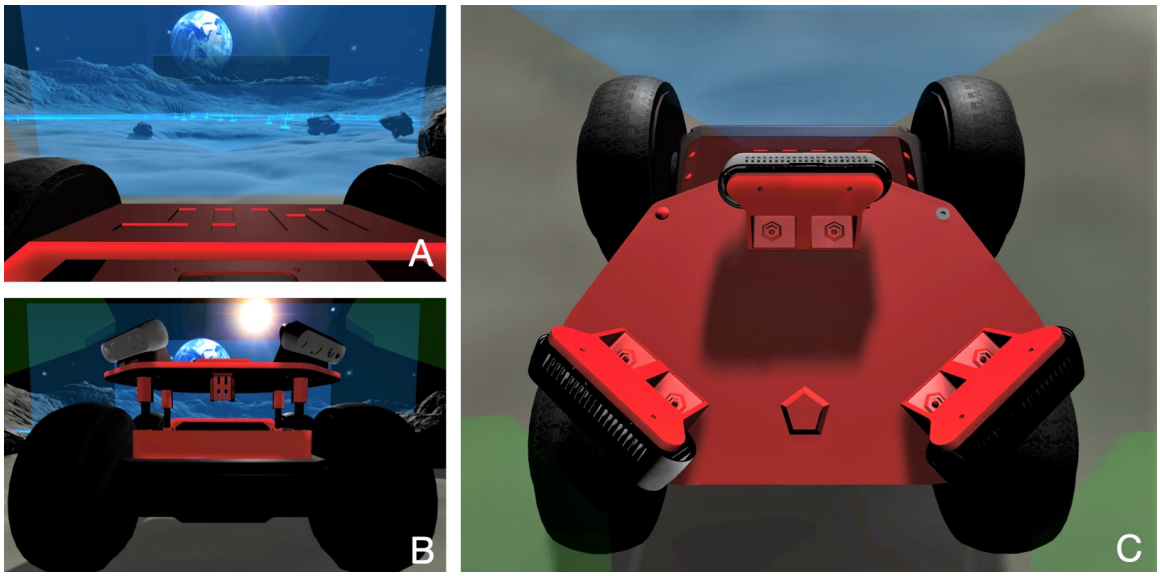

Figure 3-5: *Immersive Digital Twin of the Custom Rover Mini. This image showcases the front and side views of the digital twin, representing the Rover Mini developed by Rover Robotics. In image C, we get a top view of the rover, highlighting the placement of three real-sense D435 cameras, with one in the front and two in the rear.*

3.2.2 Technical Implementation

The implementation process commenced by importing the Rover Mini as a Rigidbody into Unity, with the wheels and payload detached. In Unity, a Rigidbody is a component that allows an object to be affected by physics, therefore the name "rigid body" commonly used in mechanics, enabling realistic interactions with the virtual environment. By configuring the Rigidbody component, the Rover Mini's physical properties, such as mass, friction, and collision, were established, making it behave like a physical object. To enhance the visual fidelity, a 3D model of the custom payload, (Figure 3-2), was integrated into the Rover Mini, along with suitable materials, such as 3D models of the Intel RealSense D435i, to achieve realistic rendering and appearance. For enabling movement of the Rover Mini in the 3D space, a custom script was developed using C Sharp, a programming language supported by Unity. This script included configurable parameters for maximum velocity, acceleration, as well as gravity, providing the flexibility to fine-tune the Rover's movement characteristics. The script also handled the rotation of the wheels based on user input, ensuring that the wheels' movements synchronized with the overall motion of the Rover. To facilitate manual control, the implementation allowed users to operate the Rover using either keyboard inputs or a PlayStation (PS) controller, offering multiple control options to suit user preferences. For automatic navigation, the algorithm

known as the A* (A star) algorithm [155] was employed. The A* algorithm is a popular pathfinding algorithm used to find the shortest path between two points in a grid-based environment, such as the Unity terrain. By using a mouse raycast system, users could set waypoints for the Rover by simply clicking on their desired location in the virtual environment, (as highlighted in Figure 3-6-B). The A* algorithm would then compute the optimal path between these waypoints, allowing the Rover to autonomously follow the planned trajectory. To provide users with visual feedback on the Rover's planned path, a line mesh renderer was implemented.

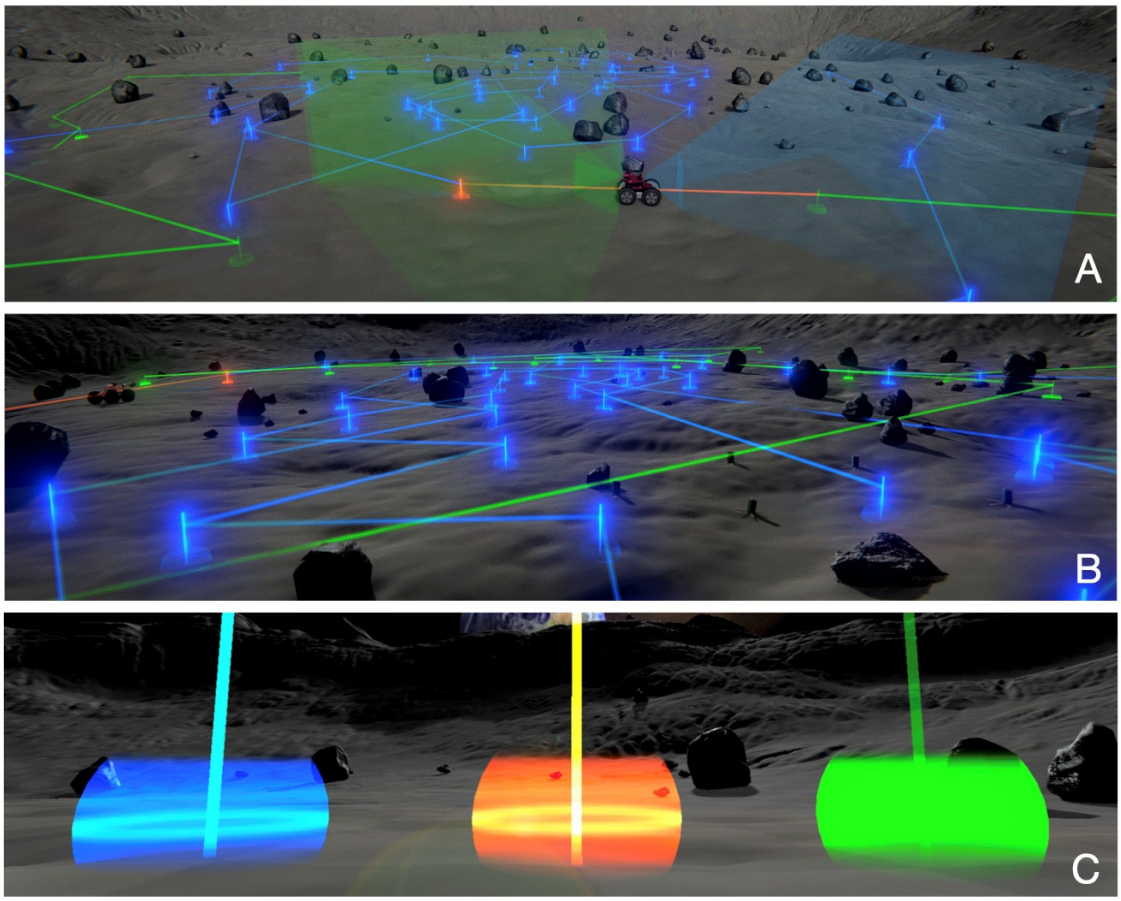

Figure 3-6: *Image A represents the rover ass seen while following its designated path within the simulated environment. Image B showcases the historical paths taken by the rover, represented by the blue lines, as well as the future waypoints depicted in green. Image C highlights past, present, and future waypoints. In this image, the waypoints are color-coded to represent different stages. The blue waypoints signify the rover's past path, the red waypoint represents the current position, and the green waypoints indicate the planned future path. This visual representation aids users in understanding the rover's journey and facilitates effective mission planning.*

This visual, (Figure 3-6), representation of the waypoints utilized three distinct line colors, each indicating the past, present, and future waypoints. This feature ensured users could easily discern the Rover's intended navigation path. The camera system

was pivotal in enhancing the user experience. Multiple camera views were incorporated, including the first-person perspective (FPS), real-time strategy (RTS), and role-playing game (RPG) views. These views allowed users to switch, (Figure 3-5), between different visual perspectives, offering a versatile and immersive experience. Additionally, the camera system supported orbit control, enabling users to rotate the camera around a pivot point, (Figure 3-4), using mouse inputs, providing greater control over their view of the virtual environment and the Rover's surroundings. Furthermore, to provide flexibility, the pivot point could be dynamically changed, empowering users to adjust the camera's focus based on their needs. Additionally, a simple follow camera was implemented to provide a smooth tracking experience for the Rover. It is important to highlight that the aforementioned implementation was developed using the Unity game engine, (specifically version 2020.1.8f1). The custom scripts were written in C Sharp, leveraging the extensive functionality provided by the Unity C Sharp API. By utilizing the Unity API, a wide range of methods, classes, and properties that facilitated the creation of the desired Doppelbot interaction and control system.

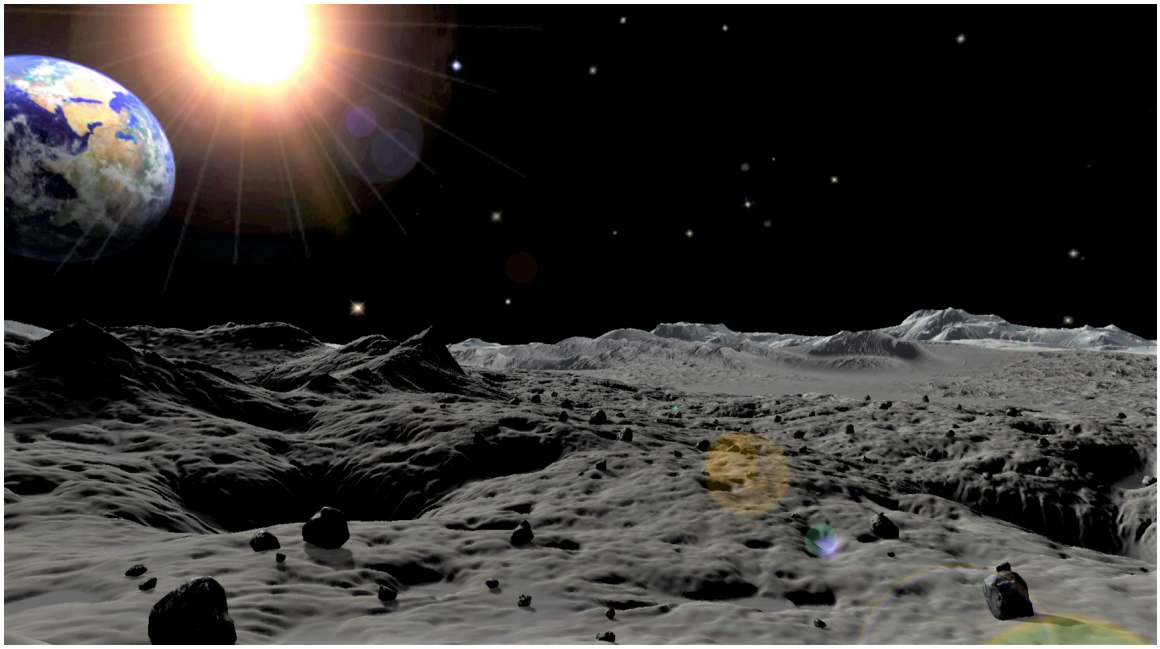

Figure 3-7: *The lunar environment in the virtual simulation was carefully created using a combination of high-resolution assets. These assets were carefully selected to provide an immersive and realistic space for testing mission planning and exploration scenarios.*

3.2.3 Synthesizing the Virtual Lunar Environment

The Unity game engine serves as the primary tool for creating and developing these virtual environments. The Unity Game engine is widely recognized for its versatility and powerful capabilities in creating interactive virtual environments. In this section, we will discuss the fundamental aspects of the Unity Game engine and explore how

it was employed to construct the lunar virtual environment. Creating a faithful representation of the lunar environment poses numerous challenges due to the Moon's unique characteristics. The lack of atmosphere, the prevalence of lunar regolith, and lunar boulders, are all aspects that must be considered in a proper simulation. The lunar environment, (Figure 3-7), was developed primarily using a variety of assets from the Unity store, thus enabling the creation of a rich and visually convincing lunar landscape. The lunar terrain was designed to reflect the Moon's unique topographical features, such as its craters, mountains, and plains, and textured with a lunar-like surface to enhance its realism.

The lunar environment consists of unique terrain features that need to be accurately represented in the virtual environment. To achieve this, a combination of 3D assets was procured from the Unity asset store. These assets include a Skybox featuring the Earth, lunar regolith simulation textures, a base terrain model depicting craters and other lunar features, as well as high-resolution 3D models of boulders and lunar debris. The careful selection and integration of these assets contribute to the overall realism and authenticity of the lunar environment. Creating an immersive lunar virtual environment involves considering the physical properties of the Moon. The materials and textures utilized in the environment were meticulously designed to mimic the unique characteristics of lunar regolith, such as its color, texture, and reflectivity. By accurately representing these physical properties, users can experience a more realistic and engaging interaction with the virtual lunar environment. Furthermore, post-processing effects were applied to enhance the visual quality and realism of the lunar virtual environment. Unity's Universal Render Pipeline (URP) includes a range of post-processing effects that were employed to achieve desired visual enhancements [156]. These effects contribute to the overall immersive experience by simulating lighting conditions, atmospheric effects, and other visual elements that are characteristic of lunar exploration. By employing these techniques and utilizing Unity's post-processing effects, the lunar virtual environment was meticulously crafted to provide users with a highly realistic and engaging experience of exploring the Moon's surface.

3.2.4 Synchronizing the Simulation

Interfacing the Rover Mini with the virtual lunar environment entails a sophisticated integration of hardware and software, including Ubuntu / ROS (Robot Operating System) [121], localization and navigation algorithms, networking and connectivity. These components work in tandem to synchronize the digital twin of the rover with the virtual environment, enabling accurate spatial representation and seamless interaction within the simulated lunar terrain.

At the core of this integration is the utilization of ROS, a flexible framework widely used in robotics applications. ROS provides a standardized platform for communication and coordination between the various hardware and software components of the Rover Mini. By leveraging ROS, the rover's sensors, actuators, and computational

Figure 3-8: *This image illustrates the preliminary efforts to establish synchronization between the real rover and its digital twin, the Doppelbot, as well as testing indoor navigation and control of the rover at the Media Lab E14 building in collaboration with Alexandra Forsey-Smerek and Cody Paige.*

modules can seamlessly interact, facilitating efficient data processing and control. This integration is vital for achieving autonomous navigation, mapping, and perception capabilities within the virtual environment. To further enhance the spatial synchronization between the physical rover and its digital twin, SLAMCORE comes into play [157]. SLAMCORE encapsulates the concept of SLAM, which stands for Simultaneous Localization and Mapping, offering a powerful suite of algorithms and tools for positioning, mapping, and perception.

Through its third-party API, SLAMCORE provides real-time SLAM operations that enable the conversion of depth information into actionable spatial understanding. By incorporating this module into the Rover Mini's architecture, the rover gains the ability to accurately determine its position in the physical space and update that information within the virtual environment. This synchronization allows for accurate navigation and interaction between the physical and virtual realms. The integration of SLAMCORE with ROS is made possible through the Nvidia Xavier board which also serves as the computational powerhouse of the Rover Mini, harnessing its robust processing capabilities to handle complex SLAM algorithms and perform real-time sensor data fusion. With the Nvidia Xavier board running ROS and communicating with SLAMCORE, the Rover Mini can achieves a reliable and efficient autonomous navigation, mapping, and perception.

In terms of connectivity, the Rover Mini is equipped with WiFi capabilities, enabling wireless communication and data transmission. This WiFi connectivity plays a vital role in the integration of the rover with the virtual lunar environment. Through the WiFi connection, the Rover Mini sends data, including its real-time position informa-

tion, to the virtual environment running in Unity. In the context of Doppelbot's data exchange, UDP (User Datagram Protocol) sockets are employed as a communication protocol. UDP is a widely used protocol in robotics and real-time applications due to its lightweight nature and low overhead [158]. It operates in a connectionless manner, meaning that data packets are sent without establishing a dedicated connection between the sender and receiver.

3.3 On Going and Future Work

3.3.1 Visualizing Time Delays in Space Operations

The challenges posed by time delays in space operations have been the focus of research for over three decades, particularly in the context of human teleoperation in space [159]. Experiments have investigated the impact of delays on human performance and explored potential solutions to mitigate their effects. Predictive displays have been demonstrated as a means to assist humans in overcoming the delay, while supervisory control offers a range of options, from local impedance control during contact with the environment to higher-level local automation. This delay can range from seconds to minutes, depending on the distance between the Earth and the target location. Time delays pose challenges in real-time decision-making and control, as operators must account for the time lag and anticipate the consequences of their actions.

To mitigate the effects of time delays and enhance the efficiency of space operations, automation plays a crucial role. However, in situations where human intervention is necessary, providing operators with a clear understanding of the temporal context becomes essential. Ideas involving possible trajectory forecasts have been considered before in aerospace (e.g., Paradiso [160] employed a search framework to unify momentum management and CMG (Control Moment Gyroscopes) steering to forecast and select possible gimbal trajectories in the face of unmodelled torques), but offer a ripe area for exploration with the sophisticated graphical environments available today. Therefore, a visualization concept is proposed, (Figure 3-10), that offers a 4-dimensional (4D) representation, encompassing the estimated, desired, and current positions of the digital twin, along with the dimension of time.

This innovative visualization approach integrates an opaque and colored representation of the digital twin within a single integrated environment, enabling users to gain a comprehensive understanding of its spatial information and temporal evolution. By incorporating time as the fourth axis, along with future UI elements such as a timeline, this visualization concept provides a dynamic view of the digital twin's movement and positioning within the virtual environment. Users can observe the estimated position, representing the projected path of the digital twin based on its intended trajectory. The desired position indicates the target location or desired path that the rover should follow. Lastly, the current position represents the real-time location of the digital twin, providing up-to-date information about its actual move-

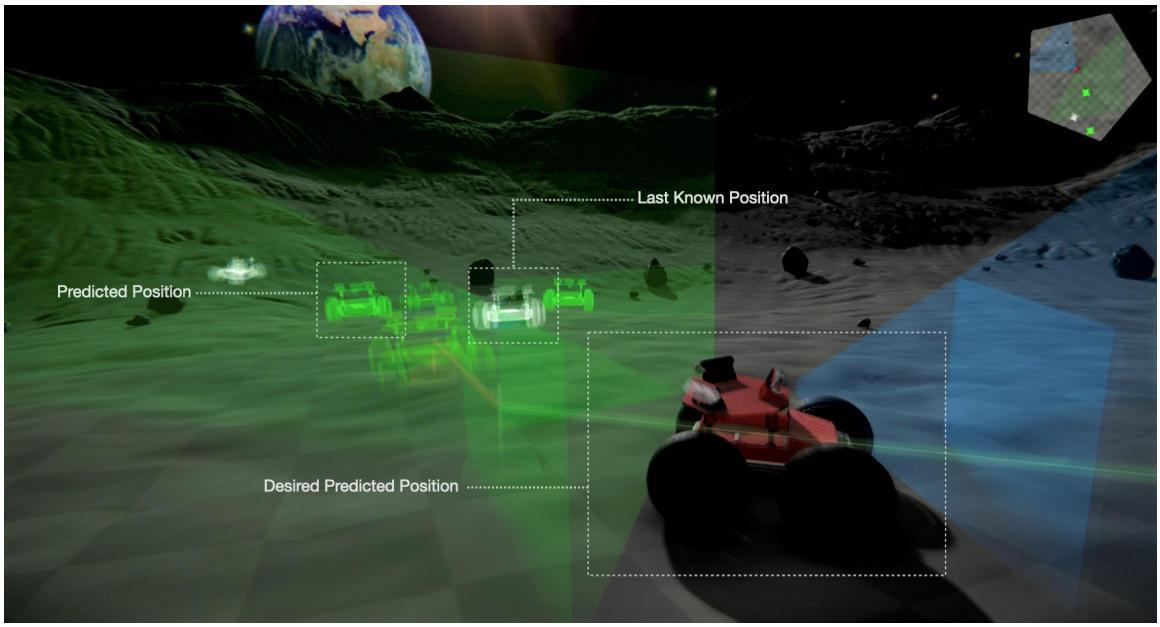

Figure 3-9: *This image proposes a concept of visualisation to pinpoint the past, present, and future possibilities of a rover's position within the virtual environment.*

ment. If the robot's autonomous operations (e.g., adjusting to immediate situations or other factors) or effects from terrain uncertainty, among other reasons, could cause the rover to deviate from its path, then the simulator might also forecast other potential trajectories based on its current knowledge and the rover's programmed behavior.

Moreover, the 4D visualization facilitates a holistic perception of the digital twin's spatial dynamics over time. Users can track its trajectory, observe any deviations from the desired path, and monitor the progression of its movement within the virtual environment. For example, if a command was sent from Earth to the rover, operators can observe the virtual states of the digital twin and assess its corresponding response based on the time stamp attached to it. The integration of the simulated and actual rover in a single environment provides a comprehensive understanding of the rover's status, actions, and including simulations of time lag associated with communication. This visualization concept aims to help operators maintain situational awareness, in a sense providing them with an abstraction tool to perform important calls that present crucial to the operation by adapt their decision-making process to accommodate time delays with anticipated moves, and ensure effective and timely control over space operations and monitoring.

3.3.2 Doppelbots and Large Language Models (LLMs)

The intersection of artificial intelligence (AI) and video games has been a focal point of interest and innovation for decades. Various AI techniques have been employed to enhance the gaming experience, ranging from non-player character (NPC) control to procedural content generation (PCG) [161]. Traditionally, NPCs have been con-

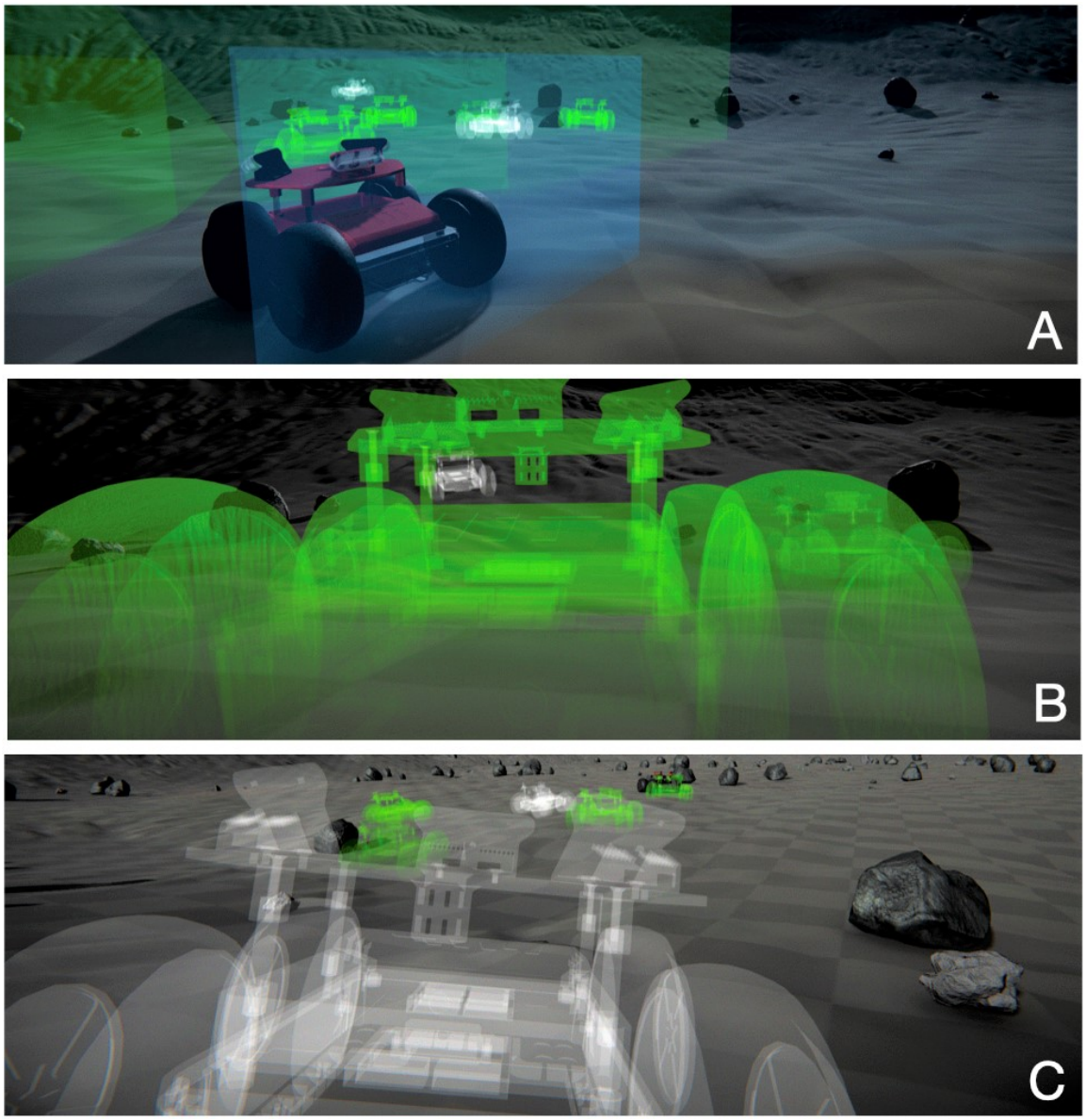

Figure 3-10: *This image presents a visualization tool designed to aid in reflecting the last known update during remote space exploration. Image A shows the desired predicted positions of the rover in solid and other possibilities as "ghosts". Image B features a Doppelbot with a green transparent texture, indicating a possible simulated position. Image C also indicates the last known position of the Doppelbot. Together, these images propose a visualisation to pinpoint the past, present, and future possibilities of a rover's position within the virtual environment.*

trolled through pre-programmed scripts, which, while effective, are inherently limited in their capacity to handle diverse and unpredictable in-game scenarios.

The rise of machine learning [162] and deep learning [163] models has opened up new possibilities in the field. Machine learning's implementation in video games is mostly known through research projects [164], as gaming companies often withhold specific information about their intellectual property. Nevertheless, the impact of these applications has been substantial. Notable examples include deep learning agents competing with professional human players in complex strategy games and the integration of machine learning in popular titles like Atari / ALE, Doom, Minecraft, StarCraft / Starcraft II, and various car racing games [165, 166, 167, 168, 169]. Additionally, traditional games like chess and Go have seen their gameplay evolve through the influence of machine learning [170]. Despite these advancements, the application of large language models (LLMs) [171], such as OpenAI's GPT-4[5], in video games and simulations is still a relatively unexplored area.

LLMs have demonstrated remarkable capabilities in natural language understanding and generation, with their applications extending to tasks involving reasoning, planning, and learning from interaction [172]. Integrating these models into video games and simulations holds immense promise for creating dynamic, responsive, and autonomous NPCs. The Integration of LLMs within virtual environments and digital twin technology can provide a more dynamic, realistic, and immersive experience compared to traditional scripted NPCs. For instance, by leveraging the data-driven and learning capabilities of LLMs, Doppelbots can adapt to changing environments and suggest alternatives to complex decisions, or even learn from past experiences. OpenAI's GPT-4, offers a powerful LLM tool with a robust third party API capable of integrating with Unity.

GPT-4 has demonstrated promising capabilities, although still at its infancy, in tasks involving reasoning, planning, and learning from interaction. By utilizing a text-based activity logging translation tool, users can contextualize information from the virtual environment and Doppelbots. This translation tool transforms cues and data collected from the virtual sensors, which are strategically placed and programmed by the user. The tool enables users to generate text-based prompts, facilitating interaction and communication with the ChatGPT API. This process allows for seamless integration of the virtual environment, Doppelbots' objectives, and user inputs. By processing the response sent back from the users' requests into this contextualized session, sensors and user interactions can generate appropriate responses, actions, and strategies for the NPC.

The concept of Doppelbots aligns with the advancements in 3D robotics simulations. The exponential increase in processing power and the availability of open software

[5] openai.com

and hardware standards have enabled more complex, versatile, and scalable simulation strategies that has been long explored [173]. This evolution has also allowed real-time simulations with hardware in-the-loop or simulations controlling mobile embedded systems. Robot simulation platforms like Open HRP [174], Gazebo[175], and Webots [176] revealed early development into versatile robotic simulations. Despite their individual strengths, these platforms often struggle with the scalability and portability of models and controllers, often requiring careful matching or recompilation on different hardware or platforms.

The integration of LLM into robotics sytems through contextualized virtual environments aims to address the ultimate challenges to create a general-purpose rover simulator platform that is flexible, scalable, dynamic, and generated from real-world data, to abstract the complexity of underlying systems. The recent integration of GPT-4 Plugins further enhances the simulator's capabilities, enabling direct computation through paradigms such as the Wolfram Alpha engine [177], adding exciting ventures to integrating AI systems within both physical and virtual environments.

3.3.3 Integration of LLMs in Space Operations

The Unity game engine serves as a robust platform for developing rich, interactive simulations [58]. With its comprehensive suite of design and scripting tools, Unity offers a versatile environment for integrating advanced technologies like LLMs. The focus of this section is the integration of OpenAI's GPT-4 into the Unity game engine, thereby programming the Doppelbots to operate in a simulated lunar environment. Unity has been chosen for its broad adoption within the gaming community, multiplatform support, robustness, and compatibility with external APIs. With its integrated scripting engine, Unity is well-suited to communicate with GPT-4 and parse its responses to drive the behaviors of the Doppelbot.

The integration of GPT-4 into Unity is initiated with a request to the Chat GPT-4 API. This initial request establishes a cascading chain of context using a detailed prompt. This prompt contains comprehensive information about the Doppelbot, including its position in the simulated lunar environment, the locations of shadowed regions, points of interest such as lunar regolith containing water, the current energy status of the rover, and its speed. The prompt also has access to a range of rover functions, such as move(destination, speed, action) and rotate(angle, speed, action). Once the initial context has been established, GPT-4 is tasked with generating responses that guide the rover's actions in the simulation. These responses are returned in a JSON format, a choice driven by JSON's wide acceptance, readability, and ease of parsing. On receiving the responses from GPT-4, a script running within the Unity game engine parses the JSON text. This parsing process translates the LLM's responses into actions within the Unity environment, triggering the corresponding functions to control the Doppelbot's movements and actions. This could include directing the rover to a particular location, adjusting its speed to conserve energy, or initiating measurements of the lunar surface.

This integration of GPT-4 and Unity forms the backbone of the Doppelbot project. It demonstrates the potential of LLMs in enhancing the capabilities of NPCs, not just in traditional gaming contexts, but also in the realm of scientific simulations. The framework offers a novel approach to data-driven NPC control, creating dynamic and adaptive responses based on real-time conditions in the simulated lunar environment. One of the key features of the Doppelbot is its ability to interact with users through natural language. Leveraging GPT-4's natural language processing capabilities, users can instruct the Doppelbot with commands like,"Survey this zone. Return data every 30 seconds", toggle "power conservation mode on", or "choose the best sunlit route to maintain solar charge"...etc. Such interactions further reinforce the Doppelbot's role as an intelligent, adaptive NPC capable of interpreting and responding to complex commands. The result is a sophisticated lunar environment simulator that combines the visual fidelity of Unity with the artificial intelligence capabilities of GPT-4, creating a dynamic, interactive, and adaptive simulation platform for lunar exploration.

3.4 Conclusion

In conclusion, this chapter has presented the Doppelbot project, which introduces an innovative approach to mission planning and user interaction within a virtual lunar environment. A core concept in this research is built upon Cross-Reality, (Section 1.3.2), applied to the field of space exploration and operations. Through an immersive environment encapsulating a simulated lunar environment and a simulated digital twin of the Rover Mini by Rover Robotics called Doppelbot. The behavior and locomotion of the actual rover is implemented within the Unity game engine and provides independent motor control for each wheel based on user inputs. Doppelbot showcases a wide range of movements that enable users to interact with it within the virtual environment.

One of the notable aspects of the Doppelbot project is the incorporation of video game metaphors and user interface elements, which significantly enhance the user's experience and navigation within the simulated environment. By drawing inspiration from video games, elements like the minimap and the utilization of a computer mouse for highlighting navigation points have been seamlessly integrated into the interface. These game-inspired elements not only contribute to the interface's intuitiveness but also make the user's interaction with Doppelbot engaging and immersive. Users can navigate and explore the virtual lunar environment, benefiting from effective interaction and a sense of control.

The detailed discussion in this chapter on the Real-Time Strategy (RTS) inspired user interface further explores the application of the Unity game engine and physics simulation to create an interface reminiscent of popular RTS games. By implementing a color-coded time-based visualization approach, users can gain a clear visual

understanding of the planned, current, and past paths of the rover. This visualization method provides valuable context for users, allowing them to track the rover's movements over time. Additionally, the inclusion of a minimap offers users a comprehensive overview of the virtual environment, enhancing situational awareness and aiding navigation. The ability to seamlessly switch between manual and automated control modes empowers users to adapt their control methods based on their preferences and requirements.

A system investigation has been conducted to explore the synchronisation of the Rover Mini with the Doppelbot, through the integration of powerful embedded systems, depth cameras, and third party SLAM platforms. These include the utilization of embedded computing and sensing platforms such as the Nvidia Xavier, Intel RealSense Cameras, Ubuntu running ROS (Robot Operating System), and SLAM (Simultaneous Localization and Mapping). Integrating these technologies will enable accurate spatial representation and seamless synchronization between Doppelbots, physical, and virtual environments.

Finally, in the future and on-going work section of this chapter, (Section 3.3.1), a proposed visualization concept, that suggests overlaying several anticipated, and current known positions of the digital twin along with the dimension of time in a virtual monitoring platforms, to provide users with a comprehensive understanding of the rover's movement and positioning in the free of the large data delays implicit in the planetary missions. Additionally, the integration of large language models (LLMs), such as OpenAI's GPT-4, holds immense potential for creating dynamic and responsive non-player characters (NPCs) within the virtual environment. By leveraging the capabilities of LLMs and the Unity game engine, more realistic, adaptive, and intelligent simulations can be transformative in several fields such as remote construction and farming, intelligent monitoring environments, manufacturing, and space exploration.

Chapter 4

Doppelspot

Virtual Possession - Applications in Human-Robot Interaction and Remote Environments through Augmented Virtuality.

4.1 Introduction

4.1.1 A Brief Overview of this Chapter

This chapter expands on the literature and concept of the Doppelbot presented in Chapter 3, with the introduction of more complex robotic systems in space analog environments. Through the implementation of a robotic telepresence system, centering its focus on the remarkable Spot, the quadruped robot designed by Boston Dynamics renowned for its exceptional mobility and agility in navigating outdoor terrains. Spot emerges as an ideal candidate for automating routine inspection tasks and data collection in a multitude of industries [178, 179]. Its versatility has rendered it indispensable for warehouse management, inspection tasks, and even construction [180], adapting to challenging terrains through its advanced mobility, accurate, and frequent data gathering. The extensive adoption of Spot spans various sectors, proving its mettle as an invaluable assistant in inspections and industrial monitoring, public safety, and scientific exploration. Astonishingly, Spot made its way in the performance art industries since its flexibility, and connection to human beings, is not confined to traditional applications; it has even graced fashion show runways [181], showcasing its adaptability and potential for diverse and innovative uses.

The following sections of this chapter introduces the Doppelspot, a telepresence exploration experience in a space analog environment. The Spot robot is augmented with a custom-built payload that facilitates the streaming of real-time data, including LiDAR data and 360° video, to a Virtual Reality (VR) headset. This integration of Spot's capabilities with VR enhances the immersive experience and allows users to remotely explore and interact with the environment. Doppelspot demonstrates a blend of visualisation techniques into remote operations and data collection, enabling humans to navigate and study challenging terrains and environments from a safe and distant location. The emphasis on autonomy in Spot's navigation offers efficient and

independent exploration, while introducing the ability to involve a human operator when needed through the telepresence system, providing a valuable combination of human intervention and robotic capabilities.

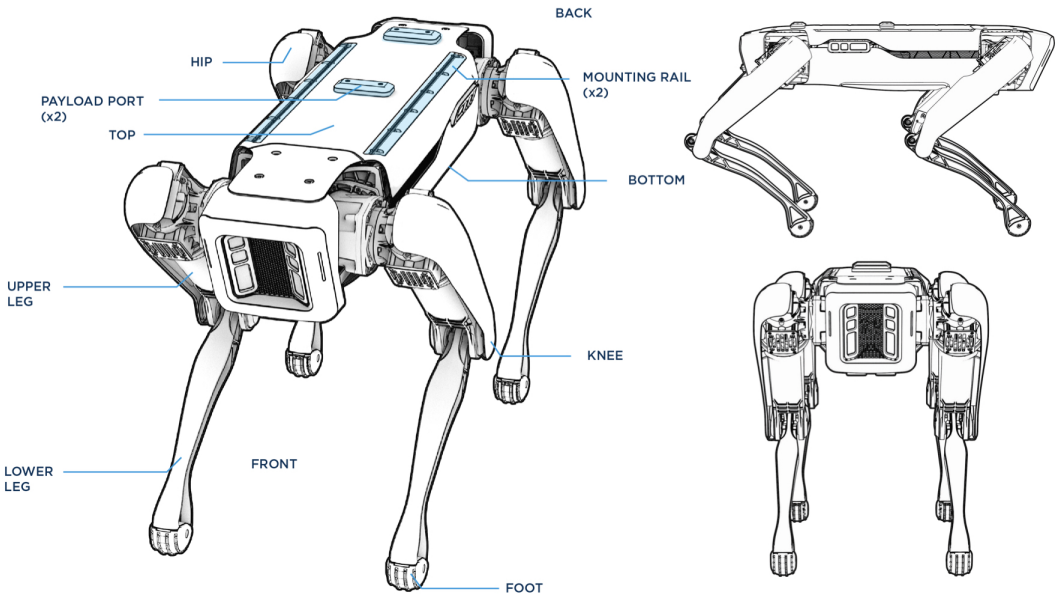

Figure 4-1: *Spot is an advanced robotic quadruped designed and developed by Boston Dynamics. This diagram illustrates the remarkable features of Spot, including its 12 degrees of freedom and actuator distribution in the hips and knees. Equipped with stereo cameras, Spot captures detailed black and white images and videos, making it a versatile and agile robotic platform for various applications. Image courtesy: Boston Dynamics.*

4.1.2 Presentation of Spot by Boston Dynamics

Spot, (Figure 4-1), is equipped with an array of advanced features that enable its efficient operation. It incorporates five pairs of stereo cameras, providing high-resolution black and white images and video for comprehensive perception and sensing capabilities. These cameras contribute to Spot's ability to navigate and interact with its surroundings effectively. Spot also has a variety of sensors, including a LiDAR scanner, which helps it to create a 3D map of its surroundings. This allows Spot to avoid obstacles and navigate through complex environments. Spot also has a variety of sensors that allow it to detect and avoid hazards, such as stairs and ledges. This makes Spot a safe and reliable robot for a variety of applications. The robot's locomotion is facilitated by its hips and joints, which consist of two actuators in each hip and one actuator in each knee. This configuration allows for precise control of Spot's movements and results in 12 degrees of freedom, with three degrees of freedom per leg. The hip joints, referred to as HX and HY, offer a wide range of motion, including rotation and flexion/extension, while the knees provide additional flexion/extension capabilities within a specific range.

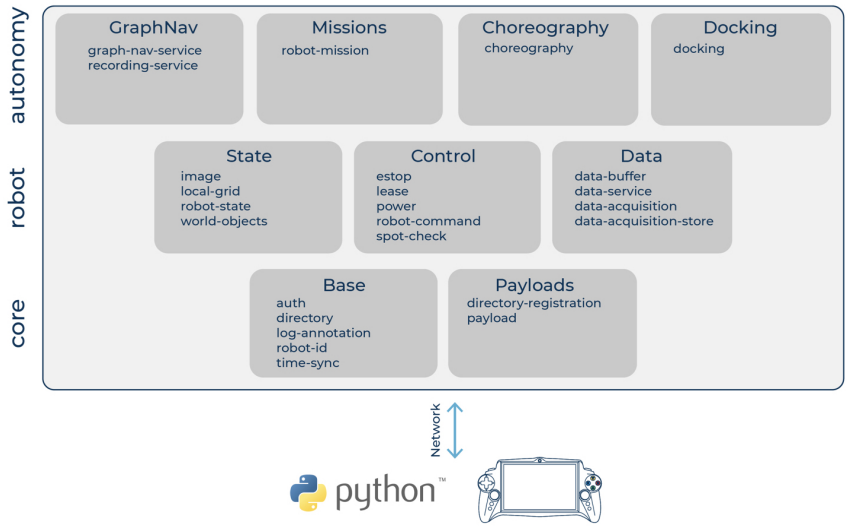

Figure 4-2: *The Spot API provides a flexible and versatile framework for controlling Spot, accessing sensor data, and integrating custom payloads. This diagram showcases the client-server model, where client applications communicate with Spot's network services over various IP networks. The architecture allows for seamless integration of client applications and expansion through payloads, such as Spot CAM, offering additional control and functionality. Image courtesy: Boston Dynamics.*

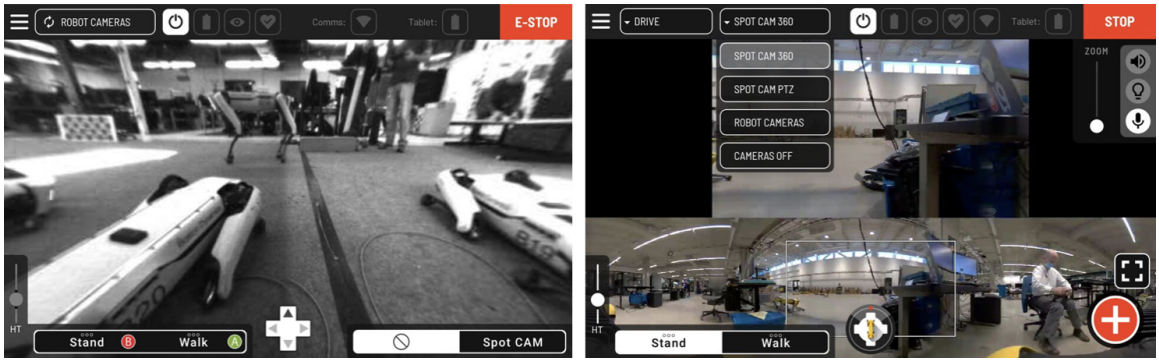

Figure 4-3: *The Android app provides intuitive features for controlling Spot. To change camera views, users can utilize the top-down perspective for a comprehensive overview, and employ touch-to-go functionality to guide Spot to specific locations. With a range of up to 50m and adaptability to different wireless configurations, the app ensures a safe teleoperation of Spot while avoiding obstacles for optimal performance. Image courtesy: Boston Dynamics.*

To facilitate seamless integration and control of Spot, Boston Dynamics has developed a proprietary Python API[1] that enables applications to interact with the robot, access sensor information, and create and integrate payloads, (Figure 4-2). The Spot API follows a client-server model, where client applications establish network connections with services running on the robot. This API offers a comprehensive set of

[1] dev.bostondynamics.com

functionalities, empowering developers to leverage Spot's capabilities and customize its behavior to suit their specific needs. The Spot API architecture consists of various network services that operate within the robot's framework. These services include the image service, responsible for handling image-related operations, and the robot-command service, which facilitates commanding and controlling Spot's movements and actions.

These services form the foundation of the API stack, with higher-level services, such as autonomy services and choreography, built on top of them. Client applications can run on a range of platforms, including tablets, laptops, cloud-based applications, or payloads connected to Spot. As long as a network connection can be established, the client application can communicate with Spot. This connection can be established through various IP networks, such as direct WiFi or Ethernet connections to the robot, intranet, or even the internet, ensuring flexibility in deployment scenarios. Furthermore, Spot's API allows for the integration of additional payloads, expanding the range of services and capabilities beyond what is provided by the robot itself. For example, Spot CAM offers services that allow control of stream quality and LED lights, enhancing the visual and communication aspects of the robot's operation. Developers utilizing the Spot API have access to a wide range of functionalities, including networking, base services, geometry and frames, robot services, e-stop functionality (safety and emergency stop button), lease management, fault handling, autonomy services, Spot Arm control, and access to Spot's data. These capabilities empower developers to create innovative applications and solutions, leveraging Spot's mobility, sensing, and manipulation capabilities for various use cases.

4.2 Overview of Spot's Payloads

4.2.1 LiDAR Enabled Payload

One of the notable payloads available for Spot is the Spot Enhanced Autonomy Payload (EAP), which incorporates a Velodyne LiDAR assembly, (Figure 4-4), to enhance the robot's sensing capabilities. The addition of the LiDAR sensor significantly enhances Spot's depth perception, extending it to approximately 120 meters compared to the 2 to 4 meters range without the LiDAR. This feature is particularly beneficial when navigating areas with limited physical features, referred to as "feature deserts," where the robot needs sufficient data to localize itself within an "Autowalk" map. The EAP assembly consists of several components, including the Velodyne Puck LiDAR, previously known as the Velodyne VLP-16, a mount plate, roll cage, and the integrated Spot CORE, which hosts Spot's General Expansion Payload (GXP) and provides additional computing capability. Spot CORE serves as the interface for the Velodyne LiDAR service to connect with the robot, offering network and data interfaces, as well as regulated power within an integrated package. By incorporating the EAP, Spot's autonomous navigation capabilities are significantly enhanced. Without the EAP, the effective range of the base platform is limited to 4 meters, constraining

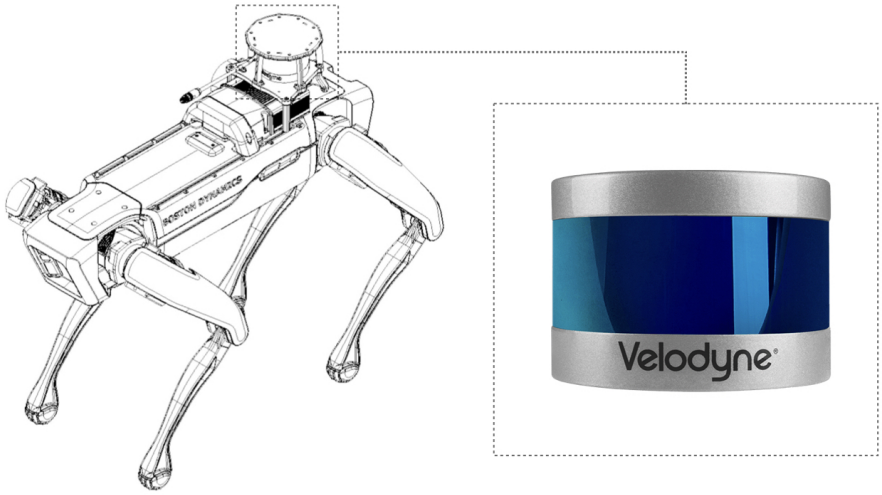

Figure 4-4: *Spot, equipped with the Enhanced Autonomy Payload (EAP) combining a Spot CORE payload and the Velodyne VLP-16 assembly. This LiDAR significantly enhances Spot's depth perception, extending it up to approximately 120 meters compared to the standard range of 2 to 4 meters. Image courtesy: Boston Dynamics / Velodyne.*

the GraphNav maps to features within this range. The EAP, utilizing the VLP-16 LiDAR, extends the mapping range to approximately 120 meters, enabling Spot to gather more comprehensive data for navigation and mapping purposes. The Spot CORE comes preloaded with the Velodyne service software, which facilitates seamless integration with the LiDAR. This service registers the LiDAR payload and provides a RemotePointCloud service, which sends processed and filtered data to one or more clients. The GraphNav service utilizes this data for map creation and navigation.

Technical Specifications (VLP-16):

The Velodyne LiDAR Puck[2] is a LiDAR depth sensor with a 360° horizontal field of view (FoV) and a 30° vertical FoV. It offers an angular horizontal resolution of 0.1°-0.4° and a vertical angular resolution of 2°. The VLP-16 achieves a depth accuracy of ±3 cm within a measurement range of up to 100 meters.

4.2.2 Modular Custom Payload

Spot is equipped with various other integrated cameras and depth sensors that provide a comprehensive range of imaging capabilities, as well as a versatile payload that was used to house several instruments for the purposes of this project. These cameras and sensors play a crucial role in data collection and telepresence applications, as well as autonomy, enhancing Spot's visual perception and enabling it to capture a variety of visual information. Spot is equipped with five stereo cameras featuring global shutter image sensors. Each camera can provide a fisheye image, a

[2]velodynelidar.com/products/puck

depth image, and a depth image adjusted within the frame of reference of the fisheye image. The ideal operating range for depth images is 4 meters. The integrated Spot cameras are strategically positioned, with two in the front, one on each side, and one in the back, providing a 360° horizontal FoV. Two fisheye images taken from the front cameras could be stitched together using Boston Dynamic's proprietary Python API. The Intel RealSense D435i is a stereoscopic depth camera with an 87° horizontal FoV and a 58° vertical FoV. It offers a depth stream output resolution of up to 1280x720 at a maximum frame rate of 90 fps. This camera module also includes an RGB camera with a 69° horizontal FoV and a 42° vertical FoV, providing a 2 MP resolution. The recommended operating range for the D435i is between 0.3 meters to 3 meters. Additionally, the integrated IMU collects time-stamped data on camera movement and orientation, contributing to more robust depth data reconstruction.

Figure 4-5: *Equipped to handle diverse terrains, Spot confidently traverses various surfaces encountered during its missions. From rugged landscapes to urban environments, Spot is designed to "go where humans go". This image was captured from a custom payload featuring the Insta360 ONE X VR camera during a tour on the MIT Campus[a] on August[th], 2021.*

[a]Spot MIT Tour in a 360° video.

During the field experiment, two additional Intel RealSense cameras, the D435i and the L515, were mounted on a custom 3D printed payload mount designed in collaboration with Ferrous Ward from the MIT RESOURCE team, (Figure 4-6). Each camera was assessed separately, (Appendix B Tables B.2-B.3), to evaluate their capabilities and suitability for data collection in space analog environments [182]. The D435i camera, equipped with RGB and stereo-depth capabilities, was utilized to capture data, by Alexandra Forsey-Smerek, at each waypoint along the traverse. A python script was developed to collect images from the D435i camera, ensuring efficient data

gathering and organization. The camera was manually rotated 360° at each waypoint, capturing both RGB and depth data. The L515 camera, incorporating Time-of-Flight (ToF) technology, was also employed, by Cody Paige, to capture data at each waypoint. Similar to the D435i, a separate python script was developed to control and capture data from the L515 camera. The camera was activated at each waypoint, and a recording was initiated while manually rotating the camera 360° to capture RGB and high-resolution depth data.

