

TELEOPERATION METHODS FOR HIGH-RISK, HIGH-LATENCY ENVIRONMENTS

by

Will Pryor

A dissertation submitted to Johns Hopkins University
in conformity with the requirements for the degree of
Doctor of Philosophy

Baltimore, Maryland

October, 2020

© 2023 by Will Pryor

All rights reserved

Abstract

In-Space Servicing, Assembly, and Manufacturing (ISAM) can enable larger-scale and longer-lived infrastructure projects in space, with interest ranging from commercial entities to the US government. Servicing, in particular, has the potential to vastly increase the usable lifetimes of satellites. However, the vast majority of spacecraft on low Earth orbit today were not designed to be serviced on-orbit. As such, several of the manipulations during servicing cannot easily be automated and instead require ground-based teleoperation.

Ground-based teleoperation of on-orbit robots brings its own challenges of high latency communications, with telemetry delays of several seconds, and difficulties in visualizing the remote environment due to limited camera views. We explore teleoperation methods to alleviate these difficulties, increase task success, and reduce operator load.

First, we investigate a model-based teleoperation interface intended to provide the benefits of direct teleoperation even in the presence of time delay. We evaluate the model-based teleoperation method using professional robot operators, then use feedback from that study to inform the design of a visual planning tool for this task, Interactive Planning and Supervised Execution (IPSE). We

describe and evaluate the IPSE system and two interfaces, one 2D using a traditional mouse and keyboard and one 3D using an Intuitive Surgical da Vinci master console. We then describe and evaluate an alternative 3D interface using a Meta Quest head-mounted display. Finally, we describe an extension of IPSE to allow human-in-the-loop planning for a redundant robot. Overall, we find that IPSE improves task success rate and decreases operator workload compared to a conventional teleoperation interface.

Thesis Committee

Primary Readers

Peter Kazanzides (Primary Advisor)

Research Professor

Department of Computer Science

Johns Hopkins Whiting School of Engineering

Simon Leonard

Assistant Research Professor

Laboratory for Computational Sensing and Robotics

Johns Hopkins Whiting School of Engineering

Louis Whitcomb

Professor

Department of Mechanical Engineering

Secondary Appointment, Department of Computer Science

Johns Hopkins Whiting School of Engineering

Acknowledgments

First, I would like to thank my primary advisor, Prof. Peter Kazantzides. He offered me guidance when I needed guidance and flexibility when I needed flexibility. He offered research directions at the beginning of my Ph.D, then supported the direction I wanted to take my research once I found it. He also supported me taking two summers off to take internships in industry, which I found invaluable experiences. And he supported me with patience and compassion when I was struggling with my research, especially during the COVID pandemic. Prof. Kazantzides is and will continue to be one of my best role models for his excellent leadership.

I would like to thank Prof. Simon Leonard for his advising and assistance with my research. Prof. Leonard helped me determine my intentions in the Ph.D program as well as helping me with the details of my research. I am also glad I got the opportunity to be a TA for Prof. Leonard's class, where in office hours and even in grading I enjoyed my ability to help new students of Robotics.

I would like to thank Prof. Louis Whitcomb for his advising, in areas ranging from technical to organizational to career. I would also like to thank Profs. Whitcomb, Leonard, and Kazantzides for serving on my thesis committee.

I would like to thank the other members of our research group throughout

the years. Balazs Vagvolgyi, for contributing the registration, reconstruction, and much of the visualization technology that closely integrated with my own research, and for his guidance throughout my Ph.D. Anton Deguet, for sharing his technical expertise. His help in integrating with the da Vinci was invaluable, and although my research moved away from areas in which he could directly contribute, his advice helped me throughout my Ph.D. Arko Chatterjee, for his contribution to the design and development of the VR interface, which informed our future work but sadly was not itself published. Liam Wang, for design and development of the VR interface that was published. Nick Greene, for many areas of assistance throughout the latter part of my work here, and for taking on the mantle when I am gone. And to all the others whose valuable contributions were not as entwined with my own.

I would also like to thank Prof. Axel Krieger for the experience I had working in his lab and with him as a PI. Prof. Krieger was my first introduction to medical research, and my experience was so positive I asked to expand my qualifying project into an entirely new direction. Prof. Krieger supported me with infectious enthusiasm throughout, and was understanding when I unfortunately could not bring it to completion. I am also grateful to the other excellent members of the IMERSE lab, who are too numerous to name.

I am grateful to Jie Ying Wu for her enthusiastic welcome to the lab and her drive to create the friendly and social atmosphere that has persisted long after her departure. She offered me advice, experience, and even lodging that I appreciate to this day.

I would like to thank the robot operators we worked with, particularly Billy

Gallagher and David Carabis. Your advice, assistance, and participation contributed significantly to my research, and working with you was always enjoyable.

I would like to thank my mentors during my internships at Verb Surgical, Ellen Klingbeil and Kevin Yee, my manager Haoran Yu, intern program coordinator Mary Lynn Gaddis, and the rest of my intern cohort. You gave me very valuable experiences and two excellent summers.

Finally, I would like to thank all the students of SMARTS lab for making these an excellent six years. May you continue to make this lab, LCSR, and Baltimore a great place to be.

Table of Contents

Table of Contents	viii
List of Tables	xii
List of Figures	xiii
1 Introduction	1
2 Augmented Virtuality with Model-Based Teleoperation	4
2.1 Related Work	4
2.2 Teleoperation System Description	6
2.2.1 Scene Modeling	7
2.2.1.1 Calibration and Registration	7
2.2.1.2 Modeling Unknown Geometry	8
2.2.1.3 Definition of Cutting Path	9
2.2.2 Augmented Virtuality Visualization System	11
2.2.3 Teleoperation System Implementation	11
2.3 Experiments	15
2.3.1 Mock Satellite and MLI Hat	15
2.3.2 Teleoperation Pilot Study	16

2.4	Discussion and Conclusions	23
3	Evaluation of Augmented Virtuality with Model-Based Teleoperation with Trained Experts	26
3.1	Introduction	27
3.2	Background	28
3.3	System Description	31
3.4	System Configurations	32
3.4.1	CAM: Conventional (Camera) Visualization	33
3.4.2	AV: Augmented Virtuality Visualization	34
3.4.3	KB: Conventional (Keyboard/GUI) Control Interface . .	36
3.4.4	dV: da Vinci Control Interface	38
3.5	Experiments	40
3.5.1	Study Subjects	40
3.5.2	Experimental Setup	41
3.5.3	Conventional Teleoperation	43
3.5.4	Conventional Teleoperation with Augmented Virtuality .	43
3.5.5	da Vinci Teleoperation with Augmented Virtuality . . .	44
3.6	Results	44
3.7	Discussion and Conclusions	49
4	Interactive Planning and Supervised Execution (IPSE)	50
4.1	Background and Motivation	50
4.2	System Description	54
4.2.1	Registration and Reconstruction	54
4.2.2	Interactive Planning	55

4.2.2.1	2D Interface	57
4.2.2.2	3D Interface	59
4.2.3	Supervised Execution	60
4.3	Experiments	63
4.3.1	Experimental Setup	63
4.3.2	Experimental Task	64
4.3.3	Conditions	65
4.4	Results	66
4.5	Discussion and Conclusions	72
5	A VR HMD Interface to IPSE	73
5.1	Introduction	73
5.2	Related Work	75
5.3	Background and Motivation	77
5.4	System Description	79
5.4.1	Selection of Mixed Reality HMD	80
5.4.2	Interactive Planning: Camera Visualization	80
5.4.3	Interactive Planning: Virtual Reality Interface	81
5.4.4	Interactive Planning: Motion Preview and Execution	83
5.5	Experiments	85
5.5.1	Experimental Setup	85
5.5.2	Experimental Task	85
5.5.3	Conditions	86
5.6	Results	86
5.7	Discussion and Conclusions	89

6	Extension to 7-DOF Teleoperation	91
6.1	Background	91
6.2	Method	94
6.2.1	2D Redundancy Resolution Interface	95
6.2.1.1	SEW Angle	96
6.2.1.2	Kinematic Branches	98
6.2.2	Planner Implementation	100
6.2.2.1	Workspace projection	100
6.2.2.2	Interpolation	101
6.2.2.3	Parametric Inverse Kinematics	102
6.2.2.4	Trajectory Timing	105
6.3	Computing the SEW Graph	105
6.4	Experiments	106
6.5	Results	108
6.6	Discussion and Conclusions	110
7	Discussion and Conclusion	111
7.1	Summary	111
7.2	Contributions	113
7.3	Future Work	114
A	Installing and Running IPSE	117
	References	118
	Curriculum Vitae	126

List of Tables

2.1	Angular deviation of blade during cutting task	19
2.2	Cutting task execution times	19
3.1	Time breakdown, seconds (Conv. configuration is KB+CAM)	45
3.2	Post-Experiment survey results	47
3.3	MLI cutting success rate	49
4.1	Pickup Task Success	67
4.2	Refueling Task Success	67
4.3	Average Task Duration (minutes:seconds)	71
5.1	Task Duration (seconds)	88
5.2	Variance in Position (mm^2) and Orientation (deg^2)	89

