Back2Future-SIM: Creating Real-Time Interactable Immersive Virtual World For Robot Teleoperation

by

Sait Akturk

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

In the context of human-robot interaction (HRI), robot autonomy addresses how environmental input influences a robot's actions, spanning a spectrum from full human control to independent robot motion. The quality of HRI relies on information exchange, evaluated through factors like interaction duration, mental workload, and spatial awareness. Sensory feedback is crucial, with a growing interest in combining communication methods for more natural interactions. Challenges arise in distant proximity interactions with communication delays, where predictive sensory feedback emerges as a solution to enhance interaction efficiency by providing predictive information from the robot's environment.

In this thesis, our focus is on providing predictive haptic feedback and immersive visuals from the virtual replica of the remote scene in a physics simulator. Our system acts as a bridge between the operator and the follower robot, considering real-world constraints. We create a cyber-physical system using a real-time 3D surface mesh reconstruction algorithm and a digital twin of the Barrett WAM arm robot. The Gazebo physics simulator integrates the digital twin and an incremental surface mesh to create a real-time virtual replica of the remote scene. This virtual replica is used to provide haptic surface interaction feedback through collision detection from the physics simulator. Additionally, we address the operator's spatial awareness by using an immersive display for predictive visualization.

Preface

I begin this thesis by creating a Docker file for the CARV with the ORB-SLAM2 method implemented by Jun Jin. I became interested in SLAM and 3D mapping with the CARV algorithm through the 3D computer vision course, during which I developed a 3D tracking algorithm using MTF-based registration tracking. While reviewing previous literature on 3D mapping and robotics, I realized that researchers often underestimate the importance of real-world challenges and limitations for robotics. I believed that I could develop a system that addressed those challenges and facilitated the use of robotic systems in everyday life.

I began by exploring the available code and systems that I could use to integrate and improve upon previous work related to predictive visualization and haptic feedback systems. My supervisors, Prof. Martin Jagersand and Prof. Li Cheng guided me in identifying the system requirements and designed the system with CARV, incorporating the replication of the remote environment by integrating the digital twin of the Barrett WAM arm robot to provide predictive visualization and haptic feedback. We discussed how to incorporate haptic feedback through the interaction of the surface mesh reconstruction and the digital twin within the physics simulator. I gained knowledge about configuring the Gazebo physics simulator, its extendability, and compatibility with other programs such as libbarrett for the digital twin and synchronization with the follower robot. We also implemented real-time semi-dense CARV algorithm to generate a 3D mesh representation, with the assistance of creator Junaid Ahmad [1]. Point-based techniques were initially created by David Lovi and Neil Birkbeck [2]. These techniques were later extended to 3D line detection by He

and Qin [3]. Additionally, Ahmad extended the technique to planes, where I helped ensure that the system ran in an online manner and also set up the experiments. Additionally, we integrated a head-mounted display for video streaming obtained through a Gazebo sensor to provide predictive visualization with the operator's pose.

Further details about the system can be found in Chapter 3, while challenges and how we overcame them using different system choices are discussed in Chapter 4. We tested our system on EuRoC MAV data sequences as well as in real-world settings. These experiments culminated in the submission to the *IEEE International Conference on Robotics and Automation (ICRA 2024)*, titled *Immersive Human-in-the-Loop Control: Real-Time 3D Surface Meshing and Physics Simulation*, co-authored by Justin Valentine, Junaid Ahmad, and Martin Jagersand. Junaid Ahmad contributed to implementing the online semi-dense CARV algorithm and handling 3D representation and pose-related aspects, while Justin Valentine helped improve the paper and the submission of the paper.

Acknowledgements

I would like to dedicate my thesis to Suleyman Agirman, who taught me the importance of critical thinking and learning. I would also like to thank my family for always being with me and supporting me through the hardest times, despite their own struggles. I had the opportunity to learn and improve myself and, in a way, complete one phase of my dream. I would like to extend my gratitude to my supervisors, Prof. Martin Jagersand and Prof. Li Cheng, for always being supportive and helpful.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robot autonomy in HRI encompasses the mapping of input from the environment to robot actions. The degree of autonomy in robotics can be defined as a robot's ability to function independently without human intervention. In human-robot operations, the level of autonomy falls along a spectrum, ranging from humans providing inputs for all robot actions to the robot moving independently. In Figure 1.1, the degrees of autonomy on this spectrum can be found, providing a visual representation of the varying levels of autonomy in human-robot operations. Human-in-the-loop operation is a vital approach in robotics and automation, as full autonomy is often unattainable in complex domains. This collaborative approach leverages human expertise alongside robotic systems, enhancing performance and safety. It finds applications in various areas, including high-risk environments [4–6], medical surgery, telenursing [7–9], disaster relief scenarios [10, 11], and space exploration [12, 13]. In the context of HRI, robot autonomy is just one facet of the broader picture. Another crucial aspect that affects interaction quality is how information is shared between humans and robots. Information exchange plays a pivotal role in determining the success rate and efficiency of the interaction. Several metrics can be used to assess interaction quality, including interaction time, mental workload, situation awareness, and the shared platform between humans and robots [14–17].

The information exchange process in HRI is influenced by the communication

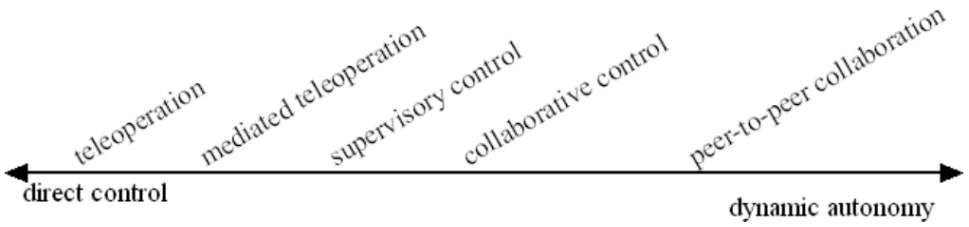

Figure 1.1: The degrees of autonomy with a focus on HRI [18]

medium and format. The communication medium encompasses human sensory perception, including sight and touch. In HRI, these sensory inputs take various forms, including visual displays, physical interactions, and haptics. Recently, there has been a growing interest in developing multimodal interfaces that combine multiple communication modalities to reduce cognitive workload and enhance natural interactions. Visual information formats range from traditional 2D displays to immersive virtual reality displays [18]. Haptic information formats include warnings through vibrations and enhancing spatial awareness through haptic hardware.

Proximity plays a fundamental role in HRI, where the simplicity of information exchange is often determined by the physical distance between the human and the robot. Close proximity interactions occur when the robot and human share the same physical space. This is in contrast with remote interaction settings, where the human and the robot are not physically co-located. In robot teleoperation, where a human operator interacts with a remote robot, communication between the operator and the robot is critical. The human operator's ability to understand the remote-robot interaction is directly tied to the proximity of the interaction; as the distance between them increases, communication delays become more pronounced, limiting the effectiveness of transparent sensory feedback in practical applications. Standard streaming of visual and haptic sensory feedback is insufficient for effective interaction due to these delays. Consequently, communication delays can lead to performance issues and a sense of disconnection between the human operator and the robot. These

delays can have serious consequences, ranging from potential damage to the robot or its surroundings to the failure of critical tasks, which could be life-threatening in fields like medicine. Thus, addressing and mitigating these delays is essential to ensuring safe, efficient, and effective human-robot interactions.

One approach to address this issue is by providing predictive sensory feedback. Predictive sensory feedback involves rendering the feedback of the robot's environment interactions in response to the operator's control without waiting for actual sensor feedback [19, 20].

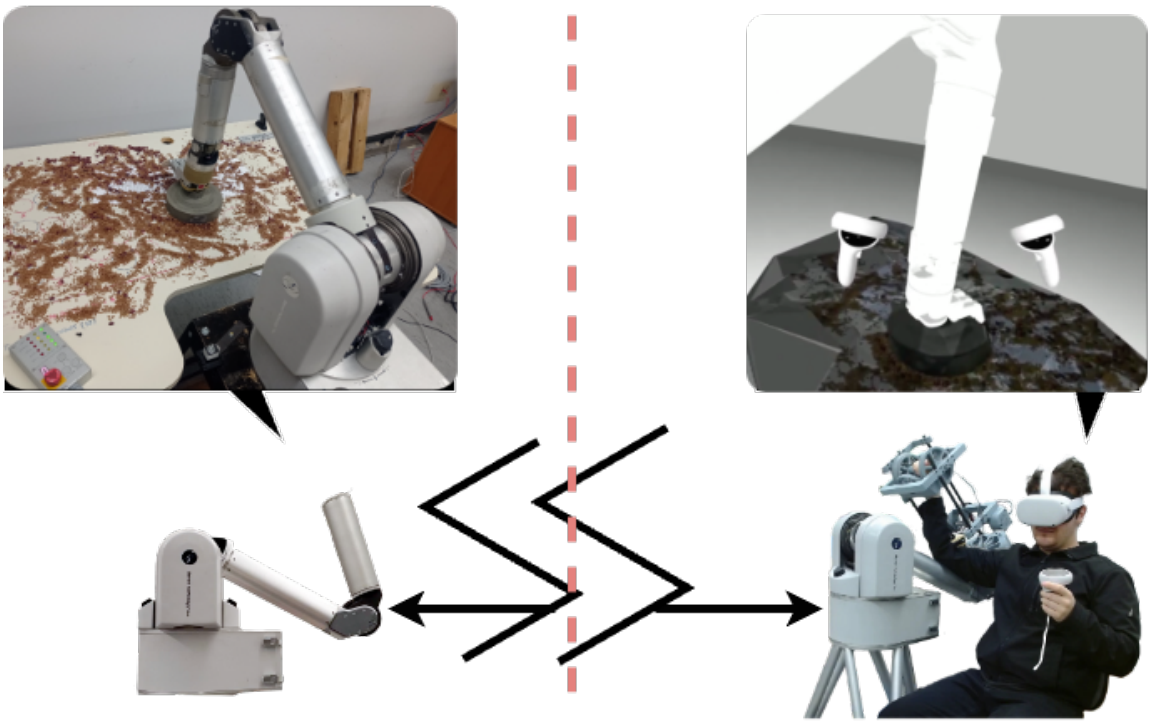

Figure 1.2: The proposed system for predictive immersive visualization and haptic feedback, using the surface mesh of the remote environment with the digital twin of the Barrett WAM arm robot.

This thesis delves into the challenges of robot teleoperation and proposes a predictive haptic and immersive visualization system. The primary focus of this thesis is to address the delay problem in visual and control feedback from the follower robot. In the bilateral robot teleoperation setting, the leader robot is controlled by the human operator to move the follower robot in the remote environment synchronously.

In conventional robot teleoperation settings, creating immediate visual and control feedback for the operator depends on low-latency and high-bandwidth networks. We propose a virtual intermediate system that updates incrementally based on remote environment states and provides predictive visual and haptic feedback. System details can be found in Figure 1.2, which illustrates the proposed system using the surface mesh of the remote environment and the digital twin of the Barrett WAM arm robot.

The central focus of this thesis is to address the question: 'How can we establish an optimal representation for human operators in human-in-the-loop operations?' This investigation delves into the development of a cyber-physical system that ensures immersive visualization and haptic feedback for human operators, incorporating the integration of an incremental surface mesh and the digital twin of the WAM arm robot within a shared virtual scene created in the Gazebo physical simulator.

We aim to explore the intricacies surrounding human-in-the-loop operations, particularly focusing on robot teleoperation. We carefully examine the essential requirements, challenges, and inherent limitations associated with this domain. Notably, cyber-physical systems have been at the forefront of robot teleoperation since the 1960s and continue to be a viable solution to this day. The development of an ideal representation in human-in-the-loop operations confronts inherent challenges stemming from the need to synchronize information between the operator and follower robots in low-latency and high-frequency settings. Ensuring this synchronization is crucial because it allows for smooth robot control while also delivering real-time visual and haptic feedback to the remote operator. The role of visual and haptic feedback assumes particular significance as it empowers the human operator with crucial insights and aids in the effective execution of tasks involving manipulation.

In the context of human-in-the-loop operations, providing visual feedback from the remote scene through a simple video feed has its limitations, as it lacks depth information, making robotic operations challenging and extending task completion time [21]. We seek a 3D representation that allows the operator to move the scene

independently from the robot camera. We consider various 3D representations, and among them, surface meshes stand out for their advantages in real-time updates, transmission, and memory requirements. The surface mesh representation is found to be an effective representation for human-in-the-loop systems to provide predictive visualization [20, 22]. In this thesis, we leverage ORBSLAM-2 [23] for localization and extracting sparse point clouds using key points. These sparse point clouds serve as the foundation for creating tetrahedrons via Delaunay triangulation and employing space carving techniques to incrementally update the surface mesh by identifying filled or empty tetrahedrons [2]. Additionally, we utilize the semi-dense module to acquire line and plane segments, thereby enhancing precision and reducing mesh complexity, leading to more efficient human-robot collaboration [1]. Through these techniques, we aim to enhance the predictive visual representation in human-in-the-loop operations, enabling smoother and more effective robotic tasks in diverse scenarios.

To provide predictive visual and haptic feedback, we used the Gazebo physics simulator to integrate the incremental surface mesh and the digital twin of the Barrett WAM arm robot. We aim to provide full predictive immersion and haptic feedback to the operator with minimal latency. The digital twin of the WAM arm robot's state is updated based on joint data obtained from the follower WAM arm robot. The Open Dynamics Engine (ODE) is used to detect collisions by analyzing the interaction between the incremental surface mesh and the digital twin WAM arm robot. The collision detection information is then used to provide haptic feedback to the operator.

Lastly, we integrated a head-mounted display with a controller to achieve a fully immersive experience with predictive visualization. The advantage of having a replica of the remote scene in full context lies in the ability to update the camera pose as desired by the operators. This helps to eliminate the requirement of increasing the bandwidth or motion of the follower robot. To provide immersion for the head-mounted display, we utilized the Gazebo wide-angle camera sensor. Predictive visualization is achieved

through the integration of head motions and the hand controller joystick. Our immersive system is intentionally designed to be independent and straightforward using web-based implementation, making it adaptable for use as an alternative type of head-mounted display in the future. By exploring and implementing these components, we aim to enhance the HRI experience, paving the way for more effective and seamless human-in-the-loop operations in various scenarios.

In summary, our proposed system provides a feasible solution for the delay problem of human-in-the-loop operations using predictive haptic and visual feedback by the virtual intermediate representation. Our developed system has practical applications in real-world scenarios, as we have taken careful considerations of trade-offs to ensure a realistic and effective implementation.

In the upcoming Chapter 2, we delve into the background research pertaining to 3D reconstruction, physics simulators, and immersive displays, which form the foundation of our work. Following that, Chapter 3 presents a detailed account of our system's architecture and design. Later, in Chapter 4, we discuss the experimental setups and results, evaluating the performance of various components and the overall system for robot teleoperation in diverse settings. Finally, we conclude by addressing the limitations of the current setup and outlining potential areas for future improvements and advancements in Chapter 5.

Chapter 2

Related Work

In this chapter, we reviewed the related work with regards to real-time 3D reconstruction in section 2.1, the physics simulator integration in section 2.2, and immersive display in section 2.3 for human-in-the-loop operations. For this chapter, we begin with 3D representations and reconstruction algorithms used for human-in-the-loop operations. The 3D reconstruction algorithms are evaluated based on memory and computing requirements, as well as the trade-off between the precision and speed. The currently used reconstruction algorithms in robot teleoperation are included in the background review as well. In the following subsection, we analyzed the physics simulator for robotics, including the integration with the robot arms, 3D models and rendering, and also interaction between the modalities. Lastly, we investigated immersive display on teleoperation and the effect on the operator and therefore the task completion rates and time.

2.1 3D Scene Reconstruction

In the context of our work, 3D scene reconstruction plays a pivotal role in enabling robotic systems to perceive and interact with their environments. This process involves the conversion of 2D images or sensor data into a comprehensive 3D representation of the scene. Various computer vision algorithms have been extensively studied and applied in the field of robotics, particularly in 3D robotic mapping, to achieve

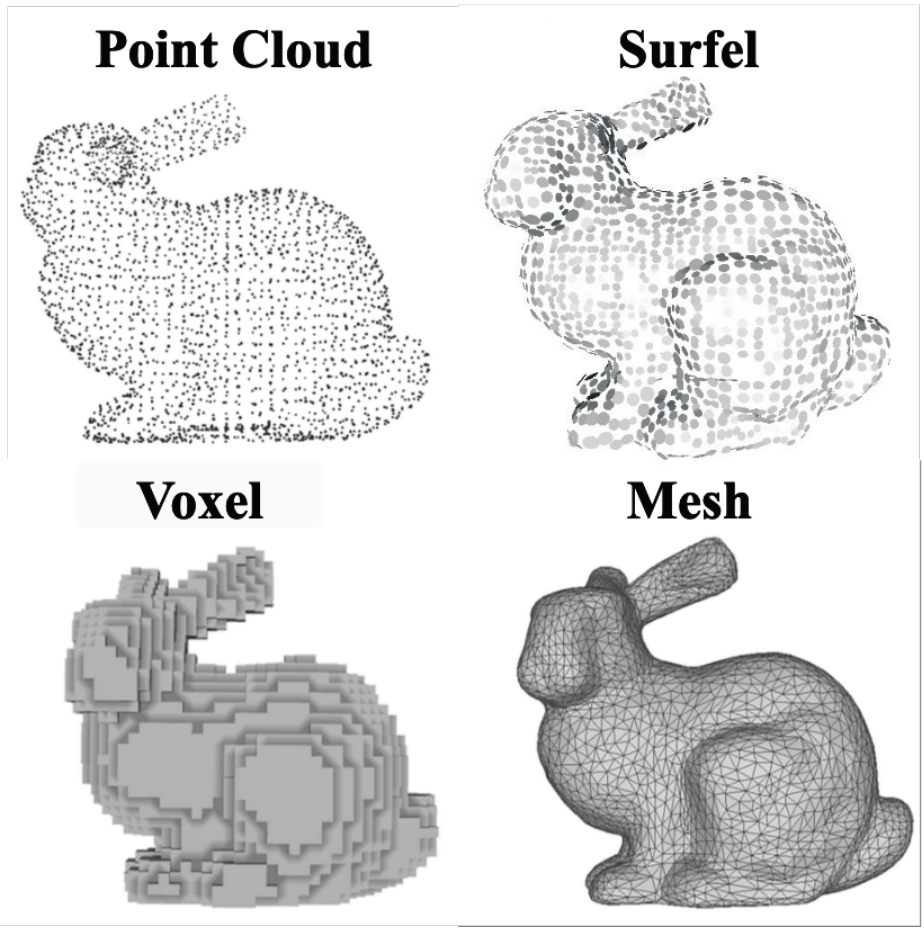

Figure 2.1: The Stanford Bunny [24] 3D model representation as a point cloud, surfel, voxel, and polygonal mesh representation.

accurate and efficient scene reconstruction. This reconstructed scene serves as the foundation for robot manipulation and navigation, allowing HRI to make informed decisions based on spatial awareness. The choice of representation for this 3D data is equally crucial, as it influences how effectively robots can understand and interact with their surroundings. Therefore, a well-designed representation system is an integral part of our approach to 3D scene reconstruction in the context of robotics.