The Insta360[3] ONE X is a 360° action camera designed to capture immersive photos and videos, (Figures 4-10, 4-5). It offers a full 360° view by using two ultra-wide-angle fisheye lenses. The camera's software stitches the two images together, creating a seamless 360° image or video. It captures RGB photos at a resolution up to 6080x3040. It can also record RGB video at a resolution up to 5.7K and 30 fps. The camera provides an immersive view with a 360° horizontal FoV and a 180° vertical FoV. These integrated cameras and depth sensors provide Spot with a rich visual perception and imaging capabilities, enabling it to gather comprehensive data for various applications, including data collection, telepresence, and visual analysis in space analog environments.

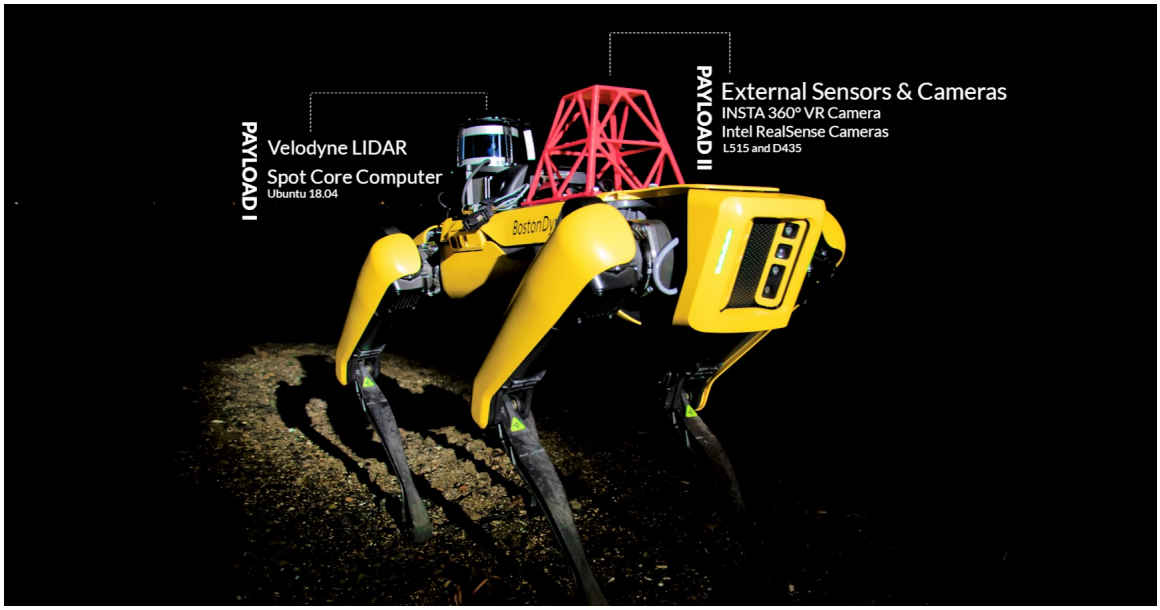

Figure 4-6: *A photograph of Spot in nighttime conditions with single spotlight lighting. Payload I consists of the Spot Core Computer and Velodyne LiDAR. Payload II is designed and developed by the MIT RESOURCE team, and carried either both Intel RealSense cameras or the Insta360 ONE X camera on the custom payload tower.*
Image courtesy: MIT RESOURCE & Nikita Borodenko.

[3]insta360.com

4.3 Analog Mission In Marblehead, MA

4.3.1 Description of the Designated Environment

A two-day field experiment was conducted on a granitic beach with exposed bedrock in Marblehead, MA to evaluate the performance of the selected cameras in a realistic and challenging setting. The chosen environment fulfilled several criteria necessary for meaningful testing, including:

- Exposed bedrock and unstable terrain: The site offered a natural setting with exposed granitic bedrock, providing geological features of interest for analysis. The unstable terrain made it difficult to set up and maintain the cameras, which tested their durability and reliability, as well as drive Spot.

- Variable lighting conditions: The beach was located in a coastal area with a variety of lighting conditions, including direct sunlight, indirect sunlight, and shade. This tested the cameras' ability to capture images in a variety of lighting conditions.

- Variable weather conditions: The beach was located in an area with a variety of weather conditions, including rain, wind, and fog. This tested the cameras' ability to capture images in a variety of atmospheric conditions.

- Minimal human activity: The site was located in a private area with minimal human activity, and offered amenities such as electricity, internet access, and other comforts. This made it an ideal location for scientific research in a controlled yet challenging environment.

- Accessible Geological Features: The selected site contained identifiable geological features such as distinct 90° fracture angles and continuous quartz veining within the granitic rock. These features served as points of interest for evaluating the cameras' ability to capture color-specific and geometry-specific geological information.

- Accessibility and Power/WiFi Availability: The site's location near a street allowed for easy transport of heavy equipment. Additionally, the site provided power and WiFi accessibility, crucial for operating the cameras and remotely accessing the data throughout the traverse.

- Daytime and Nighttime Conditions: The experiment encompassed traverses conducted both during daytime and nighttime conditions. This allowed for a comprehensive evaluation of the cameras' performance in different lighting scenarios, considering the impact of lighting conditions on image quality and depth perception.

4.3.2 Analysis of Test Results and Implications

Samples of the data collected during the field experiment, Appendix B Table B.1, yielded valuable insights into the cameras' performance and their implications for data collection in space analog environments. These key findings played a crucial role in selecting the most suitable cameras to create the immersive telepresence experience fused with depth information.

The FoV and Depth of Field (DoF): The cameras' FoV varied, with the integrated Spot cameras providing a full 360° FoV through multiple camera setups. The Intel RealSense cameras required manual rotation to capture a complete 360° FoV. The DoF varied across the cameras, with the Velodyne LiDAR offering the largest DoF of up to 100 meters. It was observed that Velodyne's large DoF, although beneficial in some scenarios, posed challenges in terms of resolution and identification of geological features. The resolution of depth data varied among the cameras, with the Velodyne LiDAR capturing a large number of points but at reduced resolution due to the viewing distance. The Intel RealSense cameras exhibited higher resolution depth data, enabling more detailed surface texture representation. Higher resolution data was particularly relevant for geological analysis tasks that required precise feature identification.

Each camera generated different file types and sizes, impacting the bandwidth requirements for data transmission and storage. The combination of RGB with ToF data, as provided by the Intel RealSense L515, posed higher bandwidth requirements compared to other cameras. Bandwidth considerations became essential for designing an efficient data pipeline and assessing the feasibility of onboard processing and transmission. Testing in both daytime and nighttime conditions revealed that the Intel RealSense stereo cameras performed surprisingly well in low-light environments, despite lower RGB image resolution.

This unexpected result indicated the stereo cameras' ability to function effectively for depth data extraction, even in challenging lighting conditions. The analysis of the test results highlighted the strengths and limitations of the Intel RealSense cameras for data collection in rocky analog environments. These findings provide essential insights for refining data pipelines, camera selection, and tool development tailored for geological analysis tasks in outdoor environments. Specifically, the cameras were systematically tested to determine their suitability for the Doppelspot telepresence application. The selection process considered multiple factors, including data bandwidth, accuracy in outdoor settings, and seamless integration with real-time functionality within the Unity application. By carefully evaluating these criteria, we aimed to identify the optimal camera solution that would enable effective and reliable operations in the challenging outdoor scenarios encountered by Doppelspot. Based on those tests, the Velodyne VLP-16 LiDAR, capable of capturing files in the format of .pcap, was selected to complement the 360° video stream from the Insta360 ONE X.

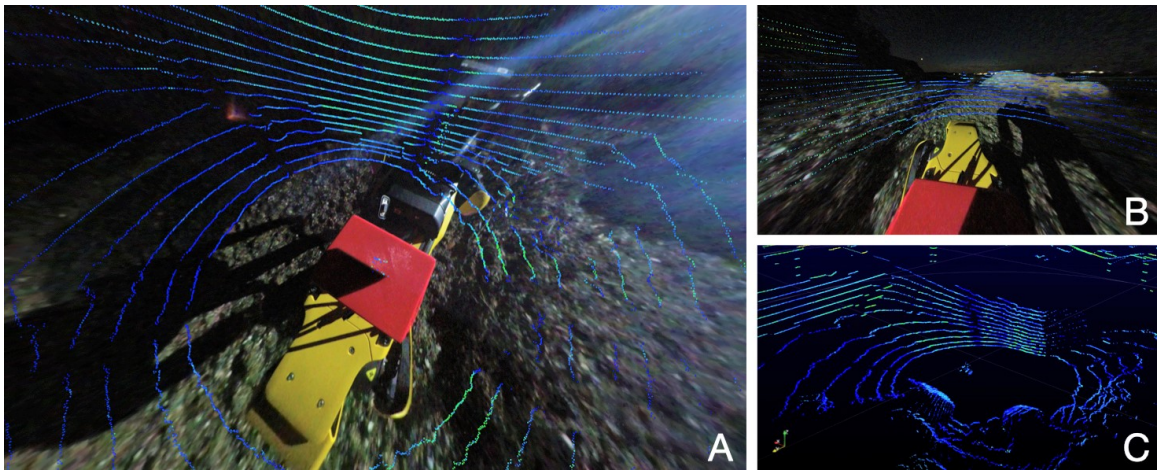

Figure 4-7: *Image A and B showcase the immersive teleoperation application developed in Unity 3D for the Oculus Quest, allowing users to experience real-time telepresence with a fusion of Velodyne VLP-16 LiDAR data and 360° video. Whereas image C highlights the raw .pcap data captured by the Velodyne VLP-16 LiDAR in the same outdoor rocky environment at Marblehead, MA.*

4.4 Mobile Immersive LiDAR Telepresence

4.4.1 Augmented Virtuality in Space Analog Environments

Augmented Virtuality (AV) is a concept that combines elements of both AR and VR, (Section 1.2.3), as a blend between visual elements to refine the telepresence experience. It involves overlaying virtual information or objects onto the real-world as perceived through a head-mounted display with an immersive live video feed, essentially providing users with additional contextual data and enhancing their understanding and interaction with remote environments. In the context of robotics and telepresence, (Section 1.2.1), AV plays a significant role in blending the physical and virtual worlds, enabling users to have a more immersive and informative telepresence experience. By leveraging AV applications, users can access real-time data, annotations, and visualizations that are superimposed onto the live video feed from the remote robot via several cameras. This additional information enhances the situational awareness and understanding of the remote environment within depth, allowing users to make informed decisions and perform tasks effectively. AV also facilitates the interaction between users and the remote environment, enabling virtual objects or tools to be manipulated and utilized for various purposes.

The utilization of AV in robotic telepresence offered several benefits. Firstly, it enhanced the users' perception and understanding of the remote environment by providing contextual information and visual cues. This improved situational awareness empowered users to make better decisions and perform tasks more efficiently. Additionally, AV facilitated collaboration and communication between remote users and on-site personnel, as both parties could interact with the same augmented content

in real-time, fostering effective teamwork and information sharing. However, implementing AV also presented certain challenges and technological limitations. Ensuring the accurate alignment and registration of virtual overlays with the real-world environment required sophisticated computer vision algorithms and precise calibration techniques. The latency introduced by processing and rendering the augmented content could potentially impact the real-time nature of the telepresence experience.

Moreover, the selection and design of intuitive and user-friendly interfaces for interacting with the virtual overlays required careful consideration to ensure a seamless and efficient user experience. The review of the benefits and challenges encountered with AV contributed to the understanding of its potential for enhancing robotic telepresence. It provided insights into the technical aspects, user experience considerations, and future directions for further refinement and optimization of AV systems in the context of robotic telepresence.

4.4.2 Technical Implementation

The telepresence aspect of the system involved the utilization of a 360° video camera to provide an immersive experience for remote users. The 360° video camera captured a panoramic view of the environment, allowing users to visualize and explore the surroundings in the VR application. This camera played a crucial role in enhancing the telepresence capabilities of the system, enabling remote users to feel as if they were physically present at the location where the robot was operating. Spot can be controlled either by a handheld tablet, (Figure 4-3), or a laptop computer using WASD controls. The type of control used depends on the type of data being collected. To collect data from Spot's stereo cameras, a laptop must be connected and controlling Spot. If data is not being collected from Spot's stereo cameras for a given traverse, the handheld tablet is used. To control Spot using WASD keyboard controls, the Boston Dynamics wasd.py script from the Spot SDK is used.

Two Python scripts were developed to collect images from Spot's five stereo cameras. The Spot SDK provided a sample script called get image.py, which takes arguments for camera names and image types and collects data of that type for that specific camera when called. This script was adapted to prevent having to pass in each camera type and each data type as arguments at every collection point. The adapted script takes in arguments of "front" or "back" to either collect all data types from either the two front and two side cameras, or to collect all data types from the back camera. Two argument options were required because the Spot image service was unable to handle all data types being requested from all five cameras at once.

A second Python script was written to efficiently gather data at each set of waypoints along a traverse. The goals of the script were to ensure that only one script call was necessary for each traverse, to allow users to uniquely label each waypoint, and to inform the user when image collection was complete at a waypoint. The script first prompts users to input where they would like to save the traverse data. Next, the

script prompts users to input the name of a waypoint. Once users enter a name, the script runs the adapted get image.py script twice, capturing stereo camera data for all cameras on Spot. Once this process is done, the script reports that data capture at the waypoint is complete, and waits for the input of the next waypoint name. A separate folder is created for each waypoint. Each folder contains the fifteen images captured at that waypoint, three for each of the five cameras [182].

The streaming process involved compressing and transmitting the recorded 360° video from the camera to a custom application running on the Oculus Quest 2 VR headset. To facilitate this streaming, an RTMP server running Nginx[4] was employed. The compressed video data was transmitted over a network connection from the camera to the VR application, allowing users wearing the Oculus Quest 2 headset to receive and experience the live 360° video feed with networked induced latency. The utilization of 360° video for telepresence purposes offered several advantages.

The immersive nature of the technology provided users with a sense of presence and an enhanced feeling of being physically present at the remote location. By offering a panoramic view of the environment, users could freely look around and explore their surroundings, mimicking the experience of being physically present. The effectiveness of 360° video for telepresence was assessed based on several factors, including the quality of the video feed, the level of immersion provided, and the overall user experience. The system's ability to transmit high-quality video in real-time without significant latency (40-100ms) was crucial for maintaining a seamless telepresence experience. Additionally, the extent to which users felt immersed and connected to the remote environment determined the success of the telepresence solution. Furthermore, offline applications were developed to replay cached LiDAR and 360° videos captured during the field experiments. These applications allowed for in-depth analysis and review of the recorded data, enabling researchers to revisit specific moments and examine the details captured by the LiDAR overlaid with the 360° video.

4.5 Doppelspot in Virtual Analog Environments

The development of the digital twin for Spot, also known as Doppelspot, presented unique challenges due to the complexity of this four-legged robot, which differs from the rover animated in Chapter 3. To accurately replicate Spot's movements and behaviors within the virtual environment, an investigation into procedural animation in Unity was conducted. Procedural animation involves defining the robot's joints, constraints, and parameters to govern its movement dynamically, ensuring precise control over its motions. By leveraging Blender, a powerful 3D computer graphics software, the team meticulously rigged the 3D model of Spot, enabling realistic animations and simulations that mirrored the actions of the physical robot.

[4] nginx.com

This highly detailed and accurate representation of Spot served as a visually appealing and interactive counterpart for users, elevating the telepresence experience and fostering a deeper understanding and engagement with Spot's interactions and movements within the Unity game engine. To achieve realistic leg animations for the digital twin of Spot, a C Sharp script was implemented in the Unity game engine. This custom implementation enables procedural animation techniques, through inverse kinematics (IK) simulation algorithms, to control the movements of the robot's legs. IK simulation algorithms are computational methods that calculate the joint angles required to position the end effector (in this case, the feet of the robot) at a desired target position. The IK simulation algorithms used in the C Sharp script aimed to achieve natural and stable leg movements for the digital twin of Spot. These algorithms considered factors such as joint limits, joint constraints, and environmental obstacles to calculate the optimal joint angles for each leg segment. By iteratively adjusting the joint angles based on the desired foot position, the IK simulation algorithms ensured that the feet of the digital twin followed a given trajectory while maintaining stability and adherence to physical constraints [183].

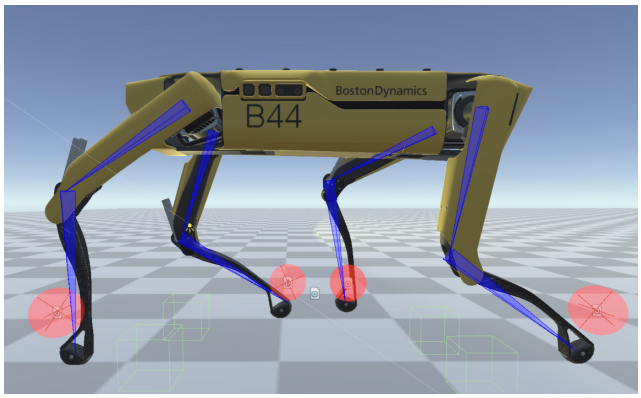

Figure 4-8: *The script utilizes inverse kinematics (IK) simulation algorithms to control the movements of the robot's legs, ensuring natural and stable leg motions. Challenges such as replicating the complex leg kinematics, handling environmental constraints, and integrating collision detection mechanisms were overcome to create lifelike leg movements in the digital environment.*

The implementation of the IK simulation algorithms for the procedural leg animation of the digital twin posed several challenges. One challenge was to accurately replicate the complex leg kinematics of the physical Spot robot in the digital environment. This required meticulous calibration and parameter tuning to ensure that the virtual legs moved in a manner consistent with the real robot. Another challenge involved handling environmental constraints and obstacles in real-time. The IK simulation algorithms needed to detect and respond to collisions or unexpected terrain conditions to maintain the stability and natural movement of the virtual legs. This necessitated the integration of collision detection and avoidance mechanisms within the script, allowing the digital twin to adapt its leg movements based on the encountered obstacles [184]. A complete reference of the procedural animation in pseudocode could be retrieved in Appendix B, Section B.2.

Doppelspot, the digital twin of the Spot robot, played a pivotal and dynamic role throughout the project, serving as a robust and lifelike virtual representation of the physical robot. Its immersive and interactive nature allowed users to closely observe

and analyze Spot's intricate movements, leg articulation, and adaptability to diverse environmental conditions. As a powerful visualization tool, Doppelspot provided a safe and controlled virtual environment for users to experiment, simulate, and fine-tune their teleoperation techniques and control strategies. This capability proved particularly valuable for remote operation training of quadruped robots in challenging and remote environments. By replicating Spot's behaviors and responses within the 3D environment, users could test various scenarios and operational concepts before real-world deployment, aiding in the development and refinement of efficient and effective teleoperation strategies.

It's worth noting that the testing space analog environment for Doppelspot was synthesized from real-world data collected during an analog mission in Svalbard, Norway. Chapter 5 provides a detailed account of this process. The integration of authentic environmental data into Doppelspot's virtual world, (Figure 4-9), enhanced its realism and provided users with a comprehensive and accurate simulation of remote terrains. This integration allowed for more realistic training scenarios, enabling users to tackle the challenges of operating in remote and unpredictable terrains, pushing the boundaries of robotic telepresence exploration in space analog environments.

Figure 4-9: *Images of the simulated Spot as it navigates a virtual analog site. The terrain is generated using a blend of high resolution aerial imaging based photogrammetry as well as other high resolution 3D assets, providing an immersive and realistic simulation. For a more detailed exploration of the terrain generation process, refer to the comprehensive insights outlined in Chapter 5.*

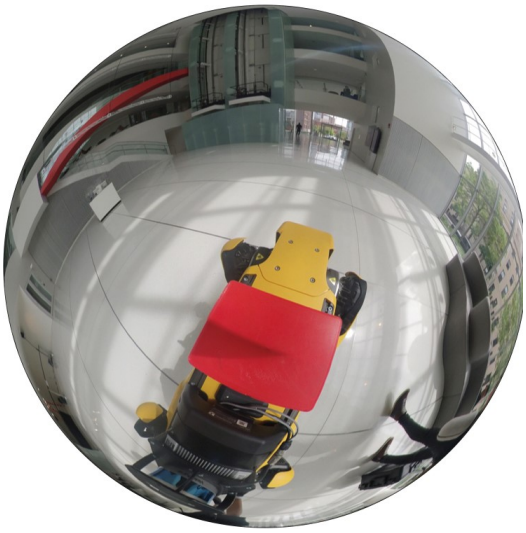

Figure 4-10: *A 360° photo captured by the Insta360 ONE X VR camera mounted on Spot that offers a panoramic perspective. With its wide 360° horizontal FoV and 180° vertical FoV, this image provides a comprehensive and immersive view of Spot's surroundings at the E14 Building of the MIT Media Lab.*

4.6 Conclusion

In summary, this chapter introduced two projects centered around robotic telepresence that make use of a versatile quadruped robot, Spot, By Boston Dynamics. The first project involves the development of a VR application with the capability to stream and replay 360° videos mixed with LiDAR data collected from a space analog environment site, drawing parallels with Quadrasense's approach [152], which integrated immersive VR cameras with unmanned aerial vehicles (UAVs) and an application designed around a real-time strategy (RTS) interface within the Unity game engine. Following a thorough evaluation of various camera options, the Insta360 ONE X camera, (depicted in Figure 4-10), and the Velodyne VLP-16 LiDAR were chosen as the optimal devices for creating this immersive experience, although many other devices were tested. Through an augmentation of the Doppelbot concept, Doppelspot provides users with an exceptional telepresence experience, empowering them to virtually explore challenging environments in an interactive and lifelike manner.

The successful implementation of this teleoperation system highlights a potential use case for robotic telepresence in scenarios unaffected by communication delays. For example, astronauts in orbit or space habitats could partake in virtual extra-vehicular activities through Doppelspot-like interfaces, capitalizing on its capabilities for exploration and research. In contrast to Chapter 3, which concentrated on employing virtual environments to simulate time delays in immersive visualization as a strategy to address remote navigation delays. Doppelspot's integration into telepresence systems introduces fresh opportunities for remote operation and data collection in demand-

ing environments. This integration presents novel avenues for remote operation and data collection in challenging contexts, as evidenced by the tests conducted in space analog environments. Throughout this chapter, the core capabilities of Doppelspot were underscored, illuminating its remarkable capacity to navigate outdoor terrains. Augmented with the Spot Enhanced Autonomy Payload (EAP) and Velodyne LiDAR, Spot emerges as an advanced mobile sensing platform, enhancing navigation, perception, and data collection in otherwise inaccessible locations.

This VR application could also be termed as an Augmented Virtuality (AV) experience. Presenting this application through this taxonomy highlights the role of enhancing the telepresence experience by seamlessly blending the real analog environment captured by the 360° video camera with LiDAR data. This fusion occurred within a Unity application, where the analog environment was streamed alongside LiDAR information. The LiDAR data was visualized with a color code to indicate depth and distances. Virtual overlays and annotations furnished real-time and cached access to data and visualizations, all superimposed on the live video feed. This approach effectively bridged the gap between the physical and virtual realms, facilitating simultaneous interaction with both domains. In parallel, the creation of a detailed 3D rigged model of Spot within a virtual space analog environment, also developed with the Unity game engine, was explored. Special emphasis is placed on the implementation of procedural leg animation through the utilization of inverse kinematics simulation algorithms built into Unity's custom rigging system. This essentially creates a virtual Spot, a "Doppelspot", that can be imported into various virtual environments, such as those synthesized from real-world data. Notably, this analog site was previously captured during a separate analog mission detailed in Chapter 5.

Looking ahead, future work for Doppelspot entails anticipated improvements to its payloads and telepresence capabilities. Advancements in 3D scanning technologies and communication present the potential for enhanced data reconstruction and alternative perspectives, which could potentially replace our current 360° based telepresence. Additionally, the integration of procedural animation into this type of telepresence system could further enhance realism and interactivity, providing users with a more immersive experience. Chapter 6 of this thesis further explores the use of commercial advanced depth cameras adapted for space environments, into robotic platforms like these, focusing on low bandwidth 3D transmission and reconstruction.

Chapter 5

Synthesizing Analog Environments

Virtual Space Analog Environments Sourced from Real-world Information Harvested within Human-Robot Operations.

5.1 Introduction

5.1.1 A Brief Overview of this Chapter

The study of terrestrial analog sites, (Figure 5-1), has become integral to scientific space exploration. These sites simulate geological, environmental, sometimes even biological conditions that resemble those found on celestial bodies [185], aiding researchers and astronauts to practice techniques and approaches on Earth's geological features that can be extrapolated to other planets. Geologists have employed a range of tools and methodologies to investigate Earth's geological processes for centuries [186]. The combination of traditional fieldwork and modern technologies has greatly advanced understanding of Earth's geological history. In this chapter, the application of 3D reconstruction and data visualization in virtual analog environments [187] is explored with specific attention to a Martian analog site situated in Svalbard, Norway. Found within the Arctic circle between Greenland and Norway, Svalbard features landscapes that closely mirror Martian terrain, making it an optimal location for geological study and research.

This chapter explores the methodologies for the synthesis [85] of virtual environments in space analog settings, using techniques like photogrammetry with high-resolution textures and 3D assets, as well as Unity's High-Definition Render Pipeline (HDRP). Photogrammetry, a process extracting three-dimensional data from two-dimensional images or videos [188], aids in constructing realistic models of real-world settings. High-resolution textures, characterized by a multitude of pixels, contribute to enhanced realism. The integration of 3D materials is pivotal for defining object appearances, encompassing lifelike textures, lighting, and reflections. Accompanying these, 3D assets—digital models of objects—complement the process, enriching the virtual environment's resemblance to Svalbard.

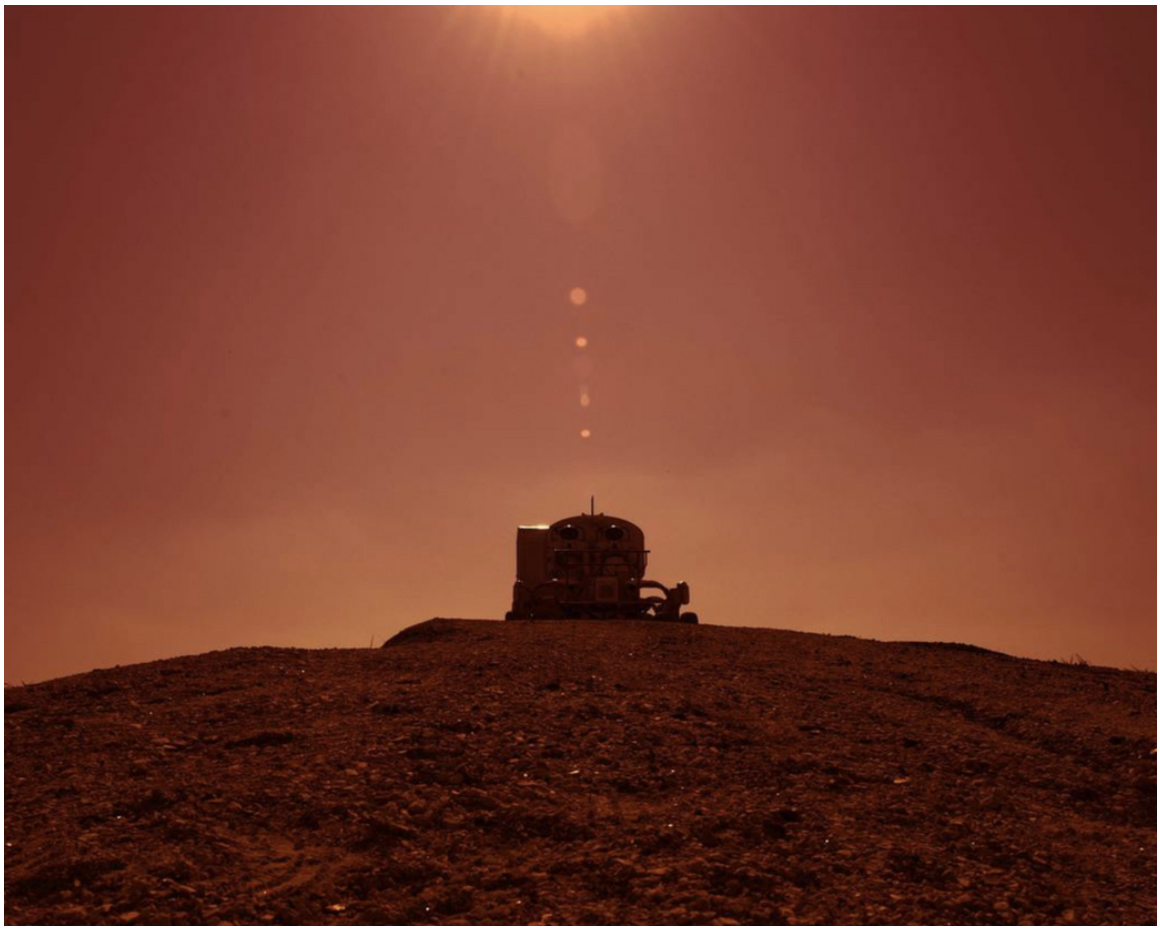

Figure 5-1: *This image captures NASA's Lunar Electric Rover ascending a hill at Black Point Lava Flow in Arizona during the 2008 Desert RATS - Research and Technology Studies. The site remarkably resembles the Martian landscape, allowing scientists to simulate extraterrestrial conditions and test rover capabilities in preparation for future missions. Image courtesy: NASA/Regan Geeseman.*

In this chapter, a comprehensive list of both hardware, (Section 5.3.1), and software, (Section 5.3.2), were used in the implementation, data collection, and data processing of this research. A wide range of cameras and sensor technologies have also been adapted to serve the purposes of geological exploration and documentation.

In addition to terrestrial imaging, aerial photography was obtained using a DJI Phantom 4 Pro V2 drone, making it possible to generate point-clouds (.ply) meshes with high resolution textures of various terrains. Environmental sensor nodes capable of measuring the temperature, humidity, illuminance and barometric pressure, among other things, were deployed at three different sites, operating for several hours.

Figure 5-2: *Satellite image of sites near Longyearbyen, Svalbard, Norway, selected by fellow RA and collaborator Cody Paige from the MIT RESOURCE on the Space Exploration Initiative 2022 analog mission to the arctic. Site 1 is a river bed at the base of a glacial valley, site 2 is a permafrost feature, and site 4 is a recently exposed glacial moraine. No data was collected at Site 3. Image courtesy: Norwegian Polar Institute.*

The study sites in Svalbard were selected based on their distinct geological features, including a glacially carved valley with a braided stream, a permafrost feature [189], and a recently exposed glacial moraine. Virtual reality (VR) might hold the potential to enhance analysis and decision-making processes for geological data in exploration missions. By providing naturalistic visualization tools that enable multiple team members to analyze, discuss, and interpret data, cross-disciplinary communication can be improved, real-time decision-making processes can be enhanced, task loads can be reduced, and flexibility in temporal and spatial planning can be achieved.

Although VR has gained traction as a tool for space applications, including Mars exploration and astronaut training [68], the specific benefits and operational contexts in which VR provides the most value are yet to be defined. These questions need to

be answered before investing in the development of VR as an operational tool. This study focuses on the use of VR for geological analysis of lunar and planetary surfaces, utilizing Svalbard, Norway, as an environmental analog [98, 190].

In particular, this chapter explores the application of 3D reconstruction and data visualization in virtual analogs, (Section 5.6), focusing on the Svalbard site as a Martian analog. By leveraging drone imagery, environmental data, and advanced rendering techniques, accurate 3D models and digital twins of the analog environments were created.

Both Desktop-based and VR application running on the HTC Vive were developed based on these reconstructions, provide users with an immersive and interactive exploration experience that facilitate geological study and research. The comparative analysis between the two applications aimed to assess the effectiveness of VR as a tool of study for space analog environments and geological surface exploration.

5.1.2 Analog Sites Selection in Svalbard, Norway

Svalbard's resemblance to Mars stems from its limited plant life and extensive exposure of rock formations [191]. These factors make it easier to discern the geologic history of the region, mirroring the conditions observed on Mars. The field sites, (Figure 5-2), in Svalbard that were explored during this study suggest some geological resemblance.

As mentioned, Svalbard offers an exceptionally intriguing location for geological study due to the preservation and exposure of its long geological history. The rock formations in Svalbard reflect mountain building processes during the Early Paleozoic, overlain by sedimentary deposits and layers that span millions of years. Glacial activity has sculpted the terrain, leaving behind sharp mountains, U-shaped valleys, and fjords that beautifully showcase the region's geological history [192]. The absence of vegetation, lack of soil, and rugged topography provide an excellent opportunity to study Earth's history and simulate conditions found on planetary surfaces [193]. Svalbard's other environmental advantages, such as a low azimuth sun angle, minimal vegetation, low temperature, and desert-like conditions with low humidity and precipitation, make it a suitable analog for Martian environments [194].

5.2 Operational Geology In a Virtual Environment

This chapter presents an innovative methodology for operational geology studies conducted within a virtual environment, designed in collaboration with Cody Paige from the MIT RESOURCE team [195]. The Operational Geology In a Virtual Environment (OGIVE) application was specifically tailored for the usage as space research through analog terrains on Earth [196]. By seamlessly integrating advanced 3D reconstruction techniques such as photogrammetry and RGBD-based reconstruction with cutting-

Figure 5-3: *This image showcases one of the synthesized virtual environments of the OGIVE application, blending cutting-edge techniques like photogrammetry and advanced shaders. Through a fusion of technologies, this chapter explores the creation of a realistic and immersive digital landscape driven by real-world data.*

edge game engine technology, digital twins of analog sites have been developed to accurately replicate the geological properties of these environments. Leveraging the immersive capabilities of Virtual Reality (VR), a comprehensive suite of tools has been designed to empower geologists and virtual explorers in conducting virtual geological surveys on distant planets. In order to evaluate the effectiveness of this approach, an extensive user study engaged participants of varying geological expertise in our Svalbard *world*, tasking them with exploring the virtual landscapes and identifying noteworthy geological features. The findings revealed the efficacy of this approach in aiding participants in identifying these features, effectively highlighting the utility of VR as a valuable tool for geologists to navigate and explore distant terrains. This research showcases the vast potential of digital twins and virtual environments in fields of space exploration and telepresence, while also providing profound insights into the application of these technologies for geological investigations.

The data generated for the OGIVE project originated from the Svalbard analog site expeditions conducted in the summer of 2022 by the MIT Media Lab Space Exploration Initiative (SEI)[1]. The analog mission to Svalbard focuses specifically on geological analysis in Martian analog environments to synthesise a virtual analog environment that was used as a base platform to conduct research on virtual operational geology [197]. By incorporating high-resolution depth data, aerial photography, and environmental data, the Svalbard project aims to assess scientists' abilities to analyze and interpret the local geology using VR compared to traditional desktop applications. While Doppellab and Doppelmarsh primarily explored the emergent functionality of digital twins and their impact on telepresence, the Svalbard project extends this exploration to the field of geology and planetary exploration. It leverages VR as a tool to enhance scientific analysis, offering immersive documentation processes in the field of virtual geology and space exploration [98, 190].

[1] media.mit.edu/groups/space-exploration

5.3 Data Collection in Svalbard

The study employed comprehensive data collection methodologies and hardware in Svalbard analog environments. Jessica Todd conducted aerial photography using a DJI Phantom 4 Pro V2 drone in collaboration with National Geographic. Environmental data collection and terrestrial imaging were also conducted. Cody Paige operated a Rover Mini augmented with the Microsoft Azure Kinect camera, and also deployed sensor nodes and selected the geological sites. The node is built of the Arduino MKR WiFi 1010 augmented with the MKR ENV shield offering an array of environmental sensors with a custom 3D-printed weatherproof enclosure designed by Ferrous Ward. These were the key components of the data collection process.

Figure 5-4: *Collaboration between terrestrial and aerial unmanned vehicles for data collection purposes. The Microsoft Azure Kinect is integrated as a custom payload atop the Rover Mini body. This dynamic setup, detailed in Chapter 3, empowers the rover with depth data collection capabilities. As the Rover Mini traverses the field sites, the Azure Kinect captures high resolution ground-view scenes for precise 3D reconstruction, while the DJI Phantom drone soars above, to capture larger terrains. Image Courtesy of Maggie Coblentz.*

5.3.1 Hardware Apparatus for Data Collection

Microsoft Azure Kinect Rover Payload:

The Microsoft Azure Kinect is a stand alone commercial product designed and developed by Microsoft after the successful and ubiquitous use of the XBox Kinect module in various fields, notably the fields of Human Computer Interaction (HCI) and robotics [198]. This special camera is equipped with a 12 megapixel RGB module supplemented with a 1 megapixel-depth module. It captures depth maps by measur-

ing the time it takes for near-IR (NIR) light to return to the camera sensor, along with recording the amount of NIR light reflected from the scene, (Section 6.1.2). The camera provided high-resolution depth data with different depth-modes. The data collected was stored as .mkv video files, which were then processed using the Open3D library [199] to generate 3D reconstructions (.ply files) [200]. These reconstructions were subsequently stitched together in the Unity game engine to create a 3D mesh of the field site for virtual environment rendering. A comprehensive illustration of the Microsoft Azure Kinect camera is expanded on in Chapter 6, Section 6.1.2.

The Microsoft Azure Kinect was integrated within a Doppelbot, (Figure 5-4), as a custom payload, which was positioned on top of a 3D printed extension of the Rover Mini body; refer to Chapter 3, (Section 3.1.2), for an in depth description of the Rover Mini by Rover Robotics. This setup facilitated the collection of data for purposes of 3D reconstruction using the camera while moving across the field sites.

Figure 5-5: *Image A - Arduino environmental sensor in custom hard-shell case. Image B and C- C. Paige collecting field notes, C. Paige collecting data using the Microsoft Azure Kinect, C. Paige, S. Image D - Auffinger and J. Todd preparing a mini rover with the Microsoft Azure Kinect mounted on a custom payload tower for data collection. Image E - C. Paige and J. Todd collecting data using a Velodyne puck on a custom 360° rotating mount (collection for a separate project, Ward et al., at ICES 2023) [190].*

Azure Kinect Development Kit (DK):

Microsoft Azure Kinect DK is a developer kit featuring advanced AI sensors that incorporate sophisticated computer vision and speech models. The Azure Kinect DK includes a depth sensor, a spatial microphone array with a video camera, and an orientation sensor, (Figure 6-3). This compact and versatile device offers multiple modes, options, and software development kits (SDKs) to support various applications, check Chapter 6 for a comprehensive list of capabilities of the Microsoft Azure Kinect device as well as its development kit [151].

DJI Phantom 4 Pro V2 Drone:

Aerial photography was conducted using a DJI Phantom 4 Pro V2 drone. The drone, a commercial off-the-shelf platform commonly used in videography and drone piloting, was operated using an iPad app by Jessica Todd from the MIT RESOURCE team in collaboration with National Geographic. The drone captured high-resolution aerial images by following preprogrammed raster, (Figure 5-12), patterns designed to cover the entire field site [201].

Environmental Sensor Node:

The Arduino MKR WIFI 1010[2] board, enhanced by the Arduino MKR Environmental Shield[3] rev2 (see Appendix A for details), forms an integrated environmental sensor suite. This comprehensive suite was designed to capture diverse environmental parameters, including color, sound, motion, temperature, humidity, pressure, light, and UV levels (see Appendix C.3 for more information). These environmental readings were stored as .CSV files, facilitating subsequent calibration and analysis. The incorporated shield conveniently accommodates a microSD card, enabling local storage of the acquired data. Furthermore, the collected data was enriched with geolocation coordinates of deployment, as well as timestamps indicating operation and data capture times. To provide a glimpse into the collected data's richness, sample plots (refer to Figures 5-7, 5-8, and 5-9) showcase data from distinct nodes and various sites over the operational days. For a comprehensive rundown of technical specifications regarding this sensor node, please refer to Appendix B Table B.4.

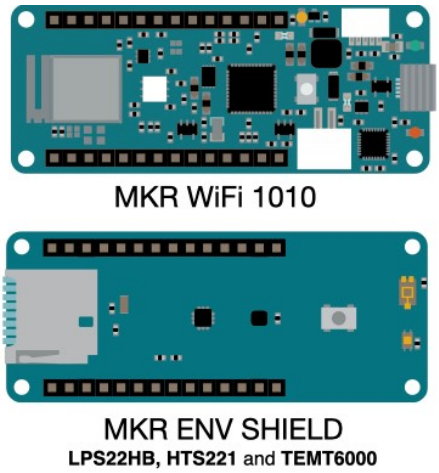

Figure 5-6: *The Arduino MKR WIFI 1010 board, enhanced with the Arduino MKR Environmental Shield rev2, provides a comprehensive sensor package. This setup captured crucial environmental data, including color, sound, motion, temperature, humidity, pressure, light, and UV levels. It Incorporated modules such as ST LPS22HB for atmospheric pressure, ST HTS221 for temperature and humidity, and VISHAY TEMT6000 for ambient light. Image courtesy of Arduino.cc*

[2]docs.arduino.cc/hardware/mkr-wifi-1010
[3]docs.arduino.cc/hardware/mkr-env-shield

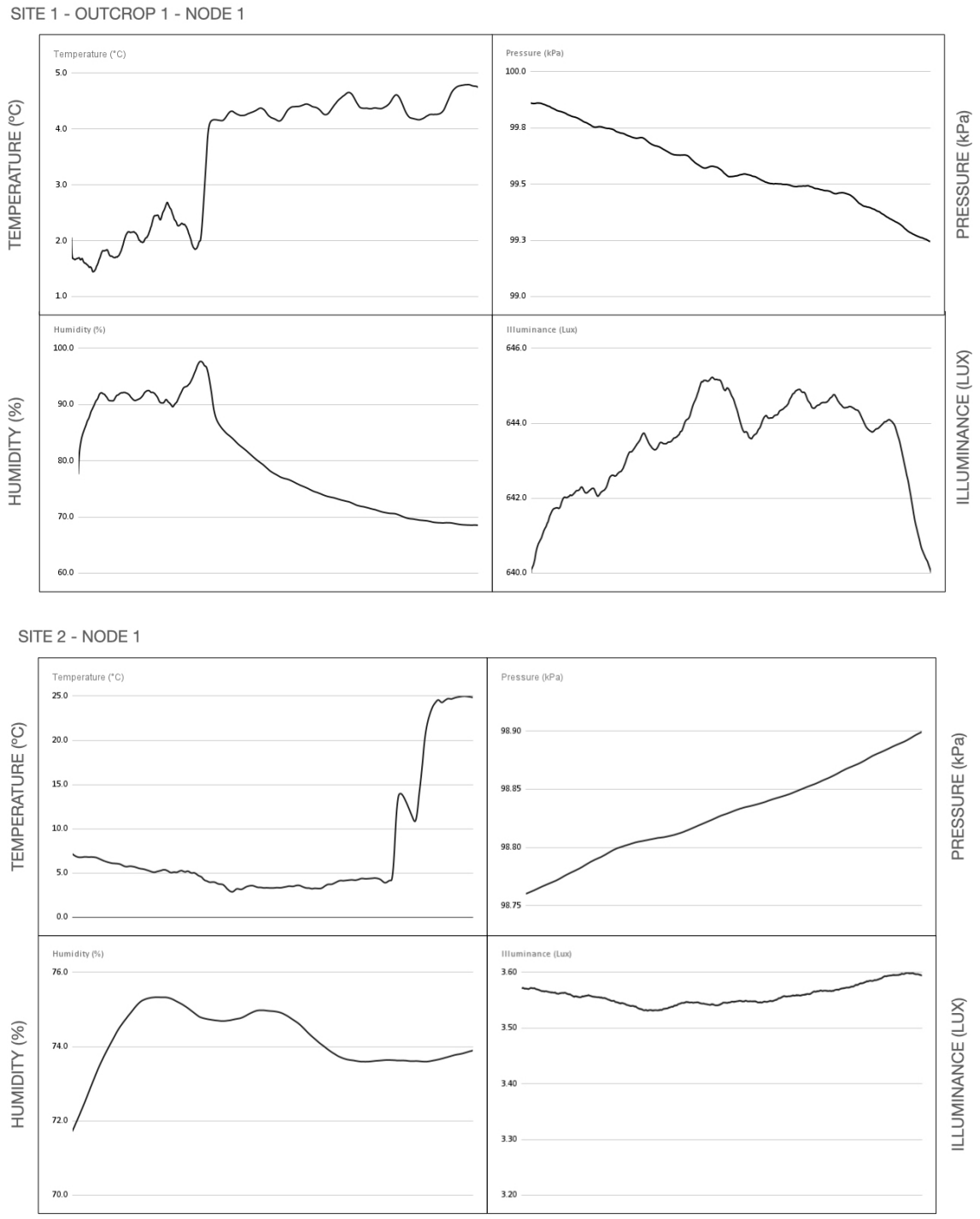

Figure 5-7: *These graphs show calibrated environmental sensor data from Node 1 on Site 1, Outcrop 1 (78.165984, 15.989823) for a period of 5 hours and 56 minutes and Node 1 on Site 2 (78.213312, 15.747577). It is important to note that the sensor Node 1 on Site 2 enclosure was closed, thus blocking the light sensors, as observed by the low amount of illuminance recorded.*

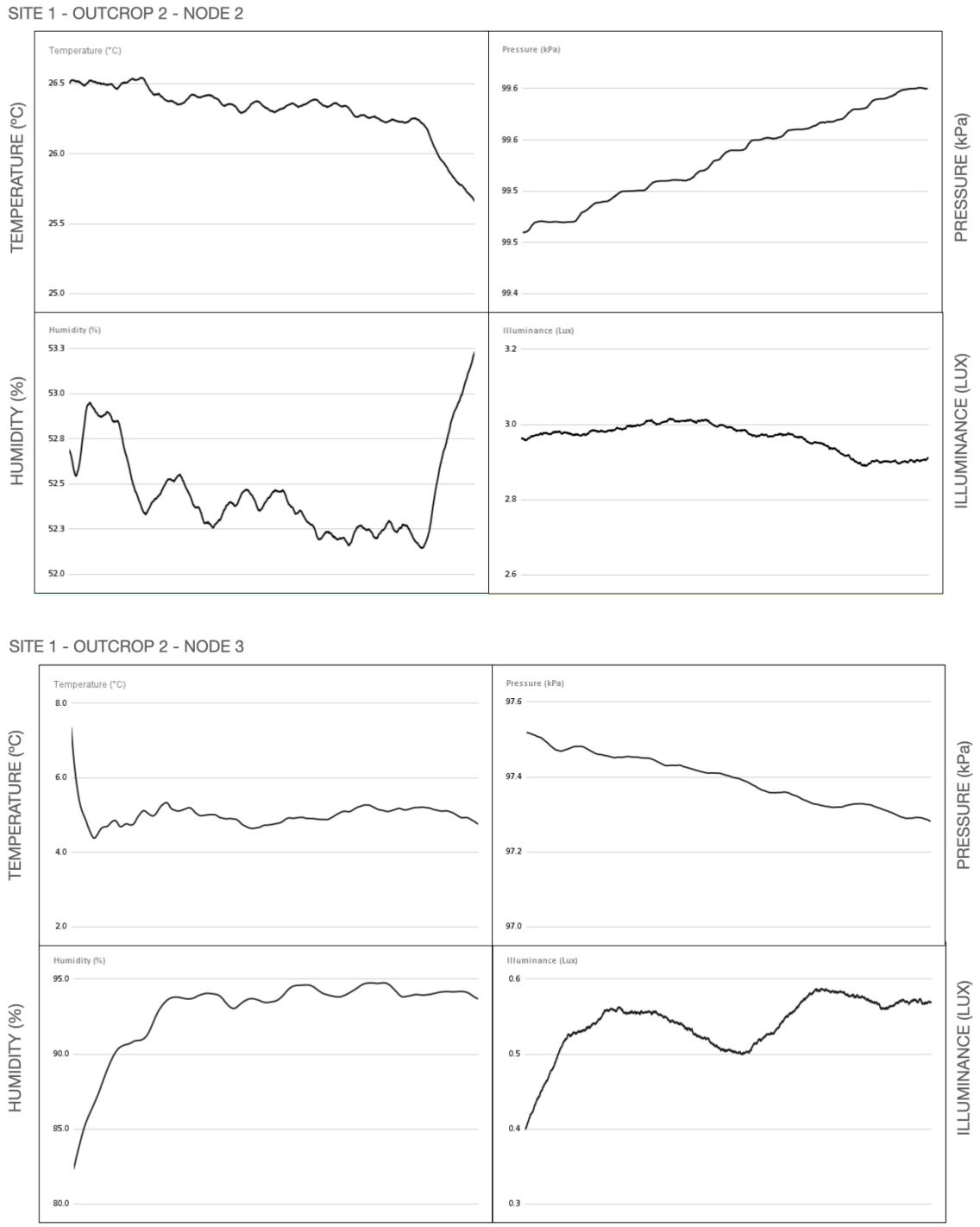

Figure 5-8: *These graphs show calibrated environmental sensor data from Node 2 (78.165974, 15.988111) and Node 3 (78.165081, 15.987596) on Site 1, Outcrop 2 and for a period of 7 hours and 42 minutes. It is important to note that both nodes' enclosures were closed, thus blocking the light sensors, as observed by the low amount of illuminance recorded.*

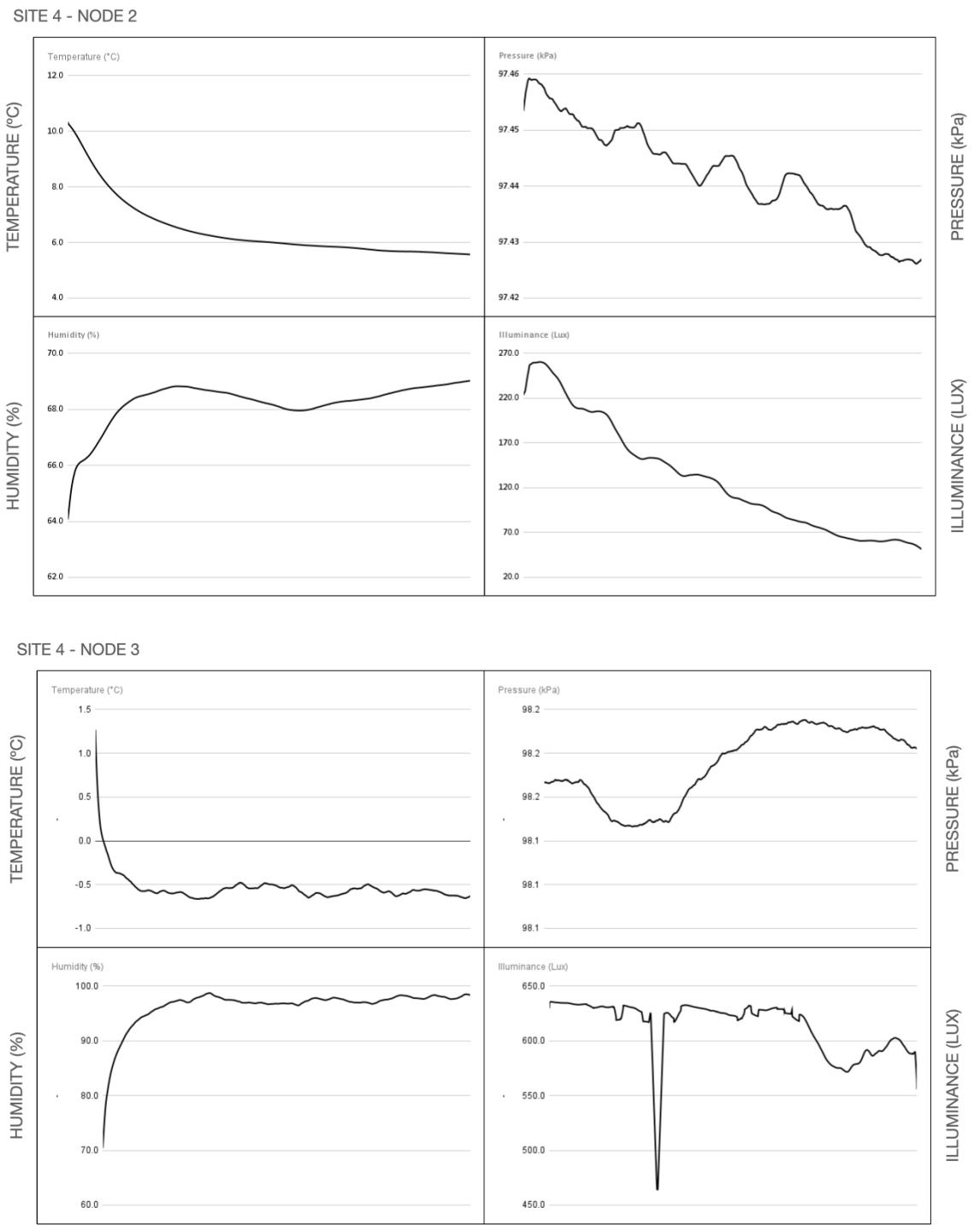

Figure 5-9: *These graphs show calibrated environmental sensor data from Node 2 (78.194296, 15.544605) and Node 3 (78,200254, 15.573699) on Site 4 for a period of 2 hours, and 03 minutes and 8 hours and 22 min, respectively.*

5.3.2 Software Apparatus for Data Collection

Agisoft Metashape:

Agisoft Metashape is a versatile software toolkit renowned for its capability to conduct photogrammetric processing of digital images, ultimately yielding three-dimensional spatial data. Its usage spans across an array of domains, including but not limited to geographic information systems (GIS), the preservation of cultural heritage artifacts, the production of visual effects in the film industry, and the derivation of indirect measurements for objects at diverse scales. Notably, this tool has found its application in numerous projects focusing on aerial 3D reconstruction, showcasing its adaptability and effectiveness in a variety of contexts where accurate and detailed spatial data is essential [202].