List of Figures

2.1	System diagram of the augmented virtuality teleoperation system	7
2.2	Modeling of MLI hat and the cutting path in Vision Assistant	9
2.3	Augmented virtuality view during cutting	10
2.4	Conventional teleoperation view during cutting	12
2.5	Mock satellite and UR5 robot equipped	16
2.6	Realistic mock-up of MLI ‘hat’	17
2.7	Crayon blade mounted on the cutting tool during the pilot study	18
2.8	Paths drawn on hat by all users	18
2.9	Visualization of blade trajectories	21
2.10	Robot and camera speed over time	22
3.1	Conceptual illustration of robotic servicing	28
3.2	Overview of operator station	29
3.3	Closeup of cutting assembly on UR10 robot	31
3.4	Augmented virtuality visualization of virtual 3D model	34
3.5	Conventional robot interface GUI	37
3.6	TLX survey results for Subjects 1-5	46
3.7	Visualization of the number of layers successfully cut in all MLI cutting trials	48

4.1	Teleoperation control interfaces	52
4.2	2D planning interface	56
4.3	A waypoint marker in the 2D user interface	57
4.4	3D user interface	59
4.5	Augmented Virtuality visualization used during Supervised Execution	61
4.6	The space-side servicing platform setup	63
4.7	Tool pickup success rate	66
4.8	Refueling success rate	68
5.1	Virtual reality planning environment using Meta Quest 2 headset	74
5.2	Overview of virtual reality planning interface architecture	79
5.3	Augmented Virtuality texture overlays displayed in the virtual reality planning environment	81
5.4	A screenshot of the operator’s perspective in the Meta Quest 2 headset	82
5.5	Motion planning in the virtual reality planning environment	82
5.6	The space-side satellite servicing setup	84
5.7	NASA TLX results and operator difficulty ratings	88
6.1	The updated 2D interface with redundancy control	94
6.2	The RViz visualisation during a planning operation	98
6.3	The experimental environment, with six spherical touch targets	107
6.4	Duration of each task segment	108
6.5	NASA TLX and difficulty rating	109

Chapter 1

Introduction

In-Space Servicing, Assembly, and Manufacturing (ISAM) can enable larger-scale and longer-lived infrastructure projects in space, with interest ranging from commercial entities to the US government. Servicing, in particular, has the potential to vastly increase the usable lifetimes of satellites, both existing satellites and those which will be launched in the future. Most present-day satellites are designed with a finite service life limited by on-board consumables—principally fuel for orbital maneuvering and attitude control. When a satellite on low Earth orbit reaches the end of its service life, or when an unrecoverable fault occurs, it gets decommissioned and placed on a decaying orbit, where it eventually succumbs to atmospheric drag and enters the Earth’s atmosphere. Beyond low Earth orbit, severe malfunctions may make it impossible to execute a deorbiting maneuver and the satellite becomes space junk.

The development of robotic technologies to enable on-orbit servicing of satellites would enable the repair and refueling of the over 1,000 satellites presently operational on Earth’s orbit, dramatically extending their service lifetimes and utility to society. These technologies could soon once again enable the servicing

of satellites in low Earth orbit at altitude 160-2000 km (the Space Shuttle was capable of servicing in the lower altitudes of LEO). Moreover, they could enable the unprecedented capability of servicing satellites on geosynchronous orbit at 35,900 km altitude and at the Sun-Earth Lagrange points at 1,500,000 km from Earth. However, the vast majority of spacecraft on low Earth orbit today were not designed to be serviced on-orbit. As such, several of the manipulations during servicing cannot easily be automated and instead require ground-based teleoperation.

Ground-based teleoperation of on-orbit robots is challenging due to high latency communications, with telemetry delays of several seconds, and difficulties in visualizing the remote environment due to limited camera views. Due to the specific constraints and priorities of ISAM, some common teleoperation strategies (such as direct teleoperation) are infeasible and others are not well-suited for the task. This thesis investigates teleoperation methods for on-orbit ISAM and describes the development of the Interactive Planning and Supervised Execution (IPSE) teleoperation system.

Chapter 2 investigates a model-based teleoperation interface with virtual fixtures and augments a previously-reported Augmented Reality (AR) visualization tool. In Chapter 3 we perform a comparative study of the model-based teleoperation interface using robot operators trained on on-orbit servicing operations. Chapter 4 describes IPSE and evaluates two interfaces to the system. Chapter 5 introduces and evaluates an improved 3D interface to IPSE. Chapter 6 expands IPSE to facilitate teleoperation of a redundant robot, including human-in-the-loop control of the redundant degree of freedom.

In this thesis we develop an architecture to support multiple visualization

and user input modalities for model-based teleoperation with time delay.

Through implementation and experimental evaluation, we determine that conveying human input directly to robot action is ill-suited for this task, that an interface that provides maximal opportunity to validate motions is preferred, that a 3D Head-Mounted Display interface has advantages in visualization and gross positioning, but a traditional mouse-and-keyboard interface is still superior for fine positioning, and that the interface can include elements that, with suitable kinematic parametrization of redundancy, help a human operator design feasible and safe trajectories for a 7 degree of freedom robot. Finally, we develop and refine the Interactive Planning and Supervised Execution teleoperation tool based on the results of these evaluations to produce a teleoperation system which allows operators to command robots in high-risk, high-latency environments with high success rates and low operator workload.

Chapter 2

Augmented Virtuality with Model-Based Teleoperation

This chapter presents an Augmented Virtuality (AV) based visualization and model-based teleoperation system for use in ISAM applications. We describe the design of a model-based method for teleoperation with time delay, and present the results of a pilot study of the AV visualization and model-based teleoperation on a Multi-Layer Insulation (MLI) blanket cutting task. This chapter is based on work published in [57]. Balazs Vagvolgyi contributed the scene modeling (Section 2.2.1) and augmented virtuality visualization (Section 2.2.2) components and assisted in conducting the study and analyzing the results.

2.1 Related Work

The research of teleoperation with time delay can be traced back to the 1960s, when Ferrell observed that the presence of significant time delay can negatively affect an operator's performance in completing a task and proposed a "move-and-wait" strategy to avoid stability issues [15]. Recent studies have shown

that delayed feedback results in degraded performance [2] and increased operator frustration [64]. For systems with several seconds of delay, one early effort was predictive display [4, 6], where the system predicts and displays the future position of the robot, often as an augmented reality overlay on the delayed camera images. Another approach is supervisory control [45] where, instead of directly teleoperating the remote robot, the operator issues high-level goal specifications, supervises the execution process and intervenes when errors occur. Model-based methods have increasingly been adopted for time-delayed teleoperation [18, 22, 35, 66]. This includes teleprogramming [18] and tele-sensor-programming [22], which allow the operator to interact with a simulated remote environment and teleprogram the remote robot through a sequence of elementary motion commands. Model-mediated telemanipulation [35] creates a model from the remote sensor data; this model is rendered haptically to the operator without delay. On the remote side, the robot controller only executes the position/force commands from the leader if the predicted state displayed to the user when the commands were issued matches the actual state of the robot when the commands are to be executed.

Separately, the challenge of teleoperating with limited camera viewpoints has been studied. One prior study demonstrated that the limited selection of camera perspectives available during conventional space teleoperation poses a significant mental workload [33]. In the medical domain, researchers have reported approaches for visualizing endoscopic camera images from alternate viewpoints [29, 30]. Draelos et al. [12] presented an Arbitrary Viewpoint Robot Manipulation (AVRM) framework, targeted at visualization of 3D optical coherence tomography (OCT) imaging during ophthalmic surgery. These prior

approaches offer visualization of real-time images from alternate viewpoints and therefore cannot provide visualization when cutting blind spots, as is possible with our model-based approach.

The two most recognized mixed reality concepts are augmented reality (AR) and augmented virtuality (AV) [34]. Both combine visual representations of real and virtual environments. In AR, virtual objects are overlaid on video streams. Since the 1990s, NASA has been experimenting with AR in teleoperation while servicing the ISS and other satellites to improve the operators' situational awareness [23].

In contrast, in augmented virtuality (AV) the result is a computer generated rendering of the environment, in which registered real-life images are overlaid on virtual objects. This approach enables visualization from arbitrary points of view, as opposed to AR, where the location of the camera is fixed. AV also enables the rendering of stereoscopic views of the scene, which has been shown to improve teleoperation performance [48].

2.2 Teleoperation System Description

The major components of the teleoperation system are depicted in Fig. 2.1, with conceptual contributions represented by the Scene Modeling and Augmented Virtuality Visualization components. Implementation details for these two components and the overall teleoperation system are provided in the following subsections.