Early attempts at 3D scene reconstruction for robot teleoperation and telepresence encountered challenges related to hardware limitations for real-time processing and transmission, as well as inadequate representation of high-fidelity details. Multi-view

reconstruction methods were also hindered by hardware limitations and insufficient 3D data representation [25–29]. Shape-from-silhouette methods, when employed, faced difficulties in achieving high-fidelity reconstruction results, particularly in surfaces containing concave regions [30, 31].

The reconstructed scene could be represented using point clouds, surfels, voxels, signed distance functions, and meshes. In Figure 2.1, examples of each representation are shown on the Stanford Bunny [24]. Each scene representation has its own unique challenges in terms of memory size, acquisition, transmission, and geometric representation capability.

2.1.1 3D Representations

2.1.1.1 Point Cloud Representation

The point cloud representation comprises a collection of individual data points situated in a three-dimensional space, with each point denoting a specific spatial position. This representation can be acquired in two ways: densely, through the utilization of depth and multi-camera systems, or sparsely, through feature-based RGB computer vision methods. The point cloud representation offers the advantage of not requiring preprocessing and being parallelizable for the representation of a 3D scene. However, it presents challenges in terms of memory requirements and the absence of geometric information. Overall, it makes processing, storage, transmission, and scalability more complex.

In the area of robotic teleoperation, transmitting point cloud data from a single depth camera poses additional hurdles due to the demand for network bandwidth. To put it into perspective, the transmission of data from a consumer-grade 3D camera typically necessitates an average bandwidth ranging from 600 to 800 Mbit/s, depending on the chosen resolution [32], while RGB video stream requires 1 to 50 Mbit/s. Despite these challenges, researchers have explored the application of point cloud representation in conjunction with RGB images to determine features like distance from

Figure 2.2: Example of visualization through a colored point cloud[32].

the target and robot state [33]. Moreover, as can be seen in Figure 2.2, the point cloud representation finds utility in identifying unknown objects within a given scene [32]. This is possible since the point cloud representation is directly obtained from depth cameras.

2.1.1.2 Surfel Representation

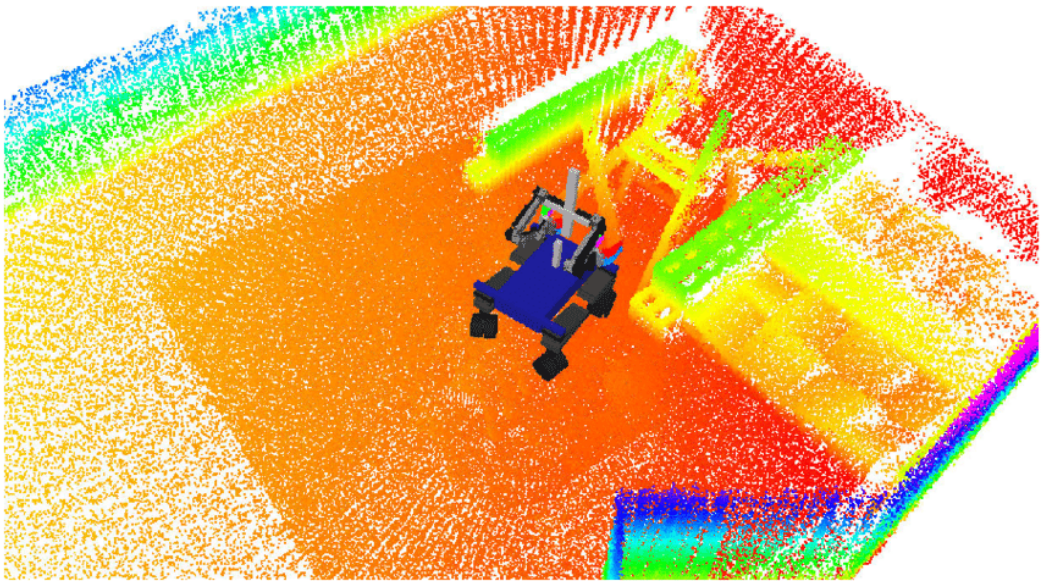

Figure 2.3: The point cloud from the egocentric multiresolution surfel map, as observed through the robot operator's perspective.[34].

The surfel representation consists of flat point patches with orientation, size, and color [35]. Compared to the point cloud representation, the surfel representation requires less memory and contains more information about the surface. The surfel representation can efficiently handle changes in the topology. Consequently, the surfel representation maintains a description of the scene that includes objects undergoing nonrigid deformation, potentially facilitating interactions with the dynamic working environments [36], as demonstrated by the mobile robot environment example shown in Figure 2.3. Despite these benefits, the surfel representation lacks connectivity and requires millions of surfels to represent a small area [37].

2.1.1.3 Volumetric Representation

The incorporation of a volumetric representation into a given framework involves the explicit modeling of volumetric attributes within the scene. This method enables a comprehensive depiction of the scene's intricate geometric properties, facilitating a deeper analysis.

Volumetric representation can be constructed using various techniques. One such technique involves using a voxel grid to discretize the scene into a three-dimensional grid of voxels, as shown in Figure 2.4. However, for improved efficiency and memory utilization, an alternative approach utilizes an octree structure [38]. This hierarchical subdivision of the scene into smaller octants allows for the adaptive allocation of computational resources. The intermediate scene representation obtained from the octree or voxel grids enables the creation of a signed distance function(SDF) [39], which quantifies the distance between each point in space and the scene's surface.

Following the widespread adoption of commodity depth cameras, a distinct class of 3D reconstruction algorithms emerged, drawing inspiration from the KinectFusion framework [40]. These algorithms leverage the concept of truncated signed distance function (TSDF) to achieve robust and precise reconstructions of 3D scenes. Although KinectFusion can handle high sensor noise to create small scenes in real-time, the voxel

grids are bound to a predefined volume and resolution and are highly inefficient in terms of memory usage. In real-time scene reconstruction, the voxel grid size and resolution are typically restricted by the available GPU memory. The voxel hashing algorithm is a memory-efficient implementation that enables larger scenes. [41]. To improve the accuracy of the 3D volumetric map, the bundle adjustment is used for loop-closure to enhance camera poses for larger scenes [42]. In robotics, SDFs can efficiently calculate the distance to obstacles, which makes the representation suitable for collision avoidance [43, 44]. Although the high fidelity of the reconstructed results makes it suitable for precise telemanipulation, due to the bandwidth requirements of around 175 Mbit/s [45] the volumetric representation is not suitable for the low-latency robot teleoperation.

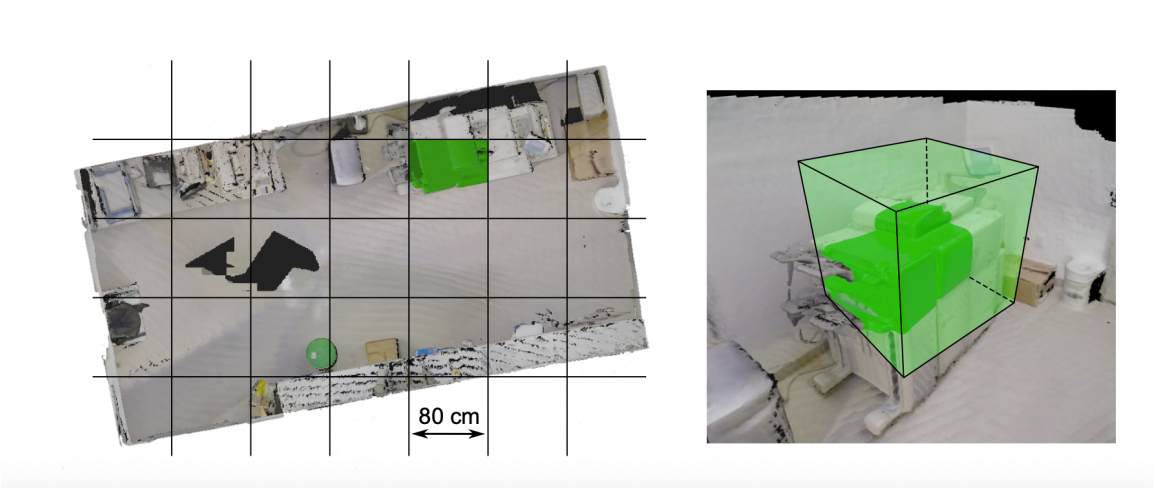

Figure 2.4: From Dividing the scene into separate mesh blocks, with each block having its own individual mesh.[45].

2.1.1.4 Mesh Representation

The mesh representation is a collection of vertices, edges, and faces that define the shape of a polyhedral object. The faces usually consist of triangles, quadrilaterals, or other simple convex polygons since this simplifies rendering. The meshes save important information related to texture, illumination for rendering, and geometric connectivity. In comparison to the aforementioned scene representations, surface mesh

reconstruction stands out as a highly efficient representation that balances memory usage and adjustable precision while preserving high fidelity. Additionally, an important advantage of this representation lies in the surface normals derived from the triangle faces, which offer valuable information for contact interaction and manipulation tasks. Obtaining an accurate mesh representation from a real-time 3D point cloud remains a persistent challenge.

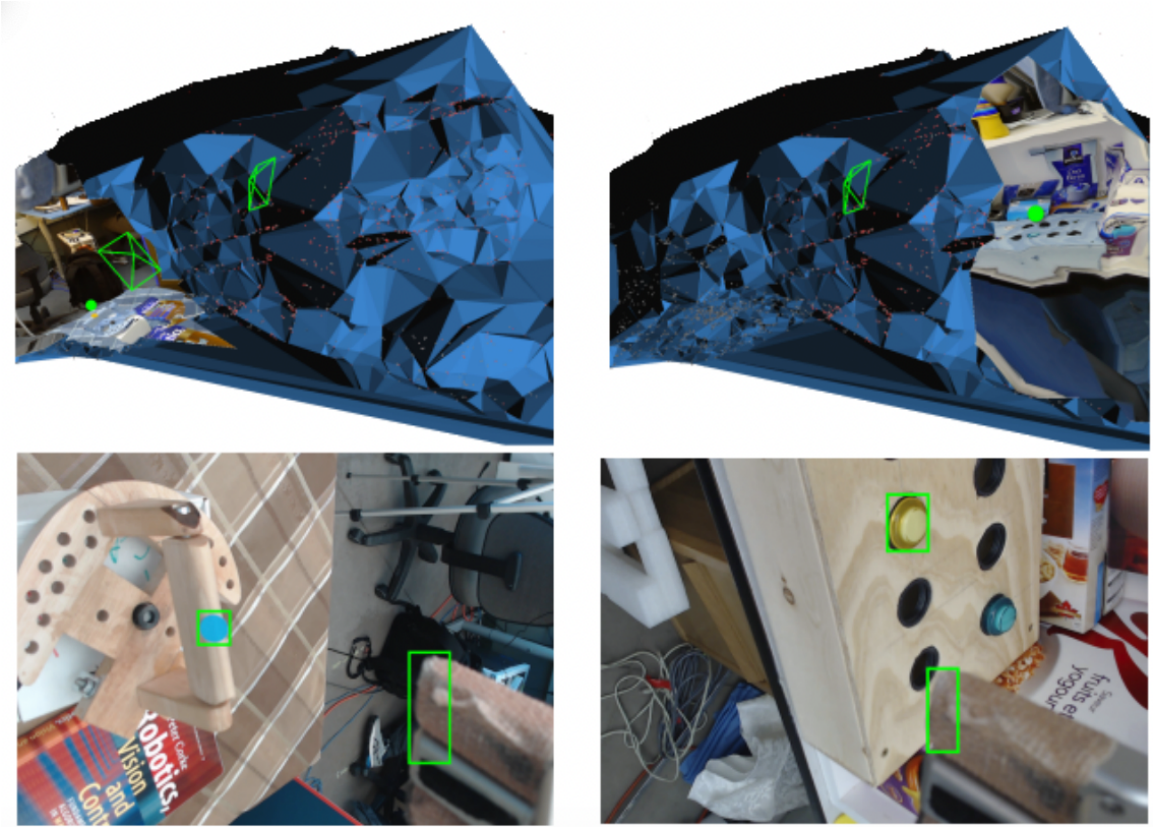

Figure 2.5: An interface for task specification that utilizes visual telepresence with mesh representation.

Existing algorithms, such as marching cubes [46] and Delaunay triangulation, exhibit time complexities that are quadratic or cubic in nature, posing difficulties in reconstructing surface meshes for larger scenes. To mitigate this issue, one approach involves utilizing incremental updates instead of updating the entire mesh surface for each scene [47]. The level of detail in the resulting mesh reconstruction varies depending on the density of the sparse point cloud [2, 48, 49]. Figure 2.5 illustrates

the incremental surface mesh reconstruction for task specification in the context of visual telepresence [50], which uses a sparse point cloud. Notably, the surface mesh representation proves to be more user-friendly and imposes a lesser cognitive load compared to the point cloud representation [51].

2.1.2 Sparse Feature Based Reconstruction

2.1.2.1 Simultaneous Localization and Mapping(SLAM)

Simultaneous localization and mapping (SLAM) algorithms enable real-time localization of a moving camera and the creation of the scene map. These algorithms can be categorized into two main groups based on the usage of feature detection. Direct SLAM algorithms, exemplified by LSD-SLAM [52], utilize pixel information and multi-frame photometric error for pixel depth calculation and camera pose estimation. Figure 2.6 shows the example output for outdoor scene reconstruction with LSD-SLAM. On the other hand, indirect SLAM algorithms such as PTAM [53] and ORB-SLAM [54] rely on feature detection algorithms to identify key points and corresponding descriptors for tracking across multiple images.

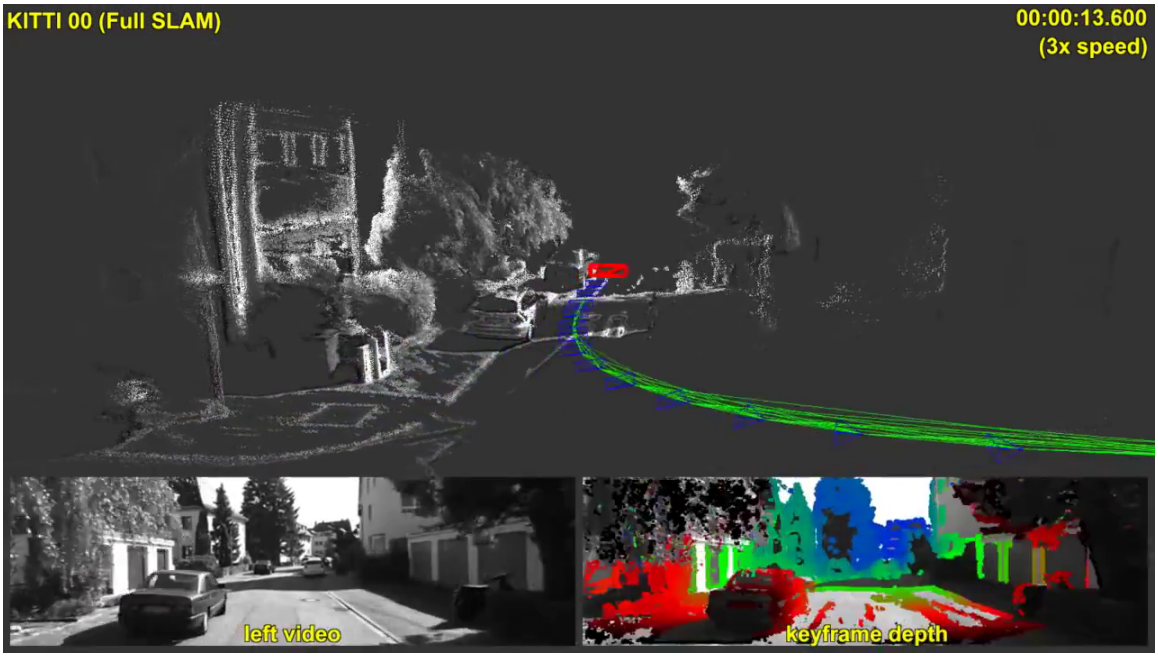

Figure 2.6: An example output generated by LSD-SLAM.[52].

Sparse feature-based reconstruction models leverage indirect SLAM algorithms to estimate the camera's pose and identify sparse features. Through the utilization of a moving camera and multiple images, features detected in these images are matched via tracking. Subsequently, triangulation is employed to determine the camera's pose based on the tracked features. Nonlinear optimization techniques, such as the bundle adjustment algorithm [55], are then used to refine the camera pose and matched features, either locally or globally. Indirect SLAM algorithms excel in scenes characterized by rich textures and distinctive corners, as these elements facilitate feature detection for reliable tracking.

2.1.2.2 Indirect Monocular Slam

Indirect SLAM algorithms are compatible with single, stereo, and depth cameras. Specifically, the monocular SLAM system operates with a single moving camera. The pioneering real-time monocular system, MonoSLAM, employs an extended Kalman filter and over a hundred probabilistic features to achieve reliable performance [56], as demonstrated in the example results shown in Figure 2.7.

(a) (b)

Figure 2.7: (a)An image capturing the probabilistic three-dimensional map. (b) Visually salient feature patches identified as visual landmarks, along with the corresponding 3D planar regions.[56].

PTAM employs a multi-threaded approach to enhance accuracy and speed by running SLAM and bundle adjustment locally and globally on separate threads. Leveraging this parallelization, PTAM achieves the capability to track numerous distinct FAST features across multiple images. Keyframes, comprising images with promi-

nent features based on spatial and temporal criteria, are created by PTAM. Through bundle adjustment optimization and loop closure using keyframes, PTAM globally optimizes the camera pose and features[57]. Figure 2.8 showcases the point cloud generated using the PTAM algorithm with the provided images.