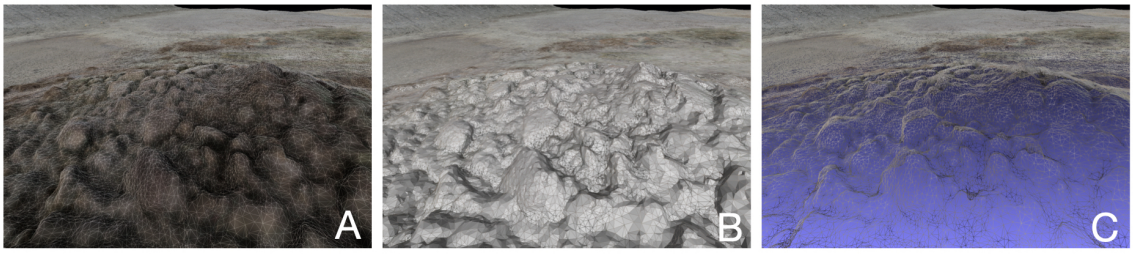

Figure 5-10: *3D reconstruction of a site featuring a pingo, a periglacial landform sculpted by the forces of colder climates. These three images, viewed in MeshLab, provide a closer look at the textured terrain. Each rendering offers a different overlay after precise model processing to optimize mesh face counts through decimation filters.*

MeshLab:

MeshLab, (Figure 5-10), is a specialized software system designed for processing and manipulating 3D meshes. Unlike general-purpose modeling software, MeshLab focuses on the management and refinement of large, unstructured mesh models. This software provides a comprehensive suite of tools tailored to tasks such as editing, cleaning, inspecting, rendering, and converting these mesh structures [203, 204]. Its unique capabilities make it an essential resource for professionals and researchers across various domains, including computer graphics, archaeology, and biology, who require powerful tools to effectively handle, enhance, convert, process, compress and analyze complex 3D mesh data.

Open3D:

Open3D is an open-source library that provides a comprehensive platform for developing software focused on processing 3D data. It offers a well-curated selection of data structures and algorithms available in both C++ and Python, enabling a wide spectrum of 3D data manipulation tasks. The library's backend is optimized for efficient parallel processing, ensuring smooth performance [199]. Open3D is not only

prominent in research projects but is also actively deployed in cloud environments. This versatility underscores its relevance in contemporary computing contexts. This library stands as an invaluable resource for researchers, developers, and professionals working with 3D data, providing a robust toolkit for various applications, and was extensively applied to generate 3D reconstructions of RGBD cameras throughout this research, specifically in Chapter 5 and 6 [205, 206].

Tablet Application - Field Notes:

Cody Paige from the MIT RESOURCE team, also trained as a field geologist, collected field notes using a specialized tablet application, custom-developed by the University of Western Ontario. The application was designed to streamline the data collection process during fieldwork by enabling the recording of annotated images, textual field notes, voice recordings, and mapping across multiple locations. By integrating these various data types into one platform, the application enhanced the efficiency of recording observations, detailed site characteristics, and distinctive features. The ability to collect such metadata was crucial, as it not only enabled a good documentation of the experiments but also allowed for a systematic collation and export of the collected information. This comprehensive approach was instrumental in developing the virtual scenes, setting the stage for subsequent in-depth analysis, and ensuring the accuracy and realism of the virtual representations.

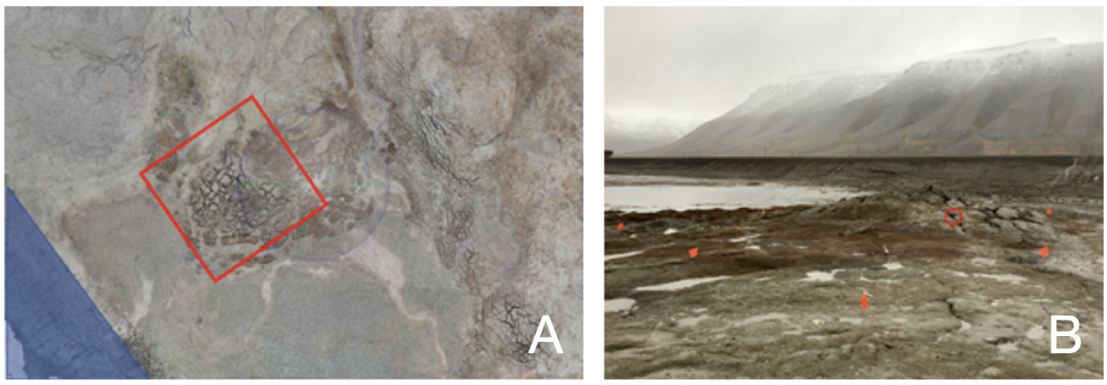

Figure 5-11: *Image A is of Site 2 - Adventdalen Valley, highlighting the permafrost feature captured by aerial imaging. Image B - a closer look at the site location with a (10 x 10 square meters) highlighted area of study with orange markers that were also used to calibrate the scale of the virtual environment.*

5.4 Methodology

Each field site was demarcated using orange flags to establish a (3.3 x 3.3 square meters) grid within a (10 x 10 square meters) area, (as depicted in Figure 5-11-B). The tablet application facilitated recording field notes encompassing essential details

like date, time, GPS coordinates, weather conditions, site identification, and distinguishing features. The Microsoft Azure Kinect camera mounted on the Rover Mini was utilized in open spaces, while it was manually maneuvered by Cody Paige across challenging terrains. This approach ensured the generation of a high-resolution, comprehensive 3D depiction of the field site to be captured, with high quality instruments. The drone executed depth data collection through a raster pattern flight, as operated by Jessica Todd, at a 15 meters height, covering the entire site. Additionally, data was gathered at a height of 25 meters to capture a less detailed view of the site.

By leveraging a range of data collection methods, encompassing drone imagery (Figure 5-11-A), readings from environmental sensors, and data captured by the Microsoft Azure Kinect camera, a comprehensive dataset was amassed. This dataset serves as a foundational resource for the synthesis, (Section 5.6), of virtual environments. The environmental sensor, housed in a custom weatherproof enclosure, was positioned at the north-east corner of the grid. It passively collected data for 2-3 hours while other data collection procedures were performed.

Subsequently, the sensor housing was closed, and data was collected overnight with a designated flag. The duration of data collection depended on factors like team availability and battery life, usually ranging from 12 to 15 hours. The sensor data encompassed temperature (℃), humidity (%), pressure (kPa), and illuminance (Lux), with readings taken every second. Calibration of the data was accomplished using archived weather data from Svalbard Airport, involving temperature at 2 meters above ground, relative humidity at 2 meters, mean sea level pressure, and total cloud cover. Graphical samples of the calibrated data are illustrated in Figures 5-7, 5-8, and 5-9. This collected data is integrated into virtual environments as done in previous work, visually representing environmental conditions using sensor nodes time-lapse [85, 69].

The Microsoft Azure Kinect camera was held approximately 1 meter above the ground, (as illustrated in Figure 5-5-E), and was manually guided through a designated pattern. This manual approach proved crucial at Site 2 due to its soft and wet ground conditions, unsuitable for tripod use. Similarly, at Site 3, high winds posed a challenge to tripod stability. Finally, drone-based depth data collection occurred at each site. Flags, with the exception of corner markers, were removed to minimize visual obstruction within the drone's field of view while retaining the site's external parameters. The drone executed a raster pattern flight at 15 meters above ground, capturing 1 meter by 1 meter sections across the entire site. Additionally, lower-resolution data was acquired at a 25 meter height, covering a (10 x 10 square meters) area surrounding each site.

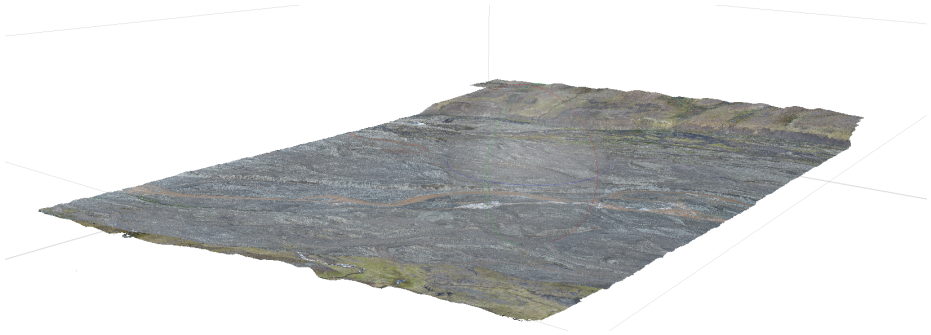

Figure 5-12: *Photogrammetry based 3D reconstruction using Metashape by Agisoft of the river bed located at Site 1. Image courtesy of Jessica Todd & National Geographic.*

5.5 Data Processing and 3D Reconstruction

Photogrammetry techniques were employed to generate high-resolution 3D models of the terrain based on the aerial imagery captured by Jessica Tod, who also played a key role in utilizing Agisoft Metashape for this purpose, in collaboration with National Geographic. The process involved matching features across images to create a dense point cloud, which accurately mapped the field site. From the point cloud data, a 3D mesh was generated and subsequently enhanced with image data to produce a 3D texture or orthomosaic. The outcome was a digital twin of the analog environment, capturing the geological features and ensuring a high level of accuracy. The 3D models were then optimized for performance, enabling real-time rendering and interaction within the Virtual Reality system. Agisoft Metashape, a renowned photogrammetry software, was instrumental in this process, allowing for the creation of 3D point clouds and meshes that represented the surface and shape of the terrain. The final 3D textures and orthomosaics were realized by skillfully combining the point cloud data with image data [201].

The GPS coordinates from the drone were used to provide an initial estimate of the camera locations. The extrinsic and intrinsic camera properties together with the detected and matched features were then used to exactly determine the location of each photo. Once a point cloud was generated, the point cloud data was used to generate a 3D mesh of the field site, or combined with image data to generate a 3D texture or orthomosaic. An example of this data for a field-site, by Jessica Todd is shown in Figure 5-12. The figure shows a 3D mesh of a field-site that was created using Agisoft Metashape [207]. The mesh is accurate and detailed, and it can be used for a variety of purposes, such as Virtual Reality and 3D printing. To create highly detailed and accurate 3D models of objects and features in the environments, 3D reconstruction techniques using the Azure Kinect camera and the Open3D library were employed.

These techniques can benefit a wide range of applications, including Virtual Reality, robotics, and computer-aided design. First, by leveraging the capabilities of the Azure Kinect camera and a modern library for 3D data processing provided by

Open3D [205], objects of geological interest were captured and reconstructed. On the other hand, the photogrammetry was performed to generate high-resolution 3D models of the overall terrain, (as described in Section 5.4).

The second step involved refining the 3D model to optimize its performance in a virtual environment. The initial 3D model, with its high polygon count, resulted in performance issues when integrated into the VR system. To address this, we used the quadratic decimation filter in MeshLab [208], an open-source 3D processing software. The quadratic decimation filter was applied iteratively to reduce the polygon count of the mesh while preserving its overall geometry and topology. This process optimized the model for real-time rendering and interaction within the game engine, ensuring a smooth and immersive experience for users in the virtual environment. The resulting digital twin of the analog environment not only maintained a high level of accuracy in terms of geological features but also proved to be efficient in terms of computational resources, thus providing an optimal balance between realism and performance in the VR system. To perform a 3D reconstruction with Open3D [206], specific commands have been employed, and can be found in Appendix B.4.

The quadric for a given plane (defined by equation $ax + by + cz + d = 0$)
is represented by the 4×4 symmetric matrix below:

$$Q = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

The decimation process iteratively collapses edges, repositions vertices, or removes vertices to reduce the overall complexity. The decision of which operation to perform is typically based on the error metric – the action that minimizes the quadric error is chosen [208]. The error for a given vertex $v = [x, y, z, 1]$ is then given by:

$$\text{Error}(v) = v^T \cdot Q \cdot v$$

5.6 Synthesizing Analog Environments

The photorealism and immersion of the virtual environment were further enhanced by using built-in features of the Unity 3D game engine [209]. Unity's extensive library of standard game assets was used to supplement the digital twin with realistic water, Skybox, and surrounding mountains, contributing to a more convincing and engaging virtual experience for users. These elements were carefully integrated into the scene to maintain the consistency and authenticity of the geological context. Moreover, Unity's High-Definition Render Pipeline (HDRP) and its volumetric post-processing stack were employed to achieve advanced rendering capabilities within the virtual environment [120].

5.6.1 Unity's High Definition Render Pipelines

The High Definition Render Pipeline (HDRP) is a rendering system employed by Unity that enables the creation of highly realistic and visually stunning graphics. In the context of painting virtual environments, HDRP provides a range of features and tools that contribute to the overall visual fidelity of the scenes. One of the key aspects of HDRP is its support for physically-based lighting. The HDRP light types use physical light units (PLUs) to ensure that the lighting in the scene replicates real-world light sources accurately. The Light component in HDRP includes properties such as Intensity and Temperature, which can be adjusted to match the characteristics of different light sources. By respecting the HDRP unit convention, where 1 Unity unit equals 1 meter, lights can behave realistically within the virtual environment. The HDRP light types include Directional, Spot, Point, Rectangle, and Tube lights, each offering various properties such as color temperature, colored cookies, and shadow mask support.

In addition to physically-based lighting, HDRP also supports a variety of other features that can be used to create realistic virtual environments. These include:

- Physically-based materials: HDRP materials are based on the physically-based rendering (PBR) model, which ensures that the appearance of materials is accurate and consistent with real-world materials.
- Global illumination: HDRP supports global illumination, which is a technique that simulates the way light bounces around a scene. This can be used to create more realistic lighting effects, such as indirect lighting and reflections.
- Post-processing effects: HDRP includes a variety of post-processing effects that can be used to further enhance the visual appearance of scenes. These include effects such as depth of field, bloom, and motion blur.

By using the features and tools provided by HDRP, artists can create highly realistic and visually stunning virtual environments. Rendering Layers are an important concept in the High-Definition Render Pipeline (HDRP) for controlling the interaction between lights, decals, and meshes. Rendering Layers, or Layer Masks, are used to specify which objects are illuminated by specific lights. This allows for precise control over the lighting in the scene, ensuring that only relevant objects receive illumination from specific lights. Additionally, Rendering Layers can be utilized in shadow map settings to decouple shadows from lighting, providing further control and customization options.

HDRP supports the use of IES Profiles, which describe the distribution of light from a light source based on the Illuminating Engineering Society's file format. IES Profiles are particularly useful for Point, Spot, and rectangular Area Lights, as they enable the accurate representation of real-world lighting characteristics. These profiles can be combined with light cookies and utilized for light map baking as well, adding another layer of realism to the virtual environment. To enhance the visual quality and

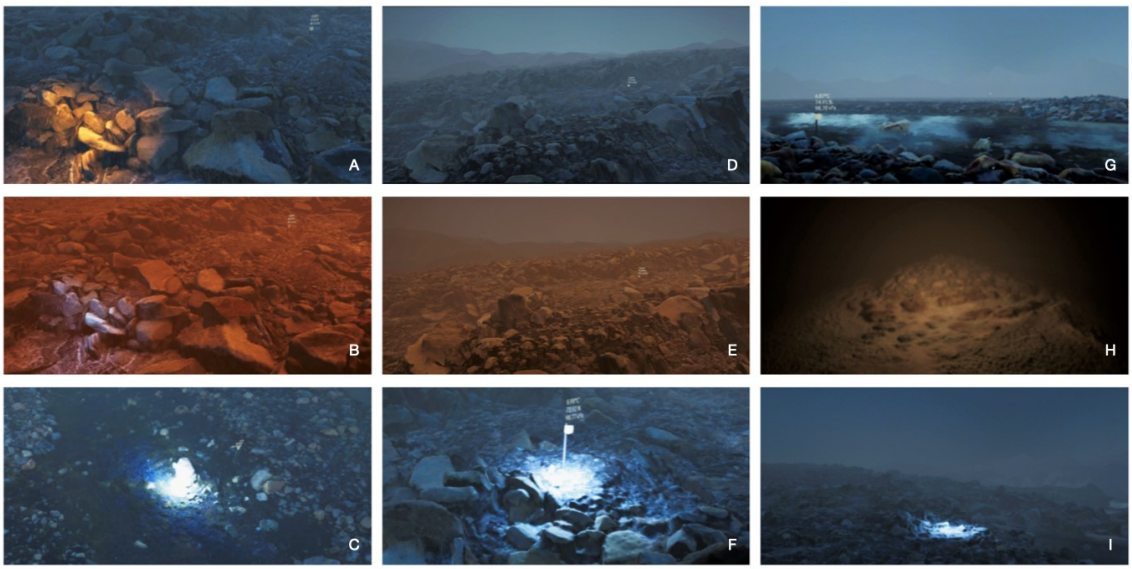

Figure 5-13: *A collage showcasing nine scenes from the three rendered sites in the Unity game engine, brought to life through a combination of photogrammetry and high-resolution assets using the Unity High Definition Render Pipeline (HDRP). Image A unveils the spatial 3D filter tool, offering a glimpse into the analog rendering of the virtual environments, whereas Image B depicts the analog version of that same site, resembling a Martian terrain. The glacier terrain and analog of Site 2 is seen in Images D and E respectively, while Image H highlights the permafrost-rich pingo site. Images G and C features The riverbed terrain of Site 1. Witness the sensor node deployed on Site 1 in Image F, complemented by the flashlight tool showcased in Image I, enabling precise focus on rocks and features while controlling the play of shadows and lights.*

realism of the scenes, HDRP offers various effects and features. One of these is the Lens Flare system, which provides customizable lens flare effects that can be applied to any GameObject in the scene. The Screen Space Lens Flare effect utilizes screen information and the Bloom texture to generate view-dependent flares, simulating bright spots and reflections. Reflection Probes in HDRP provide accurate cubemap reflections that consider surface smoothness, enabling realistic real-time reflections.

Planar reflection probes are also supported, allowing for the creation of effects such as shiny mirrors or wet floors. Screen-space techniques play a vital role in HDRP for rendering certain effects. Screen Space Reflection and Refraction techniques use the depth and color buffers of the screen to simulate accurate reflections and refractions. These techniques are particularly useful for transparent materials like windows or water. HDRP also includes Screen Space Ambient Occlusion, which approximates the intensity and position of ambient light on surfaces based on the lighting in the scene, adding depth and realism to the virtual environment. Screen Space Specular Occlusion can further improve the accuracy of specular occlusion by using a texture containing bent normal information corresponding to the light direction.

Adaptive Probe Volumes (APV) is a sophisticated feature in HDRP that automatically positions light probes according to the geometry density in the scene. APV offers per-pixel probe selection and lighting, volumetric light support, reduced light leaking, and settings for blending different lighting scenarios. This feature improves the overall lighting quality and efficiency in the virtual environment. Another notable feature in HDRP is the support for ray tracing. Ray tracing can be utilized as an alternative to certain screen space effects, shadowing techniques, and mesh rendering techniques. It provides more accurate rendering of effects such as ambient occlusion, contact shadows, global illumination, reflections, shadows, subsurface scattering, and more. Ray tracing in HDRP allows for high-fidelity rendering with realistic light interactions and accurate simulations of real-world phenomena.

Environmental effects are also an important part of the High-Definition Render Pipeline (HDRP). The sky can be configured within a Volume, allowing for dynamic changes in sky settings based on the Camera's position in the scene. HDRP offers various built-in sky types, including Gradient Sky, HDRI Sky, and Physically Based Sky, each providing different visual characteristics for the virtual environment. For example, a Gradient Sky can be used to create a simple sky with a single color gradient, while an HDRI Sky can be used to create a more realistic sky with a high-resolution image of a real-world location.

A Physically Based Sky can be used to create a sky that simulates the way light interacts with the atmosphere, creating a more realistic and immersive experience. Volumetric Clouds can be integrated into the scenes, allowing for the creation of realistic and interactive cloud formations that interact with the sky, sun, and fog. This can be used to create a more dynamic and realistic environment, as the clouds will move and change based on the time of day and the weather conditions. Additionally,

HDRP supports the setup of fog within a Volume, enabling the customization of fog settings and types based on the Camera's position. This can be used to create a more atmospheric and immersive experience, as the fog will appear thicker or thinner depending on the Camera's distance from the ground.

Post-processing is a critical component of achieving high-quality visuals in HDRP. HDRP provides its own dedicated post-processing implementation, which allows for the application of full-screen filters and effects to the Camera. Post-processing effects can significantly improve the visual appeal of scenes by providing capabilities such as tone mapping, color grading, exposure calculation, and anti-aliasing. HDRP offers multiple anti-aliasing methods, including Multisample Antialiasing (MSAA), Temporal Antialiasing (TAA), Subpixel Morphological Antialiasing (SMAA), and Fast Approximate Antialiasing (FXAA), allowing developers to choose the most appropriate technique based on performance and quality requirements.

The physically based camera system in HDRP ensures unified and realistic results by mimicking the behavior of real-world cameras. It allows for the configuration of exposure, depth of field, and other camera properties, resulting in visually accurate representations of the virtual environment. Additionally, HDRP provides options for custom post-processing and the implementation of custom passes. Custom post-processes can be integrated into the volume framework, allowing for the injection of custom shaders and scripts at specific points in the rendering pipeline. Custom passes enable the rendering of additional objects, overriding rendering properties, performing fullscreen passes, and accessing camera buffers such as depth, color, normal, and motion vectors. These features provide flexibility and customization options for developers to achieve specific visual effects and rendering behaviors.

5.6.2 Examination of the Level of Photorealism

The combination of high-quality photogrammetry, Unity's HDRP, and the Post Processing Stack contributes to achieving a high level of photorealism in the constructed virtual environment, (Figure 5-13). Photogrammetry plays a crucial role in capturing the real-world environment with great detail and accuracy. Aerial photogrammetry was compiled using images from multiple angles, allowing for the creation of high-resolution 3D models of the terrain. These models preserve the geological features and provide a realistic representation of the analog environment.

The captured imagery is then processed using Agisoft Metashape, a photogrammetric software. This software utilizes feature matching and dense point cloud generation techniques to accurately map the captured area. The resulting 3D models and point cloud data serve as the foundation for creating digital twins of the analog environments. Unity's HDRP plays a crucial role in rendering these digital twins with exceptional visual fidelity. HDRP's physically-based lighting system ensures that the virtual environment is illuminated realistically, replicating the behavior of real-world light sources. The HDRP light types, such as directional, spot, point, rectangle,

and tube lights, enable the simulation of various light sources accurately. By using physical light units and respecting the HDRP unit convention, the lighting within the virtual environment matches the characteristics of real-world lighting, further enhancing the photorealism. The Post Processing Stack is a post-processing pipeline that applies a variety of effects to the rendered image, such as tonemapping, bloom, depth of field, and motion blur. These effects can be used to further enhance the realism of the virtual environment. The combination of high-quality photogrammetry, Unity's HDRP, and the Post Processing Stack allows for the creation of virtual environments that are highly realistic and immersive.

Furthermore, HDRP's support for IES Profiles enables the accurate representation of light distribution from various light sources. These profiles can be combined with light cookies and used for light map baking, ensuring that the virtual environment accurately replicates the lighting conditions of the captured analog environment. The Post Processing Stack in Unity enhances the visual quality and realism of the virtual environment. Various post-processing effects can be applied, such as tone mapping, color grading, and exposure adjustment. These effects help to refine the lighting and color reproduction, resulting in a more realistic and visually appealing representation of the analog environment. Additionally, HDRP supports screen-space techniques, such as screen space reflection, ambient occlusion, and refraction, which further contribute to the overall photorealism. Screen space reflection accurately captures and renders reflections, while ambient occlusion adds depth and realism by approximating the intensity of ambient light on surfaces. Screen space refraction allows for the realistic rendering of transparent materials like windows or water, simulating the bending of light as it passes through these surfaces.

The combination of high-quality photogrammetry, Unity's HDRP, and the Post Processing Stack ensures that the virtual environment closely resembles the captured analog environment. The detailed 3D models generated from photogrammetry, combined with HDRP's realistic lighting and post-processing effects, create a visually stunning and immersive experience. The accurate representation of lighting conditions, the rendering of reflections and shadows, and the fine-tuning of colors and exposure all contribute to achieving a high level of photorealism in the virtual environment. Overall, through the careful integration of high-quality photogrammetry data, Unity's HDRP, and the Post Processing Stack, the constructed virtual environment exhibits a remarkable level of photorealism, allowing users to immerse themselves in a visually captivating and lifelike digital representation of the analog world.

Figure 5-14: *Sensor data are represented in the virtual environment on top of a virtual sensor node model, reminiscent of Brian Mayton's sensor node in the Doppelmarsh project, (Section 1.3.2). 3D Model of sensor node by Brian Mayton.*

5.6.3 Immersive Sensor Data Player

The review of sensor data replaying, (Figure 5-14), in the constructed virtual environment enhances the overall realism and functionality of the environment. The sensor data captured using the Arduino MKR WiFi 1010, (Section 5.3.1), augmented with its environmental shield is crucial for providing cached information about various environmental parameters such as light, temperature, humidity, and pressure. This information is replayed in the virtual environment to enhance the user experience. The captured sensor data is integrated into the virtual environment.

A user interface (UI) is overlaid spatially on top of a depiction of a sensor node within the virtual environment to visualize and present the sensor data. This UI allows users to access and interpret the recorded sensor data in a numerical manner. The replaying of sensor data within the virtual environment offers several advantages. Firstly, it provides a means to validate the accuracy and fidelity of the virtual environment by comparing the recorded sensor data with the virtual representation of the analog environment. This validation process ensures that the virtual environment accurately reflects the real-world conditions captured by the sensors. Secondly, it allows users to interact with the virtual environment in a more realistic manner by providing them with real-time feedback on the environmental conditions. This can be useful for training purposes or for simply exploring the virtual environment.

The replayed sensor data enhances the interactivity and functionality of the virtual environment. Users can monitor and analyze environmental parameters within the virtual environment by visualizing the sensor readings. This capability is particularly useful for research and analysis processes that require an understanding of real-world environmental conditions. Furthermore, the replayed sensor data contributes to a more immersive and dynamic user experience within the virtual environment.

By overlaying a UI that depicts the sensor node in a virtual environment, users can interact with and explore the sensor data, evaluating environmental conditions in different locations. This visualization enables the assessment of various scenarios and their impact, allowing users to understand the consequences of changes or interventions in the real environment. Such capability facilitates informed decision-making and exploration of potential outcomes in a controlled virtual setting. In space settings, such sensor valuing could analogously come from stand-alone sensors in the environment, sensors on rovers, remote sensory capability, etc.

5.6.4 Virtual Analog Toolkit

Virtual Reality experiences and applications offer a wide selection of tools that augment users' ability interact with the virtual analog environments. A taxonomy suggests to organize these modes of interaction into three categories: Explore, Experience, and Experiment, as proposed by S. W. Greenwald in "The Equipped Explorer: Virtual Reality as a Medium for Learning" [210], enable users to perform a variety of tasks within the virtual medium.

Three intuitive and user-friendly tools were developed and integrate with the OGIVE application to facilitate exploration and analysis within the virtual environment, The flashlight tool ,(Figure 5-13-C, 5-13-F, 5-13-I), illuminated and highlighted geological features of interest, thereby enabling users to examine detailed structures and formations more closely. Moreover, users have the option to change the functionality of this tool as a medium for visualizing different aspects of particular areas of the analog terrain, (Figure 5-18). This tool could also be utilized to render selected multi-spectral data, when available, in the virtual environment. The measurement tool, (Figure 5-16), provided users with accurate measurements of geological features, aiding in the assessment of dimensions and spatial relationships. Additionally, a filter tool, (Figure 5-17), was implemented, transforming the environment into a Martian-like landscape that simulated the challenges and conditions of conducting geological surveys on other planets.

These tools collectively enhanced the utility and applicability of the virtual exploration system, catering to a wide range of users interested in space research. The flashlight tool, (Figure 5-13-C, 5-13-F, 5-13-I), served as a means to illuminate and highlight specific areas of interest within the virtual environment. By controlling the shadows cast on geological features, users were able to draw attention to particular details and structures. The flashlight tool presented users with a range of light colors to choose from, (Figure 5-13-A, 5-13-B), allowing for customization and highlighting specific aspects of the environment. Acting as a virtual tunnel, the flashlight tool guided users through the environment, illuminating it to resemble the analog site, which in this case was Mars. This feature provided a visually immersive experience and allowed users to explore the virtual environment with greater on-demand clarity to highlight and study features of interest in the virtual environments.

Figure 5-15: *Two images highlighting the measurement tool integrated into the virtual environment, allowing users to explore, annotate, and extract precise measurements. In addition, the analog site filter removes certain high-resolution assets, such as water meshes, providing a direct comparison between the analog and real environments. Image A showcases the analog site, while Image B reveals the corresponding real site, offering a comprehensive understanding of the virtual representation and its real-world counterpart.*

The filter tool, (Figure 5-17, 5-13-E, 5-13-H), allowed users to select specific features to view within the virtual environment. This was useful for users who wanted to focus on a particular aspect of the environment, such as geological features or plant life. The filter tool could be used to isolate individual features or groups of features, providing users with a more detailed view of the environment.

Moreover, these intuitive, immersive, and user-friendly tools were developed to enhance user exploration and analysis within synthesized analog sites. These tools were designed with the intention to provide users with interactive capabilities and facilitate a deeper understanding of the virtual environments.

The measurement tool, (Figure 5-16), played a crucial role in enabling users to obtain accurate measurements of geological features within the virtual environment. By providing precise spatial information, this tool assisted users in assessing dimensions and spatial relationships between different elements. It allowed users to toggle between using the metric or imperial systems, catering to users with different measurement preferences. With a precision of two decimal points, the measurement tool facilitated the translation of real-world measurements into the virtual environment, aiding in building a better spatial intuition. Through Unity's Raycast object, the tool dynamically calculates distances by projecting an invisible ray from a designated start point to a specified end point in the virtual environment.

To ensure the development of an accurate measurement tool in the virtual environment, a careful calibration process was conducted for each scene. In the experiments methodology, (Section 5.4), flags were strategically placed within the environment and captured by aerial imaging. The distance between flags was accurately measured in the real world, establishing a reference for calibration. This calibration data was then utilized to fine-tune the measurement tool in the virtual environment, aligning

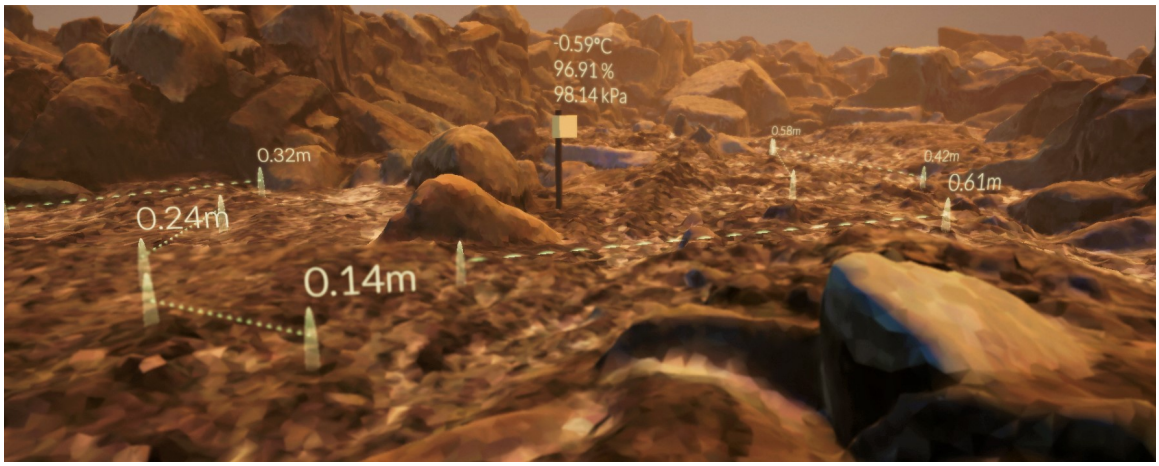

Figure 5-16: *This image showcases a range of tools designed to augment the virtual exploration experience with a virtual ruler, and replay cached sensor data.*

it with real-world measurements. By calibrating the tool in this manner, users could confidently rely on its accuracy and precision, enhancing the reliability of measurements taken within the virtual space.

The virtual lens tool, (Figure 5-15), was devised to alternate between analog and real environments, giving users diverse visual perspectives. In analog environment mode, the lighting conditions and scattering color were modified to simulate the look of Martian landscapes. This was accomplished by adjusting the color scheme based on images provided by NASA's Perseverance and Curiosity rovers. By switching to analog environment mode, users could experience the virtual environment as it would look on Mars, enhancing the authenticity and realism of the exploration experience. This tool provided users with the opportunity to explore the unique challenges and conditions of conducting geological surveys on other planets.

The user interface tools collectively enhanced the virtual exploration system's utility and applicability. They provided users with interactive features for investigating geological features, obtaining precise measurements, and experiencing the environment as it would appear on Mars. The tools catered to a wide range of users interested in space research, including researchers, scientists, and enthusiasts. By incorporating these tools into the user interface, the exploration and analysis capabilities of the digital twin environments were significantly augmented, allowing users to immerse themselves in a realistic and informative virtual experience.

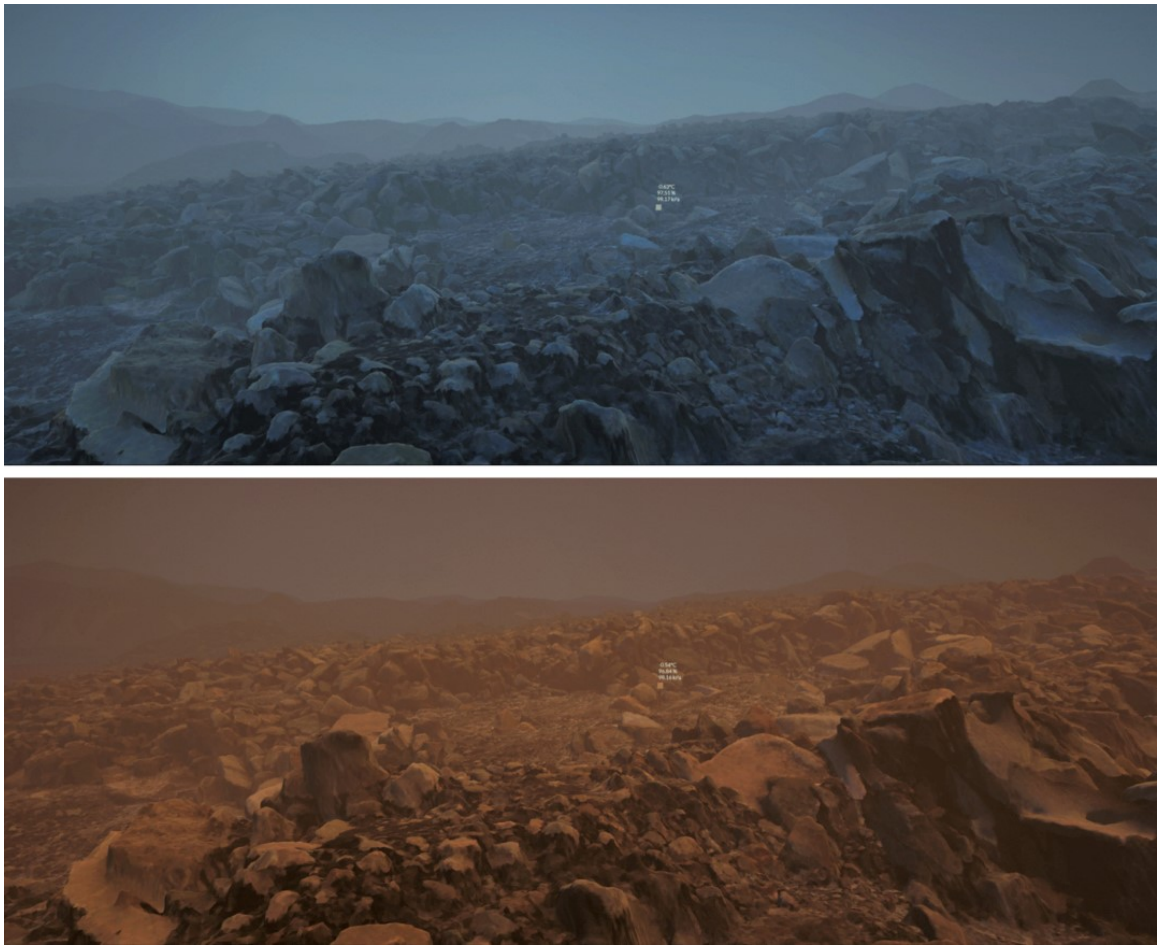

Figure 5-17: *"Earth to Mars" filter and vice-versa – Allowing users to switch back and forth from the analog / real site while maintaining the same camera perspective. Utilizing several image filters and an emulation of natural light diffusion to mimic the atmosphere on Martian landscapes through the metaphor of a virtual lens.*

Figure 5-18: *Analog spotlight tool rendering parts of the terrain as it appears in the analog version, rather than a full immersive filter. This visualization metaphor can also be extended to render different types of spatially-varying data superimposed on the terrain, such as multi-spectral data, providing a comprehensive understanding of the landscape's characteristics and enhancing spatial analysis capabilities.*

Figure 5-19: *The desktop's application navigation menu, featuring keyboard and mouse shortcuts as well as voice commands allowing a user to navigate the site, and teleport to areas of interest using natural language.*

To enhance user interaction and immersion within the virtual environment, voice commands were implemented using Unity's PhraseRecognizer capabilities, and accessed by the user from the navigation menu, (Figure 5-19). Three types of PhraseRecognizer were utilized: KeywordRecognizer, GrammarRecognizer, and DictationRecognizer. The KeywordRecognizer allowed users to speak specific phrases, such as "zoom in," "zoom out," "move left," "move right," "move up," "move down," and "jump to point of interest," triggering corresponding actions within the virtual environment. The GrammarRecognizer enabled the definition of a specific grammar using an SRGS file, providing structured voice commands for users. Additionally, the DictationRecognizer allowed users to freely speak, with their speech used for various purposes in the application.

To enable voice input, the Microphone capability was declared in the Unity project settings. Users could navigate the virtual environment by simply speaking commands, enhancing the user experience and enabling a more intuitive interaction. Furthermore, the words "analog" and "real" were utilized as voice commands to switch between the virtual analog site and the real environment, respectively. By incorporating voice commands, the virtual environment became more accessible and immersive, allowing users to explore and interact using natural language.

5.7 User Study

The study conducted a comparison between the spatial presence and immersion quality experienced in Virtual Reality (VR) and desktop applications. Participants explored the Svalbard field sites through both mediums, focusing on identifying key geological features and forming interpretations based on available facts, (as depicted in Figure 5-20). Both platforms provided access to datasets including high-resolution 3D imagery, environmental data, measurement tools, a Mars analog view, and geological fact sheets. Feedback was collected from the participants to gain additional insights into their experiences and preferences.

The research findings suggested that the VR applications present a more engaging sense of spatial presence and immersion compared to desktop applications. This perception may be influenced by the more interactive nature of the VR environment, or its resemblance to real-world experiences. Some participants also indicated that they felt more confident in making geological interpretations using the VR applications, though this was not universal across all participants. These preliminary findings hint at the potential benefits of VR in geological exploration and education as valuable tools converging scientific exploration with immersive experiences.

5.7.1 Description of the Study

This study investigated the potential of VR for virtual geological surface exploration, through a custom-made application known as OGIVE (detailed in Section 5.2). The exploration was conducted within virtual analog environments synthesized from a real-world expedition in Svalbard, Norway, (as described in Section 5.6). The primary focus was on the spatial presence and immersion quality of the applications, along with the unique capabilities offered by using a Mars analog as a tool for scientific exploration. The experiment's primary independent variable was the exploration medium, with participants engaging with either the VR or desktop application for the duration of a trial. Site 1, known as the river site, served as a training location, allowing users to familiarize themselves with the controls and toolkit for both mediums.

Subsequently, participants explored Sites 2 and 3, representing permafrost and glacial moraine terrains, through both the VR application on a modern Head Mounted Display[4] and the corresponding desktop application. During these explorations, they were provided with a geology fact sheet, (Figure 5-20), that introduced two potential geological features for each site. Key defining elements for these features were presented, and participants were required to utilize all available tools, (refer to Section 5.6.4), to deduce the features represented at the sites. This methodology allowed the study to highlight the effectiveness and immersion of VR as a medium for virtual geological exploration, offering insights into both the process and the potential educational applications.

[4] vive.com/us/product/vive-pro2

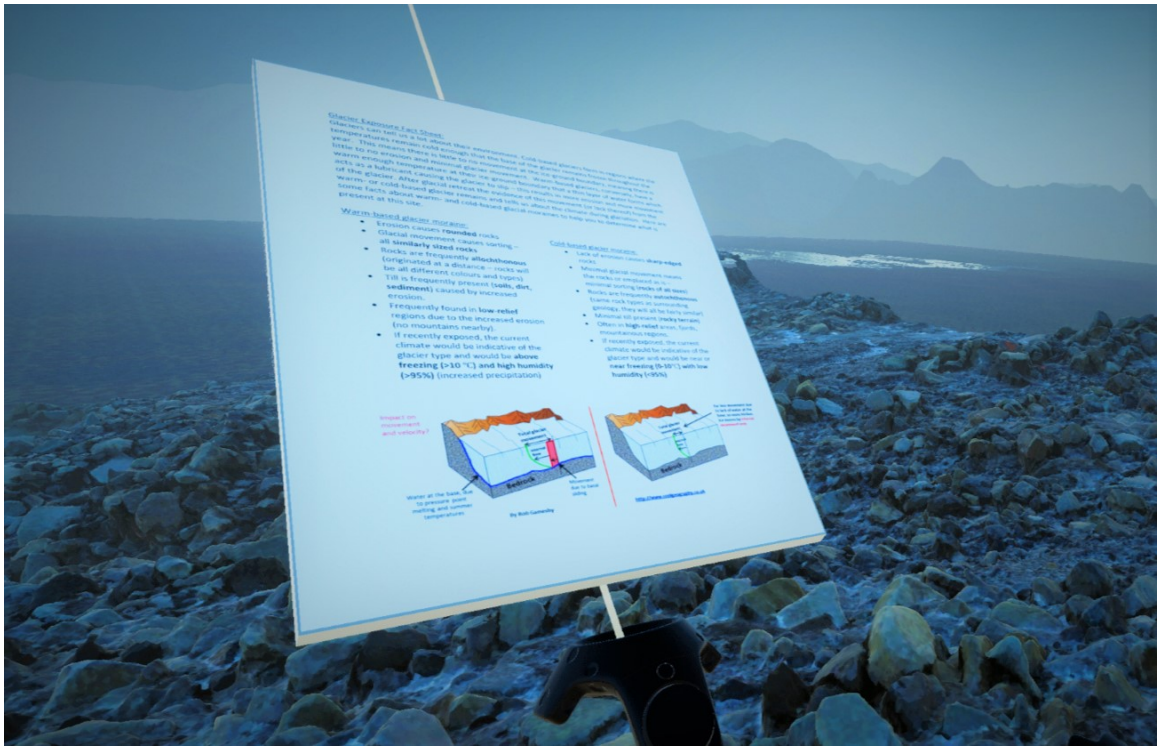

Figure 5-20: *This image illustrates the VR-based application and the accompanying User Study Requirement document. The document provides an overview of the user study, outlining the objectives and requirements. It also includes the necessary consent form for participants. The VR application serves as the platform for conducting the study, enabling users to engage with the immersive environment and provide valuable feedback.*
Image courtesy: Cody Paige and Trent Piercy.

Both applications provided the user access to the following datasets: high-resolution 3D imagery, environmental data, a series of tools, a Mars analog view, and the geological fact sheet pertaining to the field site being explored. Each subject performed the exploration task in each mode once for two field sites, for a total of four trials. Each trial contained a critical geological feature to be identified with 7-8 facts available to make the identification. The subjects all explored Sites 2 and 3, with the order of VR and desktop application and the order of sites randomized. They were then asked to provide feedback on their experience. All testing took place in a (2.5 x 2.5 square meter) open laboratory space under ambient light conditions. Twenty subjects participated in the study [195].

The results, (Figure 5-22, 5-23) showed that the VR application was significantly more effective than the desktop application for spatial presence and immersion. The VR application also resulted in significantly higher accuracy in identifying the critical geological features. The participants also reported that they found the VR application to be more enjoyable and engaging than the desktop application. These results suggest that VR has the potential to be a valuable tool for geological surface exploration. The Spatial Presence Experience Scale [211, 212] was used to assess the participant's

impression of immersion within each environment. A qualitative approach was used to assess the impact of using the Mars analog for scientific exploration.

In collaboration with Cody Paige, and Trent Piercy (MIT UROP), a mirror version of the environment was fashioned in VR to align with the desktop application, maintaining a consistent visual correspondence between the two. The VR application served as a conduit for replicating the 3D reconstructions of the analog environments, aimed at affording a more immersive user experience. Traditional tools presented in the desktop version, such as the flashlight, measurement device, and filter tool, were thoughtfully adapted to the VR context. To further amplify the user experience, VR-specific tools were developed, encompassing features like teleportation and a PDF viewing tool for geology fact sheets, (as shown in Figure 5-20). Although the VR application's intention was to enable users to traverse the synthesized analog environments within their physical space, it was primarily an exploration into new ways of immersive interaction rather than a definitive enhancement of exploration capabilities. Participants were asked the following two questions after completing their tasks in both the VR and Desktop applications:

1. What features of the environment did you notice changes when you were in the Martian analog view?

2. Did you find the Martian analog view useful in imagining the Earth feature as a Martian feature? Explain.

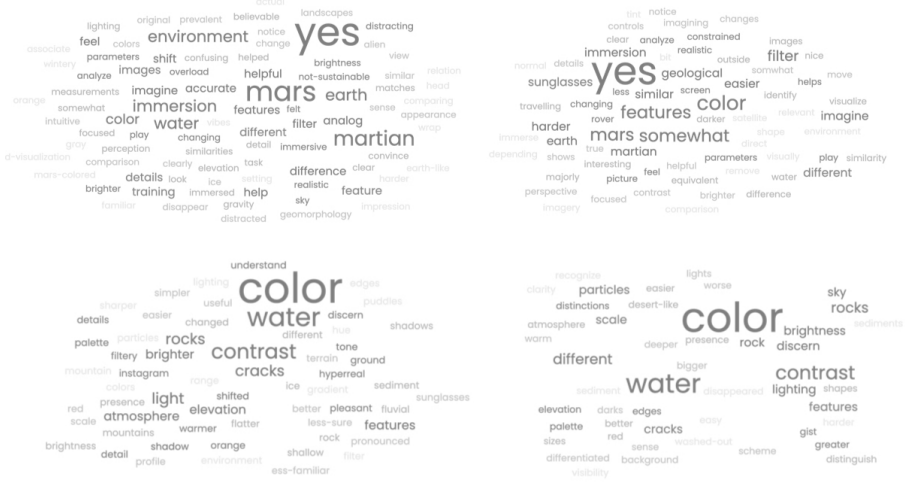

Figure 5-21: *A collection of word clouds generated from the study's questionnaire responses. The word clouds highlight the participants' perceptions of the analog environment, specifically noting the Martian-like characteristics such as color lights, absence of water, and various geological features. The prominent use of these terms in the participants' feedback indicates their strong association of the analog environment with the Martian landscape, showcasing the success of the immersive experience in the simulated analog environments.*

5.7.2 Analysis and User Feedback

The Spatial Presence Experience Scale (SPES) [211] was used to assess the participant's impression of immersion within each environment. This is a scaled response (1-5) questionnaire with 4 questions relating to the participant's concept of self-location, which has been linked to the feeling of being enveloped by or surrounded by a media environment and 4 questions relating to the participant's concept of their perceived possible actions within the environment. The questionnaire was presented to the participant after they completed their task in both the VR and Desktop applications. For both self-location and perceived action, participants were found to have a higher spatial presence in the VR application compared to the Desktop application.
The responses were then filtered to remove insignificant words (such as "and", "the", and "I") and fed into a word cloud generator. Figure 5-21 shows the word clouds generated for both the VR and Desktop applications for questions 1 and 2, respectively. For question 1, the participants most significantly noticed the water, color, and contrast for both applications. The main difference between the two applications was the number of features listed by the participants. More features were listed for VR than for the Desktop application.