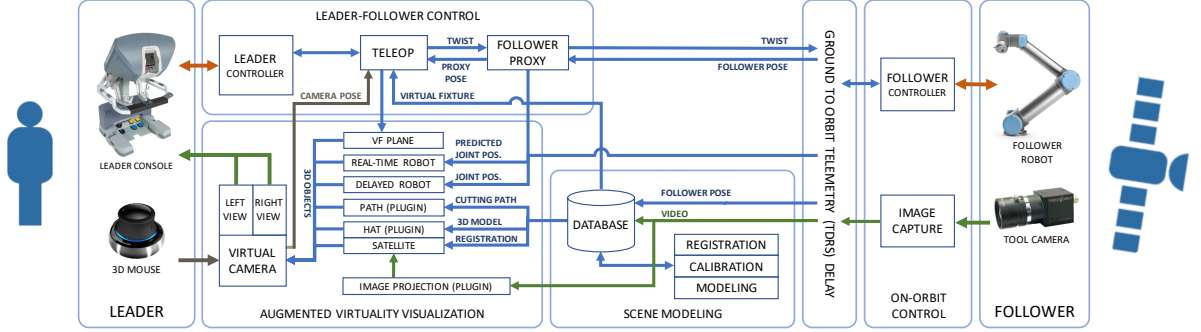


Figure 2.1: System diagram of the augmented reality teleoperation system. The telemetry delay was simulated in software, with no upstream delay and 5 seconds downstream delay. The Follower Proxy, simulated delay, and Follower Controller were combined in a single component for ease of implementation.

2.2.1 Scene Modeling

Scene Modeling is provided by the Vision Assistant software, which is responsible for image capture, camera calibration, hand-eye calibration, registration, modeling of unknown objects, and definition of cutting paths (virtual fixtures). This consists of a set of C++ applications with Qt GUI and ROS interfaces, and is deployed on the computer that handles video capture from the cameras. In an actual servicing mission, video capture would occur on-orbit and the remaining functions would be performed on the ground, as shown in Fig. 2.1.

2.2.1.1 Calibration and Registration

Accurate augmented reality visualization requires precise registration between the CAD model and the real camera images of the client satellite. This registration can be calculated by locating the satellite's natural landmarks within the images, then using pose estimation to find the satellite pose that best fits these observations. If the camera's pose (extrinsic parameters) is known from robot kinematics, then the satellite's pose with respect to the camera will

also yield a registration of the satellite to the robot’s base frame. Pose estimation is sensitive to the landmark observation accuracy; thus, we combine pose estimates from multiple camera viewpoints to obtain more accurate registration.

This registration procedure requires the camera’s extrinsic and intrinsic parameters. The camera intrinsics can be calibrated prior to launch, and they are unlikely to change during the mission. However, the Vision Assistant is capable of re-calibrating the camera during flight using a checkerboard pattern or natural landmarks. The Vision Assistant can also calculate the tool camera’s extrinsic parameters using either natural features or a checkerboard pattern. The algorithm first uses Tsai’s method [53] to solve the conventional $AX=XB$ hand-eye formulation, then refines X using reprojection error minimization.

2.2.1.2 Modeling Unknown Geometry

The MLI hat is a soft structure that is not included in the CAD model of the satellite and therefore must be modeled based on an image survey performed with the tool camera. The modeling takes place in the Vision Assistant, where the operator manually locates natural landmarks on the MLI that are unambiguously identifiable on at least two images taken from different view angles. Once the landmark observations are added, the software automatically calculates the landmark positions in 3D space with respect to the satellite’s base coordinate frame. The triangulation algorithm uses a closed-form least squares method to find the best positions given at least two observations per landmark.

Knowing the 3D coordinates of the MLI hat landmarks enables the user to create triangular or quadrilateral ‘faces’ between the landmarks and build a 3D mesh of the MLI object. The landmarks serve as vertices and the faces

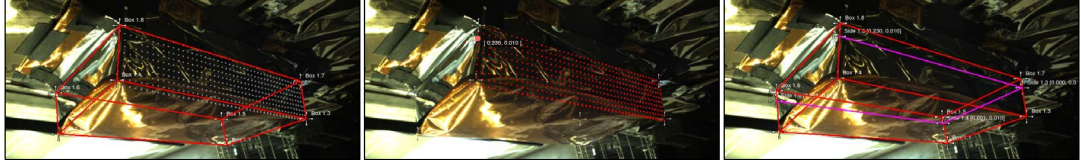


Figure 2.2: Modeling of MLI hat and the cutting path in Vision Assistant. Geometry of the hat shown with a 5 mm grid on one side (left), user selects grid point for landmark creation (center), cutting path defined and shown as a purple line (right).

are converted into triangles that form the topology of the mesh. The Vision Assistant sends the mesh to the visualization computer, where a custom RViz plugin converts it into an Ogre3D object and adds it to the OpenGL scene (Fig. 2.2).

While we only modeled the box-like MLI hat for our specific application, the method allows the construction of arbitrary 3D shapes – e.g. a deformed MLI hat – as long as 3D landmarks marking the shape’s outlines are available.

2.2.1.3 Definition of Cutting Path

The Vision Assistant provides the capability to define paths by connecting multiple landmarks with one continuous line. However, the desired cutting path may not be located between uniquely identifiable natural landmarks. In fact, for the MLI cutting task, the path lies on a flat, featureless area, where picking a landmark on multiple camera views would be nearly impossible, especially considering the highly reflective nature of the MLI. For this reason, the Vision Assistant provides a helper tool for adding 3d vertices with high precision in such featureless spots. The tool enables the user to project a metric grid on any previously modeled quadrilateral face. The resolution of the grid can be customized to arbitrary precision, and the coordinate frame of the grid can be

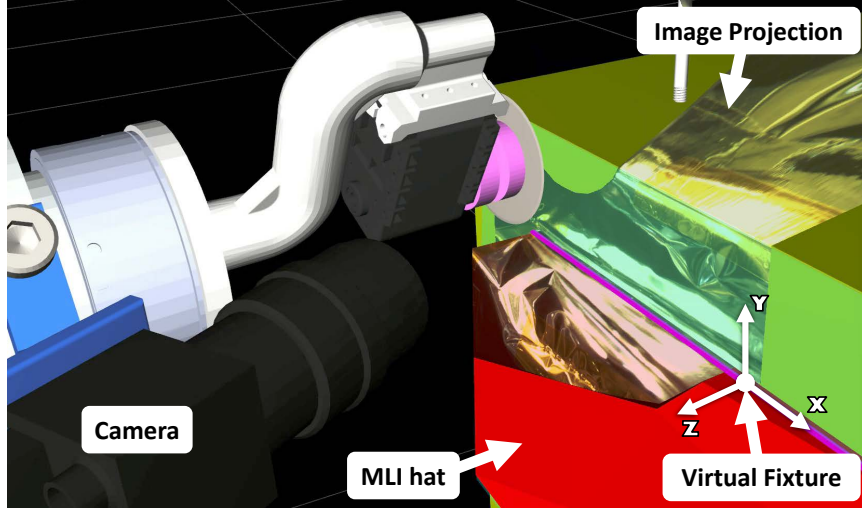


Figure 2.3: Augmented virtuality view during cutting (labels and coordinate frame added for clarity); see Fig. 2.7 for a real image of a similar scene.

aligned with any corner and side of the face. Using the grid tool, the user can pick any point on any previously modeled surface with sub-millimeter precision, then create a 3D landmark on that grid point, as shown in Fig. 2.2 (right). The quadrilaterals defined by the user need not be planar (i.e., skew quadrilaterals), therefore the projected grid can follow the 3D curvature of the faces.

In our experiments, the ideal cutting path, defined in the Vision Assistant, was displayed for the robot operators in the master console (e.g., purple line in Figs. 2.2 and 2.3). For robot control, however, we chose to represent each line in the path by a virtual fixture plane, so that the operator can easily control the depth of cutter penetration into the MLI. For the experiments reported here, all cut path lines were designed to lie in a single plane, so we defined the virtual fixture by fitting a plane to the landmarks of the ideal cutting path.

2.2.2 Augmented Virtuality Visualization System

The master side visualization system is responsible for rendering the stereoscopic augmented virtuality view for the master console. It comprises a collection of ROS RViz plugins and configuration files, and is capable of rendering the mixed reality scene featuring the following elements (Fig. 2.3):

Client satellite: A combination of known geometry (satellite CAD model) and reconstructed geometry (MLI hat).

Time-delayed robot, camera, and cutting tool: Represents the state of the robot acquired through delayed telemetry. Rendered semi-transparently.

Real-time robot, camera, and cutting tool: The command the robot operator is sending to the satellite.

Delayed video projection: Camera images arrive after a few seconds of delay. Rendered with high opacity.

Ideal cutting path and virtual fixture: Cutting path is modeled in Vision Assistant and virtual fixture is calculated from points on the path.

The visualization module is also responsible for rendering the conventional teleoperation display used for the baseline experiments. In this mode, the master console displays the unaltered tool camera image for both the left and the right eye, as shown in Fig. 2.4.