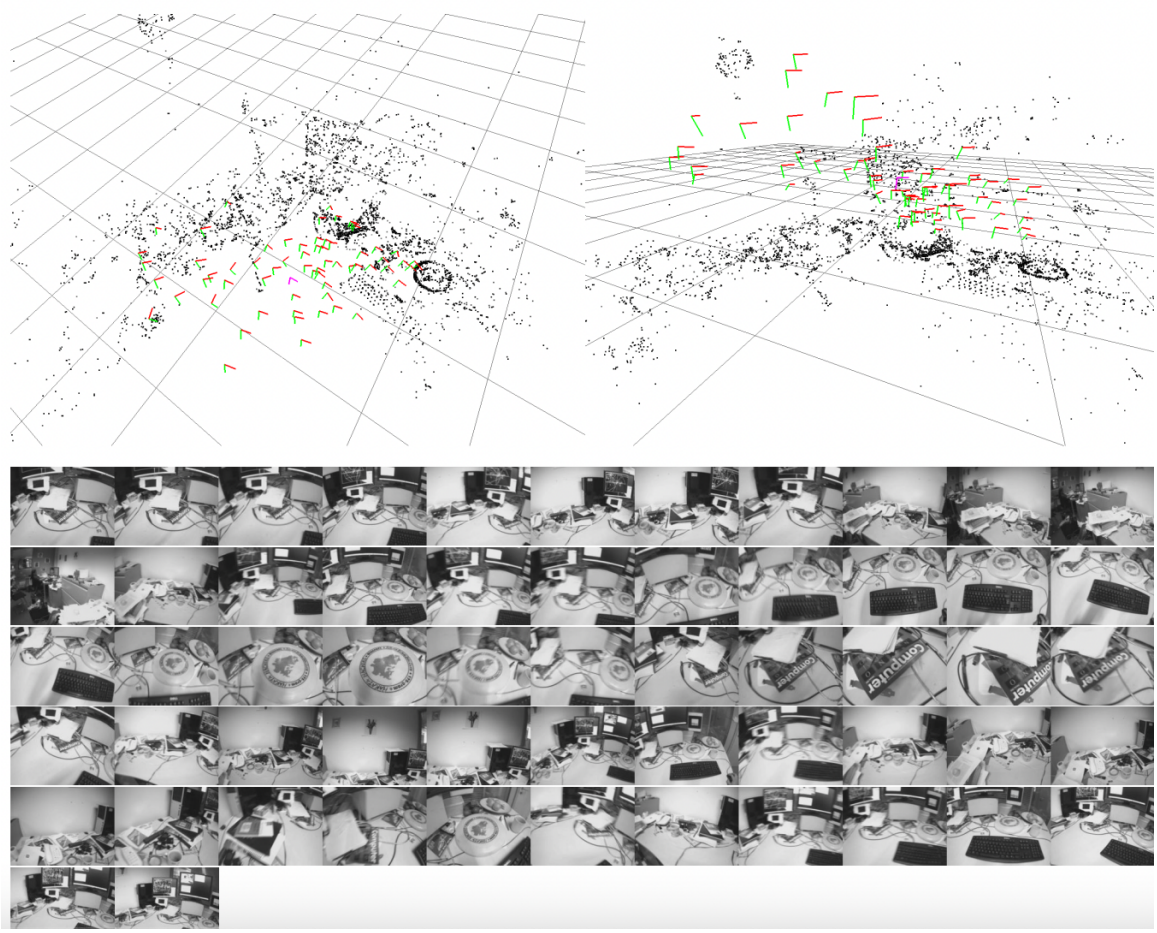

Figure 2.8: Example map and keyframes generated from the desktop video by PTAM algorithm.[57].

ORB-SLAM [54] exhibits substantial advancements in the domain of localization and mapping, offering notable improvements in scale and speed. Inspired by the multi-threaded approach of PTAM, ORB-SLAM efficiently allocates separate threads for distinct tasks such as loop closure, tracking, and local mapping, thereby optimizing performance for larger-scale environments. Diverging from PTAM, ORB-SLAM adopts the ORB feature descriptors [58]. Notably, to mitigate inaccuracies during ini-

tialization, ORB-SLAM leverages pose graph optimization techniques to refine camera pose estimation. Additionally, the loop closure algorithm is facilitated through the bag-of-words algorithm [59]. The integration of a covisibility graph based on the local area enhances ORB-SLAM's scalability and overall efficacy. Figure 2.9 illustrates the architecture and components of the ORB-SLAM system.

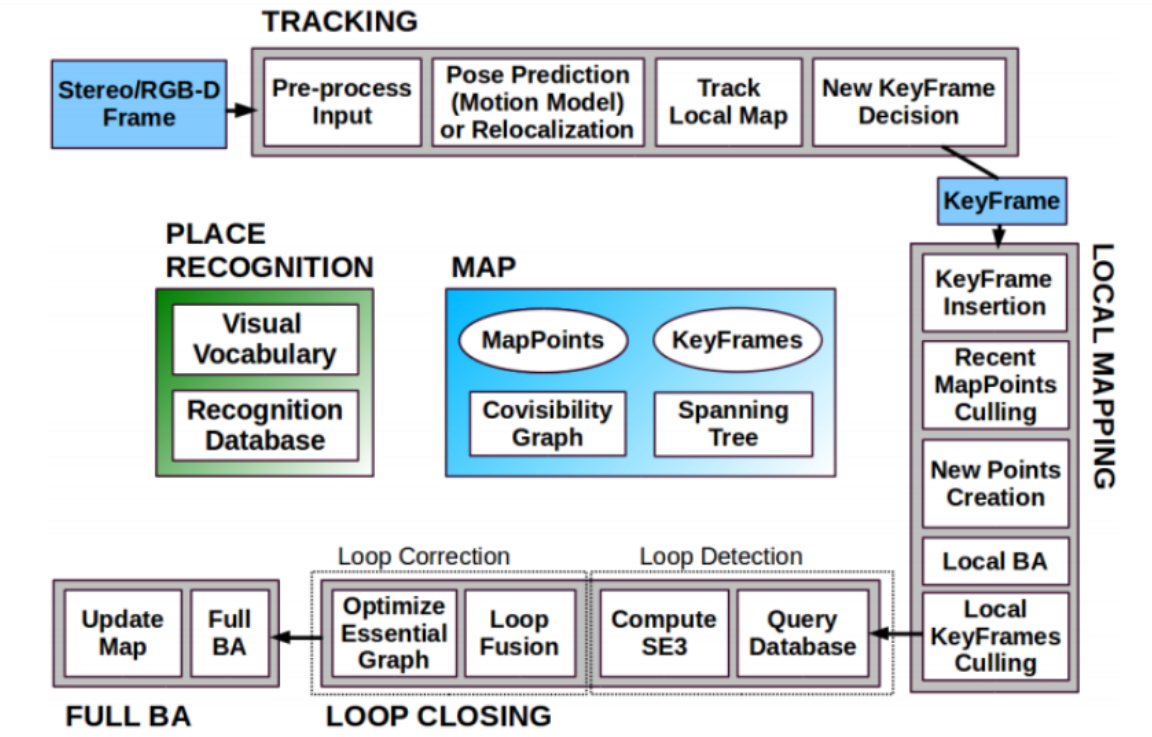

Figure 2.9: The Pipeline of ORB-SLAM System[54].

2.1.2.3 Semi-Dense Monocular Slam

Semi-dense monocular SLAM techniques estimate depth in high-gradient regions to acquire additional points within monocular SLAM keyframes. Unlike depth-based methods, semi-dense approaches operate in real-time on a single CPU without requiring extra hardware and involve a reduced number of points. Both LSD-SLAM and ORB-SLAM can be utilized to extract semi-dense points. In LSD-SLAM, high-gradient regions are used to generate a probability distribution based on estimated depth values for each pixel. During execution, the probability distribution is updated

using multiple images, leading to the elimination of pixels with lower probability and the acquisition of more accurate semi-dense points. Similarly, ORB-SLAM adopts a comparable strategy by obtaining inverse depth values from high gradient regions along epipolar lines in nearby keyframes. Multiple depth estimates are projected onto a probability distribution, which is refined across multiple images to remove low-probability pixel values and enhance the accuracy of semi-dense points. Compared to LSD-SLAM, ORB-SLAM achieves superior accuracy in semi-dense point estimation by leveraging points along epipolar lines in nearby keyframes, instead of relying solely on high-gradient regions. Additionally, the estimation of points is guided by examining the estimated depth values of neighboring pixels. Figure 2.10 showcases the output of LSD-SLAM and ORB-SLAM, highlighting the difference in semi-dense accuracy between the two methods.

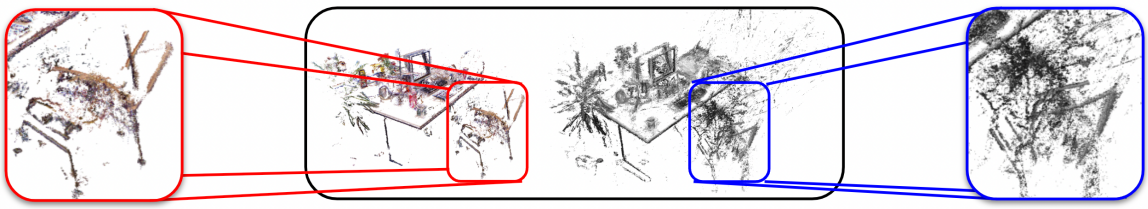

Figure 2.10: Compared to LSD-SLAM, ORB-SLAM semi-dense module is more accurate due to getting points along epipolar lines.

2.1.2.4 3D Line Detection

The point cloud obtained from SLAM algorithms does not contain high-level structural information important for surface reconstruction. One way to mitigate this issue is to extract high-level geometrical cues from the scene such as 3D lines, which can be obtained using either line matching or 3D points aided by 2D cues. Line matching algorithms use multi-view stereo matching over every frame [60] with lines detected by LBD algorithm [61]. However, such methods often fail nonplanar surfaces due to depth inconsistencies. One way to improve the performance of line detection and matching is by using epipolar lines in neighboring frames[62]. Another approach to

detecting 3D line point clouds is through intersecting planes. The normal vector of the intersecting planes is used to optimize the direction of lines. However, the number of detected lines is not adequate for the surface reconstruction due to constraints based on planar regions [63].

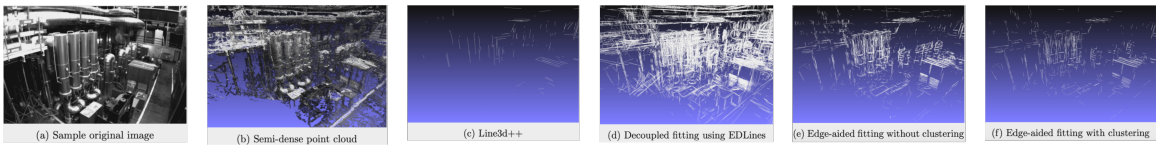

Figure 2.11: Qualitative Analysis of He et al. [3].

He et al. [3] follow a similar path to extract 3D lines from SLAM data. They use semi-dense SLAM points to determine 3D positions from 2D points forming lines in images. By selecting pixel chains with depth values, they fit two 2D lines—one in the image plane and another in the projective plane. Outliers are identified using distance measurements, and excess lines in edges are removed through clustering based on angle and distance thresholds between neighboring lines. However, the lines they obtain might be inaccurate because they could encompass points in different planes. Depending solely on clustering is not enough, as cluster centers do not consistently align with object edges. To tackle these challenges, Ahmad et al. [1] introduce a coplanarity threshold during line fitting to correct inaccuracies. Additionally, the traditional clustering algorithm is substituted with a line-based plane tracking system, focusing solely on lines observed in more than two keyframes. Figure 2.11 shows an example scene with detected 3D lines using the method proposed by He et al.

2.1.2.5 3D Plane Detection

In the context of surface reconstruction, plane detection plays an important role in scene understanding and reconstruction accuracy. Although the points and lines contain useful geometric primitives, they fail to address an accurate representation of the flat surfaces, which are the main structures of man-made structures. Most surface reconstruction algorithms often fail the textured flat surfaces. Using coplanarity

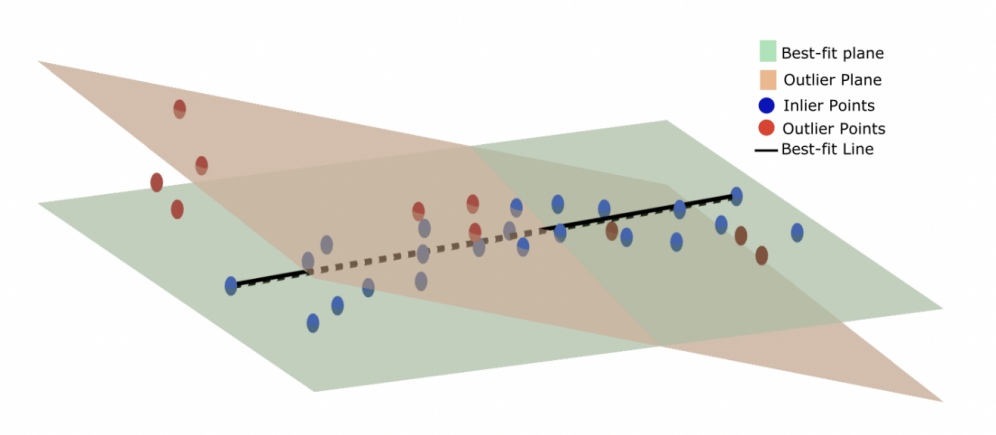

Figure 2.12: Outlier point removal process using best-fit plane point to fit 3D line segment of Ahmad et al.[1].

information, the surface reconstruction algorithms yield a more accurate and simpler representation in terms of mesh complexity. However, the plane detection algorithms that use methods such as random sampling and segmentation require intensive computing power due to the association between points and lines lying in the same plane. The previously mentioned incremental models have been used to reduce computational requirements using clustering superpixel segmentation and planes and lines obtained by semi-dense points.

Plane detection with segmentation is based on the idea of detecting planar areas based on a clustering algorithm. The detected planar areas in the frames are extended using plane normal information [64–67]. Using randomly selected points, the plane detection could be simplified to detect planes based on creating the number of plane hypotheses and checking inliner points to remove outlier planes and extend previously founded planar surfaces [68, 69].

To reduce computational complexity and make plane detection feasible for online updates, intersecting lines with predetermined angles and distances are used for plane detection. However, the 3D lines detected do not guarantee real structures and this leads to inaccuracies in plane detection. Additionally, the method utilizes stereo

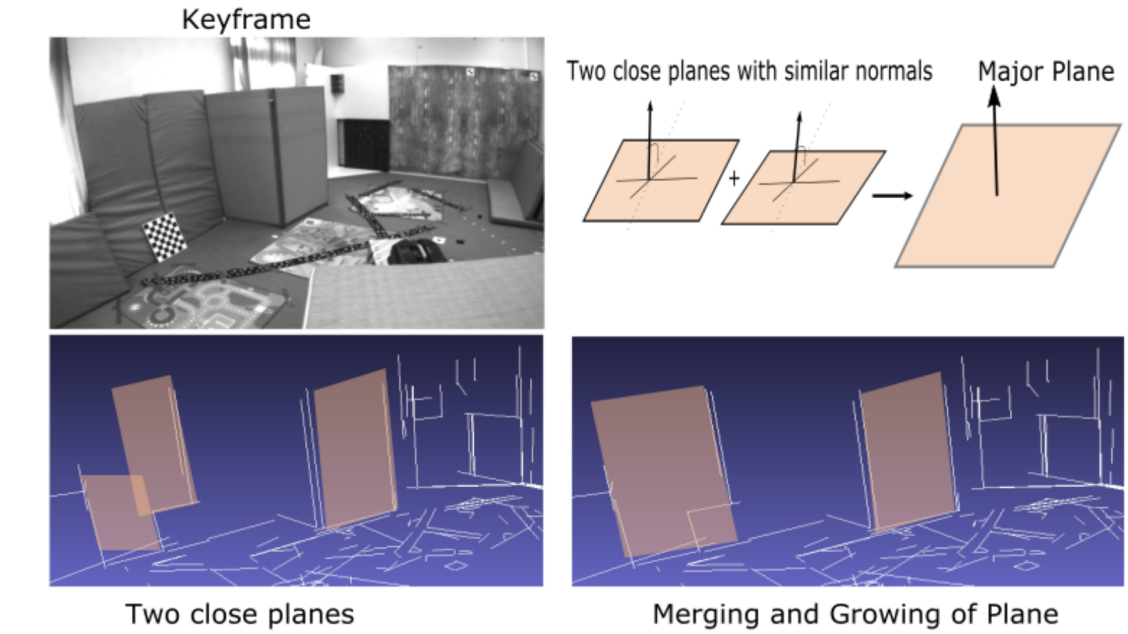

Figure 2.13: Plane growing strategy by Ahmad et al.[1].

detection is not suitable for an online setting.

Ahmad et al [1] extend the plane detection using line matching to a monocular setup using a semi-dense point cloud. The plane hypothesis is created based on the assumption that the nonparallel lines with larger than predetermined angles and minimum length belong to three maximum planes and must be unskewed. The initial plane hypothesis was pruned by the consensus of lines with similar plane normals and distances. If a plane has a sufficient number of associated lines, it's considered valid for interkeyframe matching. After plane matching, the process involves merging and expanding planes into major planar structures as new areas or parts of existing planar regions are discovered in new keyframes. However, there are constraints on the length of planar structures to prevent seamless merging of planes into larger planar regions. Figure 2.13 illustrates the plane merging strategy proposed by Ahmad et al.

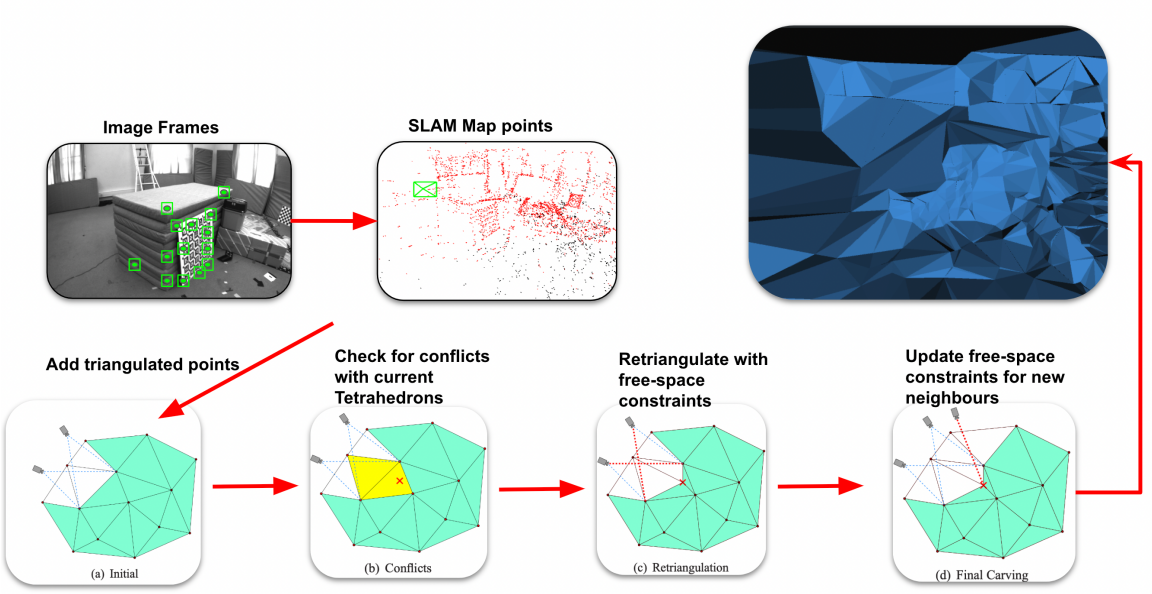

Figure 2.14: The process of obtaining surface mesh from sparse SLAM features with CARV algorithm [2].

2.1.2.6 Surface Reconstruction

Real-time surface mesh reconstruction algorithms play a crucial role in robotic applications due to their advantages in memory efficiency, ease of rendering, and the ability to create interactive surfaces. However, achieving real-time surface reconstruction from point clouds remains a challenging task for both classical and deep learning-based approaches [70]. Nevertheless, in the context of teleoperation tasks, it is not necessary to have highly realistic or finely detailed surface reconstruction for visualization, the additional texture details can complicate the determination of the relevant data [71]. Real-time incremental surface mesh reconstruction methods offer a viable solution, which can be achieved through explicit processing directly from point clouds or implicitly by utilizing intermediate representations. While Figure 2.14 illustrates one explicit example of this process, Figure 2.15 provides an example of implicit surface mesh reconstruction.