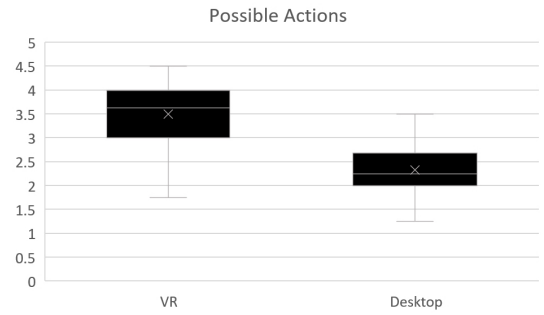

Figure 5-22: *Box and whisker plot representing the results of the four 'possible actions' questions for both environments. The findings indicate the participants reported a higher sense of perceived actions in the VR application compared to the desktop application.*

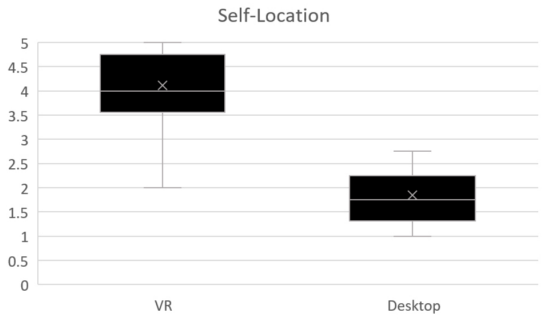

Figure 5-23: *Box and whisker plot displaying the results of the 'self-location' questions for both the VR and desktop applications. The findings indicate a higher spatial presence reported by participants in the VR application compared to the Desktop application.*

For question 2, (Figure 5-21), both the VR (left) and desktop (right) applications had "yes" as the most common answer, with "somewhat" also visible for the desktop application. Additionally, the words used for the VR analog, such as "immersion", "detail", "imagine", and "accurate", are more indicative of a representative analog, while some of the verbiage for the desktop, such as "sunglasses", and "different",

indicated a less immersive experience, similar to the SPES results. Finally, as with question 1, more diversity in the key words was observed for the VR application than for the desktop application, suggesting that the VR incited a stronger response from participants [195].

5.8 Conclusion

5.8.1 Summary

The Svalbard analog mission is a compelling example of the application of digital twin analog environments in the field of geology. The project leverages high-resolution 3D reconstructions, environmental data, and advanced visualization techniques to offer researchers and scientists a unique opportunity to explore and study the geological features of the Svalbard site as a Martian analog. The carefully selected field sites in Svalbard provide geological formations and characteristics that closely resemble those found on both Earth and Mars.

By capturing comprehensive data using aerial photography, environmental sensors, and LiDAR/RGBD cameras, the project enables accurate representation and analysis of these analog environments. This detailed exploration and analysis in a virtual setting contributes to a better understanding of geological processes and offers valuable insights into potential geological formations and habitability on other planets. The data visualization and manifestation techniques employed in the recreation of the virtual environments are essential for effectively representing and interpreting geological information within the digital twin analog environments. The project utilizes Unity's HDRP and Post Processing Stack to create photorealistic visualizations, ensuring a high level of fidelity and immersion. The integration of high-quality photogrammetry data captured from drones and the use of HDRP's physically-based lighting system results in realistic and accurate representation of the analog environments. The post-processing effects provided by the Post Processing Stack further enhance the visual quality, including tone mapping, color grading, and exposure adjustment. The combination of these techniques enables researchers to visualize and analyze geological features with exceptional detail and realism, facilitating meaningful scientific exploration and interpretation.

A coherent user study conducted by OGIVE demonstrates the significant impact and potential of utilizing digital twin analog environments in geological exploration. These virtual environments provide an immersive and interactive platform for studying geological formations, allowing researchers to overcome challenges associated with conducting fieldwork in remote or hazardous locations. The ability to navigate and analyze digital twin analog environments using intuitive tools and user interfaces enhances the efficiency and effectiveness of geological exploration and analysis. Moreover, these virtual environments can be utilized as training tools for future astronauts, preparing them for lunar and Martian field geology missions. The insights gained from

studying analog environments can be extrapolated to inform and guide geological exploration on other planets, contributing to the broader understanding of our solar system's geological history and potential habitability. Overall, digital twin analog environments have the potential to revolutionize geological exploration, enabling more accessible, efficient, and accurate study of planetary surfaces. More studies relating to geological applications of these digital twin environments are presented in Cody Paige's PhD thesis [197].

5.8.2 Discussion

This chapter has delved into the application of 3D reconstruction and data visualization in virtual analog environments, with a specific emphasis on the Svalbard site as a Martian analog. The utilization of cutting-edge technologies, including aerial photography, photogrammetry, LiDAR modules and Depth Cameras, and Unity's powerful rendering capabilities, has enabled the creation of highly accurate and immersive digital twins of the analog environments. These digital twins, coupled with intuitive user tools and interfaces, have facilitated the exploration and analysis of geological features, offering researchers and scientists an invaluable platform for studying distant environments and enhancing our understanding of other planets.

The achievement of photorealistic rendering through the integration of high-quality photogrammetry, Unity's HDRP, and the Post Processing Stack has elevated the virtual environments to a remarkable level of realism. The accurate replication of lighting conditions, the rendering of reflections and shadows, and the fine-tuning of colors and exposure have contributed to an immersive and visually captivating exploration experience. The addition of voice commands has further enhanced user interaction and immersion, enabling intuitive navigation and exploration within the digital twin analog environments. Through user studies and feedback, it has become evident that the VR application surpasses the desktop application in terms of spatial presence and immersion. The incorporation of a Mars analog view has provided users with a unique perspective, allowing them to visualize Earth features as Martian counterparts. This feature has added an additional layer of understanding and engagement to the exploration process, enabling researchers to better comprehend the similarities and differences between the two environments.

5.8.3 Future Work

The development of digital twin analog environments holds great promise for future missions and explorations. By expanding the application to other planetary surfaces and incorporating more advanced technologies, researchers can gain even deeper insights into the geological history and potential habitability of our solar system. The ongoing collaboration between geologists, data scientists, and Virtual Reality experts will drive further advancements in this field, ultimately pushing the boundaries of remote research. In addition to their relevance in remote exploration, digital twin analog environments offer immense potential for studying and preserving sensitive

environments on Earth. By providing scientists with virtual access to areas such as the Galapagos or the Antarctic, these environments minimize the ecological impact of physical exploration while still providing valuable insights. Through a virtual analog, space and time become fluid and malleable dimensions as well as shift in displayed data modality, giving explorers sensorial capabilities that promise to surpass affordances of the physical words. Furthermore, the tools and techniques developed for virtual exploration can be extended to remote and hazardous locations on Earth, aiding in research and conservation efforts.

The OGIVE application not only showcases the effectiveness of 3D reconstruction, data visualization, and virtual exploration tools in geological analysis and research but also highlights the broader impact and potential of digital twin analog environments. By offering naturalistic visualization tools and immersive experiences, these environments have the capability to enhance scientific exploration, improve decision-making processes, and increase the scientific return on both robotic and human missions. They enable researchers to analyze and interpret data collaboratively, providing a shared virtual space for knowledge exchange and discussion. The ongoing development and refinement of these virtual environments, combined with advancements in data collection and analysis techniques, will augment the way environments are explored. By bridging the gap between physical and virtual realities, digital twin analog environments pave the way for new discoveries, scientific insight, and the preparation of future astronauts for missions to Lunar and Martian surfaces.

Chapter 6

Azure Kinect à la Luna

Three-dimensional Reconstruction of Distant Environments.
In-situ Resource Reconnaissance (ISRR).

6.1 Introduction

6.1.1 A Brief Overview of this Chapter

In situ resource utilization (ISRU) is a key step in establishing a permanent human and robotic presence on the moon. It will foster a self-sustaining lunar techno-ecosystem that will prepare humanity for deep-space exploration. Human-computer interaction, particularly in mission planning, will play a major role in maximizing the scientific potential of ISRU missions [213]. Volatile prospecting missions, in particular, can benefit from advanced visualization tools that allow multiple team members to analyze, discuss, and interpret geological data through naturalistic visualization tools [8]. ISRU will be very beneficial for a number of Artemis II and III science investigations, including instrumentation to support volatile monitoring [214].

The MIT RESOURCE team has made preliminary progress in developing tools to construct a virtual reality platform for lunar surface exploration by testing low-cost techniques [215] for collecting high-resolution depth data for integration within virtual environments [196]. Current lunar surface maps available from orbital data, such as those produced by the Lunar Orbiter Laser Altimeter (LOLA) with radial elevation accuracy of 10 m, do not provide high enough depth data resolution for surface in-situ geological data analysis. This requires depth data with a higher resolution than is available from orbital data alone [216]. As previously mentioned in Chapter 1, (Section 1.3.3), project OnSight [101], provides scientists the ability to virtually work together on Mars using the Microsoft Hololens headset. The software creates an immersive 3D terrain model using images down-linked from NASA's Curiosity Rover, and processed using structure from motion (SfM) photogrammetry to create a digital outcrop model of the Martian surface.

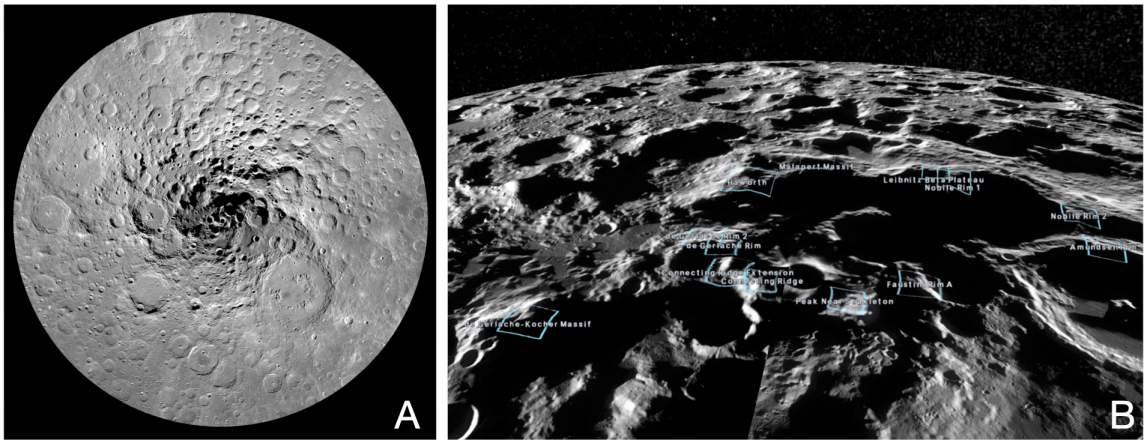

Figure 6-1: *Exploring the south pole of the Moon with NASA's Lunar Trek. Image A displays the south pole of the Moon. Image B reveals potential locations of landing for the Artemis III mission [3]. Image courtesy: NASA Trek.*

The Structure from Motion (SfM) photogrammetry method used in OnSight faces limitations due to the need for extensive data sets and a time-consuming alignment process to reconstruct three-dimensional point clouds [187]. This photogrammetric approach primarily relies on three sets of cameras: the Navcam, a pair of wide-angle stereo cameras; the Mastcam, a pair of high-resolution RGB cameras; and the Mars Hand Lens Imager (MAHLI), a color, high-resolution microscope. Although Navcam images provide significant overlap, they lack detailed high-resolution information. The varying focal lengths of the Mastcams lead to poorly overlapping images, making initial image alignment automation challenging. Consequently, the reconstruction process demands large data sets, and for instance, reconstructing the Kimberly outcrop required 638 images from the Navcam alone, with over 2000 total images used throughout the entire process [217].

The use of stereo cameras for Lunar surface applications presents an additional challenge concerning the necessity to precisely determine the distance between the two cameras [216]. During take-off and landing, these cameras experience extreme levels of vibration, making them susceptible to misalignment. As a result, re-calibration becomes necessary before use, and this process can be challenging to perform remotely. Moreover, using two cameras increases the payload mass requirements, while overlapping imagery leads to higher transmission bandwidth demands. Furthermore, this stereo camera method relies on an external or natural light source to illuminate the imaged area, posing potential challenges when mapping permanently shadowed regions and lava tubes—both of which are areas of significant interest on the Lunar surface. The team has undertaken experimentation with the use of a depth camera for 3D reconstruction [182]. Depth cameras can be classified into three main types: structured or coded light, stereo depth, and Time-of-Flight (ToF). Structured or coded light cameras leverage the deformation of a known light pattern to calculate object distance. Stereo depth cameras, as mentioned earlier, use the known distance

between two cameras and the light reflection from an object to triangulate its position. On the other hand, ToF cameras measure distance by analyzing the time taken for laser light to reflect off a surface. Stronger lasers enable ToF cameras to measure greater distances. Both ToF and structured light cameras require distinctiveness between emitted and returned light, making them susceptible to sunlight or external light source interference. However, this susceptibility proves advantageous in scenarios like lunar night exploration or 3D mapping of subsurface structures, such as lava tubes, where little to no external light is available [218].

One of the principal advantages of using ToF cameras is their simplicity as single-camera systems, which eliminates the need for complex position calibration. Moreover, ToF cameras have gained popularity in the commercial sector and have been integrated into the latest mobile devices. The availability of Commercial-off-the-Shelf (COTS) components has made it easier to adapt them for lunar applications with minimal modifications, as demonstrated in this study; There is enormous potential now in pivoting rapidly evolving terrestrial technology into space applications [71]. The recent commitment from the United States Government to return to the moon has opened new possibilities for university-scale teams to participate in lunar missions. The reduced cost of such missions and increased collaboration opportunities among institutions and research groups have contributed to this feasibility. By utilizing COTS parts, it becomes more accessible for teams to send payloads on commercial lunar rovers. COTS components to capture depth data on the Moon heralds a new era in the visualization of the lunar surface. This advancement facilitates the rendering of virtual environments, enriching the exploration for scientists and extending the reach of lunar study to a global audience [216, 195].

The team, in collaboration with NASA Ames and Lunar Outpost, has been exploring the integration of depth-mapping into a Virtual Reality (VR) platform for lunar rover exploration missions utilizing a COTS RGB plus ToF camera (RGBD camera) to provide centimeter-scale resolution imagery and depth data for geological and scientific analysis of portions of the lunar surface. During the preliminary testing phase, a combination of a ToF camera and an RGB camera was used for data collection. Through these tests, it became evident that an RGBD camera offered the most comprehensive mapping capabilities while optimizing data-return bandwidth and development costs. To this end, the team selected the Microsoft Azure Kinect, a COTS camera with integrated ToF and RGB imaging, which allowed for quick hardware adaptation for near-term lunar missions, reducing the typical prohibitive cost associated with lunar surface data collection. The utilization of a COTS RGBD camera streamlines the alignment of different camera view fields and positions when rendering the VR environment. By harnessing the capabilities of off-the-shelf technology, the team is actively developing the hardware, software, and Concept of Operations (ConOps) for a modified COTS RGBD camera, targeting its deployment on a rover aboard a Commercial Lunar Payload Services (CLPS) mission [216, 195].
This chapter introduces research that explores the use of a specific depth camera, the Microsoft Azure Kinect, which has been modified in both hardware and software

components to operate as a stand-alone payload on a lunar rover module, (as shown in Figure 6-2). Depth cameras come in three major categories: structured or coded light, stereo depth, and ToF. Among these, ToF cameras offer several advantages, including their single-camera setup without position-calibration dependency and increasing use in commercial industry. By utilizing a COTS RGBD camera, such as the Microsoft Azure Kinect, the MIT RESOURCE team aims to integrate depth mapping into a virtual reality platform for lunar rover exploration missions. This approach enables centimeter-scale resolution imagery and depth data for geological and scientific analysis of the lunar surface.

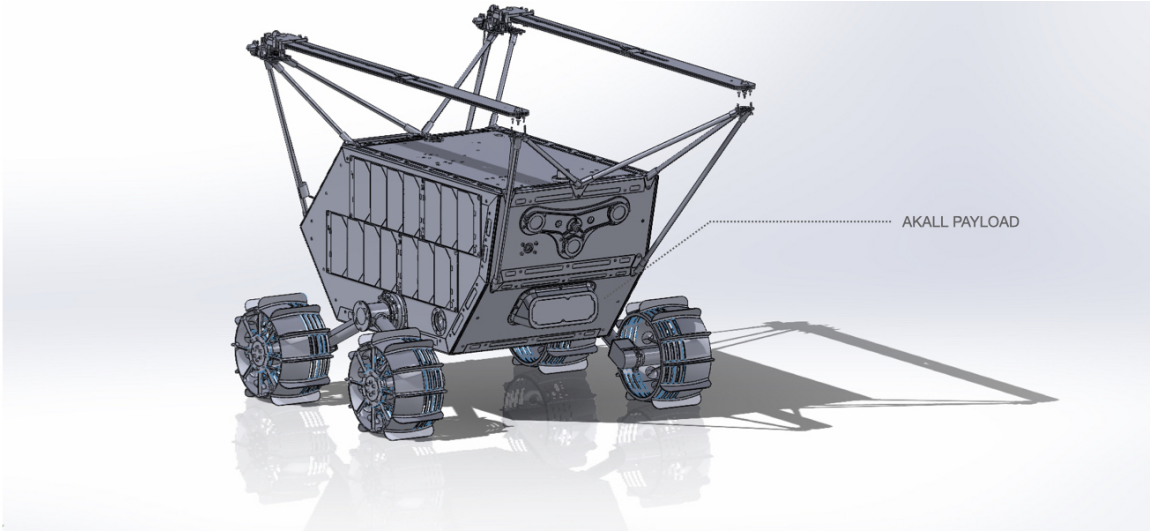

Figure 6-2: *An illustration of the Lunar Outpost's lunar rover, engineer to incorporate a modified version of the Azure Kinect as part of the AKALL payload. The design indicates the placement and integration of the camera in the rover's system.*
Image courtesy: Lunar Outpost [219].

This chapter documents the development of a custom software package, Azure Kinect à la Luna (AKALL) [215], while covering the hardware modification and testing performed by NASA Ames Research Center and the ConOps designed by the MIT RESOURCE team in collaboration with Lunar Outpost, a space systems and rovers company based in Colorado, USA [219]. The chapter begins by providing an overview of the research problem, followed by a discussion of the related work. The next section describes the hardware and software modifications that were made to the Microsoft Azure Kinect to enable its use as a stand-alone payload on a lunar rover module [220]. The following section presents the results of the hardware and software testing that was performed, followed by a discussion of the ConOps that was designed for the lunar rover exploration missions.

The chapter concludes with a discussion of the future work that is planned for this research. The collected data will provide a basis for rendering highly detailed virtual lunar environments, offering scientists and the general public a closer look into lunar

environments. Additionally, the successful implementation of depth data collection through the modified COTS camera can pave the way for more realistic astronaut training and robotic data collection in challenging lunar regions, such as craters and lava tubes.

6.1.2 Technical Overview of the Azure Kinect Device

The Microsoft Azure Kinect is a ToF depth-camera with an integrated OV12A10 12MP CMOS sensor. The camera has two separate systems for depth and RGB imagery. The ToF camera uses modulated near-IR (NIR) light to process and generate a depth-map of a scene. This is created by measuring the time it takes for the NIR light projected by the camera to return to the camera sensor. The amount of NIR light returned from the scene is also recorded, providing an IR image alongside the ToF depth-image. The ToF camera has different modes (narrow field of view - NFoV, wide field of view - WFoV, binned and unbinned) to allow for customization of the X- Y- and Z-axis range. The ToF camera has a resolution range of (320 x 288 pixels) to (1024 x 1024 pixels) and a Z-range of (0.5 to 5.5 meters) depending on the settings used. The RGB camera provides aspect ratios of 16:9 and 4:3. To have the RGB image completely overlap with the depth map, the 4:3 aspect ratio is used with the NFoV depth setting [221].

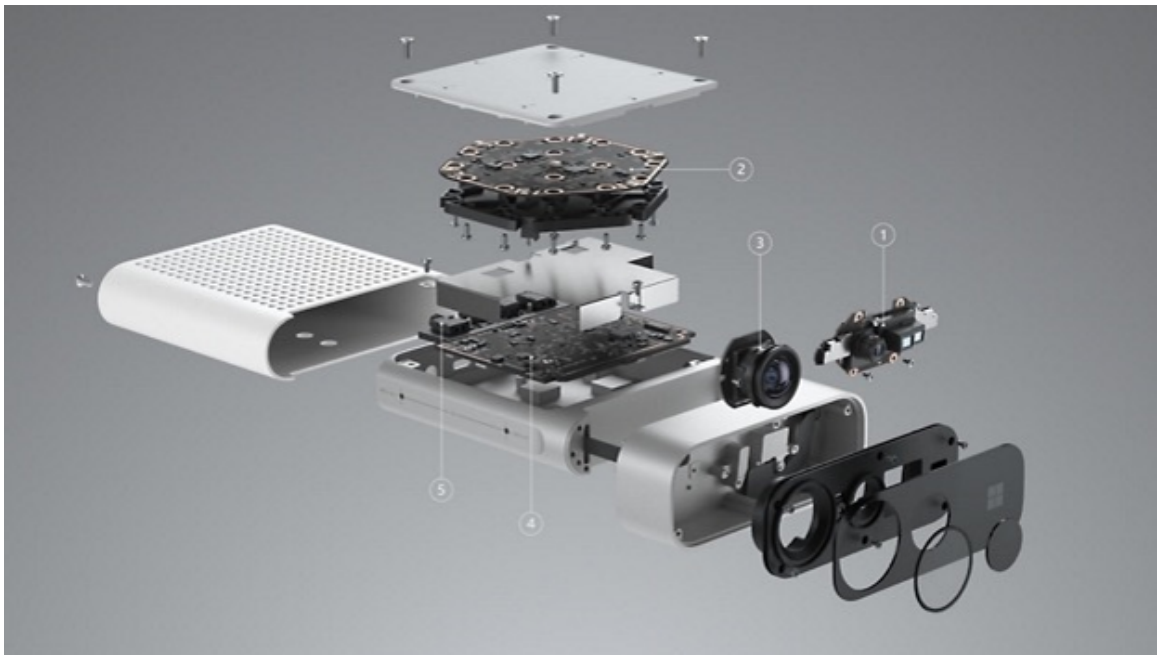

Figure 6-3: *Microsoft Azure Kinect internal hardware – 1) 1-MP depth sensor with wide and narrow field-of-view (FoV) options that help you optimize for your application 2) 7-microphone array for far-field speech and sound capture 3) 12-MP RGB video camera for an additional color stream that's aligned to the depth stream 4) accelerometer and gyroscope (IMU) for sensor orientation and spatial tracking 5) External sync pins to easily synchronize sensor streams from multiple Kinect devices. Image courtesy: Microsoft [221].*

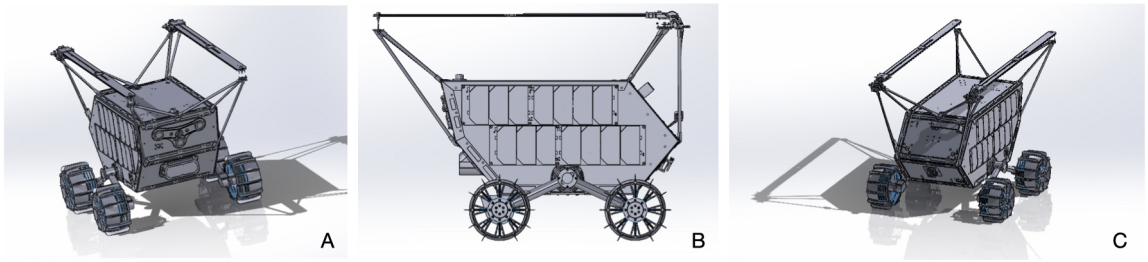

Figure 6-4: *Lunar Outpost is supporting Nokia's LTE/4G NASA Tipping Point project with Intuitive Machines, a leader in cutting-edge technologies for space, to build, integrate and test the first-ever LTE/4G network on the surface of the Moon. One of the primary objectives of the Nokia technology demonstration is to validate that Nokia's LTE/4G technology can support proximity communications for lunar operations and future human spaceflight as shown in the antennas design on the lunar rover [219]. Image courtesy: Lunar Outpost.*

6.1.3 Ruggedization of the Microsoft Azure Kinect

The selected RGBD camera, the Microsoft Azure Kinect, was evaluated for flight-compliant materials and modified to reduce mass by the team at NASA Ames, reported in Jha, V. et al [222]. The camera was modified to reduce mass by removing the outer aluminum casing, the microphone array, RF shield and front face, resulting in a mass reduction of 140g. Plastic parts were replaced with vacuum-compatible materials and cables were replaced with specialized cables for integration into the Lunar Outpost rover. The environmental testing, (Appendix B Table B.9), was performed on the modified Kinect with Random Vibration/Sine testing, per GEVS (NASA's General Environmental Verification Standard for spaceflight launch survival). Thermal Vacuum testing cycled the instrument between expected hot (+85^{o}C) and cold survival temperatures, as well as +50^{o}C and -25^{o}C operational temperatures. The instrument functioned nominally at the conclusion of vibration and thermal cycling tests [222]. The camera was flight qualified to TRL 6 including thermal vacuum and random vibration testing by NASA Ames.

6.1.4 Motivation

The Azure Kinect à la Luna (AKALL) system was motivated by the desire to utilize the powerful capabilities of the Azure Kinect, (Figure 6-3), device in a new and innovative way, particularly in the field of space exploration and robotics [223]. AKALL was designed to operate on board the flight computer of the Lunar Outpost's Mobile Autonomous Prospecting Platform (MAPP) rover to telecommand the MIT RESOURCE Kinect payload. The Azure Kinect offers a powerful platform that can be used for a variety of tasks, including object detection, tracking, and mapping. However, in the context of the upcoming IM-2 mission, (Section 6.5.3), the camera's sole purpose is to collect and send RGBD images that will later be processed and turned into textured 3D meshes of lunar environments.

The project aimed to build a custom software application that could fully utilize the Azure Kinect Development Kit (DK) to control the device's camera and capture data from its various sensors. The goal was to create a system that could be seamlessly integrated within larger computing systems, while also providing a high degree of portability and isolation. This was achieved by using Docker containers. Moreover, the AKALL software was motivated by the need for a robust communication mechanism that could allow the application to interact with other programs and devices. To meet this requirement, a UNIX socket server was implemented, enabling the application to send and receive commands using the UNIX socket protocol.

In addition, the project introduced a unique capture sequence messaging scheme to control the camera and manipulate its parameters, opening up a wide range of possibilities for data capture and analysis. The AKALL software has been successful in developing a custom software application that can fully utilize the Azure Kinect DK. The application has been tested and demonstrated in a variety of environments, including a simulated Mars rover environment. The results of the project have shown that the Azure Kinect device is a powerful tool that can be used for a variety of tasks in space exploration and robotics.

6.2 Concepts of Operation

The MIT RESOURCE conducted a series of comprehensive experiments to evaluate the ConOps of the Microsoft Azure Kinect, which offers four distinct modes [216]. These modes include narrow field of view (NFoV) and wide field of view (WFoV), each with 2 x 2 binned and unbinned options, influencing the camera's range and sensitivity. Such selections have implications for the instrument data budget and the quality of output imagery. Given the mission's constraints on power consumption and data minimization, optimizing both aspects becomes crucial. To achieve this, four capture modes were devised: Single Target Capture, Traverse Capture, Science Station Capture and Single Target Circumference Capture. Each mode serves specific purposes an is tailored to meet the mission's objectives effectively.

By default, the Azure Kinect captures and stores RGBD data into .mkv file format, but the AKALL software, (as detailed in Section 6.3), facilitates the extraction and storage of individual frames, allowing for data reduction while still enabling the creation of composite point-clouds frame by frame through stitching. Although this approach may slightly reduce image accuracy, it proves essential, given the low-bandwidth of space operations, for efficient mission operations [216]. The NFoV mode of the Azure Kinect provides a narrow field of view but high sensitivity, making it well-suited for close-range tasks, such as object identification. On the other hand, the WFoV mode offers a wide field of view but lower sensitivity, making it ideal for long-range tasks, such as mapping. Additionally, the 2 x 2 binned modes provide a lower resolution, but higher sensitivity, compared to the unbinned modes, offering flexibility in balancing data storage requirements and improving image quality, particularly in

low-light conditions. These capabilities of the Azure Kinect are instrumental in addressing the specific requirements of lunar exploration missions, where efficient data management and high-quality imagery are vital.

The Single Target Capture mode is employed to capture a single target object. This is valuable for tasks such as object tracking or scene analysis. The Traverse Capture mode is employed to capture a series of images as the camera moves through a scene. This is valuable for tasks such as 3D reconstruction or mapping. The Science Station Capture mode is employed to capture a scene at a science station. This is valuable for tasks such as monitoring experiments or collecting data. The Single Target Circumference Capture mode is employed to capture a single target object from multiple angles. This is valuable for tasks such as object recognition or 3D scanning. The Azure Kinect's capture modes are tailored to the specific requirements of various applications. The team's experiments demonstrated that the camera can be used to gather high-quality data in a variety of challenging environments. The team's findings will be utilized to create new applications for the Azure Kinect and to improve its performance.

6.2.1 Capture Modes

The single target capture mode takes individual frame RGBD images of specific areas of interest when there's limited time or data available for a full circumferential capture. This data can be used to recreate a directional 3D image of the feature. Since this is a single capture, any occlusion of data may limit the functionality of this mode. Therefore, it should not be used when complete scene reconstruction is required.

The Traverse capture mode takes single frame RGBD images along the rover's traverse path. Ideally, this should be done every 2 meters, allowing for overlap between image captures for sections of the traverse approximately 10 meters long. This data will be used to create a 3D model of the traverse path. Like the single target capture mode, occlusion will result in a directional 3D reconstruction, missing data from the backs of target objects. The science station capture mode records a complete site by performing a raster pattern approach over the selected location, capturing single-frame RGBD images at 1-meter intervals across the entire site. While this method generates the same 3D corridors as described in the Traverse mode, the use of the raster pattern also provides the reverse direction, with enough overlap between corridors to construct a complete 3D scene. This data can be used to create a 3D model of the science station environment at locations of interest, such as impact ejecta sites, approaches to challenging terrain, or sites identified as having a high probability of water content, etc [216, 195].

Finally, the single target circumference capture mode aims to capture all angles around a single target object. These captures should be taken every 30^{o} relative to the object's center, or for very large objects, as many as necessary to have approx-

imately 40° of overlap between frames. This data will be used to create a full 3D reconstruction of all visible areas of the object, including lighting conditions, color, and geometry. This capture type places the most significant demands on rover operation and is variable depending on mission constraints. For any capture, the target distance must be no greater than 4 meters from the surface of the feature, and no less than 0.5 meters for best resolution data. The ideal distance is set at 1 meter [216].

6.2.2 Mission Planning

Routine instrument command operations will adhere to the Lunar Outpost scheduling process, ensuring well-coordinated rover and lander operations with effective communication. Notably, capture operations are distinct from data cross-link and down-link processes due to their higher bandwidth requirements. The scheduling prioritizes the primary rover payload, recognizing its paramount importance on the rover. As the mission progresses, the Kinect payload is expected to gain higher priority, especially towards lunar sunset, capitalizing on its ability to operate effectively in low-light conditions. Although specific scenarios are yet to be developed, the ConOps are designed to remain fully operational, adapting to diverse lighting and timing conditions [216].

6.3 The AKALL Payload Software Module

The Azure Kinect à la Luna (AKALL) software documents the steps taken to access, compress, and archive RGBD images captured from a commercial camera, the Azure Kinect, (Section 6.1.2), and integrate them within a larger robotic system, (Figure 6-2), encapsulated within a Docker container. This section aims to highlight the primary objectives and anticipated outcomes of AKALL:

- To develop a versatile, modular, and portable framework that operates in a variety of settings, including rovers and robots.
- To expose the various sensing capabilities of the Azure Kinect device through the implementation of a novel parametric messaging service.
- To operate within an isolated Docker container and establish communication with the host machine through UNIX Domain sockets.
- To capture, and archive data captured by the Azure Kinect device in a shared directory, mounted and accessible by the host machine.
- To further optimise RGBD data collection into a new file format that is suitable for low-bandwidth scenarios such as in space exploration.

In the context of space operations, the necessity of having isolated and robust systems cannot be overstated. The AKALL software was originally created to operate on board the flight computer of Lunar Outpost's Mobile Autonomous Prospecting Platform (MAPP) rover, as a software module that remotely controls the MIT RESOURCE Kinect Payload, (Figure 6-4). As required by the company, AKALL had

to be designed and utilized in an isolated Docker container, highlighting the critical importance of maintaining a high level of system reliability to ensure that potential issues within one system do not jeopardize the overall mission's success [223].

Data compression is a crucial aspect of space exploration, as demonstrated by the Galileo NASA mission to Jupiter in 1989 [224], and continues to be a common practice in modern space missions. Given the challenges of limited communication bandwidth and vast distances, efficient data compression is essential to optimize data transmission and storage. By default, the Azure Kinect DK captures and stores RGBD frames in the .mkv format, also known as Matroska Video. While this format has been useful for various applications, there was a need to transition to a more photo-focused implementation, while performing both lossy and lossless compression, as indicated by the ConOps, (Section 6.2), which emphasizes single captures instead of videos.

6.3.1 Technical Implementation

The AKALL application is designed to operate within a Docker [225] container running on Ubuntu 18.04 LTS. This design choice provides an isolated and portable environment, making it ideal for integration within larger computing systems such as rovers and robots. The hardware requirements needed to install AKALL on a development machine include having a modern Nvidia GPU and USB3.1 Gen2, supports speeds of up to 10Gbit/s, as stated by device's technical specifications.

Using Docker containers provides a number of security benefits. It isolates the application, which means that potential issues or vulnerabilities in one container don't affect others. It also means that the application has a limited effect on the host system, as it's confined to the container. In terms of upgrades, Docker containers can be easily updated to include new versions of the application or any additional dependencies. This can be done by creating a new Docker image and replacing the running container with a new one based on the updated image. This makes it easy to roll out updates across multiple environments, as the same Docker image can be used in various computing platforms. The AKALL application utilizes custom commands or instructions to control the Azure Kinect device and capture data. This application is capable of adjusting parameters such as frame rate, compression, resolution, depth mode, exposure, contrast, sharpness, gain, white balance, black light compensation, power line frequency, and timestamp. By using these specific commands, users can customize the data capture according to their requirements, offering a high level of control and flexibility over the capture process.

The AKALL software yields a robust, flexible, and highly functional application designed to interface with the Azure Kinect device, efficiently process incoming data, and provide a streamlined user interaction interface. The application has been built using a combination of C, C++, and Python programming languages and operates within a Docker container to ensure isolation and portability. The most recent version of the AKALL code-base can be retrieved in Appendix C.4.

The next section aims to describe key aspects of the AKALL application, focusing on its interactive mode, the capture sequence console, and the specifics of data capture and storage.

- Interactive mode: The AKALL application can be run in interactive mode, which allows users to control the capture process in a Read-Eval-Print-Loop (REPL) environment. This mode is useful for debugging and testing the application, as well as for capturing data for specific purposes.
- Capture sequence console: The AKALL application also includes a capture sequence console, which allows users to specify a sequence of commands to be executed during the capture process. This is useful for automating the capture process and for capturing data in a specific order.
- Data capture and storage: The AKALL application captures data in a variety of formats, including raw data, point clouds, and RGBD images. The data can be stored locally or on a remote server.

The AKALL application is a powerful tool for capturing data from the Azure Kinect device. It offers a high level of flexibility and control over the capture process, and it can be used for a variety of purposes.

6.3.2 Docker Containers as Payload

The Docker image for the AKALL application contains all the necessary dependencies and configurations needed for the software to function. The Docker image is built using a Dockerfile, which is a text document that contains all the commands a user could call on the command line to assemble the image. Docker containers encapsulate the application and its dependencies into a single, standalone unit, which can be run on any system that has Docker installed. This makes it easier to manage, distribute, and run applications. The AKALL application specifically uses Docker to provide a controlled, isolated environment for its operation. Here's an example Docker command to launch an instance of the AKALL application [215]. This command runs the Docker container on the development machine with several flags:

```
docker run --rm --gpus all --privileged -v /storage:/storage
```

- –gpus all: This flag allows the Docker container to access all GPU devices on the host machine, which is necessary for the Azure Kinect DK to operate correctly.
- –privileged: This flag gives the container almost the same privileges to the host machine, which is sometimes necessary for certain operations during testing, although this is only implemented on the development machine, a more specific, and tailored, set of permissions on the rover's flight computer is set by the Lunar Outpost team.

- -v /storage:/storage: This flag mounts the storage directory from the current working directory on the host to the /storage directory in the container, which is used to store the captured image files.

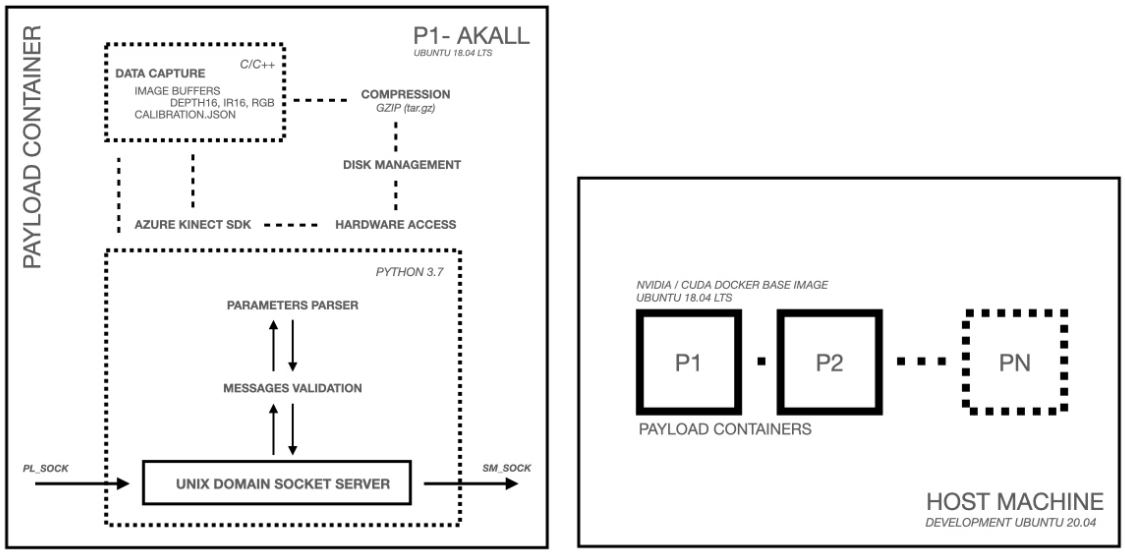

Figure 6-5: *System Architecture Diagrams of the AKALL payload.*

6.3.3 System Architecture

The overall framework utilizes a number of key concepts in its operation. The pl-sock binds messages that are incoming from the host machine to the payload's container, while the sm-sock binds messages that are incoming to the host machine from the payload's container, (Figure 6-5). This allows for two-way communication between the host machine and the payload's container. The shared directory, /storage, allows for communication between the payload's container and the host machine. This allows the payload's container to store and retrieve files from the host machine. A successful capture will generate four files (color.jpg, depth16, ir16, and calibration.json); these files are then compressed with gzip in a custom container format called ".nd3" and then stored in a shared directory. This allows the host machine to access the captured images and calibration data. The application's socket server and logger are launched using the script, (./scripts/entrypoint.sh), which enables communication with other programs and devices using the UNIX socket protocol, and allows for control of the camera using custom capture sequence messages. This allows the application to communicate with other programs and devices, and to control the camera. An in-depth review of the custom ".nd3" file format is discussed in Section 6.4.3.

The core application features several modes to control the system. Through an interactive program, users can send specific commands or instructions to control the Azure Kinect device, manipulate its parameters, and customize data capture according to

their unique requirements. This includes adjusting frame rate, compression, resolution, depth mode, exposure, contrast, sharpness, gain, white balance, back light compensation, power line frequency, and timestamp. A real-time console has also been developed, allowing the user to have a deeper level of control over all of the components of the application. This interface enables users to manage and monitor the data capture process, providing a visual guide and feedback during operation. The core application is a powerful tool that can be used to capture data in a variety of ways. The interactive program allows users to easily adjust the settings to get the perfect results for their needs. The real-time console provides valuable feedback that can help users to troubleshoot any problems that may occur. Internally, these commands interact with the Azure Kinect DK through the C++ library k4a wrapper, allowing for direct control and fine-tuning of the Azure Kinect device's capabilities. It also manages the compression of captured data frames and their subsequent storage, adding another layer of flexibility to the data capture process.

Data capture and storage in the AKALL application have been designed with efficiency and versatility in mind. The application interfaces with the Azure Kinect DK via a C++ program using the provided k4a library, which allows the AKALL application to effectively control the device's camera and capture data from its various sensors. Once the data frames (rgb, depth, ir, and calibration.json) are captured, they are stored in .nd3 format, then compressed using gzip. This step reduces the size of the captured data, making it easier to store and transmit. The data follows a naming convention, allowing the user to understand the content of each compressed folder as well as a timestamp. For storage, the AKALL application leverages Docker's capability to share directories with the host machine. The compressed data frames are stored on a disk in a directory that is mounted and shared by the Docker container with the payload host machine. This configuration enables safe and easy access to the captured data without breaching the isolation of these systems. The directories are set by the flight computer administrators and system engineers, ensuring controlled access to the data.

6.3.4 Modes of Operation

The AKALL application offers several modes of interaction, which can be utilized to affect the camera's built-in parameters and set compression and other relevant parameters. One notable mode is through a novel messaging scheme that extracts parameters from ASCII-based commands. This innovative approach enables users to interact with the camera system efficiently, using plain-text commands to modify settings and perform various actions. By employing this messaging scheme, users can easily tailor the concepts of operation, (as mentioned in Section 6.2), and adapt it to different scenarios, enhancing the flexibility and usability of the AKALL application. Additionally, this messaging system allows for seamless integration with other tools and applications, streamlining the overall workflow and facilitating seamless data capture and discovery in lunar exploration missions. AKALL employs two main capture sequences, a short capture sequence (SCS) and a long capture sequence (LSC).

```
K |  15  | MJPG | 0720 | 1          | 1671006655
K |  FPS | COMP | RESO | DEPTH_MODE | TIMESTAMP
```

```
FPS: 05, 15, 30
COMP: MJPG (Primary), NV12, YUY2, BGRA, DP16, IR16
RESO: 0720, 1080, 1440, 1536, 2160, 3072
DEPTH_MODE: 0:OFF, 1:NFOV_2X2B 2:NFOV_U 3:WFOV_2X2B 4:WFOV_U 5:P_IR
```

Figure 6-6: *Command code sequence breakdown for the AKALL software. Top shows an example code with breakdown of meaning, bottom shows all options for each command code component. FPS: frames per second, COMP: compression, RESO: resolution.*

Short Capture Sequence:

In the list of example commands below, the first part is mandatory to ensure a successful capture sequence ("K05MJPG07201", "K05MJPG10801", etc.), and it indicates the frame rate (FPS) and compression used, as well as other parameters. For instance, the "05" in the first part represents the FPS, "MJPG" represents the compression used, and "0720" is the resolution (in this case, 720p). The "1" at the end of the first part indicates the depth mode, (Figure 6-6). The camera is capable of capturing up to 4k resolution, and there are four different depth modes, (Appendix B Table B.8), available to access through the Azure Kinect DK. Appendix D.2 covers a complete list of short capture sequences.

```
K15MJPG07201-612689613
K30MJPG10802-845455484
K15MJPG14403-957520555
```

Long Capture Sequence:

The second and optional part ("EA-B128-C5-S32-H2-G0-WA-P0-L2-1671006611", etc.) specifies the Exposure, Contrast, Sharpness, Gain, White Balance, Black Light Compensation, Power Line Frequency, and timestamp. The values for these parameters may vary depending on the specific requirements of the application. For example, the "EA" in the second part sets the exposure to automatic mode, if a number is specified, such as "B128" the Contrast option is set to 128, "S" indicates the Sharpness, "G" indicates the Gain, "W" indicates the White Balance, and "1671006611" is the timestamp, (Figure 6-7). Apendix D.3 features a list of randomly generated long capture sequences, just like the list below.

```
K05MJPG07201-EA-B128-C5-S32-H2-G0-WA-P0-L2-1671006611
K30MJPG21602-E130000-B189-C3-S12-H1-G255-WA-P0-L2-1671006633
K15MJPG30720-EM11-B255-C8-S33-H3-G0-WA-P1-L1-1671006644
```

```
EA      |   B128   |   C5   |    S32   |    H2   | G0 |    WA   |      P0       |      L2
EXPOSURE|BRIGHTNESS|CONTRAST|SATURATION|SHARPNESS|GAIN|WHITE BAL|BLACKLIGHT COMP|PWR LINE FREQ
```

```
EXPOSURE TIME: A, M1-M13, M500-M130000. Default: EA (AUTO)
BRIGHTNESS: Integer from 0 to 255. Default: B128
CONTRAST: Integer from 0 to 10. Default: C5
SATURATION: Integer from 0 to 63. Default: S32
SHARPNESS: Integer from 0 to 4. Default: H2
GAIN: Integer from 0 to 255. Default: G0
WHITE BALANCE: A, Integer of 10 incremends from 2500 to 12500. Default: WA
BLACKLIGHT COMPENSATION: 0,1. Default: P0
POWER LINE FREQUENCY: 1: 50hz ,2: 60Hz. Default: L2
```

Figure 6-7: *The longer messaging scheme of the AKALL system showing optional command sequence components for RGB camera settings for the AKALL software.*

Storage management:

To remove a single capture, use the following command: SM-RM-FILENAME (timestamp). Please note that ".tar.gz" is automatically added to the filename. The files are stored in the shared directory "/tmp/payload-storage" on the host machine. Within the docker container, the software automatically assigns names to these files using the format: timestamp.tar.gz. For example, 1673370956.tar.gz.
Storage management commands SM-RM-FILENAME (timestamp) Empty /storage directory: SM-RM-ALL

```
Enter Capture Sequence # K15MJPG30721

[HOST MACHINE] K4A Image str: K15MJPG30721-1673315086
[HOST MACHINE] Sending data on socket: /tmp/payload_sockets/kinect_luna/pl_sock
[HOST MACHINE] connecting to /tmp/payload_sockets/kinect_luna/pl_sock
[HOST MACHINE] Connected
[HOST MACHINE] Done..

[HOST MACHINE]  Received data from /tmp/payload_sockets/kinect_luna/sm_sock
                DATA [17]: b'1675132016.tar.gz'

Enter Capture Sequence # K30MJPG21602-EA-B128-C5-S32-H1-G0-WA-P0-L2

[HOST MACHINE] K4A Image str: K30MJPG21602-EA-B128-C5-S32-H1-G0-WA-P0-L2-1673315140
[HOST MACHINE] Sending data on socket: /tmp/payload_sockets/kinect_luna/pl_sock
[HOST MACHINE] connecting to /tmp/payload_sockets/kinect_luna/pl_sock
[HOST MACHINE] Connected
[HOST MACHINE] Done..

[HOST MACHINE]  Received data from /tmp/payload_sockets/kinect_luna/sm_sock
                DATA [17]: b'1675132036.tar.gz'
```

Figure 6-8: *An interface view of the interactive mode designed to test the AKALL application's messaging scheme. It offers insights into the hands-on user interaction with the application and message transmission process.*

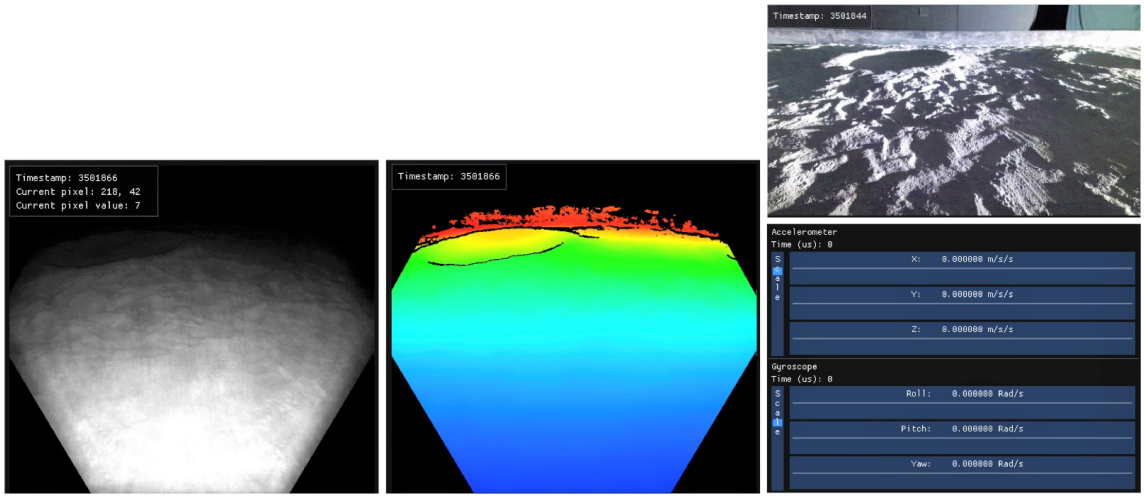

Figure 6-9: *The Azure Kinect Viewer Software (K4aviewer) is a tool that allows users to visualize and analyze data from the Azure Kinect DK. It includes both IR/Depth and RGB cameras, as well as IMU data. K4aviewer can also be used to quickly render 3D views, either from recorded data or from the real-time videos stream.*

6.3.5 Data Processing and 3D Reconstruction

In the data processing and 3D reconstruction stage, the captured .nd3 files, containing a color.jpg, a depth.b16g, an ir.b16g, and a calibration.json files, play a crucial role in constructing the Polygon File Format (.ply) object. This process is facilitated by a custom program that utilizes the transformation functions available in the Azure Kinect Development Kit (DK) which enable the conversion between color images, depth images, and point clouds. These functions form the basis for creating a comprehensive representation and 3D reconstructions of the captured scene.

```
k4a_transformation_color_image_to_depth_camera()
k4a_transformation_depth_image_to_color_camera()
k4a_transformation_depth_image_to_point_cloud()
```

To render the depth and ir data into a 2D Grayscale image, the following approach was employed. Each pixel of the DEPTH16 and IR16 data consists of two bytes (16 bits) of Big-endian unsigned depth data (b16g). To convert these b16g files into a Grayscale image, the ImageMagick software was utilized. By employing these conversion techniques, with the command below, and utilizing the transformation functions provided by the Azure Kinect DK, it becomes possible to generate a comprehensive and detailed representation of the captured scene. The resulting .ply file will contain valuable information about the geometry, color, and depth of the scene, facilitating further analysis and visualization of the captured data.

```
convert -size 640x576 -depth 16 -endian MSB
-define quantum:format=unsigned -define quantum:separate
-depth 16 gray:IR16_FILENAME -normalize IR16_CONVERTED_FILENAME.pgm
```

This command converts the depth data into a Grayscale image format, such as .pgm or .bmp, with a size of 640x576 pixels and a depth of 16 bits. The parameters "-endian MSB" and "-define quantum:format=unsigned" specify the Big-endian byte order and the unsigned format of the depth data, respectively. The "-define quantum:separate" option separates each depth value into two bytes. Finally, the "-normalize" parameter ensures the normalization of the image.