2.2.3 Teleoperation System Implementation

The master console of a da Vinci surgical robot (Intuitive Surgical, Sunnyvale, CA) is used to teleoperate a Universal Robots UR5 (Universal Robots, Odense,

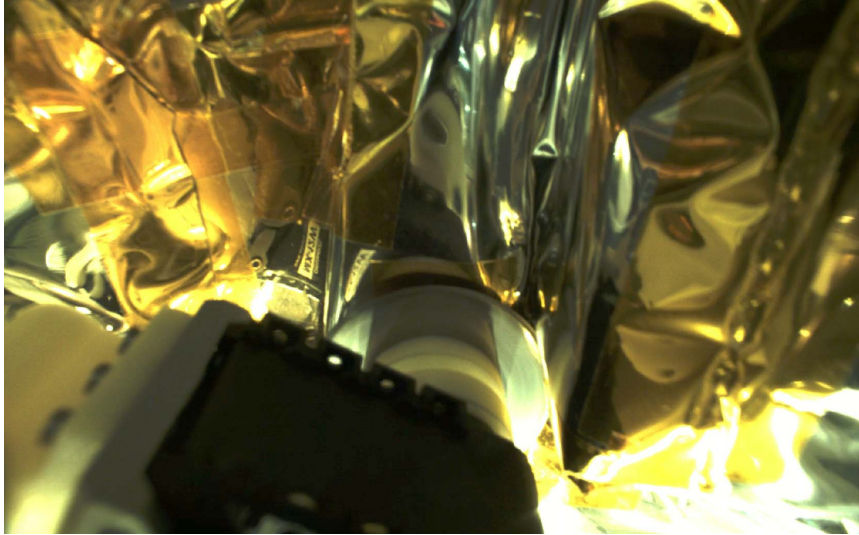


Figure 2.4: Conventional teleoperation view during cutting.

Denmark). Specifically, the operator uses the left da Vinci Master Tool Manipulator (MTM) to provide the commanded pose of the follower UR5 robot. The MTM is controlled by the da Vinci Research Kit (dVRK) open source controllers [26]. In principle, it is possible to use the da Vinci MTM to control the virtual camera, but in our implementation we use a SpaceNavigator (3Dconnexion, Munich, DE) 3D mouse, placed on the arm rest of the master console. While performing the task, the operator uses the stereo visualization system on the da Vinci master console to observe either the view from the follower robot’s tool camera or the augmented virtuality view described in Section 2.2.2.

The UR5 end-effector is equipped with a rotary cutting tool and a camera, as shown in Figs. 2.3, 2.4, 2.5 and 2.7. The cutter is composed of a 45 mm rotary blade (Arteza, Wilmington, DE) attached to a Dynamixel MX-12W servo motor (Robotis, Lake Forest, CA). The motor is attached to a 6 axis force/torque sensor (JR3 Inc., Woodland, CA) by a 3D printed curved link. The blade is inserted on a mandrel and is secured between two concentric adapters. The

overall length of the tool is 121 mm. The motor speed can be controlled in software, with a maximum no load speed of 470 RPM. Additional safety features such as maximum load and maximum torque limits are implemented to protect the hardware and the environment.

The tool camera is a lightweight 1080p High Definition device (PointGrey BlackFly, FLIR Integrated Imaging Solutions Inc., BC, Canada) mounted rigidly on the robot’s end effector, oriented towards the cutting tool, as shown in Fig. 2.7.

The teleoperation software communicates the commanded motion from the user’s manipulation of the MTM to the UR5. In augmented virtuality mode, the operator provides position input. The command position of the UR5 relative to the virtual camera is set to the measured position of the MTM relative to the da Vinci stereo viewer, with a translation offset to allow the smaller MTM workspace to accommodate the larger UR5 workspace:

$$p_{\text{cmd}} = T_{\text{vc},s}T_{\text{m},\text{sv}}p_{\text{mtm}} + p_{\text{off}} \quad (2.1)$$

where p_{mtm} and p_{cmd} are the measured MTM pose and commanded UR5 pose, $T_{\text{vc},s}$ is the transform from the UR5 to the virtual camera, $T_{\text{m},\text{sv}}$ is the transform from the MTM to the da Vinci stereo viewer, and p_{off} is the stored offset, with orientation component fixed at identity. Operators can “clutch” to adjust the position offset at any time by depressing one of the foot pedals in the master console. As shown in Fig. 2.1, commands to the robot are sent as a twist derived

from the desired position,

$$v_{\text{cmd}} = \min(k_p x_{\Delta}, v_{\text{max}}) \quad (2.2)$$

$$\omega_{\text{cmd}} = \min(k_{\theta} \theta_{\Delta}, \omega_{\text{max}})$$

where v_{cmd} and ω_{cmd} are the linear and angular velocity components of the commanded twist, x_{Δ} and θ_{Δ} are the position and orientation components of $p_{\Delta} = p_{\text{cmd}} - p_{\text{ur5}}$, p_{ur5} is the measured position of the UR5, and k_p , k_{θ} , v_{max} , and ω_{max} are constant.

In conventional mode, because the real camera moves with the blade, it is unintuitive to use position input. We therefore use rate input, where the MTM is servoed to a fixed position p_{fixed} and the user displaces it proportionally to the desired rate, similar to a joystick. The command twist is computed using (2.2) with $p_{\text{cmd}} = T_{\text{vc,s}} T_{\text{m,sv}}(p_{\text{fixed}} - p_{\text{mtm}})$.

The teleoperation software is aware of all virtual fixtures that have been defined in the virtual environment (see Sec. 2.2.1.3). For this application, each virtual fixture is assumed to be a plane, but the software could be extended to support more complex fixtures. The operator may choose to use the virtual fixture as simply a visual cue to aid in aligning the blade or can press a foot pedal to “attach” to the virtual fixture. When the tool is attached to a virtual fixture, the control mode is modified to assist the operator in keeping the cutting blade parallel to the virtual fixture plane, while still allowing the operator to override this assistance if necessary (i.e., a soft virtual fixture). The soft virtual fixture is achieved by the use of non-isotropic gains. Specifically, lower gains are used for the direction perpendicular to the plane (along the z axis in Fig.

2.3) and for the rotations out of the plane (about the x and y axes in Fig. 2.3), resulting in slower motion in any direction not parallel to the virtual fixture.

Assistance in alignment of the blade to the virtual fixture is provided by an imposed force gradient that guides the blade into alignment with the virtual fixture. The force gradient is implemented by projecting the robot’s current pose into the virtual fixture plane to obtain a p_{guide} , then computing a corresponding twist as in (2.2) with p_{guide} replacing p_{cmd} . The resulting twist is added to v_{cmd} and ω_{cmd} . This results in the blade drifting into alignment with the virtual fixture if the operator allows the motion, but enables the operator to override the motion if necessary.

2.3 Experiments

2.3.1 Mock Satellite and MLI Hat

It is desirable to evaluate MLI cutting performance under realistic conditions, since minor details of material selection and visual appearance may significantly affect the success of execution. Thus, we constructed realistic models of an MLI hat and mounted them on our scaled down mock satellite platform [56] shown in Fig. 2.5.

The natural features of the mock satellite, and their known locations in the corresponding CAD model, were used to establish registration between the robot platform and the satellite, as described in Section 2.2.1.1. The MLI blanket covering the frame of the satellite provided a realistic and stable platform for the MLI hat.

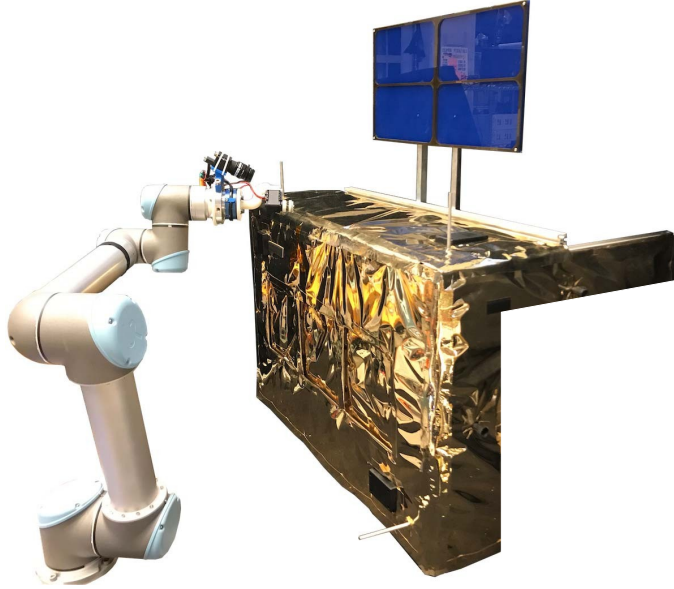


Figure 2.5: Mock satellite and the UR5 robot equipped with the tool camera and the rotary cutting tool.

The hats were fabricated from publicly available, non-aerospace-rated materials, but the materials were carefully selected and assembled to accurately simulate the rigidity, thickness, texture, and visual appearance of the flight-rated blanket (Fig. 2.6).

2.3.2 Teleoperation Pilot Study

The purpose of this pilot study is to evaluate whether the scene modeling and AV visualization makes it easier for operators to perform motions that are representative of cutting. The operators’ performance while using a conventional teleoperation interface was compared to their performance using our proposed AV visualization interface and the virtual fixture. The cutting task was simulated by a non-destructive drawing task, using a circular crayon “blade” (Fig.