Explicit methods for real-time surface reconstruction leverage sparse point clouds obtained through techniques such as SLAM (Simultaneous Localization and Mapping)

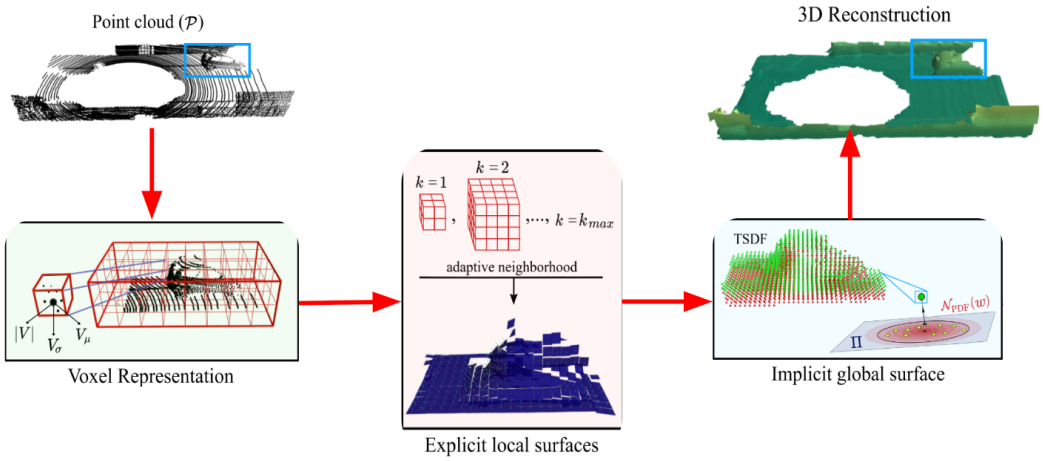

Figure 2.15: The process of obtaining surface mesh implicitly using volumetric representation.

or SFM (Structure from Motion) [2, 72]. These methods employ Delaunay triangulation to represent the continuous space as tetrahedrons, with the point clouds serving as the vertices. Incrementally, the tetrahedrons are labeled as empty or filled based on adjustments to point poses across keyframes. The resulting filled tetrahedrons are used to construct surface meshes. Different strategies, including the forgetting heuristic, [2] and graph-cut on the dual graph [47], are employed to optimize surface mesh quality while satisfying real-time constraints. To enhance the accuracy of the surface meshes, semi-dense point clouds enriched with geometric information from lines [3] or a combination of lines and points [1] are utilized, depending on the specific algorithm employed.

Implicit methods use volumetric representation to create a TSDF. The TSDF is utilized to extract surface meshes using the marching cubes algorithm. Although implicit surface reconstruction methods operate in real-time, they are constrained by GPU memory and rely on depth cameras, which suffer from inherent noise and limited resolution and range, leading to compromised reconstruction quality. Another approach for constructing volumetric intermediate representations involves employing neural networks to model small-scale scenes and objects [74]. While these models

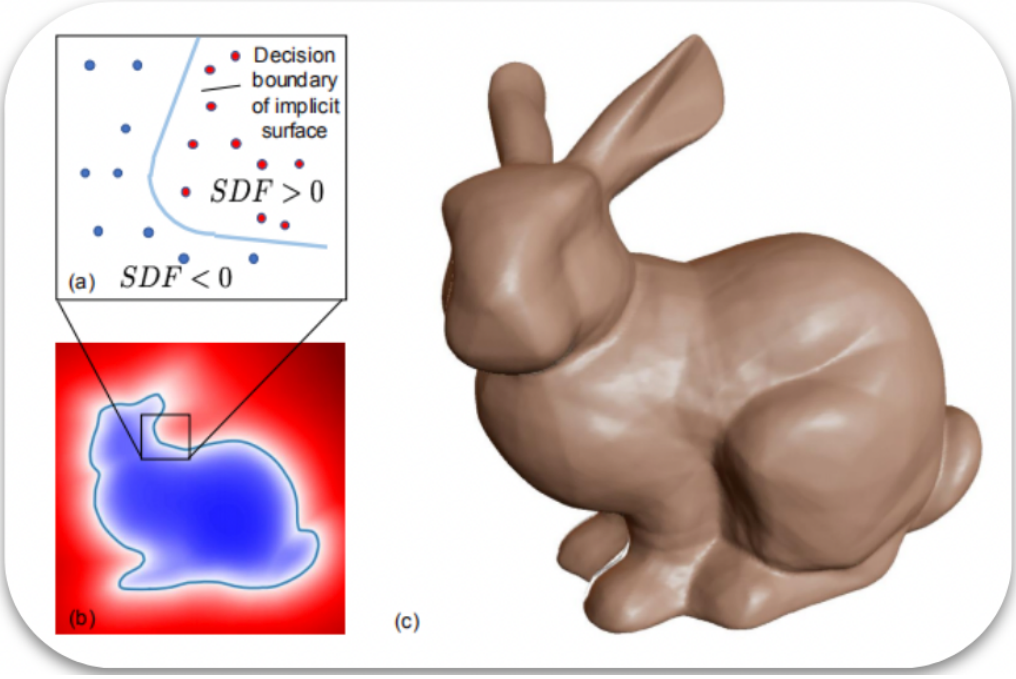

Figure 2.16: An example output generated by DeepSDF [73].

excel in rendering and reconstructing complex environments, their generalizability to diverse scenes and objects is limited, and their training and rendering times are unsuitable for real-time applications in surface mesh reconstruction [73], as seen in Figure 2.16, which provides an example object utilizing the DeepSDF approach.

2.2 Physics Simulator

Physics simulators have been used to test the various robotic tasks in different environment settings. Physics simulators enable cheaper, safer, and faster testing of the robotic task. However, choosing the right physics simulator for different tasks might be challenging for researchers. Although the physics simulator is widely used in the industry for testing purposes, the key challenges inhibit wide usage [75]. Mainly, the developers and researchers indicate that physics simulators have problems related to reality gap, complexity, lack of capabilities, reproducibility, and environment reconstruction [75]. Another inhibitor factor for using a physics simulator for robotics is

choosing the correct physics simulator for a particular task. Physics simulators' performance and reliabilities vary depending on the test scenarios [76]. Another critical factor in choosing physics simulators is support for various toolsets and the integrability of different programs. Gazebo is a well-known open-source physics simulator that supports multiple sensors, and SDF mesh models including robot types and objects to create and extend the testing environments.

2.2.1 Gazebo

Gazebo, an open-source physics simulator, supports four physics engines: Bullet [77], Simbody [78], DART [79], and Open Dynamics Engine (ODE) [80]. It utilizes the Object-Oriented Graphics Rendering Engine (OGRE) for rendering 3D environments. Gazebo consists of two main architectures: gzclient and gzserver. Gzserver is responsible for simulating physics, rendering, and sensors, while gzclient provides a graphical interface for user interaction. Communication between these architectures is facilitated through Google Protobuf serialization and boost::asio for message transportation. Figure 2.17 illustrates the system architecture details and the graph depicting dependencies between its components. Gazebo's functionality is encompassed by model, actor, world, and launch files. Models represent 3D static or dynamic rigid objects, while actors are models with animation capabilities. The world file defines the simulation environment, including entities such as models, actors, and simulation environment values [81]. Gazebo offers extensibility through plugins, which enable access to and extension of its functionalities and entity states. It can be integrated with both ROS1 and ROS2 using plugins.

When compared to other robot simulators such as CoppeliaSim [83], MORSE [84], and Webots [85], Gazebo demonstrates superior performance in terms of CPU resource consumption, achieving a higher real-time factor. The real-time factor represents the ratio between the sum of the simulated time step and the sum of the desired real-time step. Gazebo also exhibits greater accuracy in real-world mobile robot tests

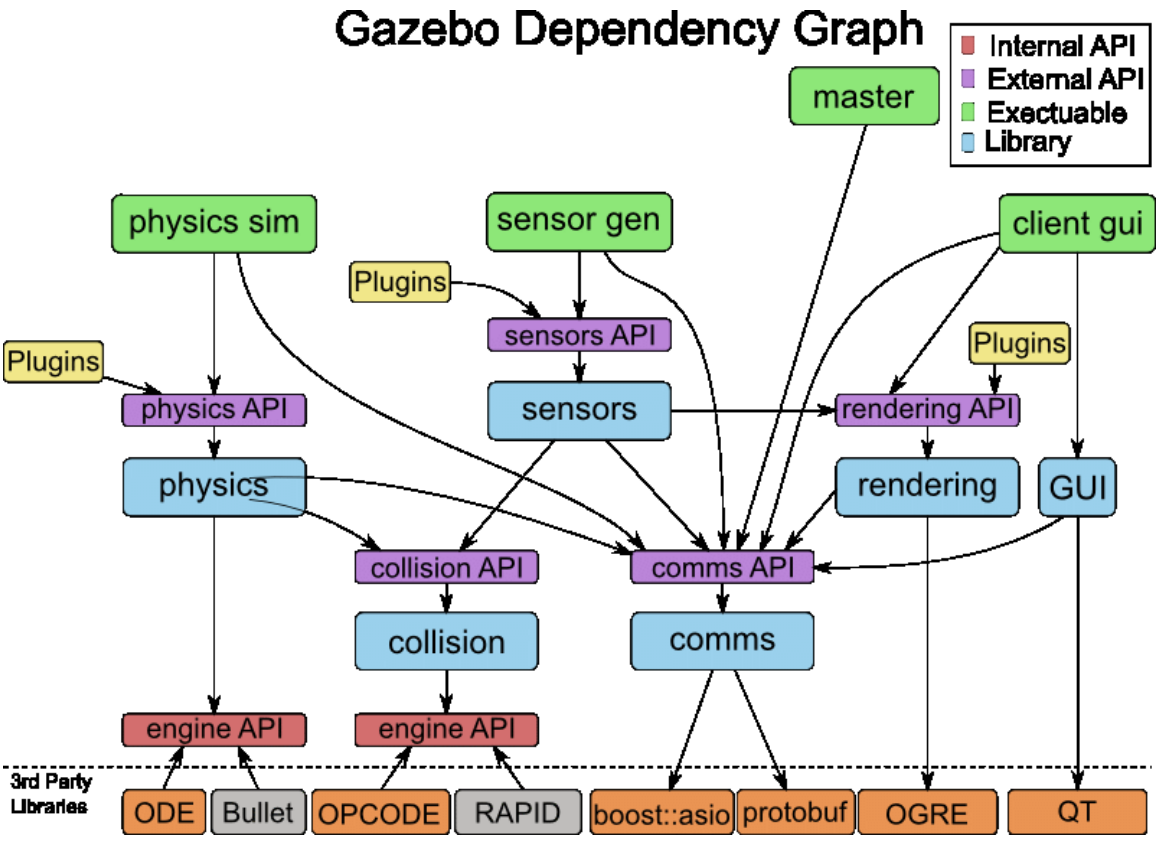

Figure 2.17: The Gazebo Architecture [82].

[86]. Additionally, Gazebo outperforms other simulators (V-REP/CoppeliaSim, AR-GoS [87]) in terms of simulator performance in large scenes with multiple robots and in headless mode, while offering comparable features to V-REP/CoppeliaSim [88].

2.2.2 Physics Engine

Although numerous physics engines are available for simulating complex manipulation tasks, the simulation results are often unstable and cannot be trusted. There is no physics engine for every scenario, making it difficult to select the appropriate engine for manipulation tasks requiring multi-contact interactions [76]. The inherent complexity and variability of real-world manipulation tasks, coupled with the approximations and limitations of existing engines, contribute to the challenge of achieving reliable and accurate simulation outcomes. Researchers and practitioners face the arduous task of carefully considering the task characteristics and desired accuracy levels

when selecting a physics engine, balancing computational efficiency with fidelity. Efforts to enhance physics engines and develop robust simulation frameworks continue, aiming to overcome these challenges and provide more trustworthy simulation results for complex manipulation tasks.

Contact dynamics pose significant challenges in computational physics due to their inherent computational complexity. The problem of calculating contact dynamics is classified as NP-Hard, primarily because of its non-convex and discontinuous nature [89]. As a result, physics engines often resort to approximated solutions using relaxed methods, which inevitably leads to a notable drop in accuracy. Even with approximate solutions for contact detection, the algorithms employed can adversely affect the real-time performance and stability of the physics engine. The compromises made in achieving real-time simulation and stability further exacerbate the difficulties in obtaining reliable and trustworthy results for contact-intensive scenarios [90, 91].

Table 1. Comparison of physics engines.

Physics Engine	Object	Collision Detection	Contact Clustering	Contact Solver	Integration	Coordinate
ODE	Primitive and mesh	Built-in	Ignores some contact points	LCP + Dantzig/PGS	Semi-implicit Euler	Maximal
Bullet	Primitive and mesh	Built-in	Maximizes contact area	LCP + Dantzig/PGS /NNCG/SI	Semi-implicit Euler	Generalized
Vortex	Primitive and mesh	Built-in	Details are not disclosed	LCP + direct/iterative method	Semi-implicit Euler	Generalized
MuJoCo	Primitive and convex mesh	libccd	Keeps only a few contact points in front	Convex optimization	Semi-implicit Euler/fourth-order Runge–Kutta	Generalized

Figure 2.18: Main comparison between ODE, Bullet, Vortex, and Mujoco physics engines [76].

The well-known physics engines in robotics employ different formulations to handle contact and multibody dynamics. ODE, for example, is utilized in simulation platforms like Gazebo and CoppeliaSim. It supports a wide range of shapes and meshes, allowing for realistic simulations of robotic systems. One of the key features of ODE is its built-in collision detection capability, which facilitates the accurate modeling of interactions between objects. Additionally, ODE is equipped with solvers such as the Dantzig solver [92] for solving the linear complementarity problem [93] to obtain an

exact solution, and the projected Gauss-Seidel [94] solver for iterative solutions using the maximal coordinate representation. Bullets, like ODE, possess similar capabilities in terms of contact dynamics. However, Bullet strives to enhance its contact area coverage, allowing for a more comprehensive modeling of contact interactions. With the generalized coordinate representation, Bullet is particularly advantageous for heavily constrained systems like robots with numerous joints. This alternative representation offers improved computational efficiency and accuracy. Lastly, Mujoco [95] shares similarities with the aforementioned methods in terms of supporting various shapes and meshes. However, it differs in its inability to handle nonconvex shapes. When it comes to contact dynamics, Mujoco adopts a different approach by converting the problem into a convex optimization problem. This conversion allows for the utilization of second-order methods, in addition to the methods mentioned earlier. Similar to Bullet, Mujoco employs a generalized coordinate system, complemented by a 4th-order Runge-Kutta integrator. In terms of accuracy in contact manipulation tasks, ODE and Bullet exhibit similar and satisfactory performance when dealing with simple-shaped objects. However, Bullet demonstrates superior performance in multi-contact scenarios. On the other hand, Mujoco performs poorly in these tasks but compensates with faster speed due to its different approach. [76, 91, 96]. Figure 2.18 compares these physics engines and their respective features.

2.2.3 Digital Twin

The application of digital twin technology traces its origins to NASA's utilization of space robotics during the 1960s. A noteworthy milestone in its development can be observed in the Apollo 13 space program, where digital twins played a pivotal role in simulating essential technologies by replicating physical infrastructure instead of generating real-time digital replicas [98]. In his 2002 presentation titled "Conceptual Ideal for PLM," Michael Grieves introduced the term "digital twin," which he defined as the integration of the physical and virtual environments, connected by a crucial

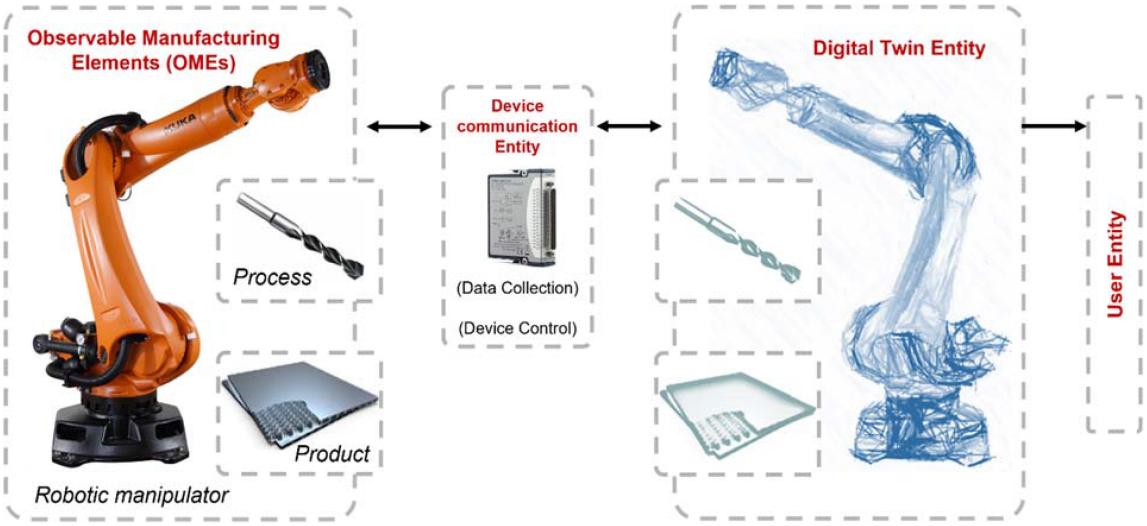

Figure 2.19: Digital twin helps to create a virtual replica of the robot's state and its environment [97].

data link [99]. In the field of robotics, digital twins provide the ability to persistently maintain a virtual replica of the physical robot system throughout its lifetime. This virtual model proves invaluable for troubleshooting and predictive inference purposes. Moreover, digital twins enable robots to provide valuable information about the condition of their hardware and forecast potential hardware issues. With the provided observability, digital twins enhance the process safety and design that contribute to more accurate simulations for future applications in robotics [100]. The application of digital twin technology in the field of robotics has been extensively researched across various domains, including space robotics, medical robotics, soft robotics, human-robot interaction, and industrial robotics [101].

Digital twins play a crucial role in teleoperation scenarios by providing real-time information about remote robots and their surrounding environment to operators through the device communication entity depicted in Figure 2.19. This transparency and visualization enable precise control over the robot's actions. For the application side, digital twins integrated with human-in-the-loop systems have been extensively studied in medical applications such as remote telemedical service robots [102], hazardous and contagious environments [103], surgical procedures [104, 105], and reha-

bilitation settings [106]. Additionally, in the industrial sector, digital twins are being utilized in hazardous environments to enhance safety and optimize operations [107].