This data processing and 3D reconstruction option highlight the powerful capabilities of the Azure Kinect camera and the sophisticated software ecosystem surrounding it. By leveraging these tools, researchers can gain a deeper understanding of the captured environment, enabling insights and discoveries relevant to Lunar exploration and beyond. The ability to transform raw sensor data into a comprehensive 3D representation opens up a wide range of possibilities for scientific analysis, visualization, and decision-making in various applications. For example, this technology could be used to create detailed maps of the lunar surface, which would be invaluable for planning future missions. Additionally, it could be used to create 3D models of objects or structures, which could be used for training purposes or to create virtual reality experiences [226].

Figure 6-10: *NASA Ames Research Center SSERVI Testbed a sandbox of simulated regolith for lunar studies. Image courtesy: Ferrous Ward.*

6.4 Hardware and Software Testing Review

A series of laboratory and field tests were conducted to assess the capabilities of RGBD cameras for lunar conditions and optimal ConOps, (Section 6.2), design. For lunar conditions, the capabilities were tested with dust interference and solar flux without atmospheric dispersion [227]. The power and data rates for the four capture modes, (Appendix B Table B.8), were tested, and each was optimized for RGB resolution and depth resolution capture sequences. A field test was then conducted in Svalbard, Norway, to design a capture methodology for optimal 3D reconstruction of the traverse and science station capture modes. Finally, the 3D reconstruction techniques and resolution capabilities were tested in the Solar System Exploration Research Virtual Institute (SSERVI) Lunar Testbeds.

6.4.1 Testing Procedures

Dust Interference in Lunar Gravity:

Lunar dust poses significant challenges for human exploration of the Moon due to its abrasive and cohesive nature, which can impair optical instrumentation. Moreover, its fine particles can easily become airborne and remain suspended for extended periods, creating breathing difficulties for astronauts and potential equipment damage. To evaluate the performance of COTS camera in lunar gravity and microgravity conditions, researchers conducted experiments during a parabolic flight campaign with Zero-G Corporation on May 20, 2022 [228, 229].

The experimental setup involved a sealed glove box containing a 1-inch layer of lunar regolith simulant [230, 231], with a mounted RGBD camera and a rover wheel. The camera was focused on a 3D-printed wall positioned across from the glove box. During the experiments, the rover wheel was activated to disturb the lunar simulant, and the camera recorded the outcomes. Five lunar gravity parabolas and ten microgravity parabolas were utilized. The lunar gravity parabolas simulated rover movement on the Moon, while the microgravity parabolas simulated a worst-case scenario where the kicked-up dust remained suspended.

The results of the experiments demonstrated that the camera effectively functioned in both lunar gravity and microgravity environments, albeit with some visibility loss caused by the lunar dust. Despite the dust interference, the camera was capable of collecting depth data successfully. These findings will play a crucial role in enhancing the design of cameras for future lunar missions, ensuring better performance and reliability. Prior to the flight, the experiment was sealed shut, necessitating an operational system accessible from outside the box. To achieve this, the wiring for both the Azure camera and the electronics of the light and wheel motor was fed through the polycarbonate walls. The RGBD camera was operated using the provided recording software from the Microsoft Azure Kinect DK, specifically employing the k4arecorder.exe operation. To avoid data loss mid-flight, thirty seconds of data

were recorded for each parabola, with each recording saved to a separate file instead of continuous recording. The light and wheel motor were operated via a custom PCB, with the wheel controller positioned outside the box. One of the flyers initiated and stopped the run to coincide with the parabola's beginning and end, ensuring better synchronization with the flight path [216].

Solar Simulation Testing:

The Microsoft Azure Kinect ToF camera emits a near-infrared image with a wavelength of 850 nanometers. Although it is not the peak wavelength for solar light, it still impacts the camera's functionality in Earth daylight due to atmospheric dispersion, which can saturate the infrared sensor. However, on the lunar surface, sunlight is functionally directional, and the lack of atmosphere enhances its power. To assess the RGBD camera's performance in worst-case lunar solar conditions, a solar simulator was utilized. The lunar surface's solar flux for 850 nm (no atmospheric dispersion) is approximately 0.97 $Wm^{-2}nm^{-1}$. A 110,000 lux (1.02 $Wm^{-2}nm^{-1}$ at 850 nm) full-spectrum halogen bulb was employed to simulate solar illumination for the lunar surface. The RGBD camera was set up within the glovebox used for dust interference, using the same lunar highland regolith simulant to emulate the lunar surface's albedo. RGBD videos were recorded for four different box positions. In Appendix B Table B.5, the estimated lux and flux values at 850 nm for each position are listed [216].

Power and Data Rates:

Two of the most critical limiting factors for the Commercial Lunar Payload Services (CLPS) mission are power and data rates. Specific values are not disclosed due to confidentiality, imposed by Lunar Outpost, but they are described as either within or exceeding the set limits. The rover has limited power available for its multiple payloads, and data transmission is constrained by download limits in the MB range. Given that a single 10-second RGBD (.mkv) video can require up to 1 GB of data, this severely restricts the data collection for the RGBD camera. For some ConOps, a minimum of 12 views from different directions is required, which could result in up to 120 GB of data for a single scene. However, much of the data in the 10-second video may be extraneous.

To address this, the AKALL software uses a data reduction technique. It initiates several single capture sequences, picks and stores the last frame, and then parses a single still RGBD image from the end of the recording. Only this image is saved for transmission to the lander, ensuring that the image has stabilized without transmitting unnecessary data. Although some data is lost using this method, such as gaps caused by dust, light reflection, and noise, it allows for increased image capture, albeit at a slightly lower quality. Initial power tests were conducted to assess the average power requirements for different camera modes: off, idle, streaming using the K4aviewer, and recording using the K4arecording software. The baseline power requirements (listed in Appendix B Table B.6) were used to ensure that the cam-

era would not draw power when off or idle and would remain within the mission's maximum power requirements during further development. The power and data requirements for each capture mode were evaluated in a laboratory setting using an unaltered Microsoft Azure Kinect mounted on a custom payload tower at the same height as the Lunar Outpost rover-mounted Kinect (6.75 inches above the ground). Capture sequences were conducted for each mode while monitoring the power. With the AKALL software, the power and data loads were tested for various combinations of RGB resolution and ToF camera resolution [216, 195].

ConOps Testing: Svalbard, Norway:

The Microsoft Azure Kinect was also tested at the same Marblehead, MA field site in 2022 [182]. The assessment was carried out using the Microsoft Azure Kinect DK software and involved manual manipulation of the camera position to evaluate the baseline ConOps for data collection. Data was collected by manually rotating the camera on a marked platform in 60^{o} increments, allowing for 5% overlap on each side of the image for the narrow field-of-view setting. A 10-second video was captured at each position while moving the camera on a tripod in a raster-style pattern with 3-meter increments over a (10 x 10 square meters) field site. This data collection approach provided valuable insights into challenges, particularly in terms of data gaps caused by the 360^{o} rotation and grid-style capture.

A second field test was conducted in Svalbard, Norway, to assess two different capture styles: grid-style data capture and raster-style data capture. Svalbard, known for its exposed geology, minimal vegetation, and tundra conditions, served as a valuable planetary surface analog. While not a true Lunar analog, these outdoor tests provided important developmental insights for the Lunar mission. The MIT RESOURCE team conducted preliminary tests of concepts of operations using the Microsoft Azure Kinect in conjunction with the development of a virtual environment to test the effectiveness of VR for Lunar and planetary surface exploration.

The tests involved two capture styles: raster and grid, to address blind spots observed in the Marblehead fieldwork. The raster pattern provided long swathes of data side-by-side, while the grid pattern offered 360^{o} blocks of data with central blind spots that could be partially eliminated by overlapping data from adjacent grids. The Open3D software was utilized for capturing, processing, and visualizing data from the RGBD camera, resulting in the creation of highly precise and detailed 3D models from individual recordings. This software allowed for a single video capture approach, streamlining the generation of comprehensive 3D point clouds without the need for manual compilation of separate images or videos.

To ensure camera stability during recording, a manual data collection approach was adopted, where the RGBD camera was held at a nearly constant height and moved in selected patterns over the field site. This method enabled faster data collection, which was crucial in low temperatures, while ensuring smoother video recording. The RGBD

camera traversed the field site in two patterns, capturing data as .ply video recordings for complete (10 x 10 square meters) science station captures at three distinct field sites. These data captures were used to assess 3D reconstruction ConOps for both the traverse capture mode and the science station capture mode [232].

Figure 6-11: *A view of the NASA Ames regolith test bed featuring the Azure Kinect camera during its testing phase. This image provides a real-world representation of the camera's deployment in a lunar analog environment*

ConOps Testing, SSERVI Lunar Sandbox:

The SSERVI Lunar Testbed, (as shown in Figure 6-11), is a (4 x 4 x 0.5 cubic meters) Testbed filled with 8 tons of Lunar lowland simulant. This Testbed was equipped with a simulated crater, small rocks, and ejecta features like rockslides and debris flows. It served as a suitable environment to test various capture modes, including the single capture mode, circumferential capture mode, and a potential future mode with gimballing capabilities for changing the camera angle. Additionally, the AKALL capture modes and Open3D software, (Figure 6-12), were evaluated to generate detailed 3D reconstructions of the environment. These reconstructions facilitated analysis of image quality and optimal rover positions in relation to the sun angle to minimize washout effects [232].

Five testing scenarios were conducted within the JSC-1A Testbed, each employing four different camera settings (as listed in Appendix B Table B.7). For tests 1 and 2, the camera was manually moved around the object at 60^{o} intervals while maintaining a 1m distance from the object. Tests 3 to 5 involved positioning the camera on the edge of the crater and adjusting it to five different camera angles: yaw left and right (+/-30^{o}), pitch up and down ($+12^{o}$/-13^{o}), and neutral central. These tests demonstrated

individual image capture possibilities for potential future missions with gimballing capabilities. Each of these scenarios was repeated for six simulated sun positions: 0^{o}, 60^{o}, 120^{o}, 180^{o}, 240^{o}, and 300^{o}, measured from the central line of the camera, with 0^{o} behind the camera [232].

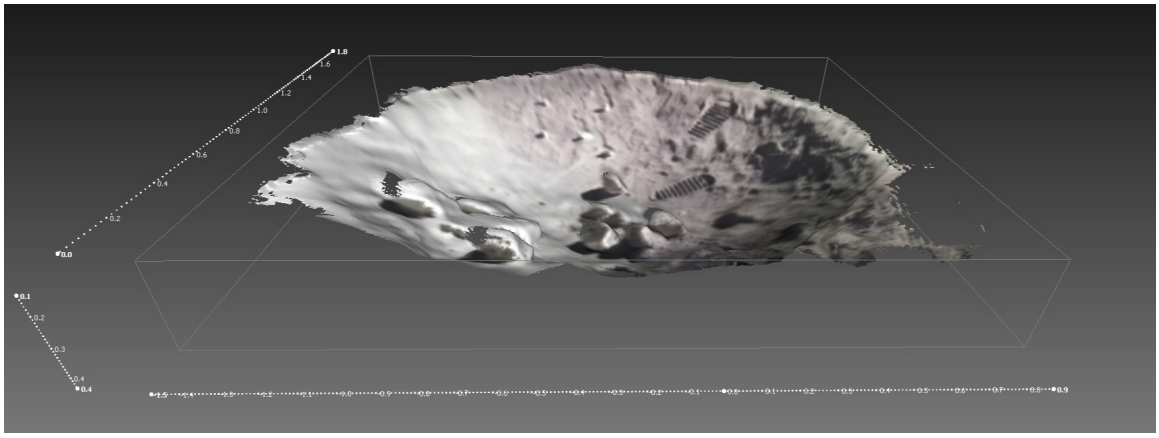

Figure 6-12: *This image showcases a detailed 3D reconstruction of the SSERVI regolith simulant Testbed at NASA Ames, achieved using the Azure Kinect camera. The camera's advanced depth sensing capabilities have enabled the accurate capture of the regolith's surface features and topography.*

6.4.2 ConOps Analysis and Results

A series of comprehensive tests, comprising both laboratory and field experiments, were conducted to assess the capabilities of the RGBD camera under Lunar conditions and optimize the ConOps design. These tests addressed crucial aspects, such as dust interference, solar flux without atmospheric dispersion, power and data rates, and capture methodologies for 3D reconstruction. The results of these tests provide valuable insights and implications for the success of the project.

To evaluate the camera's performance in the presence of Lunar dust interference, parabolic flights were utilized to simulate Lunar gravity. The camera's functionality amidst interfering particles and the impact of dust exposure on visibility were studied. Motorized rover wheels were employed to agitate the dust and simulate rover movement while recording the camera's output. These experiments provided valuable insights into the camera's performance in Lunar gravity and its ability to withstand dust interference.

Solar simulation testing was conducted to evaluate the camera's response under intense solar illumination on the Lunar surface, where atmospheric dispersion is absent. Using a solar simulator, the camera's performance was assessed under different levels of solar flux, enabling researchers to understand its behavior under challenging lighting conditions.

Power and data rates were critical factors considered for the success of the CLPS mission. To optimize camera capture modes, power requirements were evaluated to ensure compliance with specified limits. Data rates were also analyzed to manage the transmission of large data volumes effectively. The AKALL software facilitated testing power and data loads for various camera resolution combinations, providing insights into efficient data capture and transmission.

Field tests conducted in Svalbard, Norway, and the SSERVI Lunar Testbed assessed capture methodologies and 3D reconstruction techniques. Svalbard, with its characteristics resembling planetary surfaces, served as an important analog environment. The Open3D software enabled detailed 3D model generation from RGBD data, with raster and grid capture styles eliminating previous blind spots. These field tests provided crucial insights for the refinement of capture methodologies and stability improvement. The analysis of test results has significant implications for the project. The camera demonstrated its capability to capture high-resolution data in challenging Lunar conditions. Optimized capture modes, power management, and data reduction strategies ensure efficient resource utilization without compromising data quality. Insights from field tests and analog environments will contribute to the development of robust ConOps for future Lunar missions. These findings foster enhanced data collection, improved 3D reconstruction techniques, and support the successful execution of Lunar exploration missions [232].

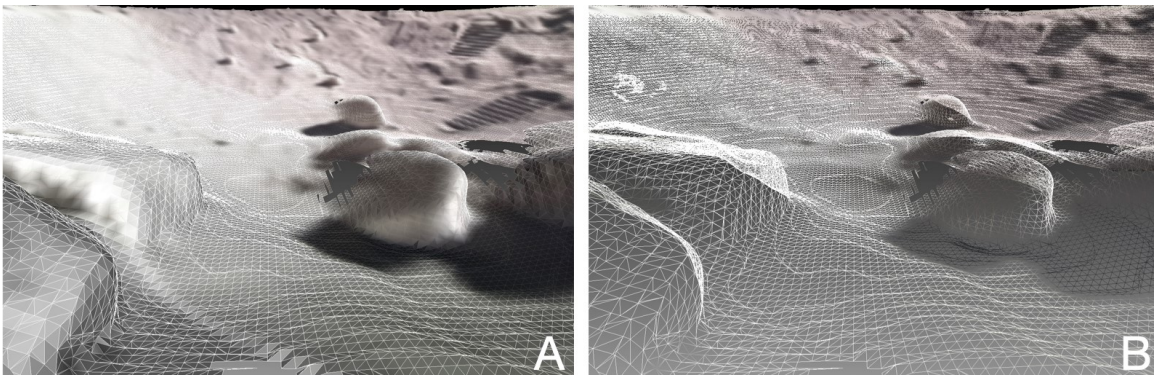

Figure 6-13: *Image A presents a close-up view of the textured mesh, capturing with details, the surface of the simulated lunar regolith. Meanwhile, Image B unveils the wire-frame representation, revealing the underlying structure and geometry of the scene.*

6.4.3 Software Testing: Analysis and Results

The AKALL software functioned as expected during the data collection at the SSERVI Lunar Testbed, successfully meeting the specified data requirements and demonstrating effective download formatting. Due to the absence of custom rendering capabilities in the AKALL software as of date, the Open3D software was utilized to capture representative still videos, simulating the single image captures taken by AKALL. These stills were subsequently processed using Open3D to generate 3D images, resembling the AKALL captures, (Figure 6-14). By employing MeshLab, (Section 5.3.2), to

stitch together these images, the creation of 3D scenes allowing us to compare the methodologies of single image capture and continuous video capture during movement. This process also established the data pipeline to be used in conjunction with the future custom rendering software.

The experiment conducted at NASA Ames provided significant insights regarding the degrees of overlap required for a comprehensive circumferential 3D reconstruction, optimal sun direction determination through the crater test, and a comparison between image reconstruction and video reconstruction. Furthermore, the analysis of the astronaut boot print, designed and printed by Ferrous Ward, (Figure 6-15), and crater data yielded valuable information regarding the obtainable feature resolution. The data set features 180 samples captured with the Azure Kinect device in various light conditions and scenarios, (as shown in the results in Figures 6-16 to 6-22), they show a direct comparison between the file sizes generated by three different software tools: K4A, AKALL, and Open3D. K4A, a software pack ,developed by Microsoft, that includes various utilities and drivers for the Azure Kinect DK, stores streams of data from the device into the Matroska multimedia container (.mkv), commonly used for video formats. AKALL, on the other hand, adopts its own custom data format (.nd3), (Section 6.4.3), that stands for the Natural Depth 3D format, which is composed of 16-bit Grayscale images and metadata compressed in a gzip container. Open3D employs a different approach, organizing data in a hierarchical folder structure that contains various scene information. This software provides functions to reconstruct 3D footage captured using depth cameras into the Polygon file format (.ply), (as described in Chapter 5, Section 5.5).

The results from these comparisons indicate that both the .nd3 file format and AKALL software result in significantly smaller file sizes compared to the K4A recordings that capture .mkv files at the smallest allowed interval (1-second video), as well as the .ply file generated by Open3D. The terminology list, in Appendix D.1, is utilized to comprehend the naming convention of the data files, distinguishing various data capture scenarios, locations, objects, and light properties. For instance, "BR_L300" denotes that this file comprises experiments featuring the Bed Rock object, with "L300" signifying the 300° angle of the light source in relation to the camera's origin. Note that the plots use a log scale to expand relative differences between .nd3, .mkv and .ply formats. The difference in File size between .mkv and the other file formats are well over an order of magnitude.

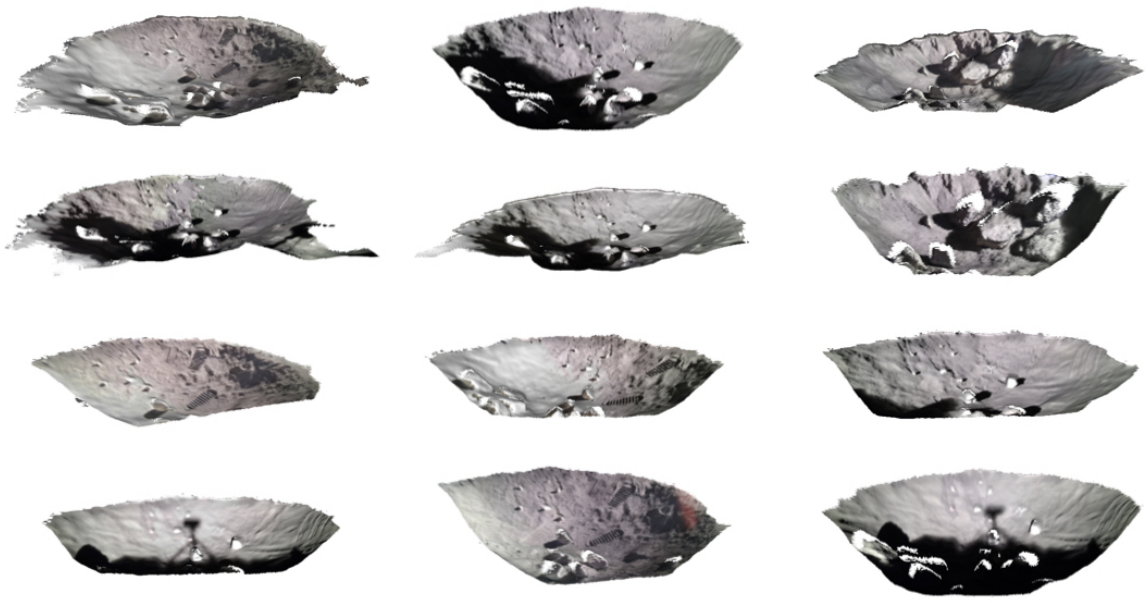

Figure 6-14: *3D reconstruction performed of various lighting conditions using the Azure Kinect device and processed with Open3D and Meshlab.*

Figure 6-15: *This captivating image showcases the intricate detail of 3D printed soles designed to replicate the footprints left by astronauts during extravehicular activities (EVAs). These custom soles faithfully recreate the distinctive patterns and textures of the original boot imprints, allowing for a precise and accurate representation of the astronaut's footsteps. Image courtesy: Cody Paige and Ferrous Ward.*

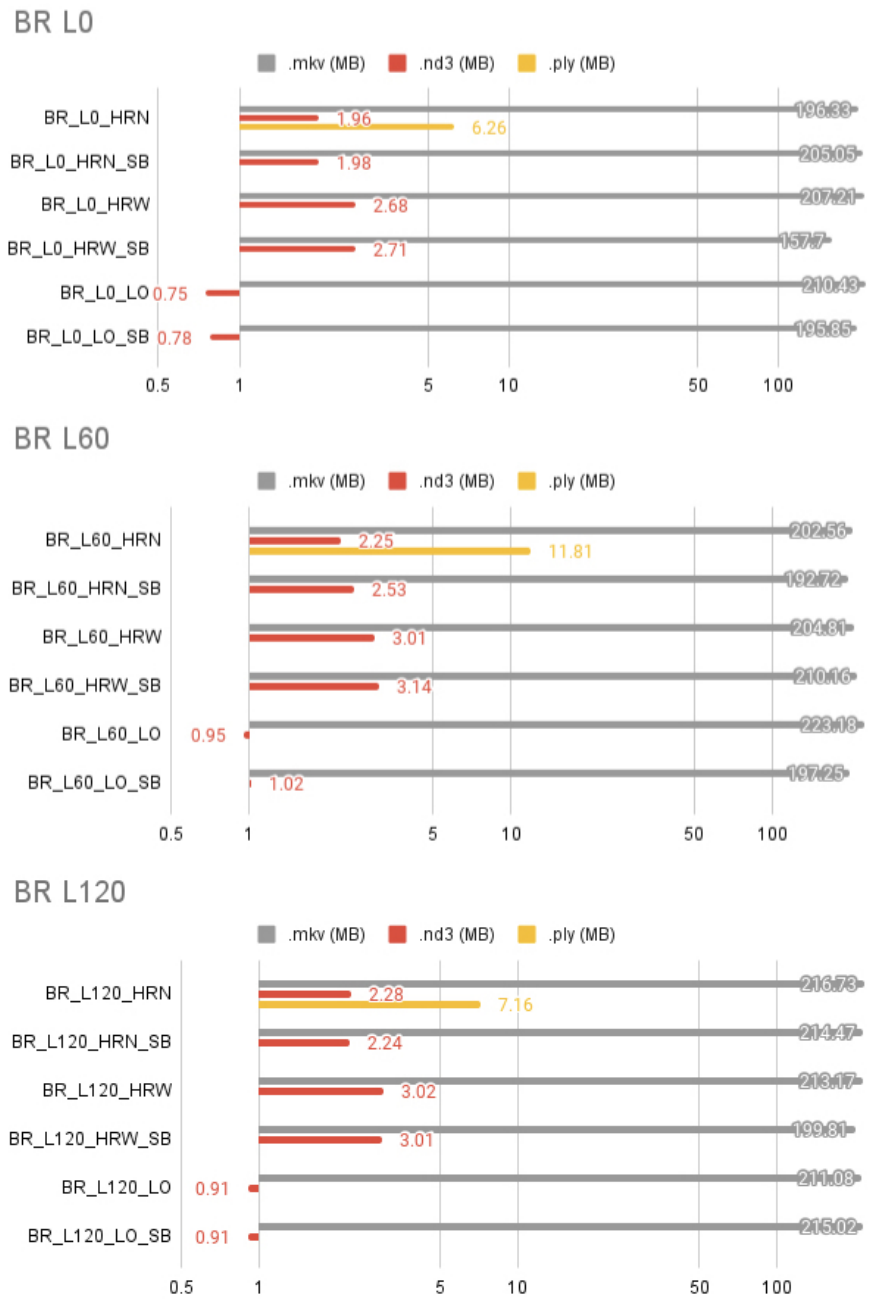

Figure 6-16: *Comparison of file sizes (MB) shown on log scale: K4A (.mkv), AKALL (.nd3), and OPEN3D (.ply) for the Bed Rock Object with Varying Light Angles (0°, 60°, and 120°)*

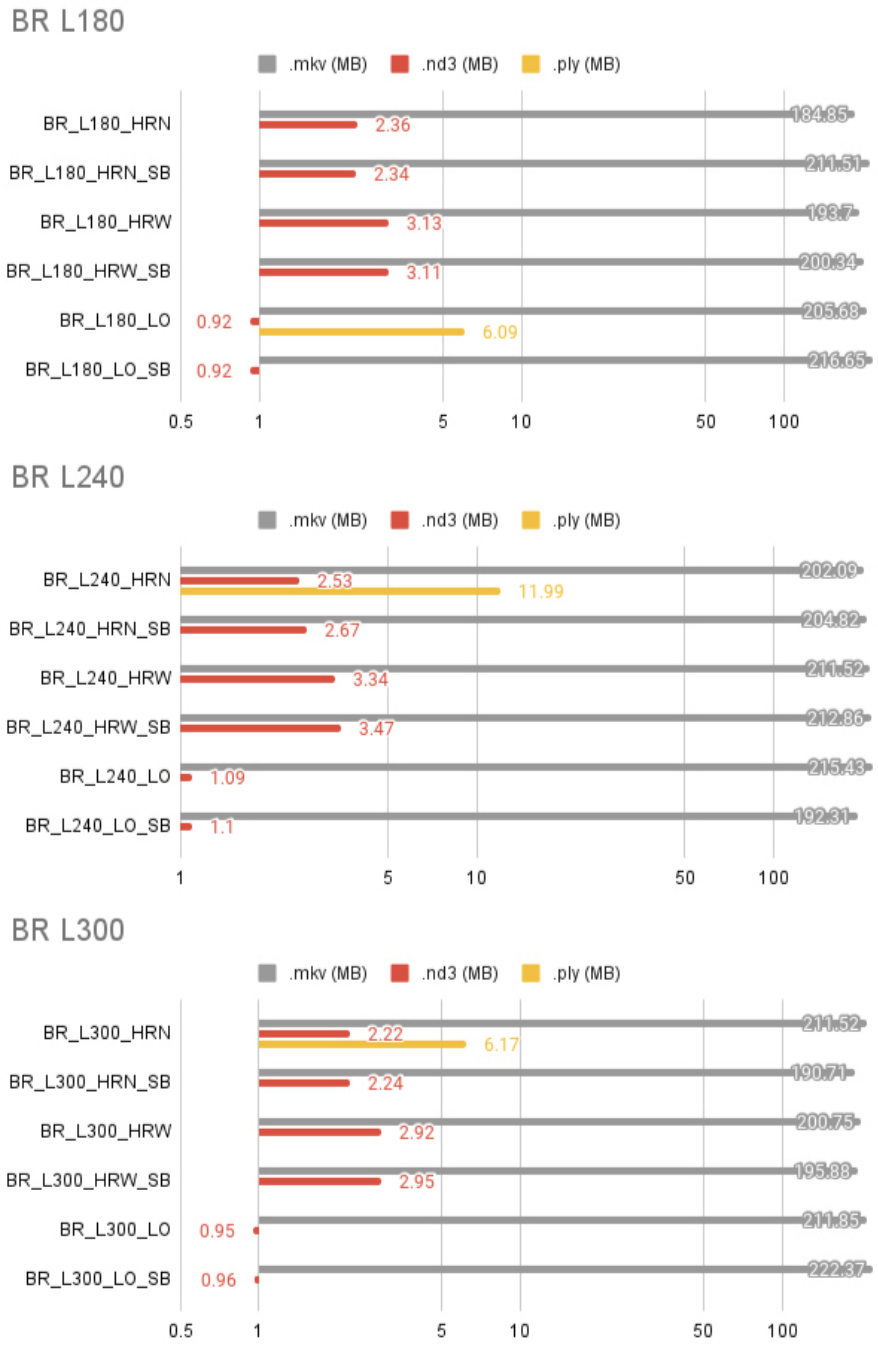

Figure 6-17: *Comparison of file sizes (MB) shown on log scale: K4A (.mkv), AKALL (.nd3), and OPEN3D (.ply) data types for bed rock object with varying light angles (180°, 240°, and 300°)*

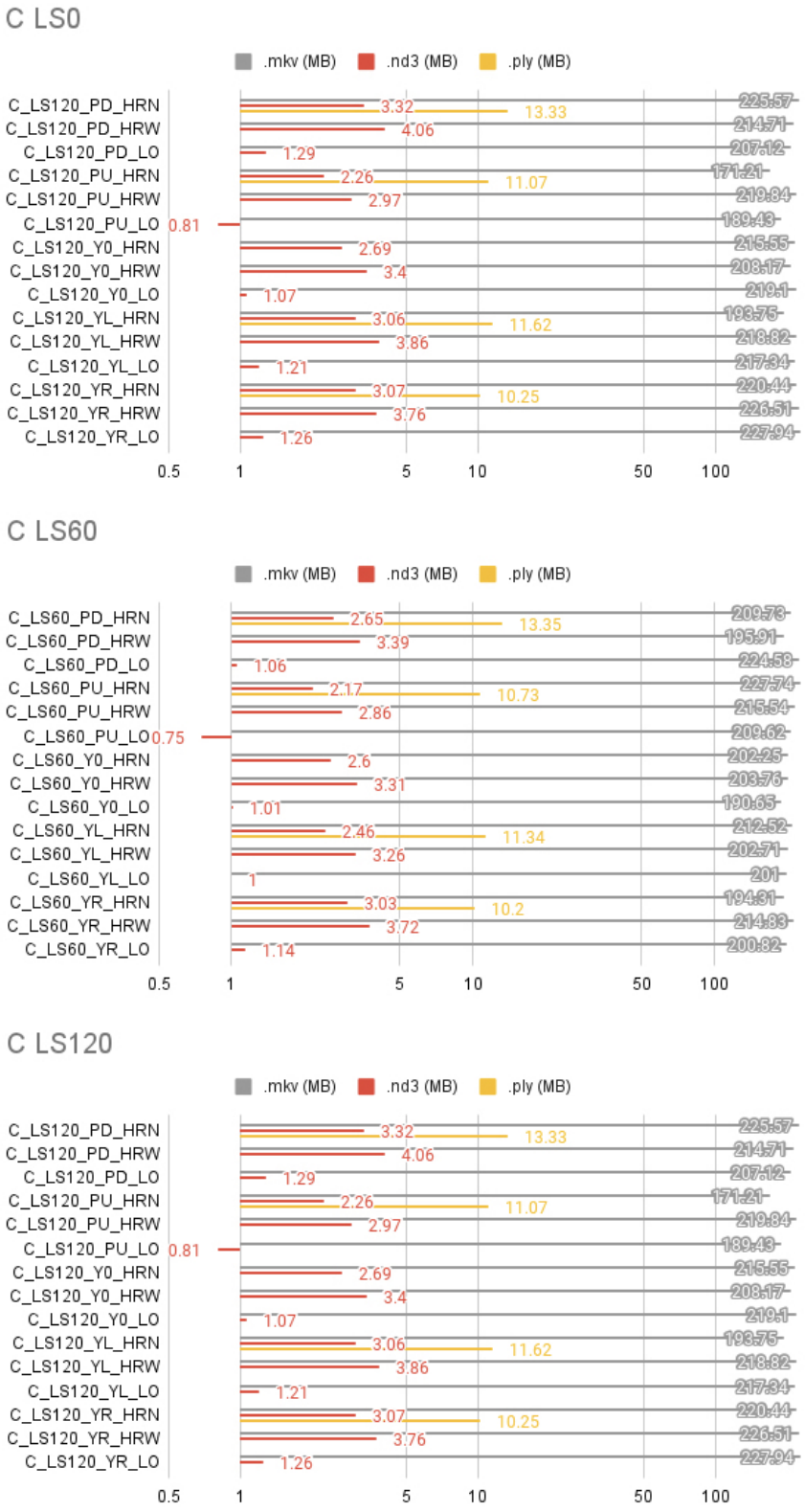

Figure 6-18: *Comparison of file sizes (MB) shown on log scale: K4A (.mkv), AKALL (.nd3), and OPEN3D (.ply) data types for crater capture with varying light angles (0°, 60°, and 120°)*

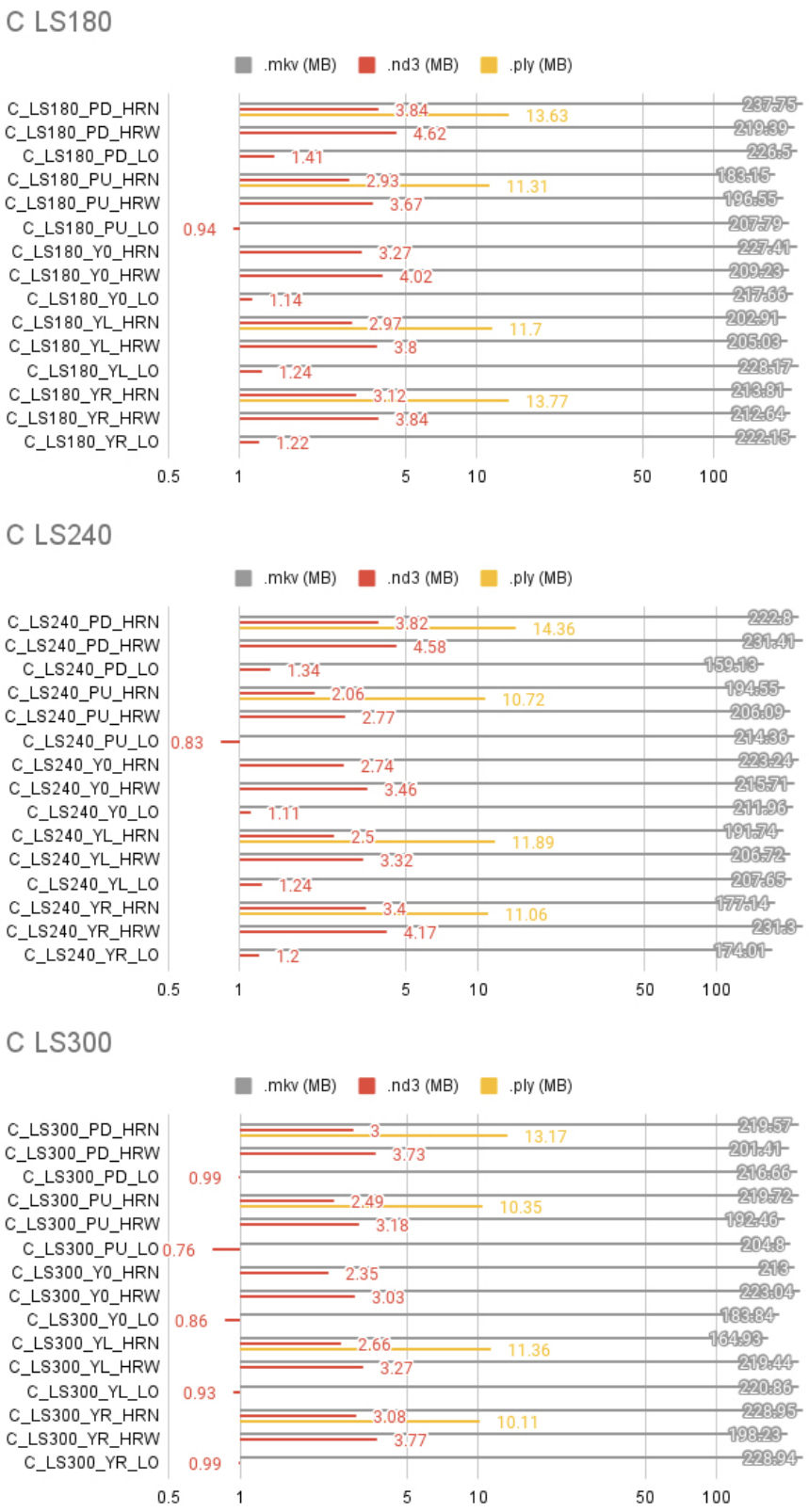

Figure 6-19: *Comparison of file sizes (MB) shown on log scale: K4A (.mkv), AKALL (.nd3), and OPEN3D (.ply) data types for crater capture with varying light angles (180°, 240°, and 300°)*

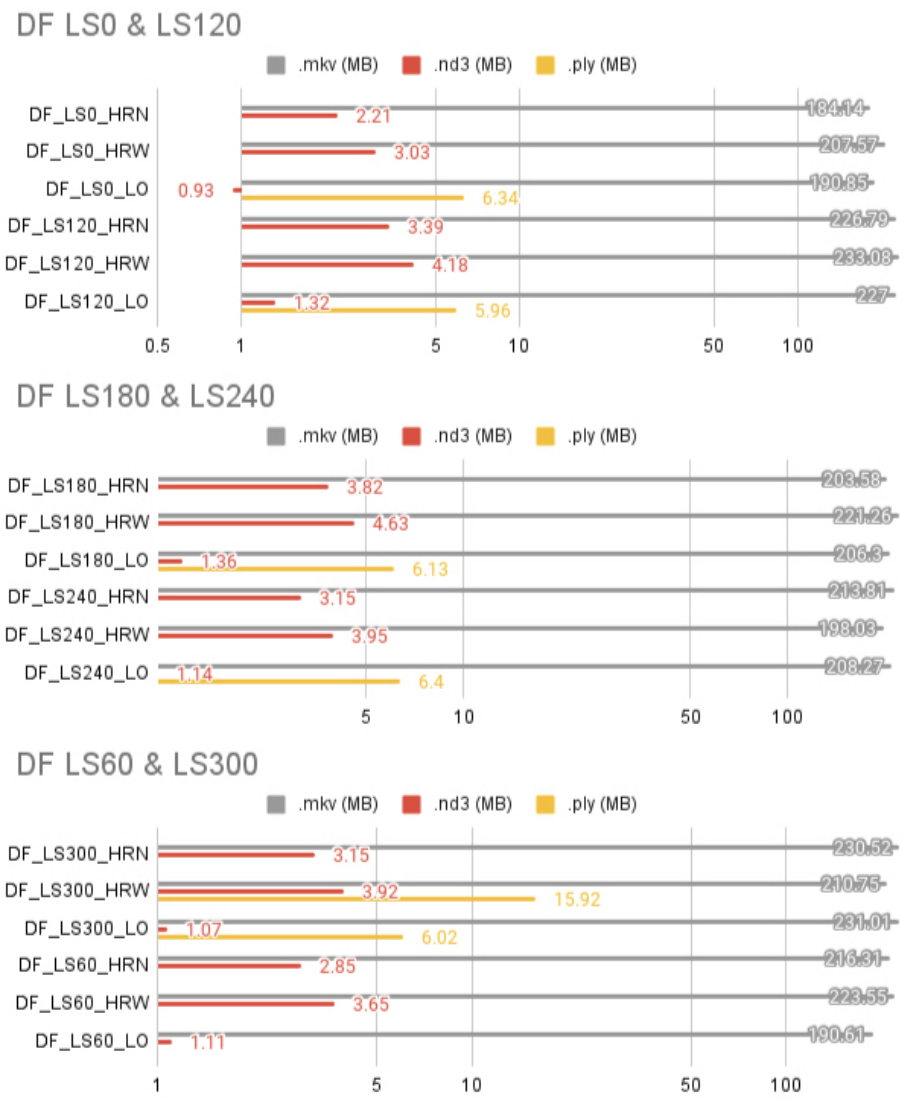

Figure 6-20: *Comparison of file sizes (MB) shown on log scale: K4A (.mkv), AKALL (.nd3), and OPEN3D (.ply) data types for debris flow capture with varying light angles (0° to 300°)*

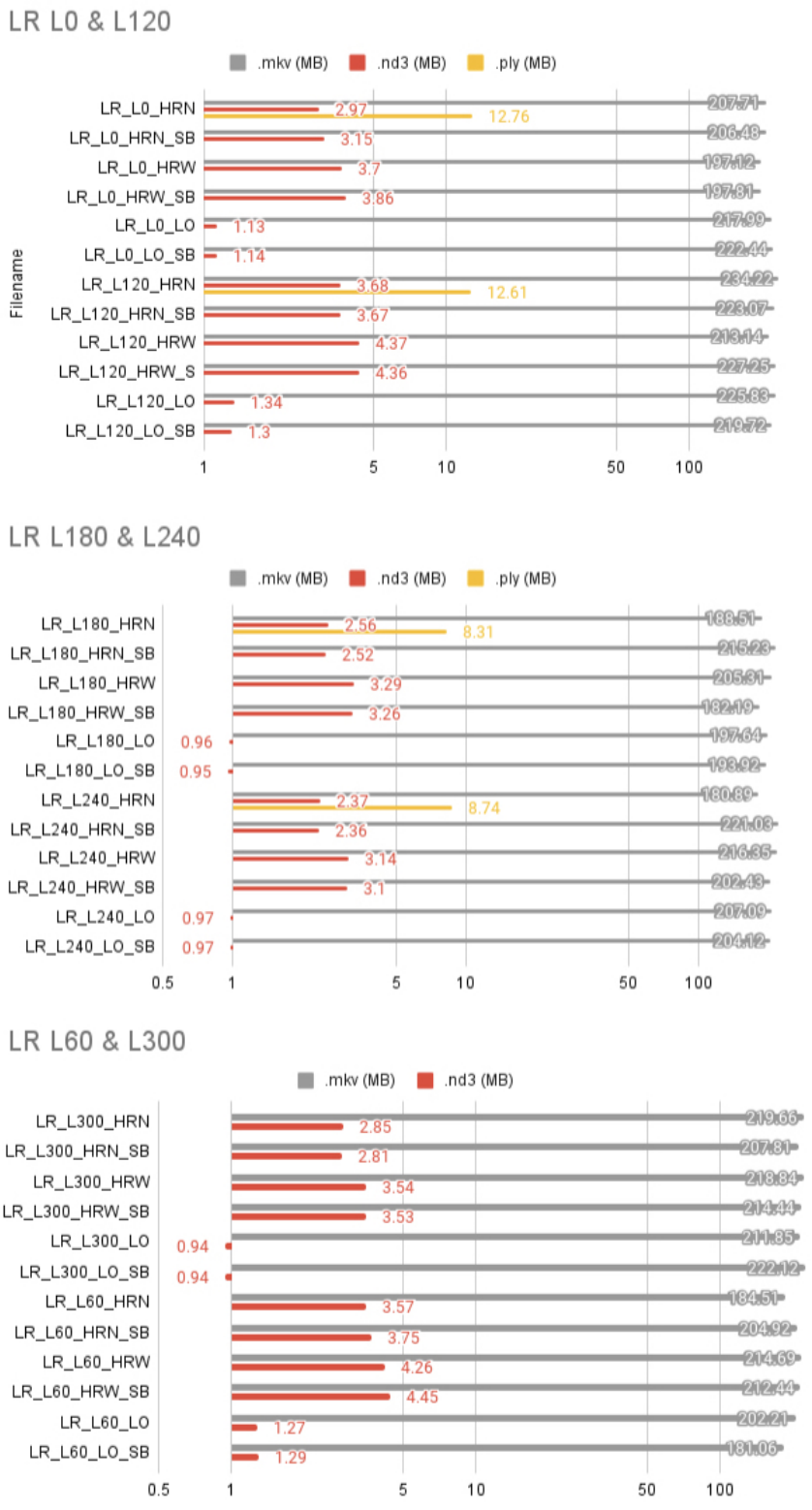

Figure 6-21: *Comparison of file sizes (MB) shown on log scale: K4A (.mkv), AKALL (.nd3), and OPEN3D (.ply) data types for little rock object with varying light angles (0° to 300°)*

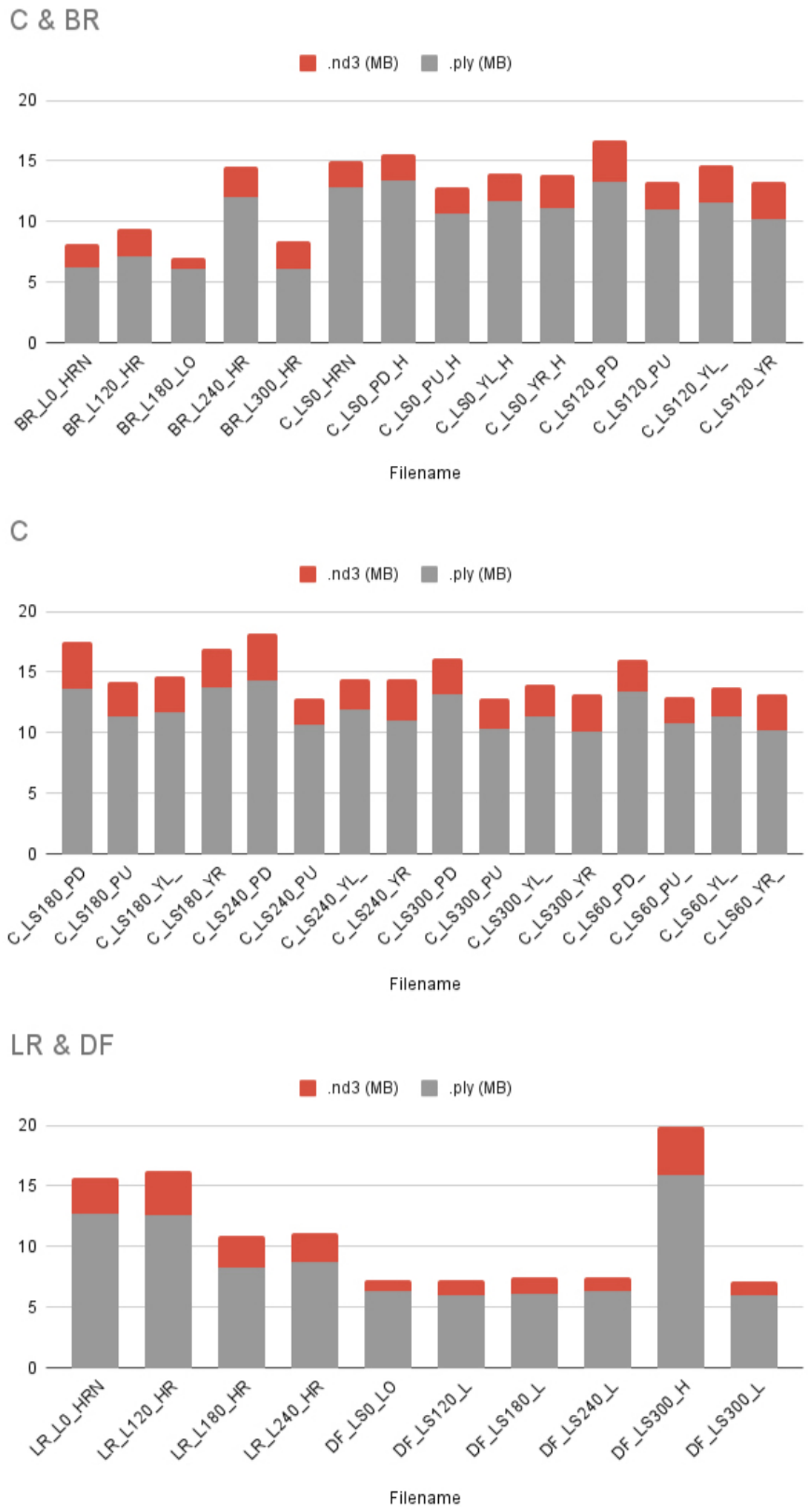

Figure 6-22: *Comparison between AKALL (.nd3) and OPEN3D (.ply) file sizes (MB) in various scenarios and lighting conditions.*

6.5 Conclusion and Future Work

6.5.1 ".ND3": File Format for RGBD Imaging

The Azure Kinect device implements the Amplitude Modulated Continuous Wave (AMCW) ToF principle for depth collection, (as detailed in Section 6.1.2). The device emits modulated illumination within the near-IR (NIR) spectrum [233]. It subsequently captures an indirect measurement of the round-trip time taken by the light to travel from the camera to the scene and back [151]. These recorded measurements are then processed to create a depth map (depth.b16g), which contains Z-coordinate values for each pixel in the image, measured in millimeters. Additionally, alongside the depth map, a so called "clean IR reading" (ir.b16g) is recorded. The pixel values in this reading correspond to the quantity of light reflected back from the scene, providing valuable information about the light in the scene. The second sensor included in the Azure Kinect allows for capturing color images with native support for JPEG and MJPEG compression. One of significant contributions highlighted in this research involved optimizing the data generated by the Azure Kinect, through AKALL, and developed a novel data format called **.ND3**. "ND" stands for "Natural Depth", metaphorically connecting it to the illumination provided by, natural or artificial sources, *Natural Light*: as it entails that the ".ND3" file extension is tailored for processing, capturing and storing the needed information to synthesize 3D Objects from 2D images [234], therefore making good use of existing image compression algorithms such as the Joint Photographic Experts Group (JPEG) [235, 236]. This uses three main components, a color image for example (**color.jpeg**) depth image (**depth.b16g**) and calibration data (**calibration.json**). Optionally, the (**ir.b16g**) image could be utilized to further focus on rending light texture maps, through advanced mapping techniques to achieve a more immersive and realistic final 3D Object.

6.5.2 Summary

The Azure Kinect à la Luna (AKALL) project was undertaken with the aim of creating a versatile and portable application that harnesses the extensive capabilities of the Azure Kinect device, specifically in the context of space exploration and robotics. The project set out to achieve multiple ambitious objectives, which are summarized in this chapter. One of the primary goals of the AKALL software was to develop an application that seamlessly integrates into existing computing systems, such as rovers and robots, without disrupting established workflows. By fitting within the architecture of these systems, the AKALL application can enhance capabilities and provide new functionalities. Another crucial objective was to ensure high portability and isolation.

This was achieved by utilizing Docker containers, which encapsulate the application and its dependencies, enabling deployment across different environments with minimal configuration efforts. By leveraging Docker, the AKALL application remains isolated from the host system, preventing any issues within the application from impacting the wider system. A robust communication mechanism was also a key objective of the

AKALL software. The application implemented a UNIX socket server, allowing seamless interaction between the AKALL application and other programs or devices using the UNIX socket protocol. This communication mechanism facilitates control of the camera and data capture using custom capture sequence messages, enabling efficient and customizable data collection. Furthermore, the AKALL software aimed to fully utilize the data capturing capabilities of the Azure Kinect device. The application offers a high level of control over the device's parameters, allowing users to customize the data capture process to their specific requirements. The ability to adjust parameters such as frame rate, compression, resolution, exposure, and more provides users with a flexible and tailored data collection experience. Upon completion, the AKALL software is expected to have significant outcomes and contributions. The application's portability, flexibility, and robust communication abilities open up numerous possibilities for developers and researchers to utilize the Azure Kinect device in innovative ways. With potential deployment in space exploration and robotics, the AKALL software holds promise for making substantial contributions in these fields.