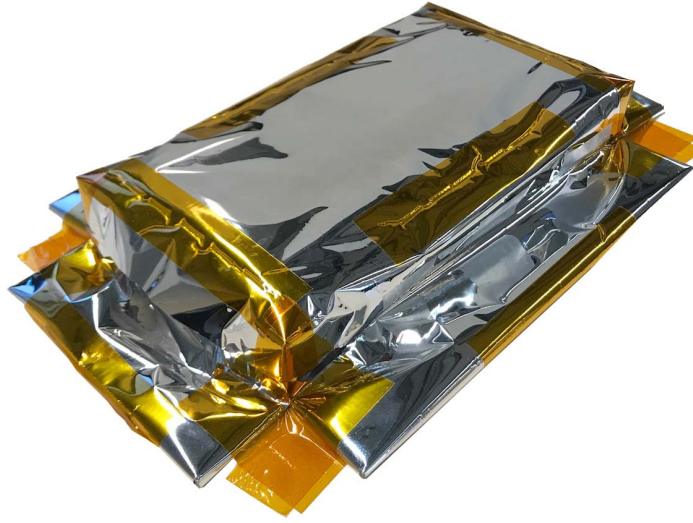


Figure 2.6: Realistic mock-up of MLI ‘hat’

2.7). We placed a foam support inside the MLI hat to provide sufficient rigidity for the drawing task; this also keeps the MLI hat from sagging due to the effects of gravity in our ground-based platform.

Correctly registered virtual fixtures are enabled by the registration and scene modeling steps performed for AV visualization, therefore in our experiments the use of virtual fixtures was limited to the AV visualization tasks.

The seven test subjects were all familiar with the design of the system, reflecting the use of skilled operators in the real-world cutting task. Operators were instructed on the use of the system and allowed to practice. For the trials with the virtual fixture, the fixture was pre-defined and always visible in the AV view. Operators were free to activate and deactivate the virtual fixture control features at will. For the conventional teleoperation trials, operators were asked to draw a straight line near the base of the MLI hat, but were not asked to follow a specific trajectory. Each operator performed the conventional teleoperation task first and the AV teleoperation task second.

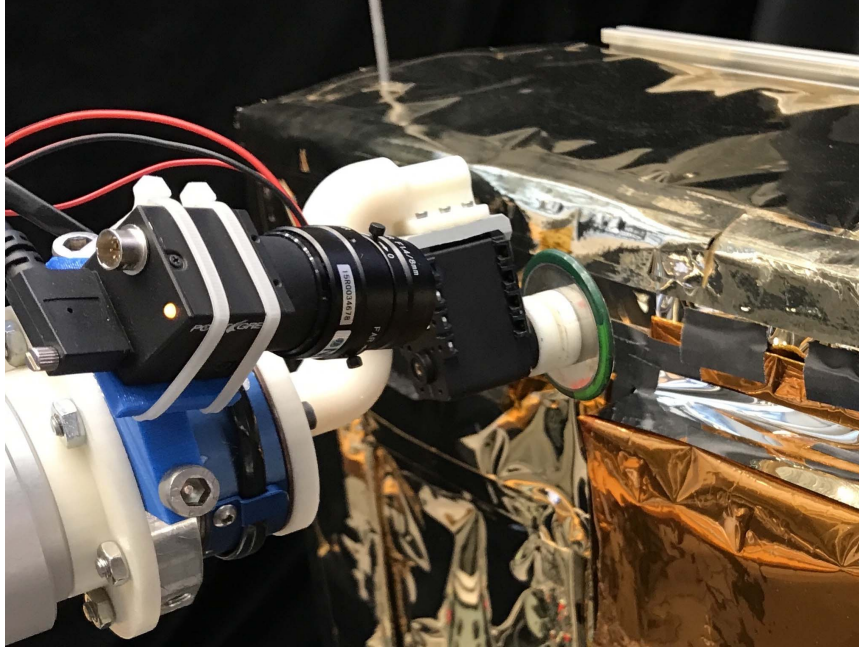


Figure 2.7: Crayon blade mounted on the cutting tool during the pilot study.

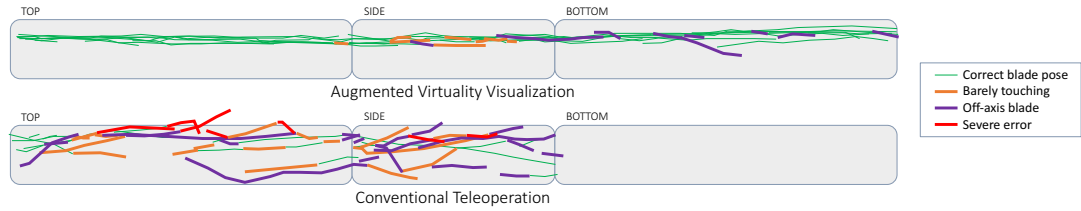


Figure 2.8: Paths drawn on hat by all users; three sides for augmented virtuality and two sides for conventional teleoperation.

For both control approaches, the operators started with the top of the MLI hat and continued smoothly to the side. When using AV, the operator also continued to the bottom of the hat but with the delayed camera image projection disabled, mimicking the expected task where one of the sides may be cut without camera visibility. In AV mode, this restriction means only that the operator has less visual feedback. However, in the conventional teleoperation mode, the task is impossible without visualization, so only the top and side are cut.

The resulting paths from all trials are shown in Fig. 2.8. Green lines indicate

Table 2.1: Mean (standard deviation) of angular deviations of the blade during the entire length of the cutting task (degrees), not including periods when the follower robot was not moving.

	Augmented Virtuality				Conventional Teleoperation			
	$\angle VFPlane$		$\angle Trajectory$		$\angle VFPlane$		$\angle Trajectory$	
User 1	8.4	(3.9)	8.3	(9.2)	6.8	(3.8)	13.6	(11.5)
User 2	2.0	(1.4)	6.5	(10.2)	11.5	(3.2)	11.5	(11.5)
User 3	3.1	(1.4)	10.1	(16.1)	9.6	(4.1)	13.8	(15.3)
User 4	7.9	(6.9)	12.9	(15.4)	17.2	(8.6)	15.0	(12.0)
User 5	3.9	(3.0)	9.8	(10.6)	20.2	(5.9)	15.6	(16.0)
User 6	4.6	(2.9)	11.5	(10.6)	19.9	(5.5)	18.0	(10.5)
User 7	3.1	(3.1)	10.1	(12.6)	22.2	(6.0)	16.1	(14.9)
All Users	4.7	(3.7)	9.9	(12.4)	15.3	(5.6)	14.8	(13.3)
z test h(p)	1 (5.7e-7)		0 (0.33)		1 (2.7e-14)		0 (0.30)	

Table 2.2: Task execution times (seconds) for each user. The augmented virtuality (AV) values correspond to the same progress along the path as the conventional teleoperation (Conv.) times.

User	1	2	3	4	5	6	7	Mean	σ	z test h(p)
AV	178	176	396	321	357	231	344	286	90	1 (p=0.026)
Conv.	316	449	382	662	1430	600	418	608	382	1 (p<0.001)

proper cutting, while other colors indicate various types of errors. While using AV mode, shown at the top, operators were able to both keep much straighter paths and keep the cutting blade properly positioned. Using the conventional teleoperation mode, operators struggled to maintain the proper drawing angle, maintain contact, and maintain a straight path. We expect this is caused by loss of situational awareness when using the narrow view from the tool camera, which is further exacerbated by the reflectivity of the MLI.

Fig. 2.9 and Table 2.1 show the difference in orientation between the blade and virtual fixture plane for both modes. Similarly, these show that the majority of operators were better able to maintain the proper blade alignment using AV

with virtual fixtures than using conventional teleoperation. These also show that some operators were not able to finish cutting the side using conventional teleoperation after becoming completely disoriented. The mean and standard deviation for all users in Table 2.1 is computed by averaging the individual means and variances, which assumes that each user’s data is an independent random variable. A Z-test analysis of the data in Table 2.1 shows that the angular deviation of the blade with respect to the VF plane is distinct between the AV and conventional teleoperation populations with a 95% confidence level, but that the deviation of the blade with respect to the trajectory is not distinct.

Table 2.2 shows the execution times for the drawing task in the AV and conventional teleoperation modes. The time listed for both trials is the time until the blade reached the endpoint of that operator’s conventional teleoperation task, which controls for both the additional bottom-side “cut” in AV mode and the fact that some operators were not able to complete both sides in the conventional teleoperation mode. For all but one operator, the AV task was completed more quickly than using conventional teleoperation. On average, the AV task was completed in less than half the time as the conventional teleoperation task. This is partially attributed to the fact that the conventional teleoperation mode did not provide visual feedback to the operators until the delayed camera feed caught up to their actions, causing them to adopt a move-and-wait approach. Using AV, operators could see the commanded position of the robot in real time, allowing them a level of feedback to perform longer stretches of the task without waiting for confirmation from the delayed camera feed. A Z-test analysis of the data in Table 2.2 shows that the AV population is distinct from the conventional teleoperation population with a 95% confidence level.