2.3 Immersive Display

Head-mounted displays (HMDs) have been extensively researched and applied in diverse fields, including computer-aided surgery [108], aviation [109], and robotics [110]. In the context of robot teleoperation, visualization plays a crucial role in perceiving and manipulating remote robots. Initially, the limitations in depth cameras and network communication speed confined early teleoperation systems to 2D displays [111]. However, advancements in depth camera technology and efficient 3D reconstruction algorithms have facilitated the adoption of immersive displays for robot teleoperation [21, 112]. In Figure 2.20, Toyota's system demonstrates the control of a follower humanoid robot using a controller suit and a head-mounted display, enabling a fully immersive teleoperation experience [113].

Figure 2.20: Immersive display helps the operator to control the view of the remote environment [113].

Extensive research has revealed that relying solely on a simplistic video feed from

a remote environment presents considerable challenges in establishing a meaningful relationship between the robot and its surroundings. This limitation leads to decreased spatial awareness and imposes a significant cognitive load on the operator [21, 114]. Conventional 2D displays, which lack depth information, are inadequate for accurately representing remote environments. However, accurate representation is a crucial aspect for various robot teleoperation tasks [115, 116]. Furthermore, the absence of retained information about previously explored areas during teleoperation forces operators to heavily rely on their mental models, further increasing their cognitive load [117].

By utilizing immersive head-mounted displays, operators can control their viewpoint and gain a better understanding of the environment. The level of environmental information available also impacts task performance and efficiency [118, 119]. Immersive displays that provide comprehensive information, including task-related details and the remote environment, lead to shorter task completion times compared to situations where only task-related information is provided [120, 121]. Empirical evidence demonstrates the impact of situational awareness, mental load, and task performance in multi-robot setups across various environmental settings, encompassing indoor and outdoor environments, as well as different types of aerial and ground robots with immersive displays [122]. The spatial awareness gained through 3D representation and immersive display has proven effective for robot teleoperation in the presence of network latency, compared to a simple video stream[123].

Depending on how the scene is represented and the task information involved, the immersive display can induce motion sickness. It has been observed that there is a direct correlation between latency and both task performance and motion sickness [125]. To address this problem, one possible solution is to enhance the degree of immersion and the careful design of the system [126]. By decoupling the view, the operator gains more freedom and can adjust their perspective by moving their head, which leads to improved success rates in teleoperation tasks compared to a fixed

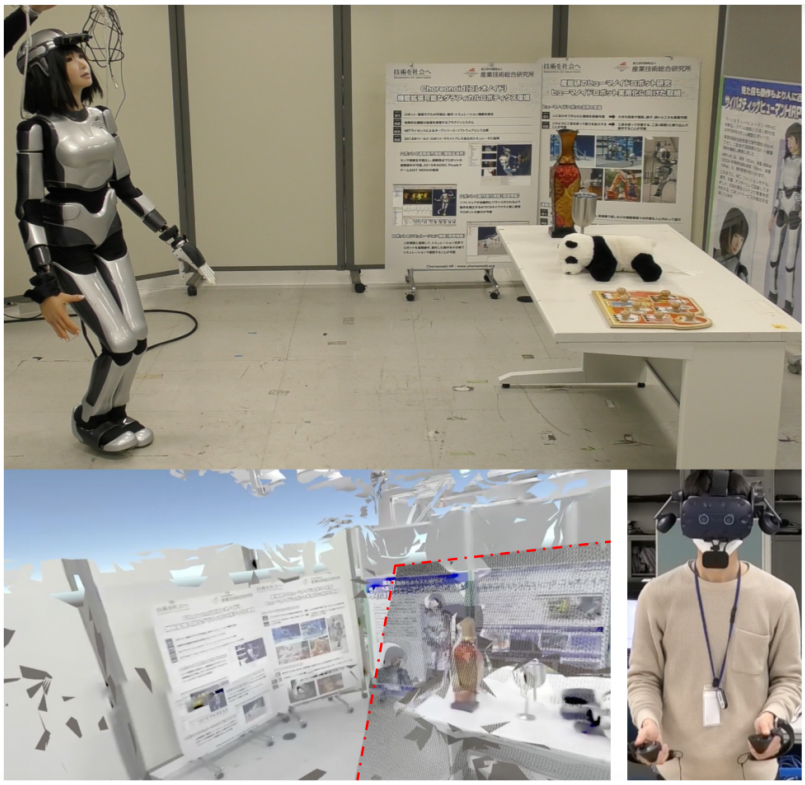

Figure 2.21: Immersive display can induce motion sickness, the decoupled view helps to reduce motion sickness caused by the latency [124].

camera view [127]. Furthermore, the decoupled view can be used to not only update the visual perspective based on the operator's head movements but also incorporate pose updates through additional controllers [51, 128]. Decoupling the view for robot teleoperation has also been found to have a positive effect on reducing motion sickness [50]. However, it is important to note that simply updating the video feed and adjusting the camera to decouple the view introduces additional latency and can exacerbate motion sickness for the operator [129, 130]. Therefore, achieving real-time 3D reconstruction of remote environments becomes crucial for ensuring successful immersive displays with decoupled views. In Figure 2.21, researchers have explored the use of incremental mesh representation based on point cloud data in various scenarios [50, 124].

Chapter 3

Method

In this section, our attention turns to the system and its parts. Our goal is to develop an immersive predictive visualization and haptic feedback setup for controlling robots remotely while considering practical considerations. To achieve a captivating predictive experience with minimal delay, our setup integrates the subsequent components to construct a simulated version of the remote environment and the follower robot in real-time.

The proposed system parts are:

- **Surface Mesh Reconstruction:** We utilize the semi-dense monocular CARV algorithm to continuously create a detailed surface mesh of the remote follower robot environment.
- **Physics Simulator:** The Gazebo physics simulator is employed to provide predictive visualization and haptic. It accomplishes this by simulating contact forces between the incremental surface mesh and the digital twin of the Barrett WAM robot arm.
- **Immersive Display:** This module employs a VR headset to control a pair of wide-angle stereo cameras within the Gazebo simulation environment. This setup immerses the operator in the virtual world, offering an immersive experience.

For a visual representation of all the system components and their dependencies, please refer to Figure 3.1.

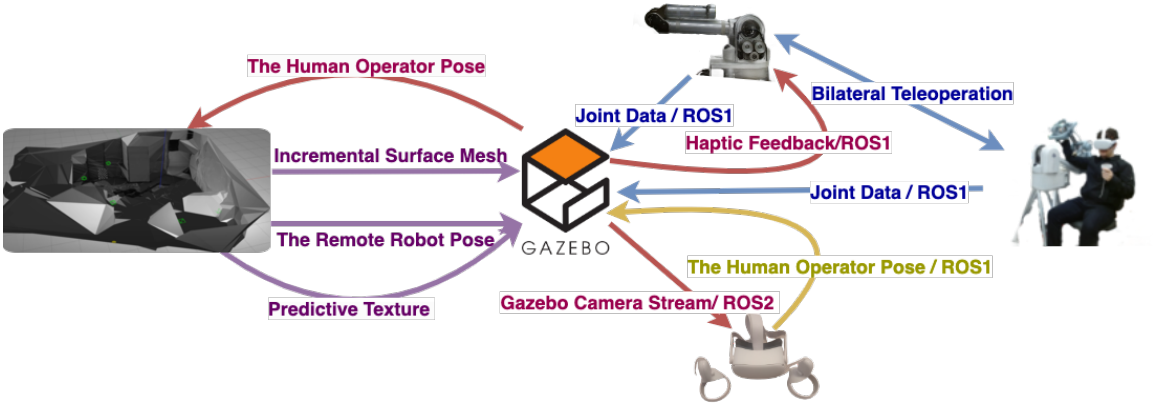

Figure 3.1: The proposed system architecture.

The visual feedback often requires a large network bandwidth for low-latency visual updates needed for remote robot manipulation. Traditional teleoperation setups often face challenges in replicating the intuitive control and perception of direct physical interaction due to this delay. To address this, our research aims to dissect the system's underlying mechanisms and synthesize an integrated framework that seamlessly combines human input and robotic execution. Additionally, we use advancements in predictive visualization techniques, aiming to provide operators with a more immersive and intuitive understanding of the remote environment. Through the exploration of our system's architecture and methodologies, we unlock new possibilities in the area of robot teleoperation, improving the way humans interact with and control robotic systems.

Furthermore, we delve into the area of immersive display, where we aim to enhance the immersion of the operator in the remote environment by improving the system's visualization capabilities. By enabling viewpoint-free rendering for the Gazebo camera sensors and utilizing directional light sensors, we provide operators with improved camera projection and scene illumination based on camera pose. These advancements contribute to a more immersive and detailed visual representation of the remote en-

vironment.

Finally, we incorporate a physics simulator within our system, leveraging the power of the Gazebo. This simulator serves multiple purposes, including collision detection for predictive haptic feedback, real-time interaction between the digital twin and the physical robot, and immersion through camera sensors. We discuss the challenges associated with maintaining real-time performance while ensuring accurate and reliable collision detection and haptic feedback. Through optimization techniques and the integration of plugins and services, we achieve a seamless and efficient interaction between the digital twin and the physical robot.

By addressing these key areas and their respective challenges, our system aims to improve the predictive visualization and haptic feedback of the field of robot teleoperation by addressing these key areas and their respective challenges, offering operators a more intuitive and immersive experience. Through combinations of 3D surface reconstruction, immersive display, and the utilization of a physics simulator in real-time, we push the boundaries of human-robot interaction and open up new possibilities for teleoperation applications.

3.1 3D Surface Renconstruction

We used the incremental surface mesh reconstruction algorithm not just to provide 3D information but to address the delay issue of robot teleoperation. The visual feedback often requires low latency and high bandwidth networks. Independent intermediate representation that is incrementally updated using remote scenes eliminates the delay problem using predictive immersive display. Additionally, we used the incremental surface mesh for creating the collision mesh in the physics simulator. Thus, the surface mesh reconstruction is not just important for providing predictive visual, but also predictive haptic feedback through interaction in the virtual replica.

We chose to use a surface mesh because it is easier to update, transmit, and save for use in real-time operations. Surface meshes serve the purpose of generating col-

lision meshes within the physics simulator, making surface mesh representation the most suitable solution for our system. This is due to the essential role it plays in the current collision detection requirements. When collecting point cloud data, two primary methods are available: using depth cameras to acquire dense point clouds or utilizing RGB cameras with SLAM or SFM methods to obtain sparse point cloud data. Depth cameras are designed to measure distances or depths in the scene, relying on emitted signals and returning reflections, which can result in limitations in their working range depending on signal strength and design. While RGB cameras rely on visible light and capture 2D color images without directly measuring depth. Additionally, depth cameras are more susceptible to noise arising from signal strength fluctuations and multi-path reflections, which are not typically encountered in RGB cameras, leading to potentially less noisy and more straightforward color images. A sparse point cloud is well-suited for online processing and does not require specialized hardware because it's more efficient, has reduced data size, lower computational demands, and is ideal for real-time applications like tracking and navigation. To facilitate the incremental surface mesh reconstruction of remote environments, we employ a semi-dense monocular CARV algorithm, which has real-time performance using a single CPU processor. Our system's capabilities were put to the test in an online manner with a real-world environment, as depicted in Figure 3.2. This algorithm uses the point cloud and camera tracking of ORB-SLAM2, a well-established method for obtaining accurate camera pose estimates and generating sparse point clouds. These sparse point clouds serve as the foundational data for reconstructing the surfaces of the remote environment.

The integration of the semi-dense module into the reconstruction pipeline significantly enhances the precision and completeness of the resulting surface mesh. By extracting line and plane information from the semi-dense module, the algorithm effectively refines the reconstructed surface mesh by identifying and eliminating outlier points. To eliminate outlier points, a RANSAC-based approach randomly selects a

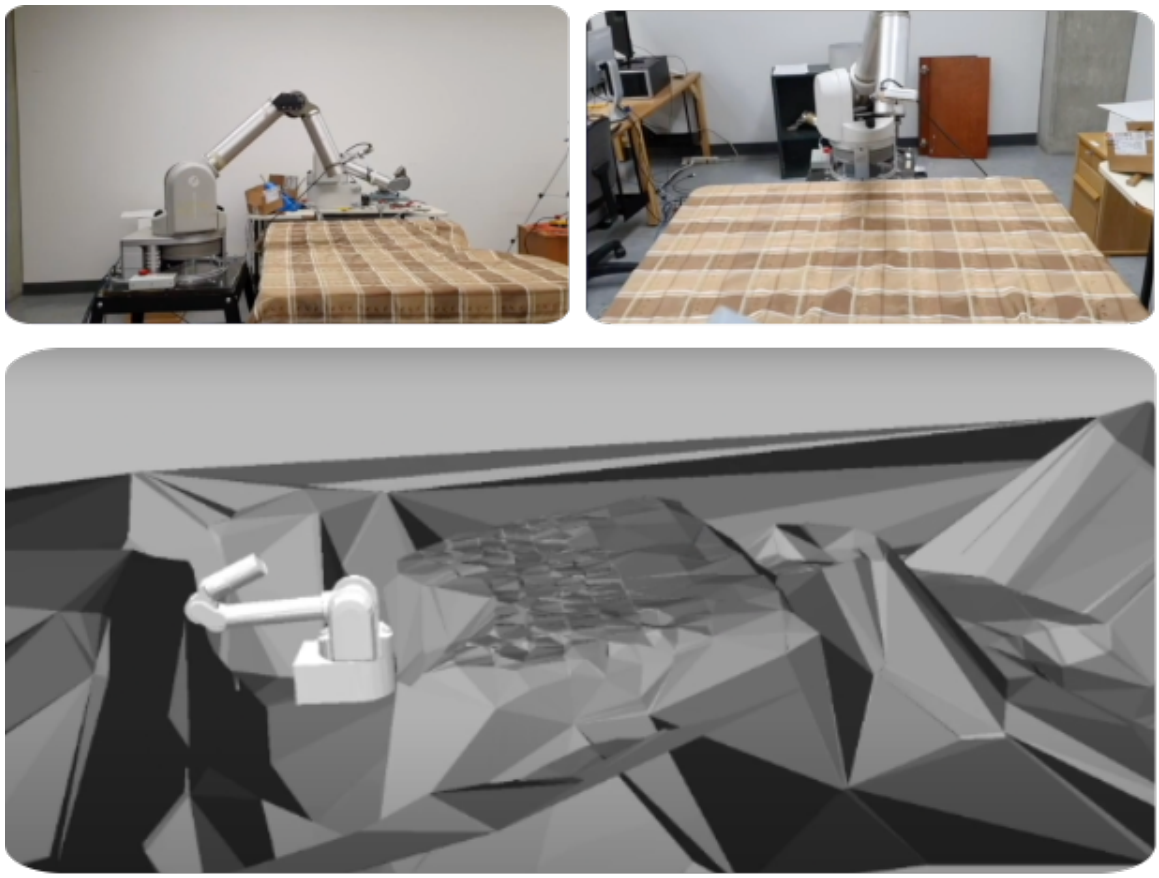

Figure 3.2: The surface mesh reconstruction the remote environment.

subset of pixels from the pixel chain. These selected pixels are checked to see if their perpendicular distance from the best-fit plane exceeds a predetermined threshold, and if so, they are eliminated from consideration as outliers. This process plays a critical role in improving the overall quality and accuracy of the reconstructed model.

The removal of outlier points is particularly beneficial in terms of optimizing the complexity of the surface meshes. By eliminating these outliers, the reconstruction process becomes more streamlined, resulting in a reduction in computational complexity. This reduction not only enhances the efficiency of the algorithm but also leads to a more visually pleasing representation of the remote scene.

The surface model is updated incrementally by the semi-dense CARV algorithm. This occurs when new keyframes are registered or when existing keyframes are adjusted through bundle adjustment. Such updates ensure that the surface mesh model

remains up to date with a dynamically changing remote environment. Following each update, the resulting surface mesh model is written in an OBJ file format which facilitates easy sharing and compatibility with different programs and platforms.

The OBJ files, which contain the reconstructed surface mesh, can be seamlessly integrated with the Gazebo physics simulator to create a virtual representation of the collision mesh model. This integration enables improved control, interactions, and immersion within the reconstructed environment. Additionally, in order to shorten the process of updating the textured mesh, texture keyframes and their corresponding coordinates, which represent the 3D rendered vertices, are saved in image files and smaller-sized OBJ files. This approach streamlines the updating process by efficiently storing the essential data in both image and OBJ formats, making it easier to manipulate and modify the texture and geometry of the 3D model without the need for extensive data processing during updates. By storing these key data components, the system ensures efficient and fast updates to the textured mesh representation.

The design framework aims to prioritize the timely update of the textured mesh by selecting the appropriate texture model based on the specified camera pose. This camera pose can be derived from the latest pose generated by the SLAM algorithm or from a predictive texture generated using the Gazebo camera pose, camera pose, which accounts for the operator's head pose. By dynamically adapting to the camera pose, the current texture model accurately aligns with the remote scene, enhancing the visual realism and fidelity of the reconstructed environment. Figure 3.3 illustrates how the camera sensor supplies the textured mesh with varying poses, showcasing this dynamic alignment in action.

To enable real-time tracking of the remote robot's pose within the remote scene, the current camera pose is published as a ROS1 (Robot Operating System) topic. This ROS topic ensures the continuous dissemination of the camera pose information, enabling external systems or applications to track and synchronize their operations with the follower robot. By providing an accurate and up-to-date representation of

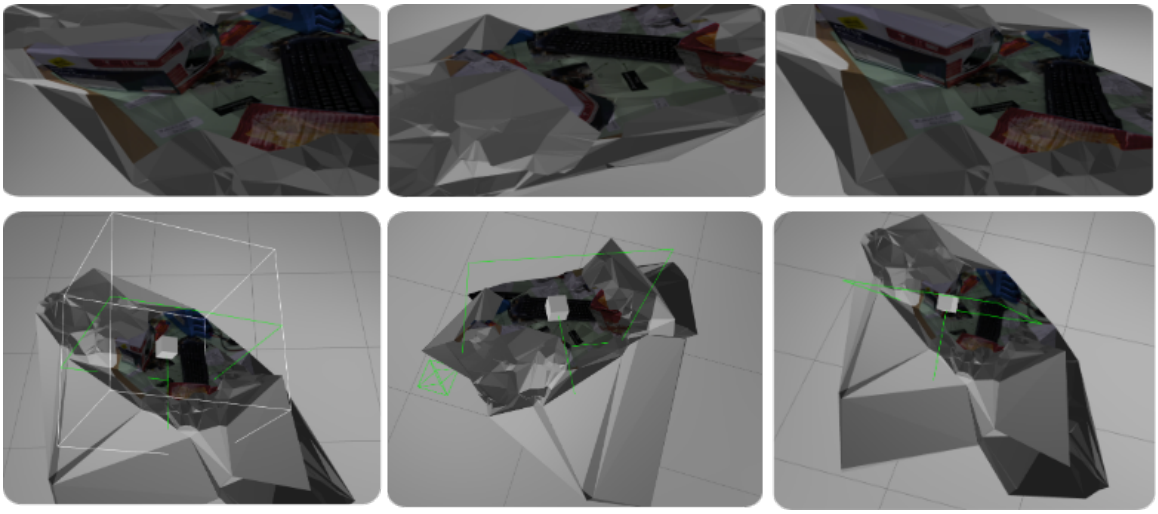

Figure 3.3: Texture mesh and surface mesh with different camera poses.

the robot's position and orientation, the published camera pose assists in coordinating actions and interactions with the follower robot within the reconstructed scene.