To evaluate the capabilities of the Azure Kinect device under Lunar conditions, a series of tests were conducted, including laboratory and field experiments. These tests addressed various aspects, such as dust interference, solar flux, power and data rates, and capture methodologies for 3D reconstruction. Results from these tests provided valuable insights into the camera's performance and implications for the project. Dust interference, a significant challenge in Lunar conditions, was evaluated by conducting tests using a parabolic flight to simulate Lunar gravity. The camera's functionality and visibility in the presence of dust interference were assessed, providing insights into its performance and settling time required for capturing depth-data after dust agitation. Solar simulation testing was conducted to evaluate the camera's response to intense solar illumination on the Lunar surface, considering the absence of atmospheric dispersion. These tests aimed to ensure the camera's robust performance under such conditions. Optimizing power and data rates was crucial for the success of the CLPS mission. The AKALL software was utilized to assess power requirements and data management strategies for various camera modes. Capture sequences were optimized based on specific mode requirements, ensuring efficient use of limited resources while maintaining data quality. Field tests conducted in analog environments, such as Svalbard, Norway, and the SSERVI Lunar Testbed, provided insights into capture methodologies and 3D reconstruction techniques. These findings contributed to the development of a robust ConOps for future Lunar missions.

6.5.3 IM-2 Mission Updates

Recent details have emerged [237] about the upcoming IM-2 mission, which marks a significant milestone in commercial space exploration. Nokia Bell Labs [1] is at the forefront of this groundbreaking endeavor, deploying the first-ever cellular network on the Moon. The mission aims to demonstrate that cellular technologies can fulfill

[1] bell-labs.com

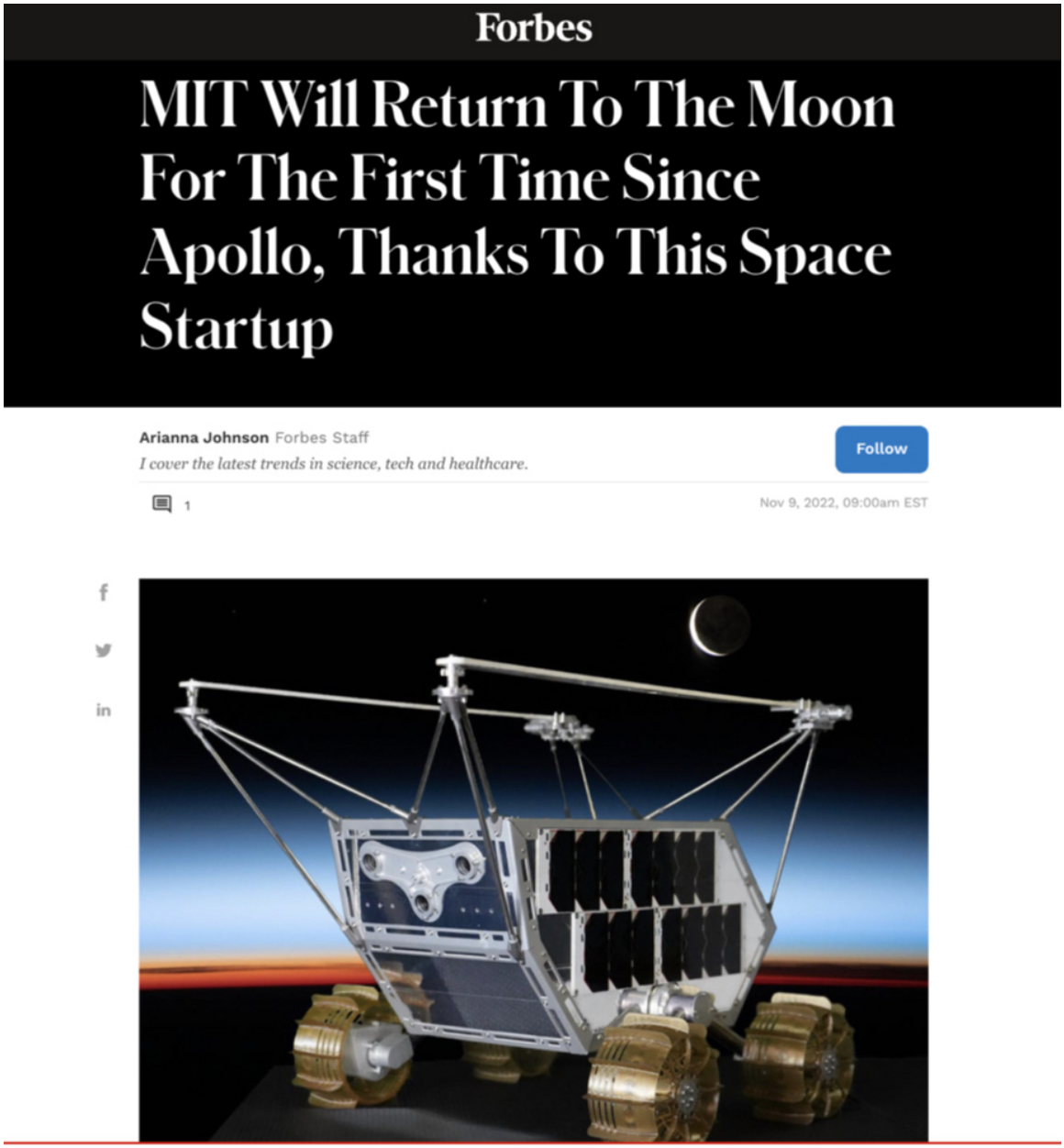

Figure 6-23: *MIT's Lunar Payloads: Announced as of Nov 8, 2022 MIT contributes two innovative payloads to the Lunar Outpost's Mobile Autonomous Prospecting Platform (MAPP) rover. Image courtesy: Forbes*[a].

[a]MIT Will Return to the Moon for the First Time since Apollo - Forbes.com

the critical communications needs of future lunar and Martian missions. Collaborating with Intuitive Machines[2] and Lunar Outpost[3], Nokia has developed a low-power, space-hardened version of its 4G/LTE MicroCell, specially designed to withstand the extreme conditions of the lunar surface. Scheduled for launch in November 2023, the IM-2 mission holds great promise for advancing space technologies. The integration of a cellular network will play a vital role in enabling reliable, high-capacity, and efficient connectivity for both crewed and uncrewed missions to the Moon and other planets. This connectivity will be crucial for astronauts, who will require advanced communication capabilities to support their missions and daily activities [237].

Through several collaborations, MIT was able to incorporate two payloads, (Figure 6-23), on board the MAPP rover. One of these is the RESOURCE camera, presented in this chapter, running AKALL, to enable MAPP to capture ".nd3" files that will later be transformed into 3D Objects of distinctive rock formations and craters on the moon. It will diligently search for indications of lunar ice in the shadows near the crater edges. Additionally, MIT Media Lab's Responsive Environments group has developed the AstroAnt, a miniature rover that will ride atop the Lunar Outpost rover. Comparable in size to a Matchbox toy car, the AstroAnt will explore the MAPP's roof, collecting essential temperature data while the MAPP rover traverses the lunar surface. The valuable images, data, and telemetry collected by MAPP will be transmitted via the 4G/LTE network to the lander and then relayed back to Earth. Through this connection, Lunar Outpost mission control center in Colorado will efficiently issue commands to the rover, ensuring smooth operations during the lunar mission [237].

As part of the IM-2 mission, Nokia's 4G/LTE network will consist of a base station unit integrated into the Nova-C lander and radio equipment installed on the Lunar Outpost's MAPP rover, (Figure 6-2), and Intuitive Machines' Micro-Nova hopper. These components will form a network allowing seamless communication between the lander and the vehicles on the lunar surface. The direct-to-Earth radio connection will enable mission controllers to receive essential data, images, and remotely operate the vehicles over the cellular network. The insights gained from the IM-2 mission[4] hold significant potential for both lunar and Martian economies. Cellular networks will serve as a fundamental infrastructure, linking sensors, transport vehicles, scientific payloads, drones, and rovers. The network will also enable remote operation of essential machinery, such as mining and construction equipment, vital for astronauts' survival. The IM-2 mission will provide invaluable data and knowledge for advancing communications technologies in space exploration. As Nokia Bell Labs has a rich history of contributing to space science, this mission adds another milestone to their legacy of innovation and exploration in the realm of space technology [237].

[2]intuitivemachines.com

[3]lunaroutpost.com

[4]An Inside Look at Nokia's Moon Mission - Spaceref.com

6.5.4 Future Work

Data compression plays a crucial role in space exploration, as exemplified by the Galileo NASA mission to Jupiter launched in 1989 [224], and remains a common practice in modern space missions. Facing challenges of limited communication bandwidth and vast distances, efficient data compression is vital to optimize data transmission and storage. By utilizing lossless and lossy compression techniques [238], space missions can transmit and analyze vast amounts of scientific data, maximizing mission success and enabling deeper insights into celestial bodies across the solar system. The .nd3 file format holds great potential for shaping the landscape of RGBD devices and their corresponding software environments. As hardware devices continually evolve and adapt, the seamless encapsulation of data types becomes crucial in facilitating the creation of innovative hardware that aligns with the needs and availability of current software capable of processing such data files. Developing a companion media player, editing, file export, and processing tool tailored specifically for .nd3 files can play a pivotal role in advancing research conducted on RGBD devices in general. While the 3D industry predominantly operates using established formats like .stl, .obj, .ply, .fbx, and others, each format has been shaped by the requirements of specific software and hardware environments that standardized them.

However, in the context of space exploration, where data size and efficiency are paramount, there emerged an urgent need to devise a mechanism for efficiently storing RGBD data in the most compact format possible. This necessity laid the groundwork for the development of the .nd3 file format, as demonstrated through the research presented in this chapter. As a novel and optimized format, .nd3 provides an ideal solution for storing and processing RGBD data in space exploration missions. Further research can investigate even better use of compression techniques and algorithms to enhance the .nd3 file compression and 3D reconstruction capabilities. This ongoing exploration can lead to even more efficient data storage and transmission for different depth cameras in single capture mode, benefiting space missions and pushing the boundaries of what is possible with RGBD devices in the context of space exploration.

6.5.5 Acknowledgment

This chapter presented a corpus of research that owes its realization to the synergistic collaboration among multiple entities and individuals, including members of the MIT RESOURCE team supervised by Prof. Dava Newman, her students Cody Paige, and Ferrous Ward, as well as the contributions of the talented engineers and scientists at NASA Ames Research Center and Lunar Outpost.

Moreover, Dr. Jennifer Heldmann and Dr. Amanda Cook from NASA Ames Research Center, Dr. Ariel Ekblaw and Sean Auffinger from The MIT Media Lab Space Exploration Initiative, and Ben Brokaw and Matt Mitchel from Lunar Outpost played instrumental roles in shaping and supporting this research endeavor.

Chapter 7

Conclusion

7.1 Summary

In summary, this thesis highlighted the combination of multiple fields of research, weaving together virtual environment design, robotics, sensor systems, and space operations. By utilizing real-world data to generate virtual environments, it has forged a novel futuristic concept for mission planning and execution. This research explored the synergies and challenges associated with human-robot operations in virtual space analog environments, thereby shedding light on new methodologies and technologies. These innovative approaches not only offer practical solutions for current challenges, but also set the stage for future explorations and discoveries. This blend of technology and creativity serves as a testament to the power of interdisciplinary research and underscores the potential for continued advancement and innovation in space exploration.

The Doppelbot project, (Chapter 3), represents an innovative frontier in the field of virtual space exploration, introducing an immersive approach to mission planning and user interaction within a simulated lunar environment. Utilizing the concept of cross-reality, the project creates a digital twin of the Rover Mini by Rover Robotics, named Doppelbot, implemented within the Unity game engine. It simulates the actual rover's behavior and locomotion by offering independent motor control for each wheel based on user inputs, allowing for a wide range of interactive movements. A standout feature of the Doppelbot project is the incorporation of video game metaphors and user interface elements, including minimaps and mouse-controlled navigation, thereby enhancing user experience and navigation. Drawing inspiration from Real-Time Strategy (RTS) games, the interface applies a color-coded time-based visualization method, along with physics simulation, to offer users a conceptual visual understanding of the rover's planned, current, and past paths. This visualization is complemented by the inclusion of a minimap to enhance situational awareness.

Furthermore, a detailed system investigation has synchronized the Rover Mini with Doppelbot through the integration of advanced technologies such as Nvidia Xavier, In-

tel RealSense Cameras, Ubuntu running ROS (Robot Operating System), and SLAM (Simultaneous Localization and Mapping). This allows accurate spatial representation and seamless synchronization between physical and virtual environments. The chapter also proposes new visualization concepts, overlaying anticipated and current positions of the digital twin with the dimension of time, and explores the transformative potential of integrating large language models (LLMs) such as OpenAI's GPT-4. The inclusion of these LLMs within the Unity game engine promises to create more realistic, adaptive, and intelligent simulations

Two concepts revolving around robotic telepresence were introduced and explored in Chapter 4, specifically using a quadruped robot. The first project focuses on the creation of a Virtual Reality (VR) application that streams and replays 360° videos with LiDAR data, enabling a lifelike telepresence experience. The selected devices for this immersive project were the Insta360 ONE X VR camera and the Velodyne VLP-16 LiDAR, integrated into the Doppelspot concept. The success of this teleoperation system showed potential for use in scenarios unaffected by communication delays, including astronauts' virtual extra-vehicular activities. Doppelspot's integration introduces new possibilities for remote operation and data collection in challenging environments, and the chapter highlights the various capabilities, including integrated cameras, depth sensors, and Python API for customization.

Augmented Virtuality played a critical role by blending real captured environments with simulated ones, using a 360° video camera and LiDAR data. Virtual overlays and annotations provided real-time data and visualizations on the live video feed, enhancing the telepresence experience. The digital twin of Spot, named Doppelspot, was presented with features like procedural leg animation, contributing to a realistic telepresence experience. Future applications may involve offloading extra-vehicular activities to robots, improving situational awareness, and facilitating collaborative research. Future work for Doppelspot includes improvements to payloads and telepresence capabilities, with advancements in 3D scanning technologies and communication, offering the potential for enhanced data reconstruction and realism. The chapter also anticipates a shift towards low bandwidth 3D transmission and reconstruction. Overall, these insights aim to expand the applications and capabilities of robotic telepresence in space missions and exploration, focusing on optimization and potential future developments.

The Svalbard analog mission exemplifies the innovative application of digital twin analog environments in geological research, utilizing high-resolution 3D reconstructions, environmental data, and advanced visualization techniques to study the Svalbard site as a Martian analog. By employing aerial photography, environmental sensors, LiDAR/RGBD cameras, and Unity's HDRP and Post Processing Stack, the project offers a realistic and accurate representation of geological features that resemble both Earth and Mars, enabling insightful exploration and analysis. The virtual environments created are instrumental in enhancing the understanding of geological processes and potential extraterrestrial formations. Researchers can visualize and

analyze these features in great detail, with post-processing effects augmenting the visual quality. A study was conducted to evaluate OGIVE that highlights the impact of digital twin analog environments in geological exploration, providing an immersive platform to study formations remotely and efficiently, with future applications such as training tools for astronauts. Chapter 5 explores the application of 3D reconstruction and data visualization in virtual analog environments, emphasizing the Svalbard site. Technologies such as photogrammetry, LiDAR, and Unity's rendering capabilities have enabled the creation of accurate digital twins, facilitating exploration and understanding of other planets.

Photorealistic rendering and voice commands have enhanced the user experience, and user studies indicate that VR applications offer more spatial presence and immersion than desktop applications. Transporting the user to distant planets with an added feature of visualizing Earth features as Martian counterparts further enriches the exploration process, as well as equipping them with two fundamental tools for exploration, a ruler and a modality-shifting "flashlight". The OGIVE application demonstrates the effectiveness of these tools in geological analysis, highlighting their broader potential for scientific exploration, decision-making, and collaboration. Ongoing development, advancements in technology, and interdisciplinary collaboration will continue to push the boundaries of remote research, transforming the exploration of environments and aiding in the preparation of future Lunar and Martian missions.

The Azure Kinect à la Luna (AKALL) project was developed to create a versatile and portable application for space exploration and robotics, harnessing the capabilities of the Azure Kinect device. The key objectives of the project included seamless integration with existing systems like rovers and robots, high portability and isolation, robust communication, and parameterizing data capturing. The project achieved integration by fitting within existing architectures, and enhancing their functionalities. Portability and isolation were ensured using Docker containers, which allowed the application to be deployed across various environments without affecting the host system. The custom Python server handling UNIX domain sockets enabled seamless communication, facilitating control and access over the camera's underlining sensing modalities. The AKALL project took full advantage of the Azure Kinect's data capturing capabilities, providing control over parameters such as frame rate, compression, resolution, and exposure. This enabled a tailored data collection experience, and the application's flexibility and robust communication abilities promised substantial contributions to space exploration and robotics.

A series of laboratory and field tests were conducted by the MIT RESOURCE team, and by NASA Ames, to evaluate the device's performance under Lunar conditions, while providing the necessary protection to make this device viable in space operations. These tests addressed challenges like dust interference, solar flux, power and data rates, and capture methodologies for 3D reconstruction. Specific tests simulated Lunar gravity and intense solar illumination to assess the camera's performance and develop strategies for power and data management. Field tests in locations such as

Svalbard, in Norway, and the SSERVI Lunar Testbed at NASA Ames contributed insights into 3D reconstruction techniques and prepared a robust Concept of Operations (ConOps) for future Lunar missions.

A significant contribution of the research was the optimization of data generated by the Azure Kinect, through AKALL, and the introduction of a novel data format called .ND3, designed for processing, capturing, and storing information needed to synthesize 3D Objects from 2D images. It utilizes components such as color and depth images, calibration data, and optionally, an IR image for rendering light texture maps, aiming for more immersive and realistic final 3D Objects. Overall, the AKALL project demonstrates a forward-thinking application of technology in the field of space exploration, promising to enhance both robotic capabilities and our understanding of Lunar conditions.

7.2 Discussion

The pursuit of exploring distant space environments through robotics presents a complex challenge. The further we venture, the slower the data transmission becomes, and efficiency becomes paramount. This research has shed light on the necessity of bridging the gap between distant robotic exploration and human understanding by harnessing advanced techniques. Through combining automation and predictive modeling with cutting-edge visualization, the research emphasizes keeping the human in the loop. It explores innovative ways to synthesize sensorially scanned, and sampled environments from space, allowing us to comprehend and interact with far-off locations. The focus on making processes more efficient and better adapted to the constraints of distance highlights a promising direction in space exploration. This integrated approach ensures that as we push the boundaries of space, our ability to understand and explore does not lag behind. It sets a path toward more intelligent, responsive, and human-centric exploration of the cosmos, aligning our technological advancements with our innate human curiosity and adaptability.

One of the remarkable aspects of modern space exploration is the adaptation of commercially available technologies and products for space applications. This strategy not only leverages existing technological advancements but also fosters collaboration between commercial industries and space agencies. Adapting commercial products like the Azure Kinect, as seen in the AKALL project, transforms tools initially designed for other purposes into powerful instruments of exploration and discovery. By integrating these readily available resources, space exploration becomes more cost-effective, agile, and inclusive. It invites a broader community of innovators to contribute to the endeavor of exploring the unknown, bridging the gap between commercial technology and the specific needs of space exploration. This approach signifies a democratization of space technology, aligning commercial interests with the noble pursuit of knowledge and exploration, and ensures that space remains an accessible frontier for all.

This integration of commercial products into the space exploration landscape further emphasizes the holistic and interdisciplinary nature of space research. It demonstrates that the pursuit of understanding distant environments is not confined to specialized equipment and bespoke technology alone. Instead, it's a dynamic and flexible field that continually evolves, drawing from various sectors and adapting to the ever-changing challenges and opportunities of space exploration. It underscores the concept that space is not an isolated domain but an interconnected aspect of our technological, commercial, and intellectual landscape.

Virtual environments play a pivotal role in bridging the vast distances of space, allowing us to explore, comprehend, and interact with distant worlds as if they were within our reach. Through advanced 3D reconstruction techniques, high-definition rendering, and immersive virtual reality, space exploration transcends physical limitations, offering unprecedented access to far-off landscapes. These virtual realms not only facilitate scientific research and mission planning but also serve as a powerful tool for public outreach, education, and inspiration. By synthesizing remote environments, space agencies and researchers can share the thrill of discovery with a broader audience, inviting everyone to partake in the journey of exploration. Virtual environments democratize space, turning it from an exclusive domain of scientists and astronauts into a shared experience that ignites curiosity and fosters a global appreciation for the wonders of the universe.

This integration of virtual environments symbolizes a profound shift in how we approach space exploration. It acknowledges the importance of not just reaching distant locations but also making them accessible and tangible to all. It's a fusion of technology, art, science, and human experience that enriches our connection to space, turning distant planets and moons into places we can virtually visit, study, and even feel. It's an embodiment of the limitless possibilities that lie in the synergy between human imagination and technological innovation, transforming space exploration from a remote and abstract endeavor into a personal and shared journey.

Appendix A

Appendix A: Hardware

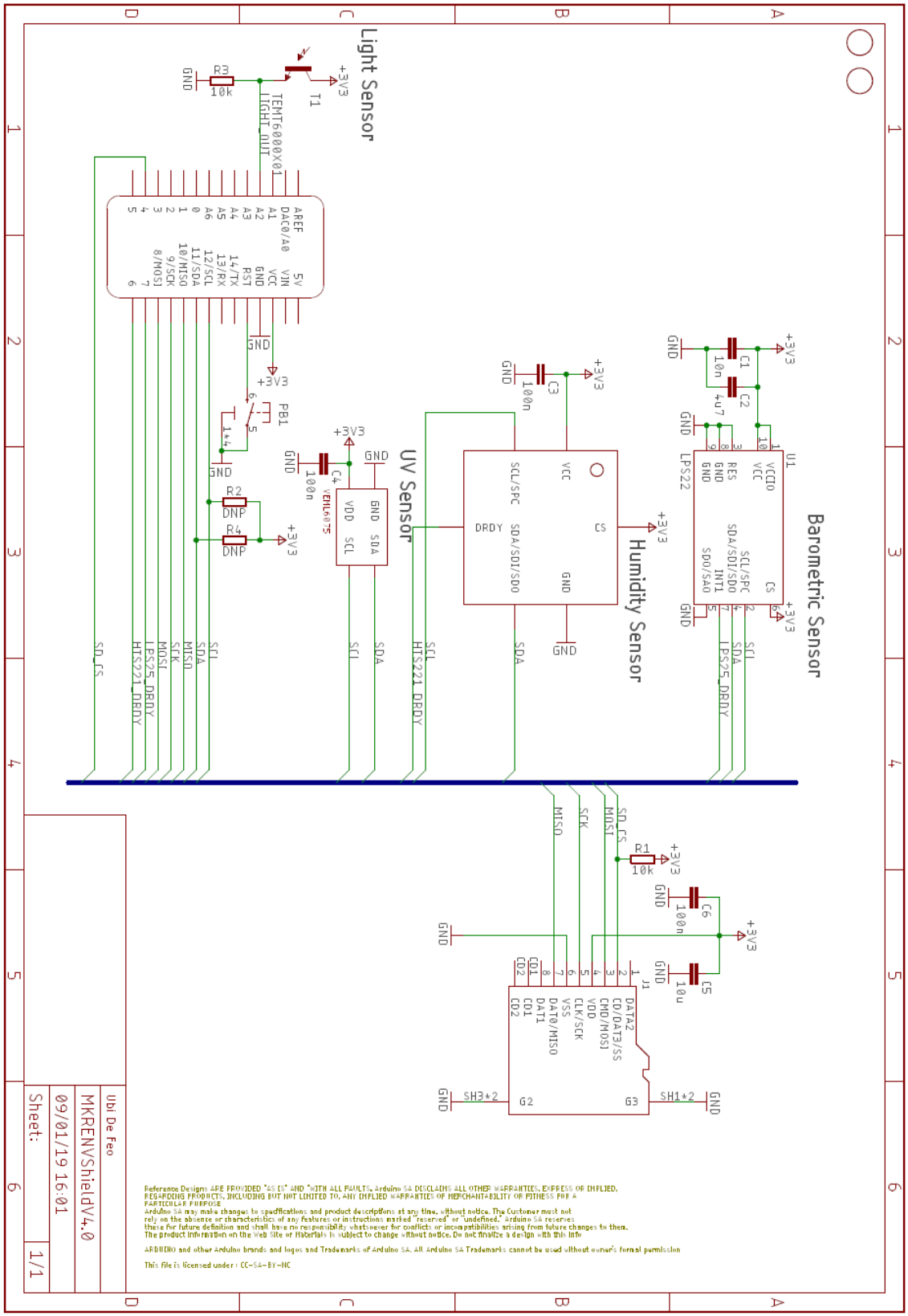

Figure A-1: *Environmental Sensing Arduino Shield designed by Ubi De Feo.*

Appendix B

Appendix B: Supporting Material

B.1 Chapter 4: Cameras Assessment

Camera	File Type	Total Size	Reduced Size
Integrated Spot cameras	.png	1.2 MB	1.09 MB
Intel RealSense D435i	.ply	4.33 GB	7.975 MB
Intel RealSense L515	.ply	2.57 GB	21.35 MB
Velodyne VLP-16	.pcap	2.84 MB	0.87 MB

Table B.1: *Further information about the data collected such as the file type, size as well as compressed size [182].*

Camera	Type	Field of View	Range
Integrated Spot cameras	Stereo, B&W, video	360	4m
Intel RealSense D435i	Stereo, RGB, video	70	9m
Intel RealSense L515	ToF (laser), RGB, video	70	9m
Velodyne VLP-16	LiDAR (Class-1 laser)	360 (30 vertical)	100m
Insta360 One VR	RGB video	360 (180 vertical)	N/A

Table B.2: *Features the various cameras installed on Spot for data collection, specifically the FoV and Range [182].*

Camera	Camera Type	Resolution
Integrated Spot cameras	B&W image	424 x 240
Integrated Spot cameras	stereo-depth	424 x 240
Intel RealSense D435i	RGB	1920 x 1080
Intel RealSense D435i	stereo-depth	1280 x 720
Intel RealSense L515	RGB	1920 x 1080
Intel RealSense L515	ToF	1024 x 768
Velodyne VLP-16	ToF	V-2.0° H-0.1°,0.4°

Table B.3: *Features the various cameras installed on Spot for data collection, specifically camera type and resolution [182]. Distinguishing the Velodyne VLP-16[a] from other devices due to it's difference is nature as a depth computation device with an Angular Resolution (Vertical): 2.0° and (Horizontal/Azimuth): 0.1° – 0.4°*

[a] Velodyne LiDAR PUCK User Manual.

B.2 Chapter 4: Inverse Kinematics Procedural Animation Pseudocode

- Use Inverse Kinematics (IK) to Control Leg: Implement inverse kinematics algorithms to calculate the joint angles required to position the leg correctly. IK allows for dynamic positioning of the leg based on the desired target location.

- Fix Bottom of the Leg to the Ground: Ensure that the bottom part of the leg remains fixed to the ground or any other surface it interacts with. This prevents the leg from penetrating the ground and provides stability during movement.
- Make a Target Point Attached to the Body: Create a target point that is attached to the body of the character or object. This target point represents the desired location for the leg to move towards.
- Raycast from the Target Point Attached to the Body: Use raycasting to detect the ground or other surfaces from the target point attached to the body. This helps determine the contact point for the leg and ensures accurate placement.
- Raycast from the Target Point Downwards to Move It Up and Down: Perform a downward raycast from the target point to adjust its height based on the detected ground or surface. This allows the leg to adapt to uneven terrain or inclined surfaces.
- Check Distance from Target Point: Measure the distance between the leg's current position and the target point. This distance is used to determine whether the leg needs to move towards the target or stay in place.
- Move Leg Towards Target When Distance Gets Too Big: If the distance between the leg and the target point exceeds a threshold, apply motion to move the leg towards the target. This helps maintain a desired leg position relative to the target.
- Put Legs in Zigzag Pattern: Arrange the leg movements in a zigzag pattern to simulate a natural walking or running gait. This alternating pattern provides balance and stability during locomotion.
- Only Move Leg When Opposite Legs are Grounded: Coordinate leg movements so that they occur when the opposite legs are grounded. This mimics the coordinated motion of real walking or running and adds to the realism of the animation.
- Use Average Leg Position + Offset for the Body Position: Calculate the average position of the legs and apply an offset to determine the position of the body. This ensures that the body remains balanced and stable during leg movement.
- Rotate Body Based on Difference Between Left and Right Leg Height: Determine the height difference between the left and right legs and use it to rotate the body. This adds a subtle sway or tilt to the character's motion, enhancing the naturalness of the animation[1].

[1] https://youtu.be/e6Gjhr1IP6w

B.3 Chapter 5: Sensor Node Specifications

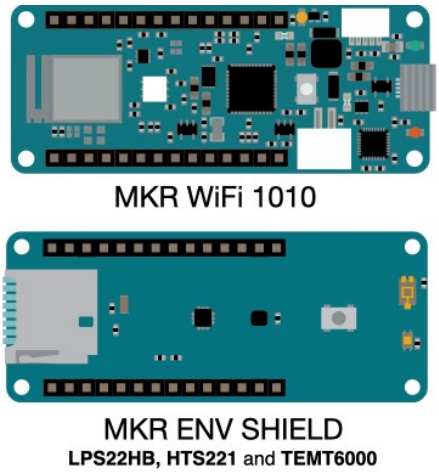

Figure B-1: *The Arduino MKR WIFI 1010 board, enhanced with the Arduino MKR Environmental Shield rev2, provides a comprehensive sensor package. This setup captured crucial environmental data, including color, sound, motion, temperature, humidity, pressure, light, and UV levels. Incorporating modules such as ST LPS22HB for atmospheric pressure, ST HTS221 for temperature and humidity, and VISHAY TEMT6000 for ambient light. Image courtesy of Arduino.cc*

Parameter	**Value**
Operating voltage	3.3 V
Dimensions	25.4 x 53.3 mm
Weight	32 g
Sensors	ST LPS22HB, ST HTS221, VISHAY TEMT6000
Data storage	microSD card 32GB
Communication	Wi-Fi, Analog/i2c
Software	Arduino IDE

Table B.4: *Arduino MKR WiFi 1010 + MKR ENV Shield Technical specifications (ST LPS22HB: Atmospheric pressure, ST HTS221: Temperature and humidity, VISHAY TEMT6000: Ambient light)*

B.4 Chapter 5: 3D Reconstruction with Open3D

```
python azure_kinect_mkv_reader.py --input "
...\[folder]\[capture].mkv" --output "...\[folder]\[capture]"

python run_system.py --make "...\[folder]\[capture]\config.json"
python run_system.py --register "...\[folder]\[capture]\config.json"
python run_system.py --refine "...\[folder]\[capture]\config.json"
python run_system.py --integrate "...\[folder]\[capture]\config.json"
```

B.5 Chapter 6: Testing Procedures and Results

Position	Distance (cm)	Lux	W/m²/nm
Horizontal	127	3500-4000	0.04
-21° Angle	105	7900-8200	0.08
Horizontal	58	22000-24000	0.22
+55° Angle	48	34000-37000	0.34

Table B.5: *Solar simulation experimental setup details [216].*

Mode	Maximum Power (W)	Settings
Off	0	N/A
Idle	0.577	N/A
Stand By	0.566	N/A
Streaming	1.082	WFOV, Binned, 2160p
Recording	1.077	NFOV, Unbinned, 2160p

Table B.6: *Baseline Power Draw requirements for various settings on the Microsoft Azure Kinect, streaming and recording at 30 FPS [216].*

Scenario	Frame	Data	P (W)	Settings
1	1	2.20 MB	1.02 W	30MJPG15362
2	1/m	0.92 MB / m	1.08 W	30MJPG15361
3	1/m	0.92 MB / m	1.08 W	30MJPG15362
4	3	2.20 MB / img	1.02 W	30MJPG15364

Table B.7: *Results from power and data rates experiments with optimized camera settings for each capture mode [216]. Scenario: 1 - Single. 2 - Traverse. 3 - Science station. 4 - Circumferential (10" object) [216].*

ID	Mode	RES	FOI	RANGE	EXP
0	OFF	N/A	N/A	N/A	N/A
1	NFOV 2x2 binned (SW)	320x288	75°x65°	0.5-5.46	12.8 ms
2	NFOV unbinned	640x576	75°x65°	0.5-3.86	12.8 ms
3	WFOV 2x2 binned	512x512	120°x120°	0.25-2.88	12.8 ms
4	WFOV unbinned	1024x1024	120°x120°	0.25-2.21	20.3 ms
5	Passive IR	1024x1024	N/A	N/A	1.6 ms

Table B.8: *AKALL depth modes and their corresponding specifications, including Resolution (RES), field of Illumination (FoI), Range (meters), and Exposure (ms).*

Test Type	Plan	Requirements/Location
Vibration table	Open COTS part, stake down parts and add supports where failure occurs	NASA Ames - one day on shake 3-axis shake table
Vacuum testing	1-week in vacuum, then remove and power on (prelim testing). Then 1-week in high-vacuum, assess any parts that may impact vacuum	Prelim - MIT, High-vacuum - NASA Ames
Thermal testing	Prepare for need to provide heating on rover - thermal heat tape tests	NASA Ames and MIT - test power requirements of heat tape
Laser function	Assess the visibility of having the field of view in the rover's shadow to avoid increasing laser power	MIT field testing - varying ground reflectivity and light source angles
Power supply	Modify to 28 V power regulated by rover - meet 5V at 6W to meet camera peak power draw	MIT in collaboration with Microsoft - hardware modification
Data rates	Rover maximum data transmission with limited budget in the range of (MB) for the entire mission - heavy optimisation and compression needed	MIT RESOURCE, in collaboration with NASA Ames

Table B.9: *Original plan to adapt the Microsoft Azure Kinect in both hardware and software for space exploration [182, 195].*

Appendix C

Appendix C: Code

C.1 Unity Projects Code Base

- Doppelmarsh (https://github.com/mitmedialab/Doppelmarsh)
- Doppelbot (https://github.mit.edu/ddh/Doppelbot)
- Doppelspot (https://github.mit.edu/ddh/Doppelspot)
- OGIVE (https://github.mit.edu/ddh/OGIVE)

C.2 Chapter 3: Rover Mini Configuration

```
<!---
Author : Don D . Haddad
Description : These instructions outline configuring the
    Nvidia Xavier NX, which operates the Doppelbot, to
    handle various scenarios, including building and
    installing Open3D on ARM64, and more.
-->

*Resize Partition with gparted
```
sudo apt install gparted
sudo gparted
```

*Add Cuda to PATH (append to ~/.bashrc)

```
export CUDA_HOME=/usr/local/cuda
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/
    lib64:/usr/local/cuda/extras/CUPTI/lib64
export PATH=$PATH:$CUDA_HOME/bin
```

*Required for USB issues on Nvidia Jetson Xavier NX (append
    to ~/.bashrc)

```
export OPENBLAS_CORETYPE=ARMV8
```

source: https://stackoverflow.com/questions/65631801/
    illegal-instructioncore-dumped-error-on-jetson-nano

*check cmake version

```
cmake --version
```

*upgrade cmake to version 3.20

```
#!/bin/bash

version=3.20
build=1
mkdir ~/temp
cd ~/temp
wget https://cmake.org/files/v$version/cmake-$version.
    $build.tar.gz
tar -xzvf cmake-$version.$build.tar.gz
cd cmake-$version.$build/
```

*install and extract source by running
```
./bootstrap
make -j$(nproc)
sudo make install
```

```
cmake --version
```

source: https://askubuntu.com/questions/355565/how-do-i-
    install-the-latest-version-of-cmake-from-the-command-
    line

Build and Install Open3D from source.
Tested on Nvidia Jetson Xavier NX supporting Azure Kinect.
*Open3D (0.15.1)
*Azure Kinect SDK (1.4.1)
*cmake (3.24.1)
*JetPack (4.4)
*Ubuntu (18.04)
*Python3 (3.6.9)

```
git clone --recursive https://github.com/isl-org/Open3D.git
git checkout v0.15.1
```
```

```

*install dependencies

```
./util/install_deps_ubuntu.sh
sudo apt-get install -y libc++-7-dev libc++abi-7-dev clang-7 python-pip3 ccache gfortran
sudo -H pip3 install --upgrade pip==20.3
pip3 install matplotlib
```

*build
```
cd Open3D && mkdir build && cd build
cmake -DBUILD_AZURE_KINECT=ON -DBUILD_CUDA_MODULE=ON -DBUILD_GUI=ON ..

#Alternatively
cmake -DCMAKE_BUILD_TYPE=Release -DBUILD_SHARED_LIBS=ON -DBUILD_CUDA_MODULE=ON -DBUILD_GUI=ON -DBUILD_TENSORFLOW_OPS=OFF -DBUILD_PYTORCH_OPS=OFF -DBUILD_UNIT_TEST=ON -DCMAKE_INSTALL_PREFIX=~/open3d_install -DBUILD_AZURE_KINECT=ON -DUSE_SYSTEM_LIBREALSENSE=ON -DGLIBCXX_USE_CXX11_ABI=ON -DCMAKE_C_COMPILER=gcc -DCMAKE_CXX_COMPILER=g++ -DBUILD_FILAMENT_FROM_SOURCE=ON -DPYTHON_EXECUTABLE=$(which python3) -CMAKE_CUDA_COMPILER=/usr/local/cuda/bin/nvcc ..

#Alternatively
cmake -DCMAKE_BUILD_TYPE=Release -DBUILD_SHARED_LIBS=ON -DBUILD_CUDA_MODULE=ON -DBUILD_GUI=ON -DBUILD_TENSORFLOW_OPS=OFF -DBUILD_PYTORCH_OPS=OFF -DBUILD_UNIT_TEST=ON -DCMAKE_INSTALL_PREFIX=~/open3d_install -DBUILD_AZURE_KINECT=ON -DUSE_SYSTEM_LIBREALSENSE=ON -DGLIBCXX_USE_CXX11_ABI=ON -DCMAKE_C_COMPILER=gcc -DCMAKE_CXX_COMPILER=g++ -DBUILD_FILAMENT_FROM_SOURCE=ON -DPYTHON_EXECUTABLE=$(which python3) ..

make -j$(nproc)
sudo make install
make install-pip-package -j$(nproc)
```

source: http://www.open3d.org/docs/release/arm.html#install-dependencies

*Run Open3D GUI
```
./bin/Open3D/Open3D
```

*3D Reconstruction with Azure Kinect and Open3D

```
python3 ~/Open3D/examples/python/reconstruction_system/sensors/azure_kinect_recorder.py --output frames.mkv

python3 ~/Open3D/examples/python/reconstruction_system/sensors/azure_kinect_mkv_reader.py --input frames.mkv --output frames

python3 ~/Open3D/examples/python/reconstruction_system/run_system.py --config ./frames/config.json --make

python3 ~/Open3D/examples/python/reconstruction_system/run_system.py --config ./frames/config.json --register

python3 ~/Open3D/examples/python/reconstruction_system/run_system.py --config ./frames/config.json --refine

python3 ~/Open3D/examples/python/reconstruction_system/run_system.py --config ./frames/config.json --integrate
```

#Enable screen recording and streaming with gstreamer
```
sudo apt-get install libgstrtspserver-1.0 libgstreamer1.0-dev
wget https://gstreamer.freedesktop.org/src/gst-rtsp/gst-rtsp-server-1.14.1.tar.xz
tar -xvf gst-rtsp-server-1.14.1.tar.xz
cd  gst-rtsp-server-1.14.1
cd examples
gcc test-launch.c -o test-launch $(pkg-config --cflags --libs gstreamer-1.0 gstreamer-rtsp-server-1.0)
```

#Launch RTSP Server
```

```
```
./test-launch "ximagesrc use-damage=0 ! nvvidconv ! omxh264enc ! video/x-h264, profile=baseline ! h264parse ! video/x-h264, stream-format=byte-stream ! rtph264pay name=pay0 pt=96 "
```

source: https://forums.developer.nvidia.com/t/streaming-desktop-with-rtsp-gstreamer-server/143765/19?page=2

#Aliases
```
alias stream='/home/nini/gst-rtsp-server-1.14.1/examples/test-launch "ximagesrc use-damage=0 ! nvvidconv ! omxh264enc ! video/x-h264, profile=baseline ! h264parse ! video/x-h264, stream-format=byte-stream ! rtph264pay name=pay0 pt=96 "'
alias azure='python3 /home/nini/Open3D/examples/python/reconstruction_system/sensors/azure_kinect_recorder.py --output mini.mkv'
```

#Setup VNC server
```
mkdir -p ~/.config/autostart
cp /usr/share/applications/vino-server.desktop ~/.config/autostart/.
gsettings set org.gnome.Vino prompt-enabled false
gsettings set org.gnome.Vino require-encryption false
Replace thepassword with your desired password
gsettings set org.gnome.Vino authentication-methods "['vnc']"
gsettings set org.gnome.Vino vnc-password $(echo -n 'nini<3'|base64)
sudo reboot
```

## C.3 Chapter 5: Sensor Node

```
// Author: Don D. Haddad
// Description: Program to read environment data from MKR ENV shield and write to SD card on an Arduino

#include <SPI.h>
#include <SD.h>
#include <Arduino_MKRENV.h>
#include <WiFiNINA.h>
#include <utility/wifi_drv.h>

const int chipSelect = 4;
String node_id = "0x004";
String filename = "ENVD004.csv";

bool verbose = false;

// Initialize the system
void setup() {
  Serial.begin(9600);

  // Initialize RGB LED pins
  WiFiDrv::pinMode(25, OUTPUT);
  WiFiDrv::pinMode(26, OUTPUT);
  WiFiDrv::pinMode(27, OUTPUT);

  // Start the MKR ENV shield
  if (!ENV.begin()) {
    Serial.println("Failed to initialize MKR ENV shield!");
    while (1);
  }

  // Initialize SD card
  Serial.print("Initializing SD card...");
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    LED(0, 255, 0, 300);
    while (1);
  }
```
```

```
  Serial.println("card initialized.");

  // Create a file or clear the existing one
  File dataFile = SD.open(filename, FILE_WRITE);
  if (dataFile) {
    dataFile.println("-,-,-,-,-,-,-");
    dataFile.close();
  }
}

// Main loop to gather and store environmental data
void loop() {
  String dataString = "";

  // Read environmental data
  float temperature = ENV.readTemperature();
  float humidity    = ENV.readHumidity();
  float pressure    = ENV.readPressure();
  float illuminance = ENV.readIlluminance();
  float uva         = ENV.readUVA();
  float uvb         = ENV.readUVB();
  float uvIndex     = ENV.readUVIndex();

  // Create comma-separated string of data
  dataString = String(temperature) + ", " + String(humidity
      ) + ", " + String(pressure) + ", " + String(
      illuminance) + ", " + String(uva) + ", " + String(uvb
      ) + ", " + String(uvIndex);

  // Verbose output to Serial monitor
  if(verbose) {
    Serial.print("Temperature = ");
    Serial.print(temperature);
    Serial.println("  C ");

    Serial.print("Humidity    = ");
    Serial.print(humidity);
    Serial.println(" %");

    Serial.print("Pressure    = ");
    Serial.print(pressure);
    Serial.println(" kPa");

    Serial.print("Illuminance = ");
    Serial.print(illuminance);
    Serial.println(" lx");

    Serial.print("UVA         = ");
    Serial.println(uva);
    Serial.print("UVB         = ");
    Serial.println(uvb);
    Serial.print("UV Index    = ");
    Serial.println(uvIndex);
    Serial.println();
  }

  // Write data to SD card
  File dataFile = SD.open(filename, FILE_WRITE);
  if (dataFile) {
    dataFile.println(dataString);
    dataFile.close();
    Serial.println(dataString);
    LED(255, 0, 128, 100); // Success LED pattern
  }
  else {
    Serial.println("error opening " + filename);
    LED(0, 255, 0, 300); // Error LED pattern
  }

  delay(900);
}

// Function to control RGB LED
void LED(int r, int g, int b, int freq) {
  WiFiDrv::analogWrite(25, g);
  WiFiDrv::analogWrite(26, r);
  WiFiDrv::analogWrite(27, b);
  delay(freq);
  WiFiDrv::analogWrite(25, 0);
  WiFiDrv::analogWrite(26, 0);
  WiFiDrv::analogWrite(27, 0);
}
```

C.4 Chapter 6: AKALL Code Base

```
###
### To Build:
### docker build -t <payload_image_name> <path_to_folder_containing_Dockerfile>
###
### To Run:
### ./scripts/launch_container.sh <payload_container_name> <payload_image_name>
###

### Set the base image using Nvidia CUDA on Ubuntu
FROM nvidia/cuda:11.4.0-base-ubuntu18.04

LABEL maintainer="Don Derek Haddad <ddh@mit.edu>, Dogi <stefan@unterhauser.name>"
LABEL version="1.1"
LABEL description="Docker image for Azure Kinect SDK with CUDA 11.4.0 on Ubuntu 18.04"

### Install essential packages and dependencies
RUN apt update -y && apt install -y \
    python3 \
    python3-pip \
    curl \
    software-properties-common \
    build-essential

### Add Microsoft's package signing key and repository
RUN curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add - && \
    apt-add-repository https://packages.microsoft.com/ubuntu/18.04/multiarch/prod

### Install Azure Kinect SDK and development libraries
RUN apt update -y && ACCEPT_EULA=y apt install -y k4a-tools && \
    apt install -y \
    libk4a1.4-dev

### Copy the source code to the container
COPY src /home/payload/workspace/src
COPY include /home/payload/workspace/include
COPY scripts /home/payload/workspace/scripts
RUN chmod +x /home/payload/workspace/scripts/*.sh

### Compile the payload software
RUN gcc -g -Wall  /home/payload/workspace/src/k4a-capture.cpp -o /usr/local/bin/check-device -lk4a -lstdc++; \
gcc -g -Wall  /home/payload/workspace/src/get-calibration.cpp -o /usr/local/bin/calibrate -lk4a -lstdc++

### Set environment variables for Nvidia container runtime
ENV NVIDIA_VISIBLE_DEVICES all
ENV NVIDIA_DRIVER_CAPABILITIES graphics,utility,compute

### Set entrypoint for the container
WORKDIR /home/payload/workspace
CMD ["/bin/bash", "/home/payload/workspace/scripts/entrypoint.sh"]
```

```
### * ****************************************** * ###
### * Docker Container Shell Script Debugging &  * ###
### * Logging                                     * ###
### * Author: Don D. Haddad <ddh@mit.edu>        * ###
### * File Name: entrypoint.sh                   * ###
### * Location: /scripts                         * ###
### * Description: Initializes UNIX Domain socket * ###
### * Python server for local communication with * ###
### * Docker container & logs output. Useful for * ###
### * debugging and data transmission.           * ###
### * ****************************************** * ###

#!/bin/sh

touch /output.log
python3 /home/payload/workspace/include/socket_coms_local.py &>> /output.log &

#Used for debugging
bash
```

```
### * ****************************************** * ###
### * Docker Shell Script for Launching Payload * ###
### * Docker Container                          * ###
### * Author: Don D. Haddad <ddh@mit.edu>       * ###
### * File Name: launch_container.sh            * ###
### * Location: /scripts                       * ###
### * Description: Launch Docker container in   * ###
### * privileged mode with custom parameters    * ###
### * on the development machine.               * ###
### * ****************************************** * ###

#!/bin/sh

# Expose the X server on the host, only needed on dev machine.
sudo xhost +local:root

export PAYLOAD_STORAGE=/tmp/payload_storage

export PAYLOAD_SOCKETS=/tmp/payload_sockets/kinect_luna

export ENTRYPOINT=/home/payload/workspace/scripts/entrypoint.sh

sudo docker run -it --rm \
    \
    --gpus all --privileged \
    -e DISPLAY=$DISPLAY \
    -v ${PAYLOAD_SOCKETS}:/tmp/payload_sockets/ \
    -v /tmp/.X11-unix:/tmp/.X11-unix \
    -v ${PAYLOAD_STORAGE}:/storage \
    -w /home/payload/ \
    \
    --name=$1 \
    $2 \
    ${ENTRYPOINT}
```

```
/* ********************************************** *
 * Custom C++ Program for Azure Kinect Data Capture *
 * Author: Don D. Haddad <ddh@mit.edu>              *
 * File Name: k4a-capture.cpp                       *
 * Location: /src                                   *
 * Description: This code is designed to capture    *
 * a single frame JPEG image, depth, and IR image   *
 * (b16g), then apply gzip compression to these     *
 * three fragments along with a calibration.json    *
 * file generated from get-calibration.cpp.         *
 * Compiled and built within the instantiation of   *
 * the Docker container.                            *
 * ********************************************** */

#include <k4a/k4a.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <sys/socket.h>
#include <sys/un.h>
#include <string>
#include <vector>
#include <unistd.h>
#include <chrono>
#include <cerrno>

using namespace std;

// Function to write binary data to a file
// Parameters:
// - fileName: name of the file to write to
// - buffer: pointer to the data buffer that contains the data to be written
// - bufferSize: size of the data buffer in bytes
// Returns:
// - 0: indicating successful write operation
long WriteToFile(const char *fileName, void *buffer, size_t bufferSize) {
  ofstream hFile;
  hFile.open(fileName, ios::out | ios::trunc | ios::binary);
  if (hFile.is_open()){
    hFile.write((char *)buffer, static_cast<streamsize>(bufferSize));
```

```
    hFile.close();
  }
  return 0;
}

int main(int argc,char* argv[]){
  // Declare configuration strings for capturing Azure Kinect data
  string fps, color, resolution, depth, unix_time;
  string exposure, brightness, contrast, saturation;
  string sharpness, gain, white_balance, blacklight_comp, powerline_freq;

  // Read command line arguments for configuring image capture
  if(argc >= 2){
    fps = argv[2];
    color = argv[4];
    resolution = argv[6];
    depth = argv[8];
    unix_time = argv[10];
    printf("[AZURE KINECT] K4A Image config: \nfps: %s, color: %s, resolution: %s, depth: %s\n", fps.c_str(), color.c_str(), resolution.c_str(), depth.c_str());
  }

  // Read additional command line arguments for configuring color settings if provided
  if (argc > 12){
    exposure = argv[12];
    brightness = argv[14];
    contrast = argv[16];
    saturation = argv[18];
    sharpness = argv[20];
    gain = argv[22];
    white_balance = argv[24];
    blacklight_comp = argv[26];
    powerline_freq = argv[28];
    printf("[AZURE KINECT] K4A Image config: \nfps: %s, color: %s, resolution: %s, depth: %s\n[AZURE KINECT] color settings: exposure: %s, brightness: %s, contrast: %s, saturation: %s, sharpness: %s, gain: %s, white_balance: %s, blacklight_comp: %s, powerline_freq: %s\n", fps.c_str(), color.c_str(), resolution.c_str(), depth.c_str(), exposure.c_str(), brightness.c_str(), contrast.c_str(), saturation.c_str(), sharpness.c_str(), gain.c_str(), white_balance.c_str(), blacklight_comp.c_str(), powerline_freq.c_str());
  }

  // Check the number of connected Kinect devices
  uint32_t count = k4a_device_get_installed_count();
  k4a_capture_t capture = NULL;
  int returnCode = 1;

  const int32_t TIMEOUT_IN_MS = 1000;

  if (count == 0) {
    printf("\n[AZURE KINECT] No k4a devices attached!\n");
    return 1;
  }

  // Open the first plugged-in Kinect device
  k4a_device_t device = NULL;
  if (K4A_FAILED(k4a_device_open(K4A_DEVICE_DEFAULT, &device))) {
    printf("[AZURE KINECT] Failed to open k4a device!\n");
    return 1;
  }

  // Retrieve device serial number
  size_t serial_size = 0;
  k4a_device_get_serialnum(device, NULL, &serial_size);

  // Allocate memory for the serial, then acquire it
  char *serial = (char*)(malloc(serial_size));
  k4a_device_get_serialnum(device, serial, &serial_size);
  printf("[AZURE KINECT] Opened device: %s\n", serial);
  free(serial);

  // Set up default camera settings
  k4a_device_configuration_t config = K4A_DEVICE_CONFIG_INIT_DISABLE_ALL;
  config.camera_fps       = K4A_FRAMES_PER_SECOND_15;
  config.color_format     = K4A_IMAGE_FORMAT_COLOR_BGRA32;
  config.color_resolution = K4A_COLOR_RESOLUTION_1080P;
  config.depth_mode = K4A_DEPTH_MODE_NFOV_2X2BINNED;