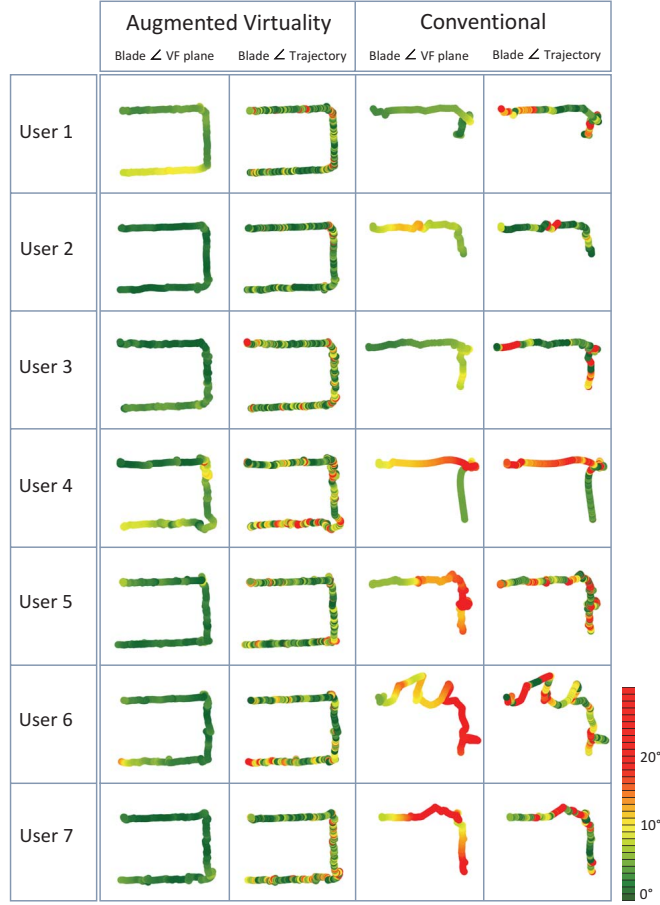


Figure 2.9: Visualization of blade trajectories projected on the virtual fixture plane. Colors indicate the angular deviation of the blade from the virtual fixture plane (columns 1 and 3) and the angular deviation of the blade from the direction of motion (columns 2 and 4).

Fig. 2.10 shows the robot speed over time for a single representative trial. The move-and-wait strategy is clearly visible for the entire duration of the conventional teleoperation, while the augmented virtuality teleoperation had periods of sustained motion. The extra time spent waiting for the delayed visual feedback more than compensates for the time spent moving the virtual camera. An analysis of the Cartesian velocities for all trials reveals that operators using conventional teleoperation paused, for 5 or more seconds, 3.25 times more often

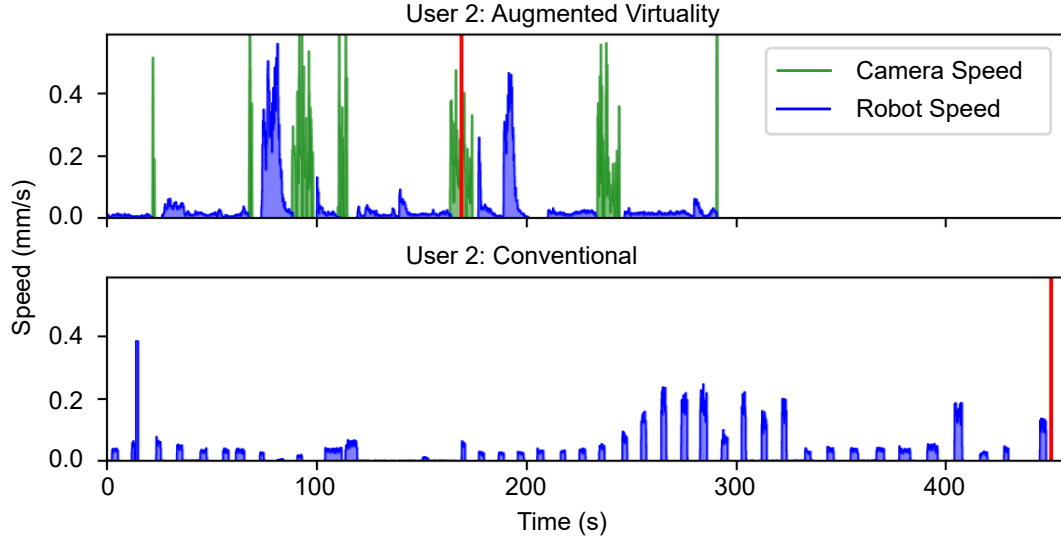


Figure 2.10: Robot and camera speed over time during a typical run. In AV mode, the velocity of the follower robot is smooth and continuous, and the operator only stops moving the robot to adjust the virtual camera. In Conventional mode, the operator moves the robot a short distance at a time, then waits for the delayed video feedback; this results in a characteristic move-and-wait pattern. Red lines represent the bottom corner, which is the stopping point for the conventional teleoperation mode. As the location of the red lines indicate, in this particular run, the operator reached the bottom corner of the MLI hat in a significantly shorter time in AV mode.

than operators using AV ($p < 0.001$).

The reported system, including an augmented virtuality view and virtual fixtures, allowed the operators to perform the task more quickly and accurately than with conventional teleoperation. Using AV, the paths were straight with minimal gaps, while with conventional teleoperation the paths deviated significantly from a straight line and contained large gaps. Additionally, the blade angle deviation from the ideal cutting plane was minimal using AV and often extreme using conventional teleoperation. It should be noted that the effect of blade angle deviation with respect to the desired trajectory would be greater when using this system to cut MLI, as compared to the drawing task, because

a skewed path may force the blade off the desired trajectory or rip the MLI.

2.4 Discussion and Conclusions

Due to the high risk of remote operations in space, a human operator will remain in the loop whenever feasible. Our approach, therefore, was to develop a system to improve the performance of the human operator in the challenging case of teleoperation with multi-second time delay and limited camera views. This system relies on three capabilities: (1) creation of a 3D model of the satellite that is registered with respect to the coordinate system of the remote robot, (2) visualization of the 3D model from arbitrary viewpoints, augmented by projections of the (delayed) camera feedback, and (3) motion assistance, in the form of virtual fixtures. Clearly, the latter two capabilities rely on the model created by the first. The experiments demonstrate that the proposed system can significantly improve user performance for time-delayed teleoperation, compared to a baseline case with no modeling, a fixed view from the tool camera, and no motion assistance. Moreover, this augmented virtuality system enabled the human operators to reliably perform otherwise infeasible teleoperation tasks in “blind spots”, in this case, the bottom of the MLI hat, that cannot be imaged directly by the camera at the remote site. Note that our experiments did not identify the relative benefits of augmented visualization and motion assistance, which remains a topic for future studies.

In addition to teleoperation, a primary role for the human operator is to carefully monitor operations and handle unexpected conditions. For the MLI cutting task, this is important because the MLI hat is not rigid and is likely

to deform and/or shift. Because the real video feedback is registered with, and projected onto, the model, the augmented virtuality interface enables the operator to detect these changes by observing the discrepancy between the projected camera image and the underlying model. For small discrepancies, the operator can compensate by adjusting the path, which is allowed by the soft virtual fixture constraints. If the discrepancy becomes large, the operator can pause the cutting operation, perform another image survey to update the model, and then resume cutting.

Our pilot studies of drawing on the MLI surface revealed several opportunities for improvement, both with the augmented virtuality and conventional interfaces. With the augmented virtuality interface, it was sometimes challenging for the operator to obtain the desired virtual view using the 3D mouse. One simple improvement would be to allow the operator to quickly select from a set of views that are pre-defined with respect to the cutting blade; for example, views from above or in front of the blade wheel. This set of pre-defined views could be augmented by the operator, who could “save” a particular view and then recall it later. For the conventional interface, it would have been helpful to include a predictive display [6] to give the operator a better sense of how the commanded robot motion compared to the currently visualized position. In practice, satellite servicing systems may be equipped with situational awareness (SA) cameras mounted on the base platform of the robot arms. While the remote locations of the SA cameras would prevent them from providing high quality close-up imaging of the immediate surrounding of the robotic tools, operator performance during our experiments would likely have benefited from having additional SA camera views available, especially during conventional

teleoperation. Another limitation of the reported experiments was the use of a fixed round-trip telemetry delay of 5 seconds; in an actual mission, this delay is expected to vary by several seconds, which would further degrade teleoperation performance.

Chapter 3

Evaluation of Augmented Virtuality with Model-Based Teleoperation with Trained Experts

In this chapter we present an evaluation of the Augmented Virtuality with Model-Based Teleoperation system described in Chapter 2 using professional robot operators trained on a real-world satellite servicing task. We also adjust the experimental setup to more closely replicate the real-world task by replacing the crayon-drawing proxy task with the actual MLI cutting task and by updating our conventional interface to more closely imitate the interface our professional operators are trained to use. This chapter is based on work published in [40]. Balazs Vagvolgyi contributed the conventional and augmented virtuality visualization (Sections 3.4.1 and 3.4.2) and assisted with the experimental procedure and data analysis.

3.1 Introduction

We previously described the development of an *augmented virtuality* interface (see Chapter 2), where the operator interacted with a *virtual* 3D model of the satellite that was augmented by projecting the *real* images from the robot tool camera onto the satellite model. The 3D model was obtained by performing 2D/3D registration between a 3D model of the satellite and multiple 2D images (from a robotic survey) [56] and by reconstructing “unknown” objects (i.e., objects not in the satellite CAD model) from the 2D images [57]. A multi-user study was performed to evaluate the system, subject to a telemetry time delay of 5 seconds between leader and follower, for a task that emulated a satellite servicing operation [57]. Limitations of the prior study were that the human subjects were not trained robot operators and the task, drawing on the satellite surface, was not an actual servicing task. This chapter reports the results of an experimental study to compare the proposed augmented virtuality visualization to the conventional camera-based visualization for an actual servicing task with trained robot teleoperators. In addition, we report a keyboard and graphical user interface (GUI) that more closely emulates the conventional robot control interface typically used by our subjects, and report an experimental evaluation of task performance with this interface in comparison to our previously reported teleoperation console, [56–59, 62, 63], based on the master console of a da Vinci surgical robot [21].