3.2 Physics Simulator

The usage of the physics simulator in our proposed system is multifaceted. We use a physics simulator not only to integrate the incremental surface mesh and the digital twin of the WAM arm robot for solving delay issues but also to improve the operator's agility and understanding of the remote scene and the follower robot state. While the interaction of the modalities provides the predictive haptic feedback, the utilization of the sensors provided by the physics simulator provides immersive predictive visuals including interaction and the follower robot state. The intermediate virtual representation gives full control of the remote environment and the follower robot by not requiring direct feedback from the physical follower robot.

For the integration of the incremental surface mesh of the remote environment and the digital twin of the WAM Robot, we selected the Gazebo physics simulator as our platform of choice. Despite the impending end of support for Gazebo Classic in 2025, it remains fully compatible with both ROS1 and ROS2. Gazebo Classic offers extensive sensor support and a wide range of ROS packages, making it seamless to

integrate different systems. To enhance Gazebo's functionality and compatibility with various models, we implemented custom Gazebo plugins.

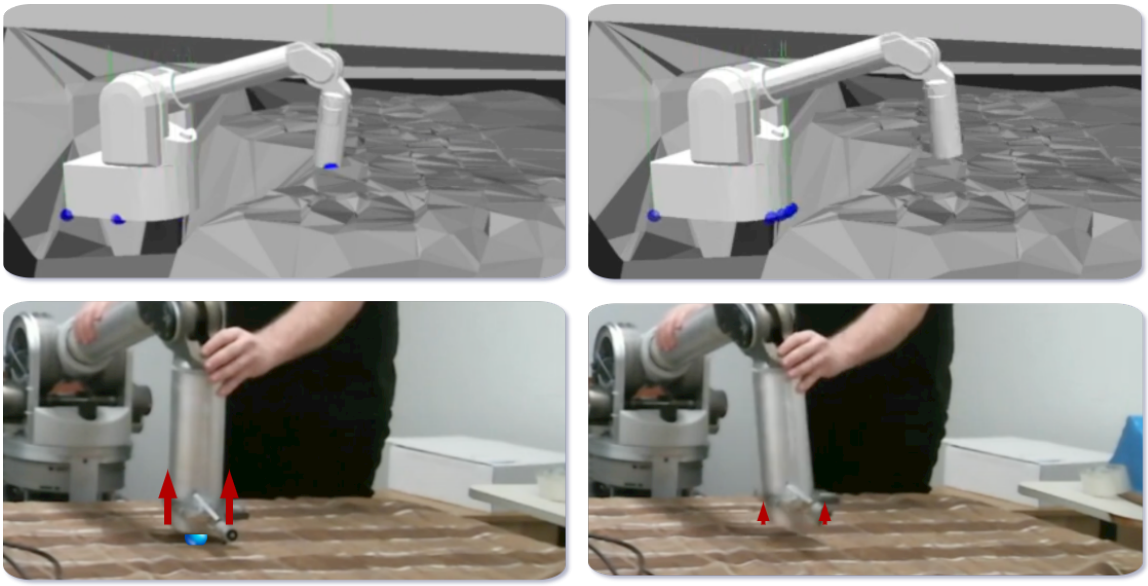

Figure 3.4: **Top :** The contact between the surface mesh and the WAM digital twin robot was calculated for haptic feedback before 20 mm of the physical contact using *min_depth* parameter. **Bottom :** Haptic feedback to WAM arm robot arm by surface collision mesh from physics simulator (**Blue Sphere**) visualizing contact between the table and the robot arm end effector, (**Red Arrow**) representing the contact forces calculated using surface mesh face normals.

To enable the interaction between the digital twin and the incremental surface model, we implemented a Gazebo model plugin specifically for the digital twin and the follower robot synchronization. This plugin allows us to obtain joint positions, velocities, and efforts which are crucial for moving the digital twin within the simulation environment. Communication between the remote WAM arm robot and the digital twin occurs through ROS1 topics published by the *libbarrett* library, which are subscribed to by the Gazebo model plugin.

To handle the physics simulation, Gazebo utilizes the ODE physics engine as its default choice. ODE provides accurate physics calculations for simulating the interaction between the digital twin and the environment within the Gazebo platform.

To ensure accurate interaction between the digital twin and the calibrated surface

collision mesh, we used contact detection provided by the ODE physics engine. By capturing the contact normals and forces, the physics engine was able to calculate force/torque magnitudes and friction values for the interaction. This information enables us to simulate realistic and responsive interactions between the digital twin and the surface mesh. In Figure 3.4, our predictive haptic feedback system undergoes testing in a tabletop setting, showcasing the haptic force generation by the physics engine as an integral part of the interaction process.

3.3 Immersive Display

To facilitate the transmission of the video stream rendered by the Gazebo stereo wide-angle camera sensor, we employed an asynchronous ROS2 client to create a dedicated topic for the video stream. By utilizing the open-source Python library called *aiortc* [131], real-time video streaming with WebRTC (Web Real-Time Communication) was established. The library enables seamless integration with ROS2, allowing us to subscribe to the video stream topic and transmit it in real-time.

3.3.1 Immersive Display Through Web

We use the immersive display to provide predictive visuals based on the virtual replica of the follower robot environments. We control the camera pose and view using the operator's head motions and the hand controller, which improve spatial awareness and reduce delay-related cognitive load.

For compatibility and easy integration with virtual reality (VR) headsets, we adopted the A-frame framework. A-frame utilizes the capabilities of Three.js, a popular library for rendering virtual and augmented reality applications within web browsers. This approach ensures that our system remains independent of specific VR headsets and simplifies the implementation process, making it accessible for simple rendering tasks.

3.3.2 Operater Motion Integration

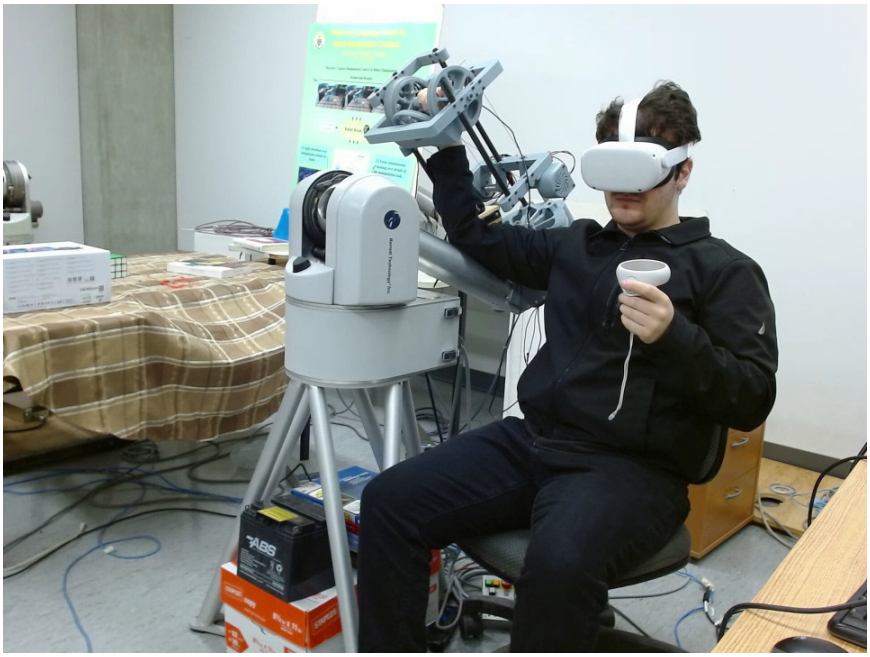

Figure 3.5: The configuration for the operator with the leader robot arm and head-mounted display.

To enable the operator to control the Gazebo camera, the A-frame framework captures the head pose of the operator. This head pose data is then used to manipulate the Gazebo camera using the Gazebo ROS1 plugin. By transmitting the pose data through the data channel alongside the video channel created with WebRTC, the operator's head motion directly controls the Gazebo camera's position and orientation. This integration allows for an immersive and interactive experience, where the operator's motions influence the rendering of the virtual scene.

To provide convenient control of the camera pose without physically moving the operator, we integrated the joysticks of hand controllers. By combining the operator's pose data with the joystick inputs, the Gazebo camera can be adjusted to update the rendering view accordingly. This functionality enables the operator to manipulate the camera pose effortlessly and explore different perspectives within the virtual scene. This functionality grants the operator the flexibility to manipulate the camera pose

while simultaneously controlling the leader robot, all through the use of a head-mounted display and controller, as demonstrated in Figure 3.5.

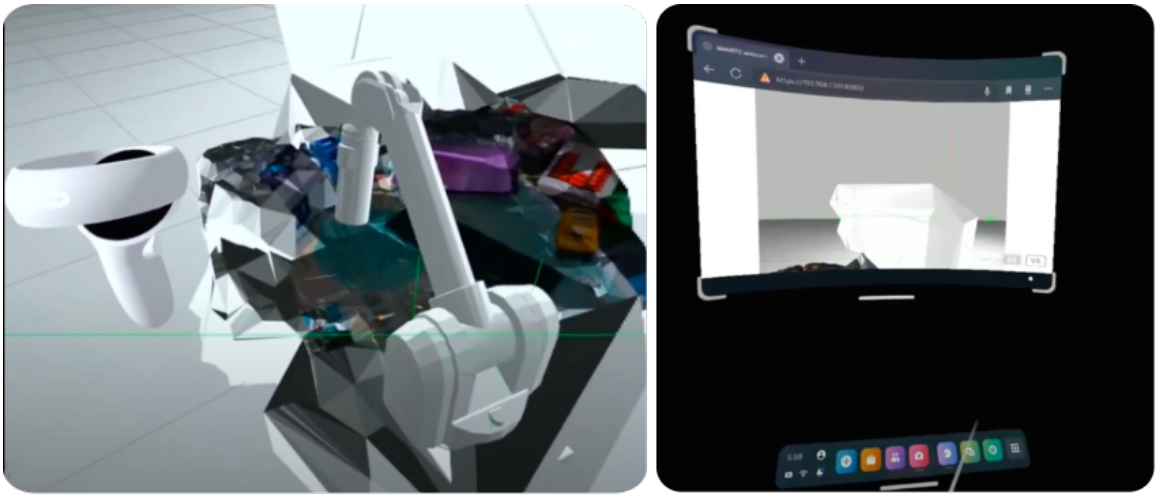

Figure 3.6: Predictive texture, surface mesh, and digital twin of the WAM robot rendered to HMD immersive display streamed by ROS2 from Gazebo wide-angle camera.

This setup offers a dynamic and interactive VR experience, allowing the operator to navigate and interact with the virtual environment using the added feeling of the digital surfaces through haptic interaction with the 3D model.

By combining ROS2, WebRTC, A-frame, and Gazebo plugins, our system enables real-time video streaming, immersive VR rendering, and interactive camera control. This integration opens up possibilities for various applications, such as teleoperation, virtual training, and remote visualization, with the flexibility to adapt to different VR setups and scenarios.

Chapter 4

Experiments

In this chapter, we evaluated the surface reconstruction algorithm. We will discuss the accuracy, precision, and the number of mesh faces resulting from the line and plane based monocular surface reconstruction, using benchmarks from the EuRoC MAV Datasets [132], including Vicon Room 101 (VR101). Additionally, we assessed the predictive haptic feedback performance based on collision detection in the physics engine and optimization results. Furthermore, we examined the synchronization between the digital twin and the follower robot, as well as the integration of the incremental surface mesh model. Lastly, we delved into the communication between the Gazebo physics simulator and the immersive display and hand controller.

4.1 3D Surface Reconstruction

During our experimental evaluation, we utilized an Intel RealSense D435 global shutter camera to capture a monocular RGB video stream with a resolution of 1280x720. Running on an Intel i9-12900K processor, the camera achieved a frame rate of 15.46 FPS, enabling real-time surface mesh reconstruction. The camera is seamlessly integrated into ORB-SLAM2 to acquire point cloud data and localization information. The produced sparse point cloud is then combined with the CARV algorithm to incrementally update the surface mesh. To enhance the accuracy of the reconstructed meshes, we leveraged the geometric cues obtained from the semi-dense module to

extract line and plane information, which facilitated the elimination of outlier mesh faces. In order to make the system compatible with the online setup, we used 50 keyframes for semi-dense points rather than using all keyframes. You can observe the resulting surface meshes depicted in the bottom section of Figure 4.1.

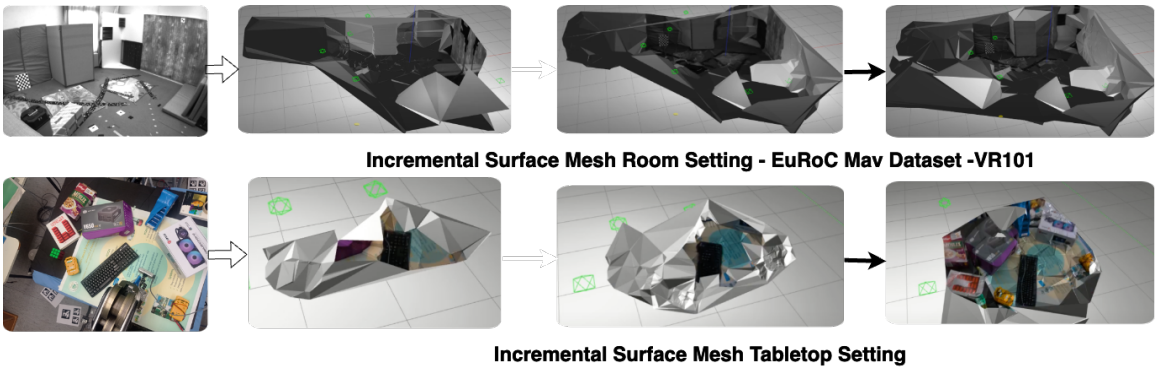

Figure 4.1: Incremental surface mesh reconstruction for room and table-top setup.

To assess the performance of our approach, we compared it with two existing monocular surface mesh reconstruction methods: Lovi *et al.* [2] and He *et al.* [3]. These methods also employed sparse point clouds and Delaunay triangulation to create surface meshes. To conduct a fair comparison, we employed the EuRoC MAV Vicon Room 101 (VR101) benchmark [132].

The metrics evaluated in this comparison were completeness, where the estimated 3D points' distances to the ground truth point are within 25 mm, and precision, where the estimated mesh surface to the ground truth scan. Lastly, the number of mesh faces, which are important for collision detection and physics simulator performance, since collision detection algorithms are computationally expensive.

Method	Vertices	Faces	Precision	Completeness
Lovi *et al.*[2]	531142	148582	74.5%	70.98%
He *et al.*[3]	50666	32012	88.33%	77.21%
Ours(Lines + Planes)	**31986**	**22998**	**96.8%**	**88.62%**

Table 4.1: Comparison between the surface mesh of **VR101** room setting created by different CARV approaches on completeness and precision.

Table 4.1 presents the results of the evaluation, including the number of vertices, faces, and the precision and completeness values for each method. Our algorithm demonstrated a superior performance achieving higher precision and completeness compared to the other methods. Furthermore, our approach significantly reduced the number of vertices and mesh faces, leading to a more optimized representation of the scene.

By leveraging the geometric cues provided by the semi-dense module, we achieved improved precision and completeness metrics. Simultaneously, we reduced the complexity of the resulting surface meshes. This showcases the potential of our approach in generating accurate and efficient surface models of remote environments.

4.2 Physics Simulator

Gazebo, an open-source physics simulator, plays a crucial role in testing robotic applications across diverse scenarios. In our experiments, we utilized Gazebo to create a real-time replica of the remote environment by configuring a Gazebo world. To ensure real-time model updates, we incorporated Gazebo ROS plugins concurrently using both ROS Noetic (ROS1) and ROS Foxy (ROS2) versions.

4.2.1 Follower Robot Integration

Initially, the communication between the digital twin of the WAM arm robot and the follower WAM arm robot involved the Gazebo ROS plugin and *libbarrett*, running at a rate of 30 frames per second. However, the Gazebo requires 1 KHz for simulation updates, while the WAM arm robot needs 500 Hz for real-world motion. This discrepancy led to issues, causing oscillations due to latency constraints.

To address this problem, we developed a Gazebo model plugin to internally integrate *libbarrett* ROS topics and services with Gazebo. This plugin leveraged *libbarrett* to obtain joint data from the follower robot and deliver predictive haptic feedback. The communication between the digital twin and the follower WAM arm robot was

executed through ROS1. Joint data from the physical robot was obtained by subscribing to the *\wam\joint_states* ROS topic at a 500 Hz frequency. This data included joint position, velocity, and effort, which updated the digital twin of the WAM arm robot.

4.2.2 Predictive Haptic Feedback

Predictive haptic feedback was acquired from the Gazebo contact topic, while the follower WAM arm robot received feedback from the plugin via the ROS *\wam\joy_-force_torque_baseservice* service. Although it was possible to detect contact for every joint of the digital twin, we preferred feedback through the end effector due to libbarrett's implementation limitations.

However, Gazebo's default implementation lacks a joint effort calculation function for contact detection, prompting us to rely on force/torque sensors to obtain values applied to each link body. Given occasional inaccuracies in these values for the WAM arm robot, we chose to use constant force/torque values defined in the plugin for predictive haptic feedback. For enhanced safety, we enabled *collision_without_contact* in the Gazebo physics simulator configuration, allowing us to detect contact before collisions without obtaining contact force values.

Method	Memory(MB)	RTF w/o opt	RTF w opt
Lovi *et al.*[2]	18.7	0.11	0.29
He *et al.*[3]	2.1	0.225	0.471
Ours	**1.4**	**0.255**	**0.52**

Table 4.2: Comparison between the surface mesh of the from Euroc Vicon Room 101 benchmarks (**VR101**) [132] setting physics simulator real-time factor (**RTF**) with w/wo optimization, (RTF being 1.00 to be the optimal RTF.)

Collision detection in a physics simulator is computationally intensive. It slows down the real-time factor (RTF), especially with numerous mesh faces and joints in the digital twin. However, many optimizations were implemented to improve the

Gazebo world configuration: disabling wind, atmosphere, and shows from renderings; improving RTF by setting the real-time update rate to 0; and increasing max step size to fully utilize CPU resources. Results of these optimizations for different surface reconstruction algorithms are presented in Table 4.2.