```

```
113   // Customize camera settings based on command line input
114   // Customize FPS
115   if(fps=="5")
116   config.camera_fps       = K4A_FRAMES_PER_SECOND_5;
117   else if(fps=="15")
118   config.camera_fps       = K4A_FRAMES_PER_SECOND_15;
119   else if(fps=="30")
120   config.camera_fps       = K4A_FRAMES_PER_SECOND_15;
121
122   // Customize compression type
123   if(color=="MJPG")
124   config.color_format     = K4A_IMAGE_FORMAT_COLOR_MJPG;
125   else if(color=="NV12")
126   config.color_format     = K4A_IMAGE_FORMAT_COLOR_NV12;
127   else if(color=="YUY2")
128   config.color_format     = K4A_IMAGE_FORMAT_COLOR_YUY2;
129   else if(color=="BGRA32")
130   config.color_format     = K4A_IMAGE_FORMAT_COLOR_BGRA32;
131   else if(color=="DEPTH16")
132   config.color_format     = K4A_IMAGE_FORMAT_DEPTH16;
133   else if(color=="IR16")
134   config.color_format     = K4A_IMAGE_FORMAT_IR16;
135
136   // Customize resolution
137   if(resolution=="OFF")
138   config.color_resolution     = K4A_COLOR_RESOLUTION_OFF;
139   else if(resolution=="720P")
140   config.color_resolution     = K4A_COLOR_RESOLUTION_720P;
141   else if(resolution=="1080P")
142   config.color_resolution     = K4A_COLOR_RESOLUTION_1080P;
143   else if(resolution=="1440P")
144   config.color_resolution     = K4A_COLOR_RESOLUTION_1440P;
145   else if(resolution=="1536P")
146   config.color_resolution     = K4A_COLOR_RESOLUTION_1536P;
147   else if(resolution=="2160P")
148   config.color_resolution     = K4A_COLOR_RESOLUTION_2160P;
149   else if(resolution=="3072P")
150   config.color_resolution     = K4A_COLOR_RESOLUTION_3072P;
151
152   // Customize depth mode
153   if(depth=="OFF")
154   config.depth_mode     = K4A_DEPTH_MODE_OFF;
155   else if(depth=="NFOV_2X2BINNED")
156   config.depth_mode     = K4A_DEPTH_MODE_NFOV_2X2BINNED;
157   else if(depth=="NFOV_UNBINNED")
158   config.depth_mode     = K4A_DEPTH_MODE_NFOV_UNBINNED;
159   else if(depth=="WFOV_2X2BINNED")
160   config.depth_mode     = K4A_DEPTH_MODE_WFOV_2X2BINNED;
161   else if(depth=="WFOV_UNBINNED")
162   config.depth_mode     = K4A_DEPTH_MODE_WFOV_UNBINNED;
163   else if(depth=="PASSIVE_IR")
164   config.depth_mode     = K4A_DEPTH_MODE_PASSIVE_IR;
165
166   // Apply additional color settings if provided in the
          command line
167   if(argc>12){
168     // Configure exposure (Default value is A)
169     if(exposure=="A")
170     k4a_device_set_color_control(device,
            K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
            K4A_COLOR_CONTROL_MODE_AUTO, 0);
171     else if (exposure=="M1" || exposure=="M500")
172     k4a_device_set_color_control(device,
            K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
            K4A_COLOR_CONTROL_MODE_MANUAL, 500);
173     else if (exposure=="M2" || exposure=="M1250")
174     k4a_device_set_color_control(device,
            K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
            K4A_COLOR_CONTROL_MODE_MANUAL, 1250);
175     else if (exposure=="M3" || exposure=="M2500")
176     k4a_device_set_color_control(device,
            K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
            K4A_COLOR_CONTROL_MODE_MANUAL, 2500);
177     else if (exposure=="M4" || exposure=="M10000")
178     k4a_device_set_color_control(device,
            K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
            K4A_COLOR_CONTROL_MODE_MANUAL, 10000);
179     else if (exposure=="M5" || exposure=="M20000")
180     k4a_device_set_color_control(device,
            K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
            K4A_COLOR_CONTROL_MODE_MANUAL, 20000);
181     else if (exposure=="M6" || exposure=="M30000")
182     k4a_device_set_color_control(device,
            K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
            K4A_COLOR_CONTROL_MODE_MANUAL, 30000);
183     else if (exposure=="M7" || exposure=="M40000")
184     k4a_device_set_color_control(device,
            K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
            K4A_COLOR_CONTROL_MODE_MANUAL, 40000);
185     else if (exposure=="M8" || exposure=="M50000")
```

```
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
        K4A_COLOR_CONTROL_MODE_MANUAL, 50000);
    else if (exposure=="M9" || exposure=="M60000")
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
        K4A_COLOR_CONTROL_MODE_MANUAL, 60000);
    else if (exposure=="M10" || exposure=="M80000")
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
        K4A_COLOR_CONTROL_MODE_MANUAL, 80000);
    else if (exposure=="M11" || exposure=="M100000")
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
        K4A_COLOR_CONTROL_MODE_MANUAL, 100000);
    else if (exposure=="M12" || exposure=="M120000")
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
        K4A_COLOR_CONTROL_MODE_MANUAL, 120000);
    else if (exposure=="M13" || exposure=="M130000")
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_EXPOSURE_TIME_ABSOLUTE,
        K4A_COLOR_CONTROL_MODE_MANUAL, 130000);

    // Configure brightness (Default value is 128)
    int brightness_integer = atoi(brightness.c_str());
    if(brightness_integer>=0 || brightness_integer <=255)
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_BRIGHTNESS,
        K4A_COLOR_CONTROL_MODE_MANUAL, brightness_integer )
        ;

    // Configure contrast (Defaul value is 5)
    int contrast_integer = atoi(contrast.c_str());
    if(contrast_integer>=0 || contrast_integer <=10)
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_CONTRAST,
        K4A_COLOR_CONTROL_MODE_MANUAL, contrast_integer );

    // Configure sturation (Default value is 32)
    int saturation_integer = atoi(saturation.c_str());
    if(saturation_integer>=0 || saturation_integer <=63)
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_SATURATION,
        K4A_COLOR_CONTROL_MODE_MANUAL, saturation_integer )
        ;

    // Configure sharpness (Default value is 2)
    int sharpness_integer = atoi(sharpness.c_str());
    if(sharpness_integer>=0 || sharpness_integer <=4)
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_SHARPNESS,
        K4A_COLOR_CONTROL_MODE_MANUAL, sharpness_integer );

    // Configure gain (Default value is 0)
    int gain_integer = atoi(gain.c_str());
    if(gain_integer >= 0 || gain_integer <=255)
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_GAIN,
        K4A_COLOR_CONTROL_MODE_MANUAL, gain_integer );

    // Configure white balance (Defaul value is A)
    if(white_balance == "A")
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_WHITEBALANCE,
        K4A_COLOR_CONTROL_MODE_AUTO, 0 );
    else{
      int white_balance_integer = atoi(white_balance.c_str
          ());
      white_balance_integer = white_balance_integer - (
          white_balance_integer%10);
      k4a_device_set_color_control(device,
          K4A_COLOR_CONTROL_WHITEBALANCE,
          K4A_COLOR_CONTROL_MODE_MANUAL,
          white_balance_integer);
      printf("DEBUG: %i\n", white_balance_integer);
    }

    // Configure blacklight compensation
    int blacklight_comp_integer = atoi(blacklight_comp.
        c_str());
    if(blacklight_comp_integer == 0 ||
        blacklight_comp_integer ==1)
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_BACKLIGHT_COMPENSATION,
        K4A_COLOR_CONTROL_MODE_MANUAL,
        blacklight_comp_integer );

    // Configure powerline frequency
    int powerline_freq_integer = atoi(powerline_freq.c_str
```

```
        ());
    if(powerline_freq_integer == 0 ||
        powerline_freq_integer ==1)
    k4a_device_set_color_control(device,
        K4A_COLOR_CONTROL_BACKLIGHT_COMPENSATION,
        K4A_COLOR_CONTROL_MODE_MANUAL,
        powerline_freq_integer );
  }
  // Start the camera with the defined configuration
  if (K4A_FAILED(k4a_device_start_cameras(device, &config))
      )
  {
    printf("[AZURE KINECT] Failed to start cameras!\n");
    k4a_device_close(device);
    return 1;
  }
  int captureFrameCount = 30;
  const auto ltt = chrono::system_clock::now();
  int64_t timestamp = chrono::duration_cast<chrono::seconds
      >(ltt.time_since_epoch()).count() ;
  const char* path = "/storage/";
  string color_blk;
  string depth_blk;
  string ir_blk;

  while (captureFrameCount-- > 0) {
    k4a_image_t image;

    // Get a depth frame
    switch (k4a_device_get_capture(device, &capture,
        TIMEOUT_IN_MS))
    {
      case K4A_WAIT_RESULT_SUCCEEDED:
      break;
      case K4A_WAIT_RESULT_TIMEOUT:
      printf("[AZURE KINECT] Timed out waiting for a
          capture\n");
      continue;
      break;
      case K4A_WAIT_RESULT_FAILED:
      printf("[AZURE KINECT] Failed to read a capture\n");
      goto Exit;
    }

    printf("Capture");

    // Probe for a color image
    image = k4a_capture_get_color_image(capture);
    if (image)
    {
      string filename = path + to_string(timestamp) + "C" +
          fps + color + resolution + ".jpeg";
      color_blk = to_string(timestamp) + "C" + fps + color
          + resolution + ".jpeg";
      printf(" | Color res:%4dx%4d stride:%5d ",
      k4a_image_get_height_pixels(image),
      k4a_image_get_width_pixels(image),
      k4a_image_get_stride_bytes(image));
      WriteToFile(filename.c_str(), k4a_image_get_buffer(
          image ), k4a_image_get_size(image));
      k4a_image_release(image);
    }
    else
    {
      printf(" | Color None                       ");
    }

    // probe for a IR16 image
    image = k4a_capture_get_ir_image(capture);
    if (image != NULL)
    {
      string filename = path + to_string(timestamp) + "IR"
          + fps + to_string(k4a_image_get_width_pixels(
          image)) + depth;
      depth_blk = to_string(timestamp) + "IR" + fps +
          to_string(k4a_image_get_width_pixels(image)) +
          depth;
      printf(" | Ir16 res:%4dx%4d stride:%5d ",
      k4a_image_get_width_pixels(image),
      k4a_image_get_height_pixels(image),
      k4a_image_get_stride_bytes(image));
      WriteToFile(filename.c_str(), k4a_image_get_buffer(
          image ), k4a_image_get_size(image));

      k4a_image_release(image);
    }
    else
    {
      printf(" | Ir16 None                        ");
    }
```

```
313
314     // Probe for a depth16 image
315     image = k4a_capture_get_depth_image(capture);
316     if (image != NULL)
317     {
318       string filename = path + to_string(timestamp) + "D" +
            fps + to_string(k4a_image_get_width_pixels(image
          )) + depth;
319       ir_blk = to_string(timestamp) + "D" + fps + to_string
            (k4a_image_get_width_pixels(image)) + depth;;
320       printf(" | Depth16 res:%4dx%4d stride:%5d\n",
321       k4a_image_get_width_pixels(image),
322       k4a_image_get_height_pixels(image),
323       k4a_image_get_stride_bytes(image));
324       WriteToFile(filename.c_str(), k4a_image_get_buffer(
            image ), k4a_image_get_size(image));
325
326       k4a_image_release(image);
327     }
328     else
329     {
330       printf(" | Depth16 None\n");
331     }
332
333     // release capture
334     k4a_capture_release(capture);
335     fflush(stdout);
336
337     if(captureFrameCount==0){
338
339       // Shut down the camera when finished with
            application logic
340       k4a_device_stop_cameras(device);
341       k4a_device_close(device);
342       //compress captured framgents
343       string compress_cmd = "calibrate;cd /storage; tar -
            czf " + to_string(timestamp) + ".tar.gz " +
            color_blk + " " + depth_blk + " " + ir_blk+"
            calibration.json; rm calibration.json "+
            color_blk + " " + depth_blk + " " + ir_blk;
344       system(compress_cmd.c_str());
345
346       //Send data fragments over UNIX socket
347       int sock = socket(AF_UNIX, SOCK_DGRAM, 0);
348       if (sock < 0) {
349         cerr << "Failed to create socket: " << strerror(
              errno) << endl;
350         return 1;
351       }
352
353       struct sockaddr_un addr;
354       memset(&addr, 0, sizeof(addr));
355       addr.sun_family = AF_UNIX;
356       strcpy(addr.sun_path, "/tmp/payload_sockets/sm_sock")
            ;
357
358       if (connect(sock, (struct sockaddr *) &addr, sizeof(
            addr)) < 0) {
359         cerr << "Failed to connect to socket: " << strerror
              (errno) << endl;
360         return 1;
361       }
362
363       // Send data to the socket
364       string data = to_string(timestamp) + ".tar.gz";
365       if (write(sock, data.c_str(), data.size()) < 0) {
366         cerr << "Failed to send data to socket: " <<
              strerror(errno) << endl;
367         return 1;
368       }
369       close(sock);
370     }
371   }
372
373   return 0;
374   returnCode = 0;
375   Exit:
376
377   if (device != NULL)
378   {
379     k4a_device_close(device);
380   }
381   return returnCode;
382 }
```

```cpp
/* ********************************************** *
 * Custom C++ Program for Azure Kinect Calibration  *
 * Author: Don D. Haddad <ddh@mit.edu>              *
 * File Name: get-calibration.cpp                   *
 * Location: /src                                   *
 * Description: This code is designed to retrieve   *
 * the calibration data from an Azure Kinect device *
 * and save it to a file in JSON format. This       *
 * includes details such as device serial numbers,  *
 * calibration blob sizes, and other relevant data. *
 * Compiled and built within the instantiation of   *
 * the Docker container.                            *
 * ********************************************** */

#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <vector>
#include <k4a/k4a.h>

using namespace std;

static void calibration_blob(uint8_t deviceIndex = 0,
    string filename = "calibration.json")
{
    k4a_device_t device = NULL;

    if (K4A_RESULT_SUCCEEDED != k4a_device_open(deviceIndex
        , &device)) {
        cout << deviceIndex << ": Failed to open device" <<
             endl;
        exit(-1);
    }

    size_t calibration_size = 0;
    k4a_buffer_result_t buffer_result =
        k4a_device_get_raw_calibration(device, NULL, &
        calibration_size);

    if (buffer_result == K4A_BUFFER_RESULT_TOO_SMALL) {
        vector<uint8_t> calibration_buffer = vector<uint8_t
            >(calibration_size);
        buffer_result = k4a_device_get_raw_calibration(
            device, calibration_buffer.data(), &
            calibration_size);

        if (buffer_result == K4A_BUFFER_RESULT_SUCCEEDED) {
            ofstream file(filename, ofstream::binary);
            file.write(reinterpret_cast<const char *>(&
                calibration_buffer[0]), (long)
                calibration_size);
            file.close();
            cout << "Calibration blob for device " << (int)
                deviceIndex << " (serial no. " <<
                get_serial(device)
                 << ") is saved to " << filename << endl;
        } else {
            cout << "Failed to get calibration blob" <<
                endl;
            exit(-1);
        }
    } else {
        cout << "Failed to get calibration blob size" <<
            endl;
        exit(-1);
    }
}

static string get_serial(k4a_device_t device)
{
    size_t serial_number_length = 0;

    if (K4A_BUFFER_RESULT_TOO_SMALL !=
        k4a_device_get_serialnum(device, NULL, &
        serial_number_length)) {
        cout << "Failed to get serial number length" <<
            endl;
        k4a_device_close(device);
        exit(-1);
    }

    char *serial_number = new (std::nothrow) char[
        serial_number_length];

    if (serial_number == NULL) {
        cout << "Failed to allocate memory for serial
            number (" << serial_number_length << " bytes)"
            << endl;
        k4a_device_close(device);
```

```
        exit(-1);
    }

    if (K4A_BUFFER_RESULT_SUCCEEDED !=
        k4a_device_get_serialnum(device, serial_number, &
        serial_number_length)) {
        cout << "Failed to get serial number" << endl;
        delete[] serial_number;
        serial_number = NULL;
        k4a_device_close(device);
        exit(-1);
    }

    string s(serial_number);
    delete[] serial_number;
    serial_number = NULL;
    return s;
}

int main(int argc, char **argv)
{
    calibration_blob(0, "/storage/calibration.json");

    return 0;
}
```

```
Camera Frames Per Second:
  -f --fps
    K4A_FRAMES_PER_SECOND_5
    K4A_FRAMES_PER_SECOND_15
    K4A_FRAMES_PER_SECOND_30

Color Format:
  -c --color
    K4A_IMAGE_FORMAT_COLOR_MJPG
    K4A_IMAGE_FORMAT_COLOR_NV12
    K4A_IMAGE_FORMAT_COLOR_YUY2
    K4A_IMAGE_FORMAT_COLOR_BGRA32
    K4A_IMAGE_FORMAT_DEPTH16
    K4A_IMAGE_FORMAT_IR16

Color Resolution:
  -r --resolution
    K4A_COLOR_RESOLUTION_OFF
    K4A_COLOR_RESOLUTION_720P
    K4A_COLOR_RESOLUTION_1080P
    K4A_COLOR_RESOLUTION_1440P
    K4A_COLOR_RESOLUTION_1536P
    K4A_COLOR_RESOLUTION_2160P
    K4A_COLOR_RESOLUTION_3072P

Depth Mode:
  -d --depth
    K4A_DEPTH_MODE_OFF
    K4A_DEPTH_MODE_NFOV_2X2BINNED
    K4A_DEPTH_MODE_NFOV_UNBINNED
    K4A_DEPTH_MODE_WFOV_2X2BINNED
    K4A_DEPTH_MODE_WFOV_UNBINNED
    K4A_DEPTH_MODE_PASSIVE_IR

Color Settings:
  -s --settings
    EXPOSURE TIME ABSOLUTE: (AUTO || MANUAL)
      v: 1:500 to 12:130000
      v: M 500 - 130000
      V: A
    BRIGHTNESS: (MANUAL)
      v: 0-255
    CONTRAST: (MANUAL)
      v: 0-255
    SATURATION: (MANUAL)
      v: 0-255
    SHARPNESS: (MANUAL)
      v: 0-255
    WHITEBALANCE: (MANUAL || AUTO)
      v: M 1800 - 10000
      v: A
    BACKLIGHT COMPENSATION: (MANUAL)
      v: 0,1
    GAIN: (MANUAL)
      v: 0-255
    POWERLINE FREQUENCY: (MANUAL)
      v: 1,2
```

```
#!/usr/bin/python

### * ************************************************************** * ###
### * Host Machine Unix Socket Communication Console                 * ###
### * Author: Don D. Haddad <ddh@mit.edu>                           * ###
### * File Name: socket_coms_console.py                             * ###
### * Location: /include                                           * ###
### * Description: Implements a threaded UNIX socket server that runs  * ###
### * within the host machine, providing functionalities for creating,  * ###
### * binding, and communicating through UNIX sockets. This module also * ###
### * provides a console interface for manually sending messages via    * ###
### * the socket, enabling bi-directional communication between         * ###
### * processes, handling both incoming and outgoing messages, and      * ###
### * managing connections.                                           * ###
### * ************************************************************** * ###

import socket
import os, os.path
import threading
import time

PRINT_PREPEND = '[HOST MACHINE] '
UNIX_SOCKETS_BASE_DIR = '/tmp/payload_sockets/kinect_luna/'
UNIX_ADDR_OUT = UNIX_SOCKETS_BASE_DIR + 'pl_sock'
UNIX_ADDR_IN = UNIX_SOCKETS_BASE_DIR + 'sm_sock'

class UNIX_Coms():

    def __init__(self, host_addr):
        self.sock = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
        self.server_address = host_addr

    '''
        Creates and binds a new UNIX socket
        Server (listener) side
        - Removes existing socket if it exists
    '''
    def bind_to_socket(self):

        if os.path.exists(self.server_address):
            os.remove(self.server_address)

        self.sock.bind(self.server_address)
        print(PRINT_PREPEND + 'starting up on {}'.format(self.server_address))

        self.unix_start()

    '''
        Start listening for incoming messages
        sever (listener) side
    '''
    def unix_start(self):
        threading.Thread(target=self.listen).start()

    '''
        Listen for incoming messages (called by unix_start)
    '''
    def listen(self):
        print(PRINT_PREPEND + 'Listening for connections on {}'.format(self.
            server_address))
        while (True):
            data = self.sock.recv(4096)
            if data:
                print('\n'+PRINT_PREPEND + ' Received data from {} | DATA [{}]:
                    {} '.format(self.server_address, len(data), data))

    '''
        Connects to existing UNIX sockets
```

```
67          client (sender) side
68      '''
69      def connect_to_socket(self):
70          connect = False
71          try:
72              print(PRINT_PREPEND + 'connecting to {}'.format(self.server_address))
73              self.sock.connect(self.server_address)
74              connect = True
75              print(PRINT_PREPEND + 'Connected')
76          except socket.error as msg:
77              print(msg)
78
79      '''
80          Send data to the server
81          - connectsto the socket and sends the data
82      '''
83      def send(self, msg):
84          print(PRINT_PREPEND + 'Sending data on socket: {}'.format(self.
                server_address))
85
86          self.connect_to_socket()
87          self.sock.sendall(msg)
88
89      def close(self):
90          self.sock.close()
91          print(PRINT_PREPEND + 'Socket closed: {}'.format(self.server_address))
92
93  def main():
94
95      unix_in = {}
96      unix_out = {}
97
98      def closeAll():
99          [unix_in[key].close() for key in unix_in]
100         [unix_out[key].close() for key in unix_out]
101
102     try:
103         unix_in = {
104             'IN' : UNIX_Coms(UNIX_ADDR_IN),
105         }
106         [unix_in[key].bind_to_socket() for key in unix_in]
107     except:
108         print(PRINT_PREPEND + 'Error creating outgoing UNIX socket(s)')
109         unix_in = None
110
111     try:
112         unix_out = {
113             'OUT' : UNIX_Coms(UNIX_ADDR_OUT),
114         }
115     except:
116         print(PRINT_PREPEND + 'Error creating outgoing UNIX socket(s)')
117         unix_out = None
118
119     time.sleep(1)
120
121     repeat = True
122
123     while(repeat):
124         input_cmd = input("Enter Capture Sequence # ")
125         params = input_cmd.split("-")
126         ts = int(time.time())
127
128         if(len(params)==10):
129             cmd_byte = bytearray(input_cmd+'-'+str(ts), 'utf-8')
130         else:
131             cmd_byte = bytearray(input_cmd+'-'+str(ts), 'utf-8')
132
133         print('\n'+PRINT_PREPEND + 'K4A Capture cmd: ' + input_cmd+'-'+str(ts))
```

```
134
135         unix_out['OUT'].send(cmd_byte)
136
137         time.sleep(1)
138         print(PRINT_PREPEND + 'Done.. \n')
139
140     return 0
141
142 if __name__ == '__main__':
143     main()
```

```
 1 #!/usr/bin/python
 2
 3 ### * ************************************************************ * ###
 4 ### * Local Machine Python Payload Server                          * ###
 5 ### * Author: Don D. Haddad <ddh@mit.edu>                         * ###
 6 ### * File Name: socket_coms_local.py                             * ###
 7 ### * Location: /include                                         * ###
 8 ### * Description: Implements a threaded UNIX socket server that * ###
 9 ### * receives messages from the host machine and triggers capture * ###
10 ### * sequence commands.                                          * ###
11 ### * ************************************************************ * ###
12
13 import socket
14 import os, os.path
15 import threading
16 import time
17 import subprocess
18
19 PRINT_PREPEND = '[KINECT LUNA] '
20 ERROR_PREPEND = '[ERROR] '
21 UNIX_SOCKETS_BASE_DIR = '/tmp/payload_sockets/'
22 UNIX_ADDR_IN = UNIX_SOCKETS_BASE_DIR + 'pl_sock'
23
24 class UNIX_Coms():
25
26     def __init__(self, host_addr):
27         self.sock = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
28         self.server_address = host_addr
29
30     '''
31         Creates and binds a new UNIX socket
32         Server (listener) side
33         - Removes existing socket if it exists
34     '''
35     def bind_to_socket(self):
36         if os.path.exists(self.server_address):
37             print(self.server_address);
38             os.remove(self.server_address)
39         self.sock.bind(self.server_address)
40         print(PRINT_PREPEND + 'starting up on {}'.format(self.server_address))
41         self.unix_start()
42
43     '''
44         Start listening for incoming messages
45         sever (listener) side
46     '''
47     def unix_start(self):
48         threading.Thread(target=self.listen).start()
49
50     '''
51         Listen for incoming messages (called by unix_start)
52     '''
53     def listen(self):
54         print(PRINT_PREPEND + 'Listening for connections on {}'.format(self.
               server_address))
55         while (True):
56             data = self.sock.recv(4096)
```

```
57              if data:
58                  print(PRINT_PREPEND + 'Received data from {} | DATA [{}]: {}'.
                        format(self.server_address, len(data), data))
59                  cmd = data.decode('utf-8').split('-')
60                  params = list(cmd[0])
61                  input_error = False;
62
63                  if(cmd[0] == "SM" or cmd[0] == "sm"):
64                      if(cmd[1] == "RM" or cmd[1] == "rm"):
65                          if(cmd[2] == "ALL" or cmd[2] == "all"):
66                              if(os.path.exists('/storage')):
67                                  os.system('rm /storage/*')
68                                  print(PRINT_PREPEND + 'All captures deleted from
                                        /storage directory.')
69                          else:
70                              if(os.path.exists('/storage/'+cmd[2]+'.tar.gz')):
71                                  os.system('rm /storage/'+cmd[2]+'.tar.gz')
72                                  print(PRINT_PREPEND + 'File '+cmd[2]+'.tar.gz
                                        removed from /storage directory.')
73
74                  if(len(params) == 12):
75                      fps = params[1]+params[2]
76                      color = params[3]+params[4]+params[5]+params[6]
77                      resolution = params[7]+params[8]+params[9]+params[10]
78                      depth = params[11]
79
80                      if fps != '05' and fps != '15' and fps != '30':
81                          input_error = True
82                          print(ERROR_PREPEND + 'Invalid FPS value ' + fps + '.')
83                      if color != 'MJPG':
84                          input_error = True
85                          print(ERROR_PREPEND + 'Invalid color format ' + color + '
                                . Currently only supports MJPG.')
86                      if resolution != '0720' and resolution != '1080' and
                            resolution != '1440' and resolution != '1536' and
                            resolution != '2160' and resolution != '3072':
87                          input_error = True
88                          print(ERROR_PREPEND + 'Invalid resolution ' + resolution
                                + '. Supported resolution:  720, 1080, 1440, 1536,
                                2160, 3072')
89                      if depth != '0' and depth != '1' and depth != '2' and depth
                            != '3' and depth != '4' and depth != '5':
90                          input_error = True
91                          print(ERROR_PREPEND + 'Invalid Depth mode value ' + depth
                                 + '.')
92
93                      if input_error == False:
94                          if fps == '05':
95                              fps='5'
96
97                          if color == 'BGRA':
98                              color = 'BGRA32'
99                          elif color == 'DP16':
100                             color = 'DEPTH16'
101
102                         if resolution == '0000':
103                             resolution = 'OFF'
104                         elif resolution == '0720':
105                             resolution = '720P'
106                         else:
107                             resolution+='P'
108
109                         if depth == '0':
110                             depth = 'OFF'
111                         elif depth == '1':
112                             depth = 'NFOV_2X2BINNED'
113                         elif depth == '2':
114                             depth = 'NFOV_UNBINNED'
```

```
115                         elif depth == '3':
116                             depth = 'WFOV_2X2BINNED'
117                         elif depth == '4':
118                             depth = 'WFOV_UNBINNED'
119                         elif depth == '5':
120                             depth = 'PASSIVE_IR'
121 
122                         print(PRINT_PREPEND + 'K4A Image Capture: ' + cmd[0] + '
                                t:' + cmd[len(cmd)-1])
123                     else:
124                         input_error = True;
125                         print(ERROR_PREPEND + 'The following command ' + cmd[0] +
                                ' is not a valid capture sequence..')
126 
127                     if(len(cmd)==11 or len(cmd)==10):
128                         print(PRINT_PREPEND + 'K4A Color Settings: {} {} {} {} {}
                                {} {} {} {}'.format(cmd[1],cmd[2],cmd[3],cmd[4],cmd
                                [5],cmd[6],cmd[7],cmd[8],cmd[9]))
129                         try:
130                             exposure = cmd[1].split('E')[1]
131                             brightness = cmd[2].split('B')[1]
132                             contrast = cmd[3].split('C')[1]
133                             saturation = cmd[4].split('S')[1]
134                             sharpness = cmd[5].split('H')[1]
135                             gain = cmd[6].split('G')[1]
136                             white_balance = cmd[7].split('W')[1]
137                             blacklight_comp = cmd[8].split('P')[1]
138                             powerline_freq = cmd[9].split('L')[1]
139                         except IndexError:
140                             input_error = False
141 
142                         if input_error == False:
143                             os.system('check-device -f {} -c {} -r {} -d {} -t {}
                                    -xp {} -br {} -cn {} -st {} -sh {} -gn {} -wb {}
                                    -bl {} -pl {}'.format(fps, color, resolution,
                                    depth, cmd[len(cmd)-1], exposure, brightness,
                                    contrast, saturation, sharpness, gain,
                                    white_balance, blacklight_comp, powerline_freq))
144                             print(PRINT_PREPEND + 'K4A Color Settings: {} {} {}
                                    {} {} {} {} {} {}'.format(exposure, brightness,
                                    contrast, saturation, sharpness, gain,
                                    white_balance, blacklight_comp, powerline_freq))
145                     elif(len(cmd) >= 1 or len(cmd) <= 3):
146                         print(PRINT_PREPEND + 'Loading K4A Default Color Settings
                                ..')
147                         if input_error == False:
148                             os.system('check-device -f {} -c {} -r {} -d {} -t {}
                                    '.format(fps, color, resolution, depth, cmd[len(
                                    cmd)-1]))
149                             print(PRINT_PREPEND+'check-device -f {} -c {} -r {} -
                                    d {} -t {}'.format(fps, color, resolution, depth,
                                     cmd[len(cmd)-1]))
150                     else:
151                         input_error = True;
152                         print(ERROR_PREPEND + 'The following command ' + data.
                                decode('utf-8') + ' is not a valid capture sequence..
                                ')
153     '''
154         Connects to existing UNIX sockets
155         client (sender) side
156     '''
157     def connect_to_socket(self):
158         connect = False
159         try:
160             print(PRINT_PREPEND + 'connecting to {}'.format(self.server_address))
161             self.sock.connect(self.server_address)
162             connect = True
163             print(PRINT_PREPEND + 'Connected')
```

```
164             except socket.error as msg:
165                 print(msg)
166 
167         '''
168             Send data to the server
169             - connectsto the socket and sends the data
170         '''
171         def send(self, msg):
172             print(PRINT_PREPEND + 'Sending data on socket: {}'.format(self.
                    server_address))
173 
174             self.connect_to_socket()
175             self.sock.sendall(msg)
176 
177         def close(self):
178             self.sock.close()
179             print(PRINT_PREPEND + 'Socket closed: {}'.format(self.server_address))
180 
181     def main():
182         unix_in  = {}
183 
184         def closeAll():
185             [unix_in[key].close() for key in unix_in]
186 
187         try:
188             unix_in = {
189                 'IN'           : UNIX_Coms(UNIX_ADDR_IN),
190             }
191             [unix_in[key].bind_to_socket() for key in unix_in]
192         except:
193             print(PRINT_PREPEND + 'Error creating outgoing UNIX socket(s)')
194             unix_in = None
195 
196         return 0
197 
198     if __name__ == '__main__':
199         main()
```

```
#!/usr/bin/python

### * ************************************************************ * ###
### * Host Machine Unix Socket Communication Module                * ###
### * Author: Don D. Haddad <ddh@mit.edu>                         * ###
### * File Name: socket_coms_main.py                              * ###
### * Location: /include                                         * ###
### * Description: Implements a threaded UNIX socket server that runs * ###
### * within the host machine. It creates an interactive mode that    * ###
### * allows users to generate messages and send them through the     * ###
### * socket. It includes functionalities for creating, binding,      * ###
### * and communicating through UNIX sockets, handling both          * ###
### * incoming and outgoing messages, and managing connections.      * ###
### * ************************************************************ * ###

import socket
import os, os.path
import threading
import time

PRINT_PREPEND = '[HOST MACHINE] '
UNIX_SOCKETS_BASE_DIR = '/tmp/payload_sockets/kinect_luna/'
UNIX_ADDR_OUT = UNIX_SOCKETS_BASE_DIR + 'pl_sock'

class UNIX_Coms():

    def __init__(self, host_addr):
        self.sock = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
        self.server_address = host_addr

    '''
        Creates and binds a new UNIX socket
        Server (listener) side
        - Removes existing socket if it exists
    '''
    def bind_to_socket(self):

        if os.path.exists(self.server_address):
            os.remove(self.server_address)

        self.sock.bind(self.server_address)
        print(PRINT_PREPEND + 'starting up on {}'.format(self.server_address))

        self.unix_start()

    '''
        Start listening for incoming messages
        sever (listener) side
    '''
    def unix_start(self):
        threading.Thread(target=self.listen).start()

    '''
        Listen for incoming messages (called by unix_start)
    '''
    def listen(self):
        print(PRINT_PREPEND + 'Listening for connections on {}'.format(self.
            server_address))
        while (True):
            data = self.sock.recv(4096)
            if data:
                print(PRINT_PREPEND + 'Received data from {} | DATA [{}]: {}'.
                    format(self.server_address, len(data), data))

    '''
        Connects to existing UNIX sockets
        client (sender) side
    '''
```

```
67      def connect_to_socket(self):
68          connect = False
69          try:
70              print(PRINT_PREPEND + 'connecting to {}'.format(self.server_address))
71              self.sock.connect(self.server_address)
72              connect = True
73              print(PRINT_PREPEND + 'Connected')
74          except socket.error as msg:
75              print(msg)
76
77      '''
78          Send data to the server
79          - connectsto the socket and sends the data
80      '''
81      def send(self, msg):
82          print(PRINT_PREPEND + 'Sending data on socket: {}'.format(self.
                server_address))
83
84          self.connect_to_socket()
85          self.sock.sendall(msg)
86
87      def close(self):
88          self.sock.close()
89          print(PRINT_PREPEND + 'Socket closed: {}'.format(self.server_address))
90
91  def main():
92
93      unix_out = {}
94
95      def closeAll():
96          [unix_out[key].close() for key in unix_out]
97
98      try:
99          unix_out = {
100             'OUT' : UNIX_Coms(UNIX_ADDR_OUT),
101         }
102     except:
103         print(PRINT_PREPEND + 'Error creating outgoing UNIX socket(s)')
104         unix_out = None
105
106     time.sleep(1)
107
108     input_error = "Invalid choice.. using default settings."
109     camera_fps = "K4A_FRAMES_PER_SECOND_5"
110     repeat = True
111
112     while(repeat):
113         input_0 = input("Enter K4A's FPS (0:05 FPS 1:15 FPS 2:30 FPS) # ")
114         if input_0 == str(0):
115             camera_fps = "K4A_FRAMES_PER_SECOND_5"
116         elif input_0 == str(1):
117             camera_fps = "K4A_FRAMES_PER_SECOND_15"
118         elif input_0 == str(2):
119             camera_fps = "K4A_FRAMES_PER_SECOND_30"
120         else:
121             input_0 = 0
122             print(input_error)
123
124         input_1 = input("Enter K4A's Color Format (0:MJPG 1:NV12 2:YUY2 3:BGRA 4:
                DP16 5:IR16) # ")
125         color_format = "K4A_IMAGE_FORMAT_COLOR_MJPG"
126         if input_1 == str(0):
127             color_format = "K4A_IMAGE_FORMAT_COLOR_MJPG"
128         elif input_1 == str(1):
129             color_format = "K4A_IMAGE_FORMAT_COLOR_NV12"
130         elif input_1 == str(2):
131             color_format = "K4A_IMAGE_FORMAT_COLOR_YUY2"
132         elif input_1 == str(3):
```

```
            color_format = "K4A_IMAGE_FORMAT_COLOR_BGRA32"
        elif input_1 == str(4):
            color_format = "K4A_IMAGE_FORMAT_DEPTH16"
        elif input_1 == str(5):
            color_format = "K4A_IMAGE_FORMAT_IR16"
        else:
            input_1 = 0
            print(input_error)

        input_2 = input("Enter K4A's Color Resolution (0: Off 1:720 2:1080 3:1440
             4:1536 5:2160 6:3072) # ")
        color_resolution = "K4A_COLOR_RESOLUTION_720P"
        if input_2 == str(0):
            color_resolution = "K4A_COLOR_RESOLUTION_OFF"
        elif input_2 == str(1):
            color_resolution = "K4A_COLOR_RESOLUTION_720P"
        elif input_2 == str(2):
            color_resolution = "K4A_COLOR_RESOLUTION_1080P"
        elif input_2 == str(3):
            color_resolution = "K4A_COLOR_RESOLUTION_1440P"
        elif input_2 == str(4):
            color_resolution = "K4A_COLOR_RESOLUTION_1536P"
        elif input_2 == str(5):
            color_resolution = "K4A_COLOR_RESOLUTION_2160P"
        elif input_2 == str(6):
            color_resolution = "K4A_COLOR_RESOLUTION_3072P"
        else:
            input_2 = 1
            print(input_error)

        input_3 = input("Enter K4A's Depth Mode (0: Off 1:NFOV_2X2B 2:NFOV_U 3:
            WFOV_2X2B 4:WFOV_U 5:P_IR) # ")
        depth_mode = "K4A_DEPTH_MODE_NFOV_2X2BINNED"
        if input_3 == str(0):
            depth_mode = "K4A_DEPTH_MODE_OFF"
        elif input_3 == str(1):
            depth_mode = "K4A_DEPTH_MODE_NFOV_2X2BINNED"
        elif input_3 == str(2):
            depth_mode = "K4A_DEPTH_MODE_NFOV_UNBINNED"
        elif input_3 == str(3):
            depth_mode = "K4A_DEPTH_MODE_WFOV_2X2BINNED"
        elif input_3 == str(4):
            depth_mode = "K4A_DEPTH_MODE_WFOV_UNBINNED"
        elif input_3 == str(5):
            depth_mode = "K4A_DEPTH_MODE_PASSIVE_IR"
        else:
            input_3 = 1
            print(input_error)

        input_4 = input("Set K4A's Color Settings (y/n) #")
        color_settings = "D"

        if input_4 == "y" or input_4 == "Y":
            input_4_1 = input("Enter Exposure Time (A, M1-M13, M500-M130000) d:A
                #")
            input_4_2 = input("Enter Brightness (0-255) d:128 #")
            input_4_3 = input("Enter Contrast (0-10) d:5 #")
            input_4_4 = input("Enter Saturation (0-63) d:32 #")
            input_4_5 = input("Enter Sharpness (0-4) d:2 #")
            input_4_6 = input("Enter Gain (0-255) d:0 #")
            input_4_7 = input("Enter White Balance (A, 2500-12500) d:A #")
            input_4_8 = input("Enter Backlight Compensation (0,1) d:0 #")
            input_4_9 = input("Enter Power Line Frequency (1: 50hz ,2: 60Hz) d:2
                #")

            color_settings = 'E'+input_4_1+"-"+'B'+input_4_2+"-"+'C'+input_4_3+"-
                "+'S'+input_4_4+"-"+'H'+input_4_5+"-"+'G'+input_4_6+"-"+'W'+
                input_4_7+"-"+'P'+input_4_8+"-"+'L'+input_4_9
```

```
195
196         print("Settings Recorded..\n")
197         print("Capturing K4A Images with the following settings:")
198
199         print(camera_fps)
200         print(color_format)
201         print(color_resolution)
202         print(depth_mode)
203         print("CS " + color_settings)
204
205         opt = [['05','15','30'], ['MJPG','NV12','YUY2','BGRA','DP16','IR16'], ['
                0000','0720','1080','1440','1536','2160','3072'], ['0','1','2','3','4
                ','5']]
206
207         cmd = 'K'+opt[0][int(input_0)] + opt[1][int(input_1)] + opt[2][int(
                input_2)] + opt[3][int(input_3)]
208
209         print('\n'+PRINT_PREPEND + 'K4A Image str: ' + cmd)
210
211         ts = int(time.time())
212         cmd_byte = bytearray(cmd+'-'+color_settings+'-'+str(ts), 'utf-8')
213
214         unix_out['OUT'].send(cmd_byte)
215
216         time.sleep(1)
217         print(PRINT_PREPEND + 'Done.. \n')
218
219         inpeat = True
220         while inpeat:
221             input_r = input('[R] Repeat capture preserving settings \n[N] New
                    capture \n[E] Exit program \nChoice [R, N, E] # ')
222             if input_r == 'R' or input_r == 'r':
223                 print('\n'+PRINT_PREPEND + 'K4A Image str: ' + cmd)
224                 ts = int(time.time())
225                 cmd_byte = bytearray(cmd+'-'+str(ts), 'utf-8')
226                 unix_out['OUT'].send(cmd_byte)
227                 time.sleep(1)
228                 print(PRINT_PREPEND + 'Done.. \n')
229             elif input_r == 'N' or input_r == 'n':
230                 inpeat = False
231             elif input_r == 'E' or input_r == 'e':
232                 repeat = False
233                 inpeat = False
234                 closeAll()
235                 print(PRINT_PREPEND + 'END: Generating logfile.. \n')
236                 return 0
237     return 0
238
239 if __name__ == '__main__':
240     main()
```

```
#!/usr/bin/python

### * ************************************************************ * ###
### * Python Test Script for Camera Capture Sequences              * ###
### * Author: Don D. Haddad <ddh@mit.edu>                         * ###
### * File Name: socket_coms_test.py                              * ###
### * Location: /include                                         * ###
### * Description: Generates 108 capture sequences covering all the   * ###
### * options of the camera (without color settings) to test the      * ###
### * device and generate all possible outcomes.                    * ###
### * ************************************************************ * ###

import socket
import os, os.path
import threading
import time

PRINT_PREPEND = '[HOST MACHINE] '
UNIX_SOCKETS_BASE_DIR = '/tmp/payload_sockets/kinect_luna/'
UNIX_ADDR_OUT = UNIX_SOCKETS_BASE_DIR + 'pl_sock'

class UNIX_Coms():

    def __init__(self, host_addr):
        self.sock = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
        self.server_address = host_addr

    '''
        Creates and binds a new UNIX socket
        Server (listener) side
        - Removes existing socket if it exists
    '''
    def bind_to_socket(self):

        if os.path.exists(self.server_address):
            os.remove(self.server_address)

        self.sock.bind(self.server_address)
        print(PRINT_PREPEND + 'starting up on {}'.format(self.server_address))

        self.unix_start()

    '''
        Start listening for incoming messages
        sever (listener) side
    '''
    def unix_start(self):
        threading.Thread(target=self.listen).start()

    '''
        Listen for incoming messages (called by unix_start)
    '''
    def listen(self):
        print(PRINT_PREPEND + 'Listening for connections on {}'.format(self.
            server_address))
        while (True):
            data = self.sock.recv(4096)
            if data:
                print(PRINT_PREPEND + 'Received data from {} | DATA [{}]: {}'.
                    format(self.server_address, len(data), data))

    '''
        Connects to existing UNIX sockets
        client (sender) side
    '''
    def connect_to_socket(self):
        connect = False
        try:
```

```
67              print(PRINT_PREPEND + 'connecting to {}'.format(self.server_address))
68              self.sock.connect(self.server_address)
69              connect = True
70              print(PRINT_PREPEND + 'Connected')
71          except socket.error as msg:
72              print(msg)
73
74      '''
75          Send data to the server
76          - connectsto the socket and sends the data
77      '''
78      def send(self, msg):
79          print(PRINT_PREPEND + 'Sending data on socket: {}'.format(self.
                server_address))
80
81          self.connect_to_socket()
82          self.sock.sendall(msg)
83
84      def close(self):
85          self.sock.close()
86          print(PRINT_PREPEND + 'Socket closed: {}'.format(self.server_address))
87
88  def main():
89
90      unix_out = {}
91
92      def closeAll():
93          [unix_out[key].close() for key in unix_out]
94
95      try:
96          unix_out = {
97              'OUT' : UNIX_Coms(UNIX_ADDR_OUT),
98          }
99      except:
100         print(PRINT_PREPEND + 'Error creating outgoing UNIX socket(s)')
101         unix_out = None
102
103     time.sleep(1)
104
105     repeat = True
106
107     fps = ['05','15','30']
108     res = ['0720','1080','1440','1536','2160','3072']
109
110     for i, x in enumerate(fps):
111         for j, y in enumerate(res):
112             for k in range(6):
113                 ts = int(time.time())
114                 cmd = 'K'+fps[i]+'MJPG'+res[j]+str(k)
115                 cmd_byte = bytearray(cmd+'-'+str(ts), 'utf-8')
116                 print('\n'+PRINT_PREPEND + 'K4A Image str: ' + cmd+'-'+str(ts))
117                 unix_out['OUT'].send(cmd_byte)
118                 time.sleep(2)
119
120     return 0
121
122 if __name__ == '__main__':
123     main()
```

Appendix D

Appendix D: Mysc

D.1 Chapter 6: Filename Terminology

- **NoSB:** No scale Bar
- **SB:** Scale Bar
- **BP:** Boot Print
- **C:** Crater
- **LR:** Little rock
- **BR:** Big rock
- **DF:** Debris Flow
- **LS:** Light source (0, 60, 120, 180, 240, 300)
- **PD:** Pitch down
- **PU:** Pitch up
- **YL:** Yaw left
- **YR:** Yaw right

D.2 Chapter 6: Short AKALL Messages (All)

K05MJPG07200
K05MJPG07201
K05MJPG07202
K05MJPG07203
K05MJPG07204
K05MJPG07205
K15MJPG07200
K15MJPG07201
K15MJPG07202
K15MJPG07203
K15MJPG07204
K15MJPG07205
K30MJPG07200
K30MJPG07201
K30MJPG07202
K30MJPG07203
K30MJPG07204
K30MJPG07205
K05MJPG10800
K05MJPG10801
K05MJPG10802
K05MJPG10803
K05MJPG10804
K05MJPG10805
K15MJPG10800
K15MJPG10801
K15MJPG10802
K15MJPG10803
K15MJPG10804
K15MJPG10805
K30MJPG10800
K30MJPG10801
K30MJPG10802
K30MJPG10803
K30MJPG10804
K30MJPG10805
K05MJPG14400
K05MJPG14401
K05MJPG14402
K05MJPG14403
K05MJPG14404
K05MJPG14405
K15MJPG14400
K15MJPG14401
K15MJPG14402
K15MJPG14403
K15MJPG14404
K15MJPG14405
K30MJPG14400
K30MJPG14401
K30MJPG14402
K30MJPG14403
K30MJPG14404
K30MJPG14405
K05MJPG15360
K05MJPG15361
K05MJPG15362
K05MJPG15363
K05MJPG15364
K05MJPG15365
K15MJPG15360
K15MJPG15361
K15MJPG15362
K15MJPG15363
K15MJPG15364
K15MJPG15365
K30MJPG15360
K30MJPG15361
K30MJPG15362
K30MJPG15363
K30MJPG15364
K30MJPG15365
K05MJPG21600
K05MJPG21601
K05MJPG21602

K05MJPG21603	K30MJPG21602	K15MJPG30721
K05MJPG21604	K30MJPG21603	K15MJPG30722
K05MJPG21605	K30MJPG21604	K15MJPG30723
K15MJPG21600	K30MJPG21605	K15MJPG30724
K15MJPG21601	K05MJPG30720	K15MJPG30725
K15MJPG21602	K05MJPG30721	K30MJPG30720
K15MJPG21603	K05MJPG30722	K30MJPG30721
K15MJPG21604	K05MJPG30723	K30MJPG30722
K15MJPG21605	K05MJPG30724	K30MJPG30723
K30MJPG21600	K05MJPG30725	K30MJPG30724
K30MJPG21601	K15MJPG30720	K30MJPG30725

D.3 Chapter 6: Long AKALL Messages (Random)

K05MJPG07201-EA-B128-C5-S32-H2-G0-WA-P0-L2

K05MJPG10801-EM1-B100-C2-S62-H4-G10-W3611-P1-L1

K15MJPG14403-E80000-B60-C2-S40-H0-G128-W53611-P0-L1

K30MJPG21602-E130000-B189-C3-S12-H1-G255-WA-P0-L2

K15MJPG30720-EM11-B255-C8-S33-H3-G0-WA-P1-L1

K15MJPG07200-EA-B128-C5-S32-H2-G0-WA-P0-L2

K05MJPG10801-EM7-B78-C8-S22-H4-G56-W12500-P1-L2

K30MJPG07201-E70000-B255-C10-S10-H4-G0-W10000-P1-L1

K30MJPG10803-EM9-B128-C4-S32-H1-G140-WA-P0-L1

K05MJPG21602-E80000-B88-C7-S40-H3-G85-W8750-P1-L2

K30MJPG10802-EM10-B70-C5-S24-H2-G0-WA-P1-L1

K15MJPG15364-E130000-B60-C2-S38-H0-G128-WA-P1-L1

K05MJPG10801-EM1-B100-C3-S48-H2-G0-WA-P0-L2

K05MJPG21601-EM1-B128-C6-S32-H1-G200-WA-P1-L1

K05MJPG30723-EM6-B128-C5-S32-H1-G0-W3750-P1-L1

K15MJPG10802-EM1-B90-C3-S32-H2-G128-W7500-P0-L2

K15MJPG21603-E130000-B255-C9-S16-H2-G0-WA-P1-L1

K30MJPG10801-EM8-B128-C7-S48-H3-G0-W3750-P0-L2

K05MJPG15362-E70000-B70-C4-S32-H2-G0-WA-P1-L1

K15MJPG14400-EA-B128-C5-S32-H2-G0-W2500-P0-L2

K05MJPG07202-E130000-B128-C6-S32-H1-G0-W8750-P1-L1

K05MJPG21600-EM9-B128-C5-S32-H2-G0-W8750-P0-L1

K30MJPG15365-EM10-B128-C6-S40-H1-G0-W3750-P1-L2

K15MJPG07200-E130000-B80-C3-S30-H3-G0-WA-P0-L2

K30MJPG21601-EM10-B70-C5-S32-H3-G255-WA-P1-L2

K05MJPG10802-EM1-B80-C6-S48-H2-G0-W8750-P0-L2

K30MJPG30721-E80000-B128-C6-S32-H1-G0-WA-P1-L1

K15MJPG21602-E130000-B80-C4-S24-H0-G0-W8750-P1-L1

K05MJPG21600-EM6-B100-C8-S62-H4-G140-W2500-P0-L2

K30MJPG10800-EM7-B60-C4-S32-H3-G0-W10000-P1-L2

K15MJPG07202-EM11-B100-C3-S40-H1-G128-WA-P0-L1

K30MJPG14400-E130000-B128-C5-S32-H1-G0-WA-P1-L1

K05MJPG10801-EM1-B128-C5-S32-H3-G0-W2500-P0-L2

K05MJPG21601-EM6-B70-C5-S32-H0-G0-W2500-P0-L2

K05MJPG30721-EM1-B100-C2-S24-H2-G128-W10000-P1-L2

K30MJPG21600-EM1-B128-C5-S32-H2-G0-W8750-P1-L2

K05MJPG07201-EA-B128-C6-S40-H3-G0-WA-P0-L1

K05MJPG10803-EM10-B128-C5-S24-H1-G128-W3750-P0-L1

K30MJPG21603-EM9-B100-C6-S32-H2-G0-WA-P1-L2

K05MJPG21602-EM6-B100-C5-S48-H0-G0-W8750-P1-L1

K05MJPG15360-EA-B100-C3-S32-H1-G128-W8750-P0-L2

Bibliography

[1] P. Bizony, "The ageless appeal of 2001: A space odyssey," *Nature*, vol. 555, no. 7698, 2018.