The remainder of this chapter is organized as follows: Section 3.2 presents background information about the satellite servicing task that motivated this work. Sections 3.3 and 3.4 describe the platform and system configurations at

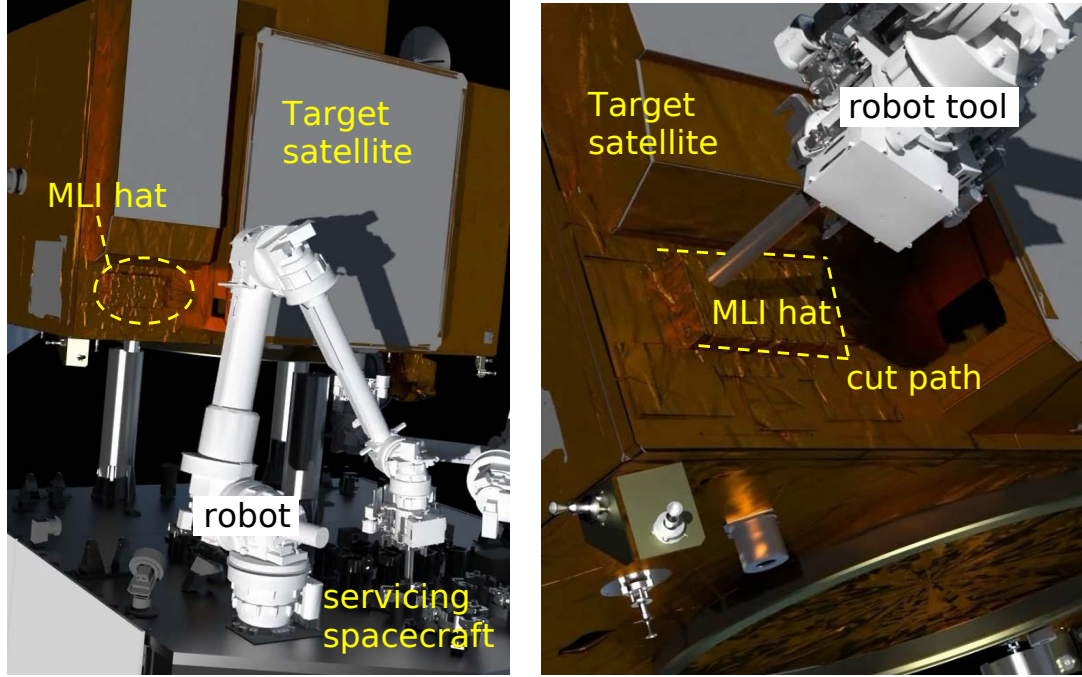


Figure 3.1: Conceptual illustration of robotic servicing. Images captured from video on NASA website [36]. Left: view showing servicing spacecraft (bottom) docked with target satellite (top). Right: closer view of multi-layer insulation (MLI) hat on target satellite, which must be removed (cut on three sides, as indicated by dashed line) to access fill/drain valves. Direct visualization may be obstructed during cutting of top path segment.

Johns Hopkins University (JHU) that are used for the experiments described in Section 3.5, with results presented in Section 3.6.

3.2 Background

The majority of spacecraft are covered in a thin blanket of thermal insulation material called multi-layer insulation (MLI) [16], which must be removed prior to servicing. MLI is composed of thin layers of insulating foil stacked together in a blanket which is then wrapped around the spacecraft to help regulate the temperature of internal components. MLI blankets have very little structural



Figure 3.2: Left: Conventional teleoperation keyboard and GUI (KB); Center: da Vinci master console (dV) with space mouse; Right: remote robot with satellite (see Fig. 3.3 for closeup of cutting assembly). Left image shows operator performing trial with augmented virtuality (AV) visualization on 3D monitor (lower left monitor). Operator is wearing noise-canceling headphones and 3D shutter glasses. Conventional (CAM) visualization is similar, except that lower left monitor shows 2D camera image. Lower right monitor shows robot control GUI (also shown in Fig. 3.5).

rigidity so that they can be easily fit to arbitrarily shaped satellites, which can complicate efforts to manipulate and remove them robotically. Further difficulties are introduced by the gradual degradation of MLI in the low Earth orbit environment from solar radiation, atomic oxygen, micro-meteor debris impact, and other effects. In addition, MLI layers often have high reflectivity, which can cause large disparities in brightness when viewed with cameras. When attempting to service a vehicle, it is often undesirable to remove entire sections of blanketing. Thus, servicing missions must be capable of cutting into a blanket to remove only a portion of it.

Different cutting tools may be required for differing MLI geometries and to accommodate a variety of underlying equipment and surfaces. Prior research at JHU [58, 59, 62, 63] used a blade to cut through tape that secured an MLI patch over the refueling valve. A similar blade was used to demonstrate telerobotic cutting of MLI tape in a cross-country experiment between the University of

Pennsylvania and the Jet Propulsion Laboratory in California [49]. In many cases, items of interest that protrude from the flat paneling of a spacecraft, such as the refueling valve in Landsat 7, are covered with MLI structures that provide a box-like shape over the items, in which blanketing is not secured directly to critical interfaces, but only to the exterior panels around it. These free-standing MLI “hats” (see Fig. 3.1-right) present a unique challenge for robotic systems to remove.

One such tool utilizes a rotary cutting blade to cut into free-standing MLI. This tool has been tested extensively by experienced operators using ground robots. During testing, operators have indicated that they rely heavily on the views provided by cameras mounted on the robot’s end effector to inform their situational awareness (SA) and make real time decisions on robot commands. However, there are several factors that reduce a teleoperator’s ability to utilize these camera views. In tight areas, these cameras often have limited visibility of the MLI being cut, as anticipated for the top segment of the cut path (dashed lines) in Fig. 3.1. In addition, remote teleoperation of in-space systems involves inherent limits on bandwidth that provide an upper bound on image quality and frame rate. Commands to, and telemetry from, remote systems are relayed through ground stations and often one or more communications satellites, introducing delays on the order of 2-7 seconds between sending a command and receiving telemetry of the robot’s response. In controlled ground-based experiments, without telemetry delays, highly skilled telerobotic operators with years of experience may take in excess of one hour to completely cut and remove an MLI hat covering the satellite fuel valves. When the factors above are considered, this time can significantly increase.

3.3 System Description

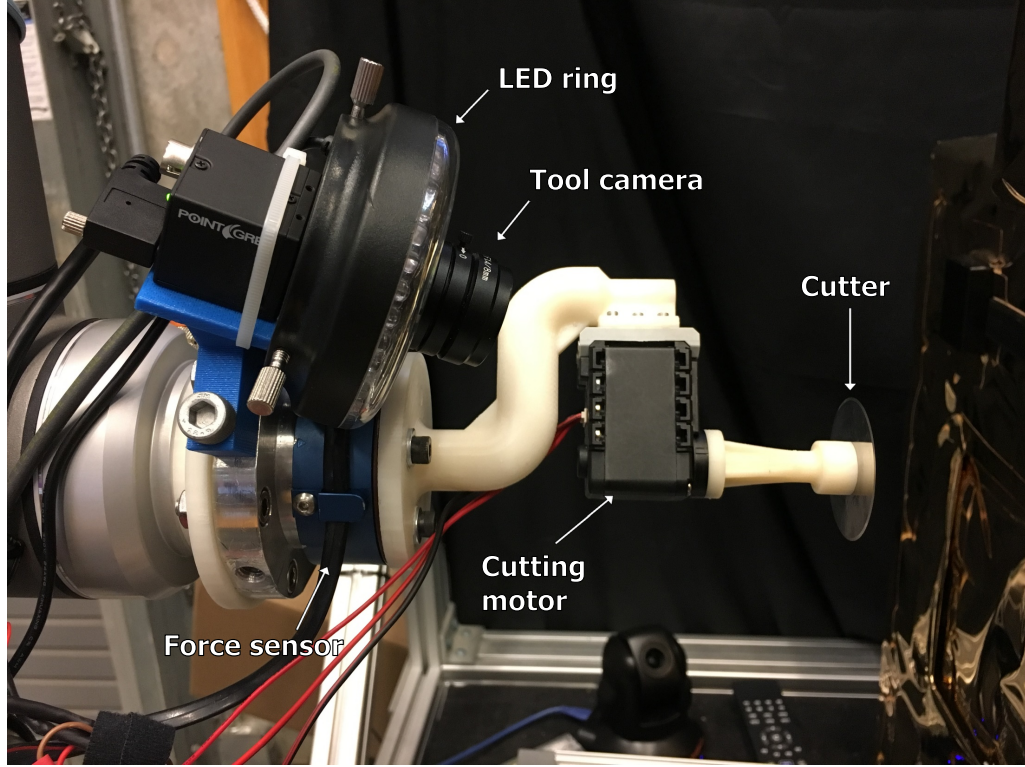


Figure 3.3: Closeup of cutting assembly on UR10 robot.

The JHU laboratory testbed, shown in Fig. 3.2, employs a UR-10 robot manipulator (Universal Robots, Odense, Denmark), equipped with a rotary cutting tool (Fig. 3.3). The tool is composed of a 45 mm circular blade (Arteza, Wilmington DE) that is attached to a Dynamixel MX-12W servo motor (Robotis, Lake Forest, CA). The overall length of the tool is 148 mm and the blade can rotate up to 470 RPM. The tool is mounted on a six axis force/torque sensor (JR3 Inc., Woodland, CA) that measures the forces applied on the blade. A BlackFly (FLIR Integrated Imaging Solutions Inc. BC, Canada) 1080p color camera is also mounted on the UR-10 end-effector to provide a close-up view

of the blade and worksite. The lens of the camera is equipped with a LED ring light.

The testbed also includes one pan-tilt-zoom (PTZ) camera (HuddleCam Downingtown, PA) and one BlackFly camera equipped with a wide angle lens (Rochester, NY) as proxies for cameras to be mounted on the servicer spacecraft platform. Both of these cameras are attached near the base of the UR-10 and directed towards the MLI hat to provide SA views of the workspace, as shown in Fig. 3.2-right.

The master input device is either a keyboard (Fig. 3.2-left) or a haptic arm (Fig. 3.2-center), where the latter consists of the Master Tool Manipulator (MTM) of the da Vinci Research Kit (dVRK) [26]. The dVRK is an open source research platform based on the da Vinci surgical robot [21]. Details of the master consoles are given in the following section.

The software modules of the teleoperation system are implemented in C++, and use the Robot Operating System (ROS) for communication, video capture, and handling of the transformation tree. Graphical user interface (GUI) modules are implemented as RQT (Qt-based GUI development framework for ROS) plug-ins. The dVRK MTM and UR10 robot are controlled using the cisst/SAW software library [11].

3.4 System Configurations

The system supports several human-machine teleoperation interfaces, based on the type of visualization and method of teleoperation. The following sections present two different visualization interfaces: conventional camera view (CAM)

and augmented virtuality (AV), and two different teleoperation interfaces: keyboard (KB) and da Vinci (dV), which lead to the three configurations tested in the experiments (i.e., all combinations except dV+CAM). We note that our prior work [57] demonstrated that dV+AV provided better teleoperation performance than dV+CAM.

3.4.1 CAM: Conventional (Camera) Visualization

For visualization, the teleoperation console our operators typically use provides a combination of video displays—real video and simulation—to guide operators while executing the robotic servicing tasks. The video feeds are captured from an array of SA cameras and from the tool cameras that are designed to provide high quality close-up views of the robotic tools and their immediate surroundings. Although this console provides a 3D simulation of the on-orbit scene, the accuracy of the model is not expected to be high enough to support precision teleoperation. Thus, the operators are trained to rely primarily on the time-delayed video feeds streaming from the servicing satellite.

A similar visualization console was implemented as a special configuration of the augmented virtuality visualization described in Section 3.4.2. Specifically, the time-delayed camera image is placed in the background of the rendered view, the virtual camera is set to the pose of the tool camera, and its intrinsic parameters are set to match the calibrated intrinsic parameters of the real tool camera. The visualization also includes two ring overlays, similar to the ones shown in Fig. 3.4, except overlaid on the camera image. The yellow ring indicates the contemplated pose of the circular cutting blade; when using the keyboard interface, it is updated whenever the operator changes the desired

position in a text input. The red ring indicates the commanded robot position [31] and immediately (i.e., without time delay) begins following the commanded trajectory when the operator initiates a motion.

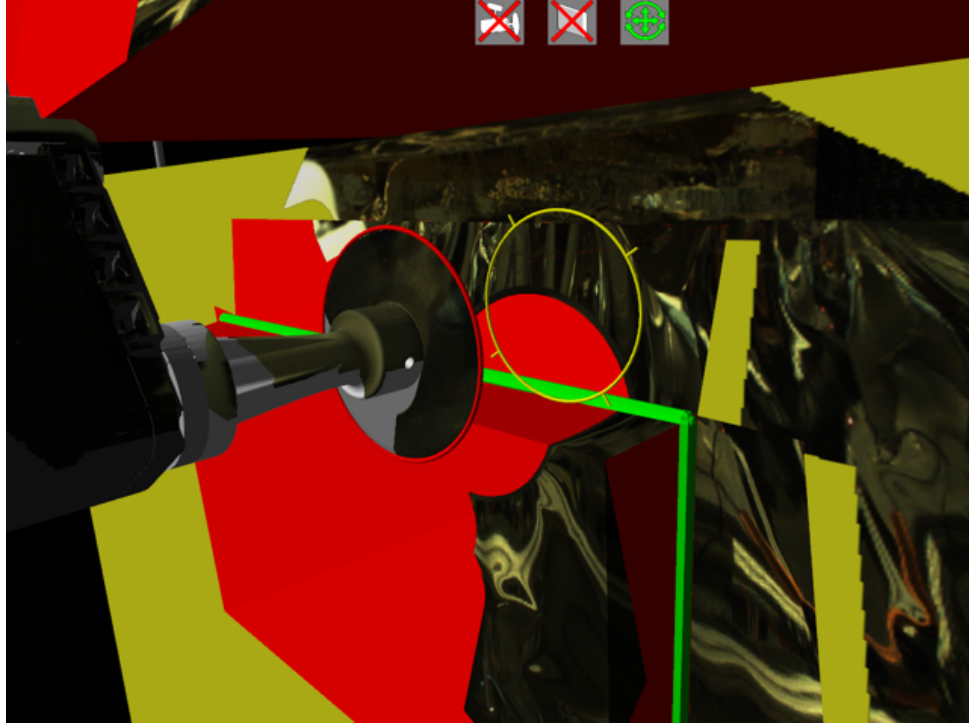


Figure 3.4: Augmented virtuality (AV) visualization of virtual 3D model, augmented by projection of real tool camera image. 3D model includes satellite CAD model (yellow), reconstructed MLI hat (red), and robotic tool. Overlays include commanded robot position (red ring), contemplated robot position (yellow ring), and cut path (green lines).

3.4.2 AV: Augmented Virtuality Visualization

The augmented virtuality visualization is based on our prior work, where we first perform a robotic image survey and then register the 2D survey images to a 3D CAD model of the satellite [56] and reconstruct features, such as the MLI hat, for which there are no accurate pre-existing models [57]. The resulting

3D model, which includes both the registered satellite CAD model and the reconstructed features, can be visualized (in stereo) from arbitrary viewpoints. Other researchers have noted that virtual displays can provide alternative views that could not be achieved with live video [31]. This approach also enables the operator to define a desired cut path with respect to the model. We go beyond virtual reality by projecting the time-delayed video captured from the tool and/or SA cameras onto the 3D model to create an augmented virtuality visualization.

In the present study, we employ an improved version of the visualization system (see Fig. 3.4) that addresses rendering flaws encountered in the previous version and implements new visualization features. Due to performance concerns and feature limitations, the new renderer software does not rely on RViz and was instead re-implemented in C++, using OpenGL. Having the flexibility to fully customize the rendering pipeline enabled a significant performance improvement and the development of advanced rendering features. It also eliminated the need to manually create an image mask for the cutting assembly which, in the previous system, was required to prevent the image of the cutting assembly from being mapped onto the satellite model.

The new renderer performs real-time ray-tracing to project the camera images with correct occlusions on the 3D scene, thereby mapping the image of the tool assembly on the tool model and the image of the satellite on the satellite model, without the need of an image mask. The 3D models in the scene are all wrapped in high resolution texture, and the renderer is capable of adding multiple camera projections to the texture using mosaicking techniques to cover the visible parts of the satellite model with registered real-life camera images.

On top of the static mosaic, the system also maps on the scene the time-delayed video streams captured from the cameras. All this is performed real-time, enabling a more realistic and dynamic 3D visualization.

The new renderer also enables the display of a variety of status indicators in the 3D view. The indicators are rendered as icons and text overlays (see icons at top of Fig. 3.4). When stereoscopic rendering is enabled, the stereo disparity of the status indicators is adjusted to reduce eye strain on the operator. The robot model in Fig. 3.4 is updated by the delayed telemetry from the remote robot. In addition, Fig. 3.4 shows the yellow and red ring overlays that were described in Section 3.4.1 and are available in both CAM and AV modes.

The updated visualization system includes a graphical user interface through which the operator can dynamically configure the visualization features, such as show/hide/reposition picture-in-picture views, set the visibility of texture layers, or change between augmented reality (conventional) and augmented virtuality modes (Fig. 3.5, bottom).

During the experiments performed in AV mode, robot operators controlled the pose of the rendered view (virtual camera) using a 3D mouse or the da Vinci master console.

3.4.3 KB: Conventional (Keyboard/GUI) Control Interface

We created a keyboard/GUI interface for the experiments at JHU that closely replicates the interface typically used by our subjects. This interface, shown in Fig. 3.5, allows operators to input a relative (*delta*) or absolute (*final*) goal pose in Cartesian space (with respect to a specified reference frame), preview