4.2.3 Gazebo Wide Angle Camera Sensor

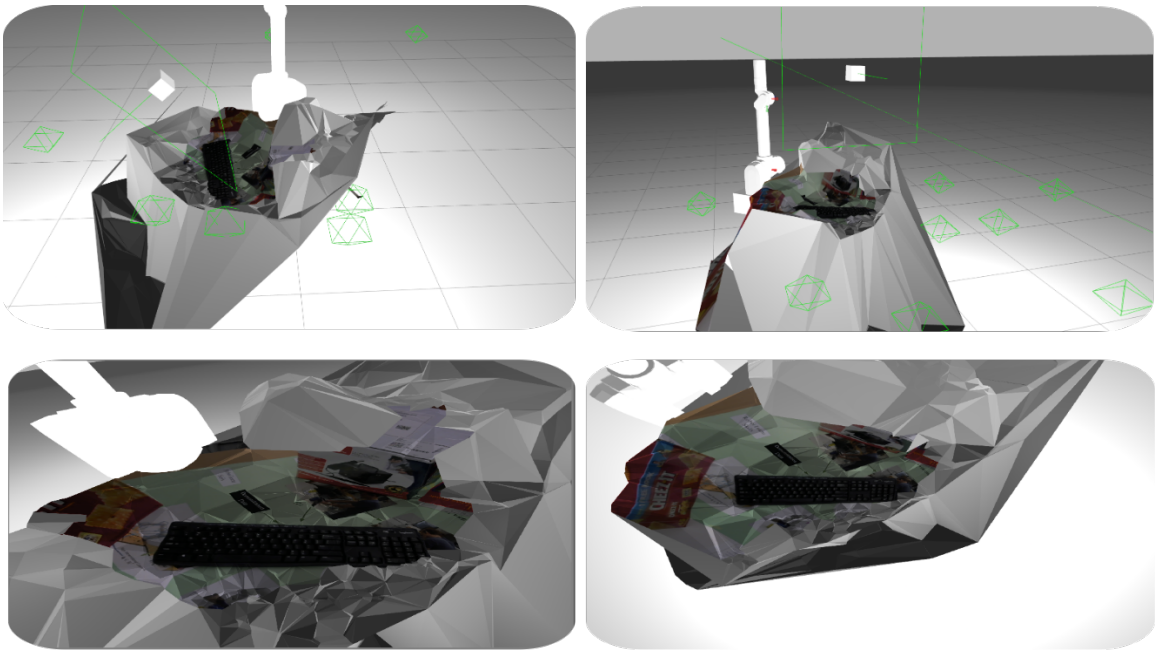

Figure 4.2: Predictive texture and surface mesh on immersive display streamed by ROS2 from Gazebo wide-angle camera(Cube).

By default, Gazebo's implementation disables projection rendering for the Gazebo camera sensor, limiting rendering to the graphical user interface. We modified the source code to enable rendering for Gazebo camera sensors, benefiting from Gazebo's open-source nature. This allows the digital twin to provide visualizations of contact, inertia, and efforts through the Gazebo camera sensor.

Our experiments revealed that the Gazebo camera sensor achieves up to 23 FPS with a 4K resolution using the Gazebo ROS2 camera plugin, while the Gazebo ROS1 camera plugin only manages 30 FPS with a 640x480 resolution. We configured the Gazebo camera sensor as a wide-angle camera with a 1.57-radian horizontal field of

view to capture sufficient information from the reconstructed scene.

Notably, we observed that camera projection improves with scene illumination. Consequently, we integrated a directional light sensor with the camera sensor to illuminate the scene based on the camera's pose. The camera sensor's pose was obtained from the human operator and updated through the Gazebo ROS1 plugin service at 30 FPS.

4.2.4 Surface Mesh Integration

The reconstructed surface mesh model in the Gazebo scene was continually updated using Gazebo ROS1 plugins. To optimize real-time model updates, we employed two different mesh models: the surface mesh of all scanned scenes and the textured mesh composed of a single frame. Disk size requirements vary, with room-scale surface mesh OBJ files needing 1.4 MB, while textured mesh OBJ files requiring approximately 300-600 KB, accompanied by a 300 KB JPEG image frame with a 1280x720 resolution. In tabletop settings, both surface mesh and textured mesh OBJ files occupied around 900 KB.

Model updates were facilitated using the *spawn_model* ROS1 service, completing the process in 60 ms for a 1.4 MB OBJ file. To remove models from the Gazebo simulation environment, the *delete_model* Gazebo ROS1 plugin service required 250 ms with *rospy*.

4.3 Immersive Display

The video stream from the Gazebo camera sensor via the ROS2 camera plugin was sent to the Oculus Quest 2 VR headset. We chose ROS2 over ROS1 for our system due to its significantly improved performance and capabilities, which are essential for providing a seamless and immersive VR experience. ROS2 offers several advantages over its predecessor, ROS1 (Noetic), making it the preferred choice for our project. This choice aligns with our goal to achieve a high-quality VR experience with the

ROS Versions \ Resolution	640x480	1280x720	1920x1080	2304x1536	3840x2160
ROS1 (Noetic)	29.996	10.019	5.031	1.720	-
ROS2 (Foxy)	173.134	123.409	77.130	50.220	23.440

Table 4.3: Comparison of media streaming frame rates (FPS) at different resolutions for ROS1 and ROS2.

Oculus Quest 2 headset, as demonstrated in Table 4.3, where ROS2 (Foxy) consistently outperforms ROS1 (Noetic) in terms of media streaming frame rates at various resolutions, ensuring smoother and more responsive video streaming.

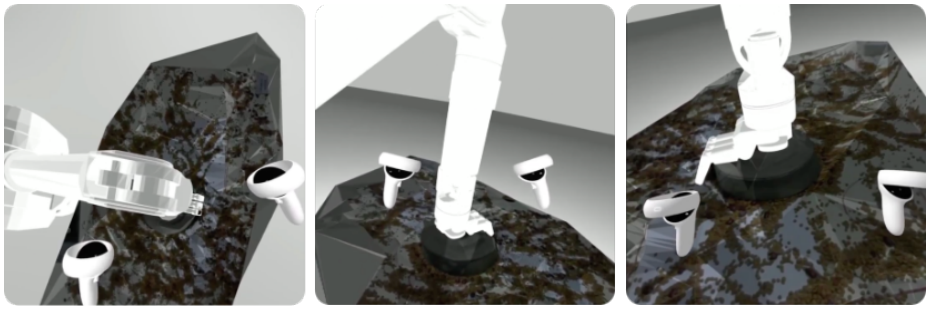

Figure 4.3: Immersive display with different operator pose.

The Oculus Quest 2 provides a per-eye resolution of 1832x1920 pixels and includes hand controllers, offering a standalone VR display without extra hardware. To ensure compatibility with various head-mounted displays, we employed the A-frame framework, allowing the video stream to be shown in a web browser. Our system supports video streaming at 1696x1600 pixel resolution over the local network, achieving a 30 FPS playback rate with approximately 10ms of latency. We used the same WebRTC stream to update the Gazebo camera pose based on the operator's pose using the VR headset and controllers, providing an immersive VR experience where the operator's movements directly control the Gazebo camera's position and orientation within the virtual environment. Figure 4.3 shows the integration of the immersive display with the simulator, with the operator's pose driving the interaction.

Chapter 5

Conclusions & Future Work

5.1 Conclusions

We begin by acknowledging the limitations of traditional teleoperation setups in replicating intuitive control and perception of direct physical interaction. To address these challenges, we utilized SLAM techniques and CARV to reconstruct the remote scene in real-time. The CARV algorithm, incorporating ORBSLAM-2 for localization and mapping, Delaunay triangulation for surface mesh reconstruction, and space carving for refinement, proved to be a suitable approach for achieving the desired reconstruction goals.

The experimental evaluation of the algorithm demonstrated promising results across various setups, including indoor rooms, and table-top scenarios. By utilizing a monocular RGB camera and a single CPU processor, the system was able to reconstruct the scene in real-time, providing operators with an up-to-date and accurate representation of the remote environment. The use of a semi-dense module further improved precision with lines and planes by removing outlier points, reducing face mesh complexity.

The integration of the 3D surface mesh reconstruction with the overall teleoperation system, including the digital twin of the WAM arm robot and the Gazebo physics simulator, resulted in a comprehensive and immersive teleoperation experience. The delay problem is fixed by using the predictive immersive display with a virtual replica

of the follower robot environment. The reconstructed surface mesh provided operators with a visual understanding of the remote environment, enabling them to make informed decisions and perform manipulation tasks effectively.

The integration of the immersive display with the teleoperation system resulted in improved immersion and easier control of the remote robot arm. By wearing the head-mounted display, operators were able to perceive the remote environment as if they were physically present, enhancing their situational awareness and enabling more intuitive manipulation of the robot. The controller provided precise and responsive input, further enhancing the operator's sense of control and dexterity.

By choosing the Gazebo physics simulator, our research leveraged a highly flexible and extensible platform that supports multiple physics engines and integrates seamlessly with ROS1 and ROS2. This choice provided us with the ability to select specific physics engines tailored to their tasks, ensuring compatibility and efficient simulation.

The integration of the physics simulator with the teleoperation system facilitated the creation of a virtual replica of the remote environment using surface meshes and the digital twin of the WAM arm robot. By controlling the digital twin robot using joint data obtained from the remote robot, the operator gets the predictive haptic feedback from the virtual replica to avoid delay-related issues.

In conclusion, this thesis contributes to the advancement of robot teleoperation in unstructured environments by addressing critical delay challenges using 3D surface reconstruction, immersive display, and physics simulation. The integration of these components paves the way for more intuitive, realistic, and predictive remote robot control, opening up new possibilities for applications in areas such as high-risk environments, medical surgeries, disaster relief, and space exploration. Through continued research and innovation, the vision of seamless and immersive human-in-the-loop teleoperation will become a reality, revolutionizing the way humans interact with and control robotic systems.

5.2 Future Work

There are several potential directions for future research and development to enhance the capabilities and performance of our integrated system.

Firstly, integrating a depth camera into our system can provide additional depth information and improve the local representation of the environment. By combining depth data with RGB imagery, more accurate 3D reconstruction can be achieved. The depth information can further enhance the precision and fidelity of the reconstructed surface meshes locally, enabling more precise manipulation and interaction with the environment.

Furthermore, the research on interactable surface meshes for robot teleoperation presented in this thesis represents an important contribution. However, certain aspects, such as contact detection and friction value calculations, currently hinder the real-time performance of the physics simulator.

Additionally, reducing the requirement for video streaming from a remote environment is a crucial aspect to consider. By utilizing keyframes and the covisibility graph obtained from the CARV algorithm, the number of frames needed for reconstructing the environment can be significantly reduced. For instance, in our experiments, the VR101 room required only around 200 keyframes, compared to the total of 2913 frames in the benchmark video. This reduction implies that the video streaming requirement can be reduced by almost 10 times. Exploring methods to leverage the obtained data, such as recreating the surface mesh and predictive textures using CARV on the operator's site, can drastically reduce the bandwidth requirement for video streaming while maintaining the necessary information for effective teleoperation.

Bibliography

[1] J. Ahmad, "Line and plane based incremental surface reconstruction," Available at https://era.library.ualberta.ca/items/adeee25c-fb1e-4dc0-a6c6-4ebc533a3eee, Masters thesis, University of Alberta, Edmonton, Alberta, CA, 2023.

[2] D. Lovi, "Incremental free-space carving for real-time 3d reconstruction," 2011.

[3] S. He, X. Qin, Z. Zhang, and M. Jagersand, "Incremental 3d line segment extraction from semi-dense slam," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 1658–1663. DOI: 10.1109/ICPR.2018.8546158.

[4] B. Stanczyk and M. Buss, "Development of a telerobotic system for exploration of hazardous environments," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, 2532–2537 vol.3. DOI: 10.1109/IROS.2004.1389789.

[5] A. Roennau, G. Liebel, T. Schamm, T. Kerscher, and R. Dillmann, "Robust 3d scan segmentation for teleoperation tasks in areas contaminated by radiation," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2419–2424. DOI: 10.1109/IROS.2010.5648926.

[6] K. Nagatani *et al.*, "Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots," *Journal of Field Robotics*, vol. 30, 2013.

[7] Z. Li, P. Moran, Q. Dong, R. J. Shaw, and K. Hauser, "Development of a tele-nursing mobile manipulator for remote care-giving in quarantine areas," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3581–3586. DOI: 10.1109/ICRA.2017.7989411.

[8] F. Gosselin, C. Bidard, and J. Brisset, "Design of a high fidelity haptic device for telesurgery," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 205–210. DOI: 10.1109/ROBOT.2005.1570120.

[9] G. Yang *et al.*, "Keep healthcare workers safe: Application of teleoperated robot in isolation ward for covid-19 prevention and control," *Chinese Journal of Mechanical Engineering*, vol. 33, 2020.

[10] R. Edlinger, M. Anschober, R. Froschauer, and A. Nüchter, "Intuitive hri approach with reliable and resilient wireless communication for rescue robots and first responders," in *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2022, pp. 75–82. DOI: 10.1109/RO-MAN53752.2022.9900739.

[11] T. Klamt *et al.*, "Flexible disaster response of tomorrow: Final presentation and evaluation of the centauro system," *IEEE Robotics and Automation Magazine*, vol. 26, no. 4, pp. 59–72, 2019. DOI: 10.1109/MRA.2019.2941248.

[12] N. Y. Lii *et al.*, "The robot as an avatar or co-worker? an investigation of the different teleoperation modalities through the kontur-2 and meteron supvis justin space telerobotic missions," 2018.

[13] E. Ackerman, *Russian humanoid robot to pilot soyuz capsule to iss this week*, 2021. [Online]. Available: https://spectrum.ieee.org/russian-humanoid-robot-to-pilot-soyuz-capsule-to-iss-this-week.

[14] J. Crandall, M. Goodrich, D. Olsen, and C. Nielsen, "Validating human-robot interaction schemes in multitasking environments," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 35, no. 4, pp. 438–449, 2005. DOI: 10.1109/TSMCA.2005.850587.

[15] T. Sheridan, "Humans and automation: System design and research issues," Jan. 2002.

[16] M. Endsley, B. Bolte, and D. Jones, *Designing for Situation Awareness: An Approach to User-Centered Design*. Jan. 2003, ISBN: 9780429146732. DOI: 10.1201/b11371.

[17] J. Johnston, S. Fiore, C. Smith, and N. Orlando, "Application of cognitive load theory to develop a measure of team cognitive efficiency," *Military Psychology*, vol. 25, May 2013. DOI: 10.1037/h0094967.

[18] M. A. Goodrich and A. C. Schultz. 2008.

[19] A. Rachmielowski, N. Birkbeck, and M. Jägersand, "Performance evaluation of monocular predictive display," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 5309–5314. DOI: 10.1109/ROBOT.2010.5509652.

[20] D. Lovi, N. Birkbeck, A. H. Herdocia, A. Rachmielowski, M. Jägersand, and D. Cobzaş, "Predictive display for mobile manipulators in unknown environments using online vision-based monocular modeling and localization," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 5792–5798. DOI: 10.1109/IROS.2010.5649522.

[21] P. Stotko *et al.*, "A VR system for immersive teleoperation and live exploration with a mobile robot," *CoRR*, vol. abs/1908.02949, 2019. arXiv: 1908.02949. [Online]. Available: http://arxiv.org/abs/1908.02949.

[22] J. Jin, L. Petrich, S. He, M. Dehghan, and M. Jägersand, "Long range teleoperation for fine manipulation tasks under time-delay network conditions," *CoRR*, vol. abs/1903.09189, 2019. arXiv: 1903.09189. [Online]. Available: http://arxiv.org/abs/1903.09189.

[23] R. Mur-Artal and J. D. Tardos, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, 1255–1262, 2017, ISSN: 1941-0468. DOI: 10.1109/tro.2017.2705103. [Online]. Available: http://dx.doi.org/10.1109/TRO.2017.2705103.

[24] G Turk, *The stanford bunny*, 2000. [Online]. Available: http://graphics.stanford.edu/data/3Dscanrep/.

[25] H. Fuchs *et al.*, "Virtual space teleconferencing using a sea of cameras," *Proceeding of First International Conference on Medical Robotics and Computer Assisted Surgery, Pittsburgh*, vol. 2, Sep. 1994.

[26] T. Kanade, P. Rander, and P. Narayanan, "Virtualized reality: Constructing virtual worlds from real scenes," *IEEE MultiMedia*, vol. 4, no. 1, pp. 34–47, 1997. DOI: 10.1109/93.580394.

[27] G. Kurillo, R. Bajcsy, K. Nahrstedt, and O. Kreylos, "Immersive 3d environment for remote collaboration and training of physical activities," Apr. 2008, pp. 269 –270, ISBN: 978-1-4244-1971-5. DOI: 10.1109/VR.2008.4480795.

[28] J. Mulligan and K. Daniilidis, "View-independent scene acquisition for telepresence," Oct. 2000. DOI: 10.1109/ISAR.2000.880933.

[29] A. E. Johnson, P. C. Leger, R. Hoffman, M. Hebert, and J. Osborn, "3-d object modeling and recognition for telerobotic manipulation," *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 1, 103–110 vol.1, 1995.

[30] C. Loop, C. Zhang, and Z. Zhang, "Real-time high-resolution sparse voxelization with application to image-based modeling," in *Proceedings of the 5th High-Performance Graphics Conference*, ser. HPG '13, New York, NY, USA: Association for Computing Machinery, 2013, 73–79, ISBN: 9781450321358. DOI: 10.1145/2492045.2492053. [Online]. Available: https://doi.org/10.1145/2492045.2492053.

[31] B. Petit *et al.*, "Multi-camera real-time 3d modeling for telepresence and remote collaboration," *International Journal of Digital Multimedia Broadcasting*, vol. 2010, Jan. 2010. DOI: 10.1155/2010/247108.

[32] S. Kohn *et al.*, "Towards a real-time environment reconstruction for vr-based teleoperation through model segmentation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain: IEEE Press, 2018, 1–9. DOI: 10.1109/IROS.2018.8594053. [Online]. Available: https://doi.org/10.1109/IROS.2018.8594053.

[33] F. Brizzi, L. Peppoloni, A. Graziano, E. D. Stefano, C. A. Avizzano, and E. Ruffaldi, "Effects of augmented reality on the performance of teleoperated industrial assembly tasks in a robotic embodiment," *IEEE Transactions on Human-Machine Systems*, vol. 48, pp. 197–206, 2018.

[34] M. Schwarz *et al.*, "Drc team nimbro rescue: Perception and control for centaur-like mobile manipulation robot momaro," in Apr. 2018, ISBN: 978-3-319-74665-4. DOI: 10.1007/978-3-319-74666-1_5.

[35] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels-surface elements as rendering primitives," in *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 2000, pp. 335–342.

[36] W. Gao and R. Tedrake, "Surfelwarp: Efficient non-volumetric single view dynamic reconstruction," *Robotics: Science and Systems XIV*, 2018. DOI: 10.15607/rss.2018.xiv.029. [Online]. Available: http://dx.doi.org/10.15607/RSS.2018.XIV.029.

[37] T. Schöps, T. Sattler, and M. Pollefeys, "Surfelmeshing: Online surfel-based mesh reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 2494–2507, 2018.

[38] C. I. Connolly, "Cumulative generation of octree models from range data," in *IEEE International Conference on Robotics and Automation*, 1984.

[39] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96, New York, NY, USA: Association for Computing Machinery, 1996, 303–312, ISBN: 0897917464. DOI: 10.1145/237170.237269. [Online]. Available: https://doi.org/10.1145/237170.237269.

[40] R. A. Newcombe *et al.*, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378.