[2] C. Paige, D. Newman, D. Haddad, F. Ward, and T. Piercy, "Lunar instrument data integration into the virtual reality mission simulation system for decision making and situational awareness," in *Proc. 50th Int. Conf. Environ. Syst*, 2021, pp. 12–15.

[3] B. Day and E. Law, "Moon trek: Nasa's new online portal for lunar mapping and modeling," in *Annual Meeting of the Lunar Exploration Analysis Group*, vol. 1960, 2016, p. 5015.

[4] M. Minsky, "Proposal for a remotely manned space station," *NASA. Lewis Research Center, Vision-21: Space Travel for the Next Millennium*, 1990.

[5] G. Sanders, W. Larson, K. Sacksteder, and C. Mclemore, "Nasa in-situ resource utilization (isru) project: Development and implementation," in *AIAA SPACE 2008 Conference & Exposition*, 2008, p. 7853.

[6] J. L. Heldmann, A. Colaprete, R. C. Elphic, G. Mattes, K. Ennico, E. Fritzler, M. M. Marinova, R. McMurray, S. Morse, T. L. Roush *et al.*, "Real-time science operations to support a lunar polar volatiles rover mission," *Advances in Space Research*, vol. 55, no. 10, pp. 2427–2437, 2015.

[7] J. L. Heldmann, A. Colaprete, R. C. Elphic, D. Lim, M. Deans, A. Cook, T. Roush, J. Skok, N. E. Button, S. Karunatillake *et al.*, "Lunar polar rover science operations: Lessons learned and mission architecture implications derived from the mojave volatiles prospector (mvp) terrestrial field campaign," *Advances in Space Research*, vol. 58, no. 4, pp. 545–559, 2016.

[8] R. Weber, S. Lawrence, B. Cohen, J. Bleacher, J. Boyce, M. Collier, D. Draper, A. Fagan, C. Fassett, L. Gaddis *et al.*, "The artemis iii science definition team report," in *52nd Lunar and Planetary Science Conference*, no. 2548, 2021, p. 1261.

[9] J. E. Kleinhenz and A. Paz, "Case studies for lunar isru systems utilizing polar water," in *ASCEND 2020*, 2020, p. 4042.

[10] F. Ward, T. Piercy, J. Heldmann, D. Lim, A. Colaprete, A. Cook, and D. Newman, "A virtual reality mission simulation system (vmss) supporting closed-loop mission control." 50th International Conference on Environmental Systems, 2021.

[11] J. M. Boyce, P. Mouginis-Mark, and M. Robinson, "The tsiolkovskiy crater landslide, the moon: An lroc view," *Icarus*, vol. 337, p. 113464, 2020.

[12] M. Minsky, "Telepresence," 1980.

[13] C. Bracken and P. Skalski, "Telepresence in everyday life," *Immersed in media: Telepresence in everyday life*, pp. 5–8, 2010.

[14] E. Demaitre. (2022, Jan) Hyundai and boston dynamics discuss metamobility concept at ces 2022. Robotics 24/7. [Online]. Available: https://www.robotics247.com/article/hyundai_boston_dynamics_discuss_metamobility_concept_ces_2022/supply_chain

[15] A. Kristoffersson, S. Coradeschi, and A. Loutfi, "A review of mobile robotic telepresence," *Advances in Human-Computer Interaction*, vol. 2013, pp. 3–3, 2013.

[16] A. Orlandini, A. Kristoffersson, L. Almquist, P. Björkman, A. Cesta, G. Cortellessa, C. Galindo, J. Gonzalez-Jimenez, K. Gustafsson, A. Kiselev *et al.*, "Excite project: A review of forty-two months of robotic telepresence technology evolution," *Presence: Teleoperators and Virtual Environments*, vol. 25, no. 3, pp. 204–221, 2016.

[17] M. G. Smith, M. Kelley, and M. Basner, "A brief history of spaceflight from 1961 to 2020: An analysis of missions and astronaut demographics," *Acta astronautica*, vol. 175, pp. 290–299, 2020.

[18] R. Lange, L. Walker, M. Lenda, C. Morantz, T. Zorn, F. Alibay, L. DuCharme, and J. Koch, "Mars 2020 perseverance rover surface operations commissioning phase overview," in *2022 IEEE aerospace conference (AERO)*. IEEE, 2022, pp. 1–20.

[19] A. Witze *et al.*, "Nasa plans mars sample-return rover," *Nature*, vol. 509, no. 7500, p. 272, 2014.

[20] M. Golombek, R. Cook, T. Economou, W. Folkner, A. Haldemann, P. Kallemeyn, J. M. Knudsen, R. Manning, H. Moore, T. Parker *et al.*, "Overview of the mars pathfinder mission and assessment of landing site predictions," *Science*, vol. 278, no. 5344, pp. 1743–1748, 1997.

[21] J. Matijevic, "Sojourner: the mars pathfinder microrover flight experiment," *Space Technology*, vol. 3, no. 17, pp. 143–149, 1997.

[22] R. Team, "Characterization of the martian surface deposits by the mars pathfinder rover, sojourner," *Science*, vol. 278, no. 5344, pp. 1765–1768, 1997.

[23] M. Lemmon, M. Wolff, M. Smith, R. Clancy, D. Banfield, G. Landis, A. Ghosh, P. Smith, N. Spanovich, B. Whitney *et al.*, "Atmospheric imaging results from the mars exploration rovers: Spirit and opportunity," *Science*, vol. 306, no. 5702, pp. 1753–1756, 2004.

[24] J. S. Norris, M. W. Powell, M. A. Vona, P. G. Backes, and J. V. Wick, "Mars exploration rover operations with the science activity planner," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation.* IEEE, 2005, pp. 4618–4623.

[25] M. Raith. (2021, Aug) History of nasa mars rovers. Blue Marble Space Institute of Science. [Online]. Available: https://bmsis.org/history-of-nasa-mars-rovers/

[26] R. Welch, D. Limonadi, and R. Manning, "Systems engineering the curiosity rover: A retrospective," in *2013 8th international conference on system of systems engineering.* IEEE, 2013, pp. 70–75.

[27] D. W. Way, R. W. Powell, A. Chen, A. D. Steltzner, A. M. San Martin, P. D. Burkhart, and G. F. Mendeck, "Mars science laboratory: Entry, descent, and landing system performance," in *2007 IEEE Aerospace Conference.* IEEE, 2007, pp. 1–19.

[28] P. R. Mahaffy, C. R. Webster, S. K. Atreya, H. Franz, M. Wong, P. G. Conrad, D. Harpold, J. J. Jones, L. A. Leshin, H. Manning *et al.*, "Abundance and isotopic composition of gases in the martian atmosphere from the curiosity rover," *Science*, vol. 341, no. 6143, pp. 263–266, 2013.

[29] J. A. Rodriguez-Manfredi, M. De la Torre Juárez, A. Alonso, V. Apéstigue, I. Arruego, T. Atienza, D. Banfield, J. Boland, M. Carrera, L. Castañer *et al.*, "The mars environmental dynamics analyzer, meda. a suite of environmental sensors for the mars 2020 mission," *Space science reviews*, vol. 217, pp. 1–86, 2021.

[30] S.-E. Hamran, D. A. Paige, H. E. Amundsen, T. Berger, S. Brovoll, L. Carter, L. Damsgård, H. Dypvik, J. Eide, S. Eide *et al.*, "Radar imager for mars' subsurface experiment—rimfax," *Space Science Reviews*, vol. 216, pp. 1–39, 2020.

[31] R. Bhartia, L. W. Beegle, L. DeFlores, W. Abbey, J. Razzell Hollis, K. Uckert, B. Monacelli, K. S. Edgett, M. R. Kennedy, M. Sylvia *et al.*, "Perseverance's scanning habitable environments with raman and luminescence for organics and chemicals (sherloc) investigation," *Space Science Reviews*, vol. 217, no. 4, p. 58, 2021.

[32] J. A. Hoffman, M. H. Hecht, D. Rapp, J. J. Hartvigsen, J. G. SooHoo, A. M. Aboobaker, J. B. McClean, A. M. Liu, E. D. Hinterman, M. Nasr *et al.*, "Mars

oxygen isru experiment (moxie)—preparing for human mars exploration," *Science Advances*, vol. 8, no. 35, p. eabp8636, 2022.

[33] J. Bell, J. Maki, G. Mehall, M. Ravine, M. Caplinger, Z. Bailey, S. Brylow, J. Schaffner, K. Kinch, M. Madsen *et al.*, "The mars 2020 perseverance rover mast camera zoom (mastcam-z) multispectral, stereoscopic imaging investigation," *Space science reviews*, vol. 217, pp. 1–40, 2021.

[34] I. E. Sutherland, "Sketch pad a man-machine graphical communication system," in *Proceedings of the SHARE design automation workshop*, 1964, pp. 6–329.

[35] W. Buxton, R. Baecker, W. Clark, F. Richardson, I. Sutherland, W. B. Sutherland, and A. Henderson, "Interaction at lincoln laboratory in the 1960's: looking forward–looking back," in *CHI'05 Extended Abstracts on Human Factors in Computing Systems*, 2005, pp. 1162–1167.

[36] I. E. Sutherland *et al.*, "The ultimate display," in *Proceedings of the IFIP Congress*, vol. 2, no. 506-508. New York, 1965, pp. 506–508.

[37] M. W. Krueger, "An easy entry artificial reality," in *Virtual Reality*. Elsevier, 1993, pp. 147–161.

[38] M. W. Krueger, T. Gionfriddo, and K. Hinrichsen, "Videoplace—an artificial reality," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1985, pp. 35–40.

[39] N. Elmqaddem, "Augmented reality and virtual reality in education. myth or reality?" *International journal of emerging technologies in learning*, vol. 14, no. 3, 2019.

[40] C. Blanchard, S. Burgess, Y. Harvill, J. Lanier, A. Lasko, M. Oberman, and M. Teitel, "Reality built for two: a virtual reality tool," in *Proceedings of the 1990 symposium on Interactive 3D graphics*, 1990, pp. 35–36.

[41] Y. Liu and K. Goebel, "Information fusion for national airspace system prognostics: A nasa uli project," in *Proceedings of the 10th Annual Conference of the Prognostics and Health Management Society, PHM, Philadelphia Center City, Philadelphia, PA, USA*, 2018, pp. 24–27.

[42] J. Fung, F. Tang, and S. Mann, "Mediated reality using computer graphics hardware for computer vision," in *Proceedings. Sixth International Symposium on Wearable Computers,*. IEEE, 2002, pp. 83–89.

[43] S. Mann, "Mediated reality with implementations for everyday life," *Presence Connect*, vol. 1, p. 2002, 2002.

[44] S. Mann and W. Barfield, "Introduction to mediated reality," *International Journal of Human-Computer Interaction*, vol. 15, no. 2, pp. 205–208, 2003.

[45] P. Milgram and F. Kishino, "A taxonomy of mixed reality visual displays," *IEICE TRANSACTIONS on Information and Systems*, vol. 77, no. 12, pp. 1321–1329, 1994.

[46] R. T. Azuma, "A survey of augmented reality," *Presence: teleoperators & virtual environments*, vol. 6, no. 4, pp. 355–385, 1997.

[47] A. Hillstead, "Simulating psychedelic therapy through mediated reality," 2017.

[48] S. K. Feiner, "Augmented reality: A new way of seeing," *Scientific American*, vol. 286, no. 4, pp. 48–55, 2002.

[49] T. Masson, Daffy, and K. Perlin, "Holo-doodle: an adaptation and expansion of collaborative holojam virtual reality," in *ACM SIGGRAPH 2017 VR Village*, 2017, pp. 1–2.

[50] J. Herling and W. Broll, "Advanced self-contained object removal for realizing real-time diminished reality in unconstrained environments," in *2010 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2010, pp. 207–212.

[51] "What is mediated reality - interactive website and immersive infographics," https://mediatedreality.info/, 2022.

[52] J. Kauffman, "A successful failure: Nasa's crisis communications regarding apollo 13," *Public Relations Review*, vol. 27, no. 4, pp. 437–448, 2001.

[53] B. D. Allen, "Digital twins and living models at nasa," in *Digital Twin Summit*, 2021.

[54] K. Bruynseels, F. Santoni de Sio, and J. Van den Hoven, "Digital twins in health care: ethical implications of an emerging engineering paradigm," *Frontiers in genetics*, vol. 9, p. 31, 2018.

[55] S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox, "Scaling digital twins from the artisanal to the industrial," *Nature Computational Science*, vol. 1, no. 5, pp. 313–320, 2021.

[56] V. Qiuchen Lu, A. K. Parlikad, P. Woodall, G. D. Ranasinghe, and J. Heaton, "Developing a dynamic digital twin at a building level: Using cambridge campus as case study," in *International Conference on Smart Infrastructure and Construction 2019 (ICSIC) Driving data-informed decision-making*. ICE Publishing, 2019, pp. 67–75.

[57] O. Foundation, "How "digital twins" protect the artist," *Medium*, Feb 2022, 2 min read.

[58] K.-J. Wang, Y.-H. Lee, and S. Angelica, "Digital twin design for real-time monitoring–a case study of die cutting machine," *International Journal of Production Research*, vol. 59, no. 21, pp. 6471–6485, 2021.

[59] P. Aivaliotis, K. Georgoulias, and G. Chryssolouris, "The use of digital twin for predictive maintenance in manufacturing," *International Journal of Computer Integrated Manufacturing*, vol. 32, no. 11, pp. 1067–1080, 2019.

[60] S. West, O. Stoll, J. Meierhofer, and S. Züst, "Digital twin providing new opportunities for value co-creation through supporting decision-making," *Applied Sciences*, vol. 11, no. 9, p. 3750, 2021.

[61] M. Singh, E. Fuenmayor, E. P. Hinchy, Y. Qiao, N. Murray, and D. Devine, "Digital twin: Origin to future," *Applied System Innovation*, vol. 4, no. 2, p. 36, 2021.

[62] L. Li, S. Aslam, A. Wileman, and S. Perinpanayagam, "Digital twin in aerospace industry: A gentle introduction," *IEEE Access*, vol. 10, pp. 9543–9562, 2021.

[63] Q. Qiao, J. Wang, L. Ye, and R. X. Gao, "Digital twin for machining tool condition prediction," *Procedia CIRP*, vol. 81, pp. 1388–1393, 2019.

[64] D.-G. J. Opoku, S. Perera, R. Osei-Kyei, and M. Rashidi, "Digital twin application in the construction industry: A literature review," *Journal of Building Engineering*, vol. 40, p. 102726, 2021.

[65] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on industrial informatics*, vol. 15, no. 4, pp. 2405–2415, 2018.

[66] A. Padovano, F. Longo, L. Nicoletti, and G. Mirabelli, "A digital twin based service oriented application for a 4.0 knowledge navigation in the smart factory," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 631–636, 2018.

[67] E. Newman, J. Huang, M. Pomerantz, and J. Sellin, "Multi-project telemetry-based digital twin environment for space-mission development, analysis, and operations," in *2023 IEEE Aerospace Conference*. IEEE, 2023, pp. 1–12.

[68] A. D. Garcia, J. Schlueter, and E. Paddock, "Training astronauts using hardware-in-the-loop simulations and virtual reality," in *AIAA Scitech 2020 Forum*, 2020, p. 0167.

[69] D. D. Haddad, "Resynthesizing reality: Driving vivid virtual environments from sensor networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2018.

[70] M. Weiser and J. S. Brown, "Designing calm technology," *PowerGrid Journal*, vol. 1, no. 1, pp. 75–85, 1996.

[71] A. Ekblaw, J. Cherston, F.-Z. Liu, I. Wicaksono, D. D. Haddad, V. Sumini, and J. A. Paradiso, "From ubicomp to universe - moving pervasive computing research into space applications," *IEEE Pervasive Computing*, 2023.

[72] J. A. Paradiso and J. A. Landay, "Guest editors' introduction: Cross-reality environments," *IEEE Pervasive Computing*, vol. 8, no. 3, pp. 14–15, 2009.

[73] S. A. Munir, B. Ren, W. Jiao, B. Wang, D. Xie, and J. Ma, "Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing," in *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, vol. 2. IEEE, 2007, pp. 113–120.

[74] F. V. Paulovich, M. C. F. De Oliveira, and O. N. Oliveira Jr, "A future with ubiquitous sensing and intelligent systems," *ACS sensors*, vol. 3, no. 8, pp. 1433–1438, 2018.

[75] Y. Zhang, L. Sun, H. Song, and X. Cao, "Ubiquitous wsn for healthcare: Recent advances and future prospects," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 311–318, 2014.

[76] G. Dublon and J. A. Paradiso, "Extra sensory perception," *Scientific american*, vol. 311, no. 1, pp. 36–41, 2014.

[77] J. Lifton, M. Feldmeier, Y. Ono, C. Lewis, and J. A. Paradiso, "A platform for ubiquitous sensor deployment in occupational and domestic environments," in *Proceedings of the 6th international conference on Information processing in sensor networks*, 2007, pp. 119–127.

[78] J. Lifton, M. Laibowitz, D. Harry, N.-W. Gong, M. Mittal, and J. A. Paradiso, "Metaphor and manifestation cross-reality with ubiquitous sensor/actuator networks," *IEEE Pervasive Computing*, vol. 8, no. 3, pp. 24–33, 2009.

[79] J. Lifton and J. A. Paradiso, "Dual reality: Merging the real and virtual," in *Facets of Virtual Environments: First International Conference, FaVE 2009, Berlin, Germany, July 27-29, 2009, Revised Selected Papers 1*. Springer, 2010, pp. 12–28.

[80] G. Dublon, "Sensor (y) landscapes: Technologies for new perceptual sensibilities," Ph.D. dissertation, Massachusetts Institute of Technology, 2018.

[81] B. Mayton, G. Dublon, S. Russell, E. F. Lynch, D. D. Haddad, V. Ramasubramanian, C. Duhart, G. Davenport, and J. A. Paradiso, "The networked sensory landscape: Capturing and experiencing ecological change across scales," *Presence*, vol. 26, no. 2, pp. 182–209, 2017.

[82] R. Kleinberger, G. Dublon, J. A. Paradiso, and T. Machover, "Phox ears: a parabolic, head-mounted, orientable, extrasensory listening device." in *NIME*, 2015, pp. 30–31.

[83] S. Russell, G. Dublon, and J. A. Paradiso, "Hearthere: Networked sensory prosthetics through auditory augmented reality," in *Proceedings of the 7th Augmented Human International Conference 2016*, 2016, pp. 1–8.

[84] G. Dublon, L. S. Pardue, B. Mayton, N. Swartz, N. Joliat, P. Hurst, and J. A. Paradiso, "Doppellab: Tools for exploring and harnessing multimodal sensor network data," in *SENSORS, 2011 IEEE*. IEEE, 2011, pp. 1612–1615.

[85] D. D. Haddad, G. Dublon, B. Mayton, S. Russell, X. Xiao, K. Perlin, and J. A. Paradiso, "Resynthesizing reality: Driving vivid virtual environments from sensor networks," in *ACM SIGGRAPH 2017 Talks*, 2017, pp. 1–2.

[86] Mangopus, "Mission iss on the oculus quest," https://www.magnopus.com/projects/mission-iss, 2023, designed and Developed by Mangopus.

[87] C. Magerkurth, A. D. Cheok, R. L. Mandryk, and T. Nilsen, "Pervasive games: bringing computer entertainment back to the real world," *Computers in Entertainment (CIE)*, vol. 3, no. 3, pp. 4–4, 2005.

[88] M. W. McGreevy, "Virtual reality and planetary exploration," in *Virtual Reality*. Elsevier, 1993, pp. 163–197.

[89] R. B. Loftin, "Virtual environments for aerospace training," in *Proceedings of WESCON'94*. IEEE, 1994, pp. 384–387.

[90] ——, "Aerospace applications of virtual environment technology," *ACM SIGGRAPH Computer Graphics*, vol. 30, no. 4, pp. 33–35, 1996.

[91] B. R. Parsons, "Nasa ksc intern final report-virtual reality in stem engagement," Tech. Rep., 2019.

[92] B. Young, "Oled displays and the immersive experience," *Information Display*, vol. 34, no. 2, pp. 16–36, 2018.

[93] J. W. Hamstra, *The F-35 lightning II: from concept to cockpit*. American Institute of Aeronautics and Astronautics, Inc., 2019.

[94] A. K. Noor, "Potential of virtual worlds for remote space exploration," *Advances in Engineering Software*, vol. 41, no. 4, pp. 666–673, 2010.

[95] R. Léveillé, "Validation of astrobiology technologies and instrument operations in terrestrial analogue environments," *Comptes Rendus Palevol*, vol. 8, no. 7, pp. 637–648, 2009.

[96] W. J. Clancey, "A closed mars analog simulation: The approach of crew 5 at the mars desert research station," 2002.

[97] M. Reagan and N. M. Director, "Nasa extreme environment mission operations (neemo)," *LPI Contribution*, vol. 8036, pp. 16–18, 2021.

[98] C. Paige, D. D. Haddad, F. Ward, G. R. Todd, Jessica Osinski, A. Ekblaw, and D. Newman, "Data collection in svalbard, norway to test the use of virtual reality for lunar and planetary surface exploration."

[99] "Juno new origin, a space simulator game by simplerockets," https://www.simplerockets.com/, 2023.

[100] A. K. Noor, "The hololens revolution," *Mechanical Engineering*, vol. 138, no. 10, pp. 30–35, 2016.

[101] M. OMAN-REAGAN, "Telexploration, onsight, and hololens" on" mars," 2015.

[102] J. Lidawer, A. Winter, R. Crocco, M. Vona, and A. Byon, "Multi-platform immersive visualization of planetary, asteroid, and terrestrial analog terrain. sp aber."

[103] L. Bass, D. Siewiorek, M. Bauer, R. Casciola, C. Kasabach, R. Martin, J. Siegel, A. Smailagic, and J. Stivoric, "Constructing wearable computers for maintenance applications," pp. 663–694, 2001.

[104] V. Kohn and D. Harborth, "Augmented reality-a game changing technology for manufacturing processes?" in *ECIS*, 2018, p. 111.

[105] P. Higgins, G. Y. Kebe, A. Berlier, K. Darvish, D. Engel, F. Ferraro, and C. Matuszek, "Towards making virtual human-robot interaction a reality," in *Proc. of the 3rd International Workshop on Virtual, Augmented, and Mixed-Reality for Human-Robot Interactions (VAM-HRI)*, 2021.

[106] D. Szafir, "Mediating human-robot interactions with virtual, augmented, and mixed reality," in *Virtual, Augmented and Mixed Reality. Applications and Case Studies: 11th International Conference, VAMR 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Orlando, FL, USA, July 26–31, 2019, Proceedings, Part II 21*. Springer, 2019, pp. 124–149.

[107] J. Xiao, P. Wang, H. Lu, and H. Zhang, "A three-dimensional mapping and virtual reality-based human–robot interaction for collaborative space exploration," *International Journal of Advanced Robotic Systems*, vol. 17, no. 3, p. 1729881420925293, 2020.

[108] J. M. Albani and D. I. Lee, "Virtual reality-assisted robotic surgery simulation," *Journal of Endourology*, vol. 21, no. 3, pp. 285–287, 2007.

[109] R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt, "Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–33.

[110] S. Chernova, N. DePalma, E. Morant, and C. Breazeal, "Crowdsourcing human-robot interaction: Application from virtual to physical worlds," in *2011 RO-MAN*. IEEE, 2011, pp. 21–26.

[111] W. Hoenig, C. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian, "Mixed reality for robotics," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 5382–5387.

[112] A. Kelly, E. Capstick, D. Huber, H. Herman, P. Rander, and R. Warner, "Real-time photorealistic virtualized reality interface for remote mobile robot control," in *Robotics Research: The 14th International Symposium ISRR*. Springer, 2011, pp. 211–226.

[113] Z. Makhataeva and H. A. Varol, "Augmented reality for robotics: A review," *Robotics*, vol. 9, no. 2, p. 21, 2020.

[114] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard *et al.*, "Sim2real in robotics and automation: Applications and challenges," *IEEE transactions on automation science and engineering*, vol. 18, no. 2, pp. 398–400, 2021.

[115] S. Bustamante, J. Peters, B. Schölkopf, M. Grosse-Wentrup, and V. Jayaram, "Armsym: A virtual human–robot interaction laboratory for assistive robotics," *IEEE Transactions on Human-Machine Systems*, vol. 51, no. 6, pp. 568–577, 2021.

[116] A. Fuste, B. Reynolds, J. Hobin, and V. Heun, "Kinetic ar: A framework for robotic motion systems in spatial computing," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–8.

[117] R. Li, M. van Almkerk, S. van Waveren, E. Carter, and I. Leite, "Comparing human-robot proxemics between virtual reality and the real world," in *2019 14th ACM/IEEE international conference on human-robot interaction (HRI)*. IEEE, 2019, pp. 431–439.

[118] M. Chen, P. Zhang, Z. Wu, and X. Chen, "A multichannel human-swarm robot interaction system in augmented reality," *Virtual Reality & Intelligent Hardware*, vol. 2, no. 6, pp. 518–533, 2020.

[119] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "Flightgoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6941–6948.

[120] J. K. Haas, "A history of the unity game engine," *Diss. Worcester Polytechnic Institute*, vol. 483, no. 2014, p. 484, 2014.

[121] A. Koubâa *et al.*, *Robot Operating System (ROS)*. Springer, 2017, vol. 1.

[122] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Autonomous Robots*, vol. 46, no. 1, pp. 307–320, 2022.

[123] G. Schroeder, "Nasa's ingenuity mars helicopter: The first attempt at powered flight on another world." *American Scientist*, vol. 108, no. 6, pp. 330–331, 2020.

[124] B. A. Aikenhead, R. G. Daniell, and F. M. Davis, "Canadarm and the space shuttle," *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, vol. 1, no. 2, pp. 126–132, 1983.

[125] M. Smith, D. Craig, N. Herrmann, E. Mahoney, J. Krezel, N. McIntyre, and K. Goodliff, "The artemis program: An overview of nasa's activities to return humans to the moon," in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–10.

[126] 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*, vol. 5, no. 1, p. 1, 2008.

[127] D. S. Katz and R. R. Some, "Nasa advances robotic space exploration," *Computer*, vol. 36, no. 1, pp. 52–61, 2003.

[128] C. E. Harriott, T. Zhang, and J. A. Adams, "Evaluating the applicability of current models of workload to peer-based human-robot teams," in *Proceedings of the 6th international conference on Human-robot interaction*, 2011, pp. 45–52.

[129] R. C. Moeller, L. Jandura, K. Rosette, M. Robinson, J. Samuels, M. Silverman, K. Brown, E. Duffy, A. Yazzie, E. Jens *et al.*, "The sampling and caching subsystem (scs) for the scientific exploration of jezero crater by the mars 2020 perseverance rover," *Space Science Reviews*, vol. 217, pp. 1–43, 2021.

[130] K. Hambuchen, J. Marquez, and T. Fong, "A review of nasa human-robot interaction in space," *Current Robotics Reports*, vol. 2, no. 3, pp. 265–272, 2021.

[131] T. Fong, J. Scholtz, J. A. Shah, L. Fluckiger, C. Kunz, D. Lees, J. Schreiner, M. Siegel, L. M. Hiatt, I. Nourbakhsh *et al.*, "A preliminary study of peer-to-peer human-robot interaction," in *2006 IEEE International Conference on Systems, Man and Cybernetics*, vol. 4. IEEE, 2006, pp. 3198–3203.

[132] T. Fong, I. Nourbakhsh, C. Kunz, L. Fluckiger, J. Schreiner, R. Ambrose, R. Burridge, R. Simmons, L. Hiatt, A. Schultz *et al.*, "The peer-to-peer human-robot interaction project," in *Space 2005*, 2005, p. 6750.

[133] D. Szafir and D. A. Szafir, "Connecting human-robot interaction and data visualization," in *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, 2021, pp. 281–292.

[134] I. Rekleitis, J.-L. Bedwani, S. Gemme, T. Lamarche, and E. Dupuis, "Terrain modelling for planetary exploration," in *Fourth Canadian Conference on Computer and Robot Vision (CRV'07)*. IEEE, 2007, pp. 243–249.

[135] M. Maurette, "Mars rover autonomous navigation," *Autonomous Robots*, vol. 14, no. 2-3, pp. 199–208, 2003.

[136] A. Johnson, J. Hoffman, D. Newman, E. Mazarico, and M. Zuber, "An integrated traverse planner and analysis tool for planetary exploration," in *AIAA SPACE 2010 Conference & Exposition*, 2010, p. 8829.

[137] L. Jenner, "Nasa eyes gps at the moon for artemis missions, june 2019."

[138] M. J. Schuster, S. G. Brunner, K. Bussmann, S. Büttner, A. Dömel, M. Hellerer, H. Lehner, P. Lehner, O. Porges, J. Reill *et al.*, "Towards autonomous planetary exploration: The lightweight rover unit (lru), its success in the spacebotcamp challenge, and beyond," *Journal of Intelligent & Robotic Systems*, vol. 93, pp. 461–494, 2019.

[139] N. Abcouwer, S. Daftry, T. del Sesto, O. Toupet, M. Ono, S. Venkatraman, R. Lanka, J. Song, and Y. Yue, "Machine learning based path planning for improved rover navigation," in *2021 IEEE Aerospace Conference (50100)*. IEEE, 2021, pp. 1–9.

[140] C. R. Stoker, D. Burch, B. P. Hine, and J. Barry, "Antarctic undersea exploration using a robotic submarine with a telepresence user interface," *Ieee Expert*, vol. 10, no. 6, pp. 14–23, 1995.

[141] C. R. Stoker, "From antarctica to space: use of telepresence and virtual reality in control of a remote underwater vehicle," in *Mobile Robots IX*, vol. 2352. SPIE, 1995, pp. 288–299.

[142] N. Thomas, J. Hamilton, A. Veillet, and C. Muir, "Biologic analog science associated with lava terrains," *Biosignature Preservation and Detection in Mars Analog Environments*, vol. 1912, 2016.

[143] K. H. Beaton, S. P. Chappell, A. Menzies, V. Luo, S. Y. Kim-Castet, D. Newman, J. Hoffman, J. Norheim, E. Anandapadmanaban, S. P. Abercrombie *et al.*, "Mission enhancing capabilities for science-driven exploration extravehicular activity derived from the nasa basalt research program," *Planetary and Space Science*, vol. 193, p. 105003, 2020.

[144] M. E. Walker, H. Hedayati, and D. Szafir, "Robot teleoperation with augmented reality virtual surrogates," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 202–210.

[145] K. Chen, B. T. Lopez, A.-a. Agha-mohammadi, and A. Mehta, "Direct lidar odometry: Fast localization with dense point clouds," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2000–2007, 2022.

[146] "Nebula-spot explores complex environments without human guidance," https://www.jpl.nasa.gov/robotics-at-jpl/nebula-spot, 2021.

[147] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[148] T. N. Titus, J. J. Wynne, M. J. Malaska, A.-a. Agha-Mohammadi, P. B. Buhler, E. C. Alexander, J. W. Ashley, A. Azua-Bustos, P. J. Boston, D. L. Buczkowski *et al.*, "A roadmap for planetary caves science and exploration," *Nature Astronomy*, vol. 5, no. 6, pp. 524–525, 2021.

[149] J. G. Blank, "Preparing for robotic astrobiology missions to lava caves on mars: The braille project's mars science mission simulation at lava beds national monument (n. ca, usa)," in *AGU Fall Meeting Abstracts*, vol. 2019, 2019, pp. P44B–09.

[150] A. Agha, K. Otsu, B. Morrell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund *et al.*, "Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge," *arXiv preprint arXiv:2103.11470*, 2021.

[151] G. Kurillo, E. Hemingway, M.-L. Cheng, and L. Cheng, "Evaluating the accuracy of the azure kinect and kinect v2," *Sensors*, vol. 22, no. 7, p. 2469, 2022.

[152] V. Ramasubramanian, "Quadrasense: Immersive uav-based cross-reality environmental sensor networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2015.

[153] M. Wertheimer, "Gestalt theory." 1938.

[154] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[155] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[156] D. Ilett, "Introduction to shaders in unity," in *Building Quality Shaders for Unity: Using Shader Graphs and HLSL Shaders*. Springer, 2022, pp. 1–8.

[157] Q. Li, "Design and implementation of indoor disinfection robot system," 2023.

[158] A. Milanovic, S. Srbljic, and V. Sruk, "Performance of udp and tcp communication on personal computers," in *2000 10th Mediterranean Electrotechnical Conference. Information Technology and Electrotechnology for the Mediterranean Countries. Proceedings. MeleCon 2000 (Cat. No. 00CH37099)*, vol. 1. IEEE, 2000, pp. 286–289.

[159] T. B. Sheridan, "Space teleoperation through time delay: Review and prognosis," *IEEE Transactions on robotics and Automation*, vol. 9, no. 5, pp. 592–606, 1993.

[160] J. A. Paradiso, "Global steering of single gimballed control moment gyroscopes using a directed search," *Journal of Guidance, Control, and Dynamics*, vol. 15, no. 5, pp. 1236–1244, 1992.

[161] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 9, no. 1, pp. 1–22, 2013.

[162] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[163] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[164] G. Skinner and T. Walmsley, "Artificial intelligence and deep learning in video games a brief review," in *2019 ieee 4th international conference on computer and communication systems (icccs)*. IEEE, 2019, pp. 404–408.

[165] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[166] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *2016 IEEE conference on computational intelligence and games (CIG)*. IEEE, 2016, pp. 1–8.

[167] S. Nebel, S. Schneider, and G. D. Rey, "Mining learning and crafting scientific experiments: a literature review on the use of minecraft in education and research," *Journal of Educational Technology & Society*, vol. 19, no. 2, pp. 355–366, 2016.

[168] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[169] C. I. Tan, C.-M. Chen, W.-K. Tai, and S.-J. Yen, "An ai tool: Generating paths for racing game," in *2008 International Conference on Machine Learning and Cybernetics*, vol. 6. IEEE, 2008, pp. 3132–3137.

[170] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[171] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[172] E. Kasneci, K. Seßler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier *et al.*, "Chatgpt for good? on opportunities and challenges of large language models for education," *Learning and Individual Differences*, vol. 103, p. 102274, 2023.

[173] L. K. Smetana and R. L. Bell, "Computer simulations to support science instruction and learning: A critical review of the literature," *International Journal of Science Education*, vol. 34, no. 9, pp. 1337–1370, 2012.

[174] K. Kaneko, F. Kanehiro, S. Kajita, K. Yokoyama, K. Akachi, T. Kawasaki, S. Ota, and T. Isozumi, "Design of prototype humanoid robotics platform for hrp," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. IEEE, 2002, pp. 2431–2436.

[175] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[176] O. Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[177] S. Wolfram *et al.*, *A new kind of science*. Wolfram media Champaign, IL, 2002, vol. 5.

[178] E. Wetzel, J. Liu, T. Leathem, and A. Sattineni, "The use of boston dynamics spot in support of lidar scanning on active construction sites," in *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, vol. 39. IAARC Publications, 2022, pp. 86–92.

[179] A. Bouman, M. F. Ginting, N. Alatur, M. Palieri, D. D. Fan, T. Touma, T. Pailevanian, S.-K. Kim, K. Otsu, J. Burdick *et al.*, "Autonomous spot: Long-range autonomous exploration of extreme environments with legged locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2518–2525.

[180] S. Halder, K. Afsari, J. Serdakowski, S. DeVito, M. Ensafi, and W. Thabet, "Real-time and remote construction progress monitoring with a quadruped robot using augmented reality," *Buildings*, vol. 12, no. 11, p. 2027, 2022.

[181] L. Maguire, "Behind the scenes: The making of coperni's next act," *Vogue*.

[182] A. Forsey-Smerek, C. Paige, F. Ward, D. D. Haddad, L. Sanneman, J. Todd, J. Heldmann, D. Lim, and D. Newman, "Assessment of depth data acquisition methods for virtual reality mission operations support tools," in *2022 IEEE Aerospace Conference (AERO)*. IEEE, 2022, pp. 1–14.

[183] L. Skrba, L. Reveret, F. Hétroy, M.-P. Cani, and C. O'Sullivan, "Quadruped animation," in *Eurographics' 2008-29th annual conference of the European Association for Computer Graphics*. Eurographics, 2008, pp. 7–23.

[184] Z. Bhatti, A. Shah, M. Karabasi, and W. Mahesar, "Expression driven trigonometric based procedural animation of quadrupeds," in *2013 International Conference on Informatics and Creative Multimedia*. IEEE, 2013, pp. 104–109.

[185] M. Reagan, B. Janoiko, J. Johnson, S. Chappell, Ph. D, and A. Abercromby, "Nasa's analog missions: Driving exploration through innovative testing," in *AIAA SPACE 2012 Conference & Exposition*, 2012, p. 5238.

[186] P. Lee, "Mars on earth: The nasa haughton-mars project," *Ad Astra: The Magazine of the National Space Society*, vol. 14, no. 3, pp. 5–8, 2002.

[187] G. Caravaca, S. Le Mouélic, N. Mangold, J. L'Haridon, L. Le Deit, and M. Massé, "3d digital outcrop model reconstruction of the kimberley outcrop (gale crater, mars) and its integration into virtual reality for simulated geological analysis," *Planetary and Space Science*, vol. 182, p. 104808, 2020.

[188] E. M. Mikhail, J. S. Bethel, and J. C. McGlone, *Introduction to modern photogrammetry*. John Wiley & Sons, 2001.

[189] O. Liestol, "Pingos, springs and permafrost in spitsbergen,• rbok 1975," *Norsk Polarinstitutt, TromsO, Norway*, 1977.

[190] F. Ward, C. Paige, D. D. Haddad, J. Todd, J. Heldmann, D. Lim, A. Ekblaw, and D. Newman, "Multi-sensor 3d data visualization in virtual reality for planetary science and mission operations."

[191] D. M. Burr, K. L. Tanaka, and K. Yoshikawa, "Pingos on earth and mars," *Planetary and Space Science*, vol. 57, no. 5-6, pp. 541–555, 2009.

[192] T. L. Péwé, D. E. Rowan, R. H. Péwé, and R. Stuckenrath, *Glacial and periglacial geology of northwest Blomesletta peninsula, Spitsbergen, Svalbard*, 1982.

[193] H. Sevestre, D. I. Benn, N. R. Hulton, and K. Bælum, "Thermal structure of svalbard glaciers and implications for thermal switch models of glacier surging," *Journal of Geophysical Research: Earth Surface*, vol. 120, no. 10, pp. 2220–2236, 2015.

[194] E. Hauber, D. Reiss, M. Ulrich, F. Preusker, F. Trauthan, M. Zanetti, H. Hiesinger, R. Jaumann, L. Johansson, A. Johnsson *et al.*, "Landscape evolution in martian mid-latitude regions: insights from analogous periglacial landforms in svalbard," *Geological Society, London, Special Publications*, vol. 356, no. 1, pp. 111–131, 2011.

[195] C. Paige, "Enabling a permanent human presence beyond low earth orbit: wearable radiation protection and enhanced science through virtual reality," Ph.D. dissertation, Massachusetts Institute of Technology, 2023.

[196] C. A. Paige, D. D. Haddad, F. S. Ward, T. J. Piercy, J. E. Todd, J. A. Paradiso, and D. J. Newman, "Operational geology in a virtual environment (ogive) novel approaches to virtualizing geological expeditions for planetary exploration," in *Conference on Human-Computer Interaction for Space Exploration (SpaceCHI 3.0)*, June 2023, june 22 -23, 2023.

[197] C. Paige, F. Ward, D. D. Haddad, J. MacNeil, P. McGaffigan, A. Ekblaw, and D. Newman, "Mit zero-g outreach initiative: using experiment design and virtual reality to inspire the next generation of space scientists and engineers," *Acta Astronautica*, 2023.

[198] M. Tölgyessy, M. Dekan, L. Chovanec, and P. Hubinskỳ, "Evaluation of the azure kinect and its comparison to kinect v1 and kinect v2," *Sensors*, vol. 21, no. 2, p. 413, 2021.

[199] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3d: A modern library for 3d data processing," *arXiv preprint arXiv:1801.09847*, 2018.

[200] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb, "State of the art on 3d reconstruction with rgb-d cameras," in *Computer graphics forum*, vol. 37, no. 2. Wiley Online Library, 2018, pp. 625–652.

[201] I. Colomina and P. Molina, "Unmanned aerial systems for photogrammetry and remote sensing: A review," *ISPRS Journal of photogrammetry and remote sensing*, vol. 92, pp. 79–97, 2014.

[202] J.-S. R. Over, A. C. Ritchie, C. J. Kranenburg, J. A. Brown, D. D. Buscombe, T. Noble, C. R. Sherwood, J. A. Warrick, and P. A. Wernette, "Processing coastal imagery with agisoft metashape professional edition, version 1.6—structure from motion workflow documentation," US Geological Survey, Tech. Rep., 2021.

[203] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia *et al.*, "Meshlab: an open-source mesh processing tool." in *Eurographics Italian chapter conference*, vol. 2008. Salerno, Italy, 2008, pp. 129–136.

[204] P. Cignoni, G. Ranzuglia, M. Callieri, M. Corsini, F. Ganovelli, N. Pietroni, M. Tarini *et al.*, "Meshlab," 2011.

[205] Q.-Y. Zhou and V. Koltun, "Color map optimization for 3d reconstruction with consumer depth cameras," *ACM Transactions on Graphics (ToG)*, vol. 33, no. 4, pp. 1–10, 2014.

[206] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5556–5565.

[207] Y. Zhou, M. Daakir, E. Rupnik, and M. Pierrot-Deseilligny, "A two-step approach for the correction of rolling shutter distortion in uav photogrammetry," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 160, pp. 51–66, 2020.

[208] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997, pp. 209–216.

[209] F. Messaoudi, G. Simon, and A. Ksentini, "Dissecting games engines: The case of unity3d," in *2015 international workshop on network and systems support for games (NetGames)*. IEEE, 2015, pp. 1–6.

[210] S. W. Greenwald, "The equipped explorer: virtual reality as a medium for learning," Ph.D. dissertation, Massachusetts Institute of Technology, 2018.

[211] T. Hartmann, W. Wirth, H. Schramm, C. Klimmt, P. Vorderer, A. Gysbers, S. Böcking, N. Ravaja, J. Laarni, T. Saari *et al.*, "The spatial presence experience scale (spes): a short self-report measure for diverse media settings. j media psychol 28: 1–15," 2016.

[212] N. Keller, N. McHenry, C. Duncan, A. Johnston, R. S. Whittle, E. Koock, S. S. Bhattacharya, G. De La Torre, L. Ploutz-Snyder, M. Sheffield-Moore *et al.*, "Augmenting exercise protocols with interactive virtual reality environments," in *2021 IEEE aerospace conference (50100)*. IEEE, 2021, pp. 1–13.

[213] NASA, "Nasa's plan for sustained lunar exploration and development," 2020.

[214] A. Plan *et al.*, "Nasa's lunar exploration program overview," *NAaS Administration*, 2020.

[215] D. D. Haddad, C. A. Paige, B. Brokaw, F. S. Ward, J. A. Paradiso, J. Heldmann, and D. J. Newman, "Azure kinect a la luna (akall): Leveraging low-cost rgb and depth-camera in lunar exploration," in *IEEE Aerospace Conference (AeroAstro)*, 2024, in Proceedings.

[216] C. Paige, D. D. Haddad, F. Ward, A. Cook, V. Jhac, A. Deutsch, J. Shimadac, A. Colaprete, J. Heldmann, and D. Newman, "Development and testing of the concept of operations for a low-cost rgb and depth-camera for a lunar south pole mission," *Nature Microgravity*, 2023, waiting review.

[217] J. Bell III, S. Squyres, K. Herkenhoff, J. Maki, M. Schwochert, A. Dingizian, D. Brown, R. Morris, H. Arneson, M. Johnson *et al.*, "The panoramic camera (pancam) investigation on the nasa 2003 mars exploration rover mission," in *Lunar and Planetary Science Conference*, 2003, p. 1980.

[218] D. Scott, M. Malenkov, J. Head, and A. Basilevsky, "Robotic and manned lunar rovers of the xx century: The view from the xxi century," in *50th Annual Lunar and Planetary Science Conference*, no. 2132, 2019, p. 2827.

[219] "Mobile autonomous prospecting platform built for extraterrestrial and extreme environments - lunar outpost," https://lunaroutpost.com/robotics/, 2023.

[220] S. Zennaro, M. Munaro, S. Milani, P. Zanuttigh, A. Bernardi, S. Ghidoni, and E. Menegatti, "Performance evaluation of the 1st and 2nd generation kinect for multimedia applications," in *2015 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2015, pp. 1–6.

[221] "Microsoft azure kinect sdk documentation and hardware specification." https://learn.microsoft.com/en-us/azure/kinect-dk/, 2021.

[222] V. Jha, M. Garrett, B. White, J. Shimada, A. Cook, A. Colaprete, J. Heldmann, A. Dave, C. Paige, F. Ward *et al.*, "Ruggedizing a commercial depth camera for novel lunar exploration," in *AGU Fall Meeting Abstracts*, vol. 2022, 2022, pp. P55E–1617.

[223] D. D. Haddad, S. Unterhauser, C. A. Paige, B. Brokaw, F. S. Ward, J. A. Paradiso, J. Heldmann, and D. J. Newman, "Leveraging docker containers for azure kinect integration in space robotics: An overview of the azure kinect a la luna (akall) framework," in *Conference on Human-Computer Interaction for Space Exploration (SpaceCHI 3.0)*, June 2023, june 22 -23, 2023.

[224] K. P. Klaasen, M. J. Belton, H. H. Breneman, A. S. McEwen, M. Davies, R. J. Sullivan, C. R. Chapman, G. Neukum, C. M. Heffernan, A. P. Harch *et al.*, "Inflight performance characteristics, calibration, and utilization of the galileo solid state imaging camera," *Optical Engineering*, vol. 36, no. 11, pp. 3001–3027, 1997.

[225] A. Martin, S. Raponi, T. Combe, and R. Di Pietro, "Docker ecosystem–vulnerability analysis," *Computer Communications*, vol. 122, pp. 30–43, 2018.

[226] Y. Wan, J. Wang, J. Hu, T. Song, Y. Bai, and Z. Ji, "A study in 3d-reconstruction using kinect sensor," in *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE, 2012, pp. 1–7.

[227] C. C. Liebe, L. Scherr, and R. Willson, "Sun-induced veiling glare in dusty camera optics," *Optical Engineering*, vol. 43, no. 2, pp. 493–499, 2004.

[228] C. Paige, F. Ward, D. Haddad, J. MacNeil, P. McGaffigan, A. Ekblaw, and D. Newman, "Mit zero-g outreach initiative: Using experiment design and virtual reality to inspire the next generation of space scientists and engineers," *Acta Astronautica*, 2023.

[229] M. R. Apodaca Moreno, C. A. Paige, J. Stober, D. Rupasinghe, D. Wood, and D. Newman, "Capturing the moon: 3d mapping and regolith collection for low-cost lunar rover missions," in *ASCEND 2022*, 2022, p. 4285.

[230] M. Isachenkov, S. Chugunov, Z. Landsman, I. Akhatov, A. Metke, A. Tikhonov, and I. Shishkovsky, "Characterization of novel lunar highland and mare simulants for isru research applications," *Icarus*, vol. 376, p. 114873, 2022.

[231] M. Everingham and N. Pelster, "Lunar analog creation: Preparation and operation of a lunar regolith simulant testbed," in *AIAA SPACE 2009 Conference & Exposition*, 2009, p. 6510.

[232] C. A. Paige, D. D. Haddad, F. Ward, J. Todd, A. Ekblaw, and D. Newman, "Data collection in svalbard, norway to test the use of virtual reality for lunar and planetary surface," in *International Conference on Environmental Systems (ICES)*, 2023.

[233] R. Salzer and H. W. Siesler, *Infrared and Raman spectroscopic imaging*. John Wiley & Sons, 2014.

[234] H. Houshiar and A. Nüchter, "3d point cloud compression using conventional image compression for efficient data transmission," in *2015 XXV International Conference on Information, Communication and Automation Technologies (ICAT)*. IEEE, 2015, pp. 1–8.

[235] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[236] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still image data compression standard*. Springer Science & Business Media, 1992.

[237] K. Cowing. (2023, July) An inside look at nokia's moon mission. Spaceref. [Online]. Available: https://spaceref.com/newspace-and-tech/an-inside-look-at-nokias-moon-mission/

[238] K.-M. Cheung and K. Tong, "Proposed data compression schemes for the galileo s-band contingency mission," in *NASA CONFERENCE PUBLICATION*. NASA, 1993, pp. 99–99.