[41] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, 2013, ISSN: 0730-0301. DOI: 10.1145/2508363.2508374. [Online]. Available: https://doi.org/10.1145/2508363.2508374.

[42] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Trans. Graph.*, vol. 36, no. 3, 2017, ISSN: 0730-0301. DOI: 10.1145/3054739. [Online]. Available: https://doi.org/10.1145/3054739.

[43] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494. DOI: 10.1109/ROBOT.2009.5152817.

[44] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574. DOI: 10.1109/ICRA.2011.5980280.

[45] A. Mossel and M. Kroeter, "Streaming and exploration of dynamically changing dense 3d reconstructions in immersive virtual reality," in *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, 2016, pp. 43–48. DOI: 10.1109/ISMAR-Adjunct.2016.0035.

[46] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, 163–169, 1987, ISSN: 0097-8930. DOI: 10.1145/37402.37422. [Online]. Available: https://doi.org/10.1145/37402.37422.

[47] C. Hoppe, M. Klopschitz, M. Donoser, and H. Bischof, "Incremental surface extraction from sparse structure-from-motion point clouds," in *British Machine Vision Conference*, 2013.

[48] J. Sun, Y. Xie, L. Chen, X. Zhou, and H. Bao, "NeuralRecon: Real-time coherent 3D reconstruction from monocular video," *CVPR*, 2021.

[49] A. Khatamian and H. Arabnia, "Survey on 3d surface reconstruction," *Journal of Information Processing Systems*, vol. 12, pp. 338–357, Jan. 2016. DOI: 10.3745/JIPS.01.0010.

[50] J. Jin, L. Petrich, S. He, M. Dehghan, and M. Jagersand, *Long range teleoperation for fine manipulation tasks under time-delay network conditions*, 2019. arXiv: 1903.09189 [cs.RO].

[51] M. Wonsick, T. Keleundefinedtemur, S. Alt, and T. Padır, "Telemanipulation via virtual reality interfaces with enhanced environment models," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic: IEEE Press, 2021, 2999–3004. DOI: 10.1109/IROS51168.2021.9636005. [Online]. Available: https://doi.org/10.1109/IROS51168.2021.9636005.

[52] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 834–849, ISBN: 978-3-319-10605-2.

[53] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852.

[54] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015. DOI: 10.1109/TRO.2015.2463671.

[55] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment — a modern synthesis," in *Vision Algorithms: Theory and Practice*, B. Triggs, A. Zisserman, and R. Szeliski, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 298–372, ISBN: 978-3-540-44480-0.

[56] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007. DOI: 10.1109/TPAMI.2007.1049.

[57] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443, ISBN: 978-3-540-33833-8.

[58] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[59] D. Galvez-Lopez and J. Tardos, "Bags of binary words for fast place recognition in image sequences," *Robotics, IEEE Transactions on*, vol. 28, pp. 1188–1197, Oct. 2012. DOI: 10.1109/TRO.2012.2197158.

[60] K. Qian, W. Zhao, K. Li, X. Ma, and H. Yu, "Visual slam with boplw pairs using egocentric stereo camera for wearable-assisted substation inspection," *IEEE Sensors Journal*, vol. 20, no. 3, pp. 1630–1641, 2020. DOI: 10.1109/JSEN.2019.2947275.

[61] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency," *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 794–805, 2013, ISSN: 1047-3203. DOI: https://doi.org/10.1016/j.jvcir.2013.05.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1047320313000874.

[62] M. Hofer, M. Maurer, and H. Bischof, "Efficient 3d scene abstraction using line segments," *Computer Vision and Image Understanding*, vol. 157, pp. 167–178, 2017, Large-Scale 3D Modeling of Urban Indoor or Outdoor Scenes from Images and Range Scans, ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2016.03.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314216300236.

[63] H. Zhou, D. Zhou, K. Peng, W. Fan, and Y. Liu, "Slam-based 3d line reconstruction," in *2018 13th World Congress on Intelligent Control and Automation (WCICA)*, 2018, pp. 1148–1153. DOI: 10.1109/WCICA.2018.8630629.

[64] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "Cpa-slam: Consistent plane-model alignment for direct rgb-d slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1285–1291. DOI: 10.1109/ICRA.2016.7487260.

[65] M. Hosseinzadeh, Y. Latif, and I. Reid, "Sparse point-plane slam," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:54066696.

[66] R. F. Salas-Moreno, B. Glocker, P. H. J. Kelly, and A. J. Davison, "Dense planar slam," *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 157–164, 2014.

[67] X. Wang, M. Christie, and E. Marchand, "Tt-slam: Dense monocular slam for planar environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 690–11 696. DOI: 10.1109/ICRA48506.2021.9561164.

[68] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-plane slam for hand-held 3d sensors," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 5182–5189. DOI: 10.1109/ICRA.2013.6631318.

[69] A. P. Gee, D. Chekhlov, W. Mayol-Cuevas, and A. Calway, "Discovering planes and collapsing the state space in visual slam," in *British Machine Vision Conference*, 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:8584240.

[70] Z. Huang, Y. Wen, Z. Wang, J. Ren, and K. Jia, *Surface reconstruction from point clouds: A survey and a benchmark*, 2022. arXiv: 2205.02413 [cs.CV].

[71] H. Smallman and M. St John, "Naive realism: Misplaced faith in realistic displays," *Ergonomics in Design: The Quarterly of Human Factors Applications*, vol. 13, pp. 6–13, Jul. 2005. DOI: 10.1177/106480460501300303.

[72] M. Lhuillier and S. Yu, "Manifold surface reconstruction of an environment from sparse structure-from-motion data," *Computer Vision and Image Understanding*, vol. 117, no. 11, pp. 1628–1644, 2013, ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2013.08.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314213001525.

[73] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, *Deepsdf: Learning continuous signed distance functions for shape representation*, 2019. arXiv: 1901.05103 [cs.CV].

[74] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *Nerf: Representing scenes as neural radiance fields for view synthesis*, 2020. arXiv: 2003.08934 [cs.CV].

[75] A. Afzal, D. S. Katz, C. L. Goues, and C. S. Timperley, *A study on the challenges of using robotics simulators for testing*, 2020. arXiv: 2004.07368 [cs.RO].

[76] J. Yoon, B. Son, and D. Lee, "Comparative study of physics engines for robot simulation with mechanical interaction," *Applied Sciences*, vol. 13, no. 2, 2023, ISSN: 2076-3417. DOI: 10.3390/app13020680. [Online]. Available: https://www.mdpi.com/2076-3417/13/2/680.

[77] Admin, *Bullet real-time physics simulation*, 2022. [Online]. Available: http://bulletphysics.org/wordpress/.

[78] *Simbody: Multibody physics api: Project home.* [Online]. Available: https://simtk.org/home/simbody/.

[79] *Dart: Dynamic animation and robotics toolkit).* [Online]. Available: http://dartsim.github.io/.

[80] [Online]. Available: https://ode.org.

[81] I. Peake, J. La Delfa, R. Bejarano, and J. O. Blech, "Simulation components in gazebo," in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, vol. 1, 2021, pp. 1169–1175. DOI: 10.1109/ICIT46573.2021.9453594.

[82] S. Ivaldi, V. Padois, and F. Nori, "Tools for dynamics simulation of robots: A survey based on user feedback," Feb. 2014.

[83] E. Rohmer, S. P. N. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1321–1326. DOI: 10.1109/IROS.2013.6696520.

[84] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular open robots simulation engine: Morse," *2011 IEEE International Conference on Robotics and Automation*, pp. 46–51, 2011.

[85] O. Michel, "Cyberbotics ltd. webots™: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, 2004.

[86] A. Farley, J. Wang, and J. Marshall, "How to pick a mobile robot simulator: A quantitative comparison of coppeliasim, gazebo, morse and webots with a focus on the accuracy of motion simulations," *Simulation Modelling Practice and Theory*, vol. 120, p. 102 629, Jul. 2022. DOI: 10.1016/j.simpat.2022.102629.

[87] C. Pinciroli *et al.*, "Argos: A modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, pp. 271–295, Dec. 2012. DOI: 10.1007/s11721-012-0072-5.

[88] L. Pitonakova, M. Giuliani, A. Pipe, and A. Winfield, "Feature and performance comparison of the v-rep, gazebo and argos robot simulators," Feb. 2018, ISBN: 978-3-319-96727-1. DOI: 10.1007/978-3-319-96728-8_30.

[89] [Online]. Available: https://leggedrobotics.github.io/SimBenchmark/.

[90] Q. L. Lidec, W. Jallet, L. Montaut, I. Laptev, C. Schmid, and J. Carpentier, *Contact models in robotics: A comparative analysis*, 2023. arXiv: 2304.06372 [cs.RO].

[91] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4397–4404. DOI: 10.1109/ICRA.2015.7139807.

[92] R. W. Cottle and G. B. Dantzig, "Complementary pivot theory of mathematical programming," *Linear Algebra and its Applications*, vol. 1, pp. 103–125, 1968.

[93] E. Drumwright and D. Shell, "Extensive analysis of linear complementarity problem (lcp) solver performance on randomly generated rigid body contact problems," Oct. 2012, pp. 5034–5039, ISBN: 978-1-4673-1737-5. DOI: 10.1109/IROS.2012.6385974.

[94] T. Liu and M. Y. Wang, "Computation of three-dimensional rigid-body dynamics with multiple unilateral contacts using time-stepping and gauss-seidel methods," *IEEE Transactions on Automation Science and Engineering*, vol. 2, pp. 19–31, 2005.

[95] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[96] M. Körber, J. Lange, S. Rediske, S. Steinmann, and R. Glück, "Comparing popular simulation environments in the scope of robotics and reinforcement learning," *CoRR*, vol. abs/2103.04616, 2021. arXiv: 2103.04616. [Online]. Available: https://arxiv.org/abs/2103.04616.

[97] A. Farhadi, S. K. H. Lee, E. P. Hinchy, N. P. O'Dowd, and C. T. McCarthy, "The development of a digital twin framework for an industrial robotic drilling process," *Sensors*, vol. 22, no. 19, 2022, ISSN: 1424-8220. DOI: 10.3390/s22197232. [Online]. Available: https://www.mdpi.com/1424-8220/22/19/7232.

[98] S. Fergusonnbsp; nbsp; nbsp; nbsp; nbsp; Siemens Digital Industries Software, *Apollo 13: The first digital twin*, 2021. [Online]. Available: https://blogs.sw.siemens.com/simcenter/apollo-13-the-first-digital-twin/.

[99] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," Mar. 2015.

[100] J. A. Douthwaite *et al.*, "A modular digital twinning framework for safety assurance of collaborative robotics," *Frontiers in Robotics and AI*, vol. 8, 2021.

[101] A. Mazumder *et al.*, "Towards next generation digital twin in robotics: Trends, scopes, challenges, and future," *Heliyon*, vol. 9, no. 2, e13359, 2023, ISSN: 2405-8440. DOI: https://doi.org/10.1016/j.heliyon.2023.e13359. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844023005662.

[102] M. Tröbinger *et al.*, "Introducing garmi - a service robotics platform to support the elderly at home: Design philosophy, system overview and first results," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5857–5864, 2021. DOI: 10.1109/LRA.2021.3082012.

[103] Y. Liu *et al.*, "Electronic skin as wireless human-machine interfaces for robotic vr," *Science Advances*, vol. 8, Jan. 2022. DOI: 10.1126/sciadv.abl6700.

[104] H. Laaki, Y. Miche, and K. Tammi, "Prototyping a digital twin for real time remote control over mobile networks: Application of remote surgery," *IEEE Access*, vol. 7, pp. 20 325–20 336, 2019. DOI: 10.1109/ACCESS.2019.2897018.

[105] K. Hagmann, A. Hellings-Kuß, J. Klodmann, R. Richter, F. Stulp, and D. Leidner, "A digital twin approach for contextual assistance for surgeons during surgical robotics training," *Frontiers in Robotics and AI*, vol. 8, Sep. 2021. DOI: 10.3389/frobt.2021.735566.

[106] P. Falkowski *et al.*, "Study on the applicability of digital twins for home remote motor rehabilitation," *Sensors*, vol. 23, no. 2, 2023, ISSN: 1424-8220. DOI: 10.3390/s23020911. [Online]. Available: https://www.mdpi.com/1424-8220/23/2/911.

[107] A. Ghosh, D. A. Paredes Soto, S. M. Veres, and A. Rossiter, "Human robot interaction for future remote manipulations in industry 4.0*****," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 10 223–10 228, 2020, 21st IFAC World Congress, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.2752. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896320335151.

[108] W. Birkfellner *et al.*, "A head-mounted operating binocular for augmented reality visualization in medicine - design and initial evaluation," *IEEE Transactions on Medical Imaging*, vol. 21, no. 8, pp. 991–997, 2002. DOI: 10.1109/TMI.2002.803099.

[109] S. Lifshitz and S. J. Merhav, "Adaptive suppression of biodynamic interference in helmet-mounted displays and head teleoperation," *Journal of Guidance, Control, and Dynamics*, vol. 14, no. 6, pp. 1173–1180, 1991. DOI: 10.2514/3.20772. eprint: https://doi.org/10.2514/3.20772. [Online]. Available: https://doi.org/10.2514/3.20772.

[110] M. Fiala, "Pano-presence for teleoperation," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 3798–3802. DOI: 10.1109/IROS.2005.1545298.

[111] R. Oboe, "Web-interfaced, force-reflecting teleoperation systems," *IEEE Transactions on Industrial Electronics*, vol. 48, no. 6, pp. 1257–1265, 2001. DOI: 10.1109/41.969406.

[112] Y. Chen, B. Zhang, J. Zhou, and K. Wang, "Real-time 3d unstructured environment reconstruction utilizing vr and kinect-based immersive teleoperation for agricultural field robots," *Computers and Electronics in Agriculture*, vol. 175, p. 105 579, 2020, ISSN: 0168-1699. DOI: https://doi.org/10.1016/j.compag.2020.105579. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169920311479.

[113] S. Hayden, *Toyota's new experimental humanoid robot uses htc vive for remote operation*, 2017. [Online]. Available: https://www.roadtovr.com/toyotas-new-experimental-humanoid-robot-uses-htc-vive-remote-operation/.

[114] F. Ferland, F. Pomerleau, C. T. Le Dinh, and F. Michaud, "Egocentric and exocentric teleoperation interface using real-time, 3d video projection," in *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, ser. HRI '09, New York, NY, USA: Association for Computing

Machinery, 2009, 37–44, ISBN: 9781605584041. DOI: 10.1145/1514095.1514105. [Online]. Available: https://doi.org/10.1145/1514095.1514105.

[115] R. Darken, M. Kempster, and B. Peterson, "Effects of streaming video quality of service on spatial comprehension in a reconnaisance task," Apr. 2009.

[116] V. Vunder, R. Valuer, C. Mcmahon, K. Kruusamae, and M. Pryor, "Improved situational awareness in ros using panospheric vision and virtual reality," Jul. 2018, pp. 471–477. DOI: 10.1109/HSI.2018.8431062.

[117] D. Labonte, P. Boissy, and F. Michaud, "Comparative analysis of 3-d robot teleoperation interfaces with novice users," *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 40, pp. 1331–42, Oct. 2010. DOI: 10.1109/TSMCB.2009.2038357.

[118] T. Kot and P. Novák, "Application of virtual reality in teleoperation of the military mobile robotic system taros," *International Journal of Advanced Robotic Systems*, vol. 15, no. 1, p. 1 729 881 417 751 545, 2018. DOI: 10.1177/1729881417751545. eprint: https://doi.org/10.1177/1729881417751545. [Online]. Available: https://doi.org/10.1177/1729881417751545.

[119] O. Liu, D. Rakita, B. Mutlu, and M. Gleicher, "Understanding human-robot interaction in virtual reality," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Lisbon, Portugal: IEEE Press, 2017, 751–757. DOI: 10.1109/ROMAN.2017.8172387. [Online]. Available: https://doi.org/10.1109/ROMAN.2017.8172387.

[120] D. Merwe, L. Maanen, F. Haar, R. van Dijk, N. Hoeba, and N. Stap, "Human-robot interaction during virtual reality mediated teleoperation: How environment information affects spatial task performance and operator situation awareness," in Jun. 2019, pp. 163–177, ISBN: 978-3-030-21564-4. DOI: 10.1007/978-3-030-21565-1_11.

[121] M. Maciaś *et al.*, "Measuring performance in robotic teleoperation tasks with virtual reality headgear," in *Automation 2019*, R. Szewczyk, C. Zieliński, and M. Kaliczyńska, Eds., Cham: Springer International Publishing, 2020, pp. 408–417, ISBN: 978-3-030-13273-6.

[122] J. Roldán Gómez, E. Peña-Tapia, P. García Auñón, J. Cerro, and A. Barrientos, "Bringing adaptive and immersive interfaces to real-world multi-robot scenarios: Application to surveillance and intervention in infrastructures," *IEEE Access*, vol. PP, pp. 1–1, Jun. 2019. DOI: 10.1109/ACCESS.2019.2924938.

[123] L. S. Yim *et al.*, "Wfh-vr: Teleoperating a robot arm to set a dining table across the globe via virtual reality," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 4927–4934. DOI: 10.1109/IROS47612.2022.9981729.

[124] Y. Chen *et al.*, *Enhanced visual feedback with decoupled viewpoint control in immersive humanoid robot teleoperation using slam*, 2022. arXiv: 2211.01749 [cs.RO].

[125] J. Zhao, R. S. Allison, M. Vinnikov, and S. Jennings, "The effects of visual and control latency on piloting a quadcopter using a head-mounted display," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 2972–2979. DOI: 10.1109/SMC.2018.00505.

[126] B. Illing, M. Westhoven, B. Gaspers, N. Smets, B. Brüggemann, and T. Mathew, "Evaluation of immersive teleoperation systems using standardized tasks and measurements," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2020, pp. 278–285. DOI: 10.1109/RO-MAN47096.2020.9223497.

[127] T. Xinxing, Z. Pengfei, and Y. Hironao, "Vr-based construction tele-robot system displayed by hmd with active viewpoint movement mode," in *2016 Chinese Control and Decision Conference (CCDC)*, 2016, pp. 6844–6850. DOI: 10.1109/CCDC.2016.7532231.

[128] G. LeMasurier, J. Allspaw, and H. A. Yanco, *Semi-autonomous planning and visualization in virtual reality*, 2021. arXiv: 2104.11827 [cs.RO].

[129] H. Cash and T. Prescott, "Improving the visual comfort of virtual reality telepresence for robotics," Nov. 2019.

[130] K. Theofilis, J. Orlosky, Y. Nagai, and K. Kiyokawa, "Panoramic view reconstruction for stereoscopic teleoperation of a humanoid robot," *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pp. 242–248, 2016.

[131] "Aiortc documentation." Accessed on 2023. (), [Online]. Available: https://aiortc.readthedocs.io/en/latest/index.html.

[132] M. Burri *et al.*, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016. DOI: 10.1177/0278364915620033. eprint: https://doi.org/10.1177/0278364915620033. [Online]. Available: https://doi.org/10.1177/0278364915620033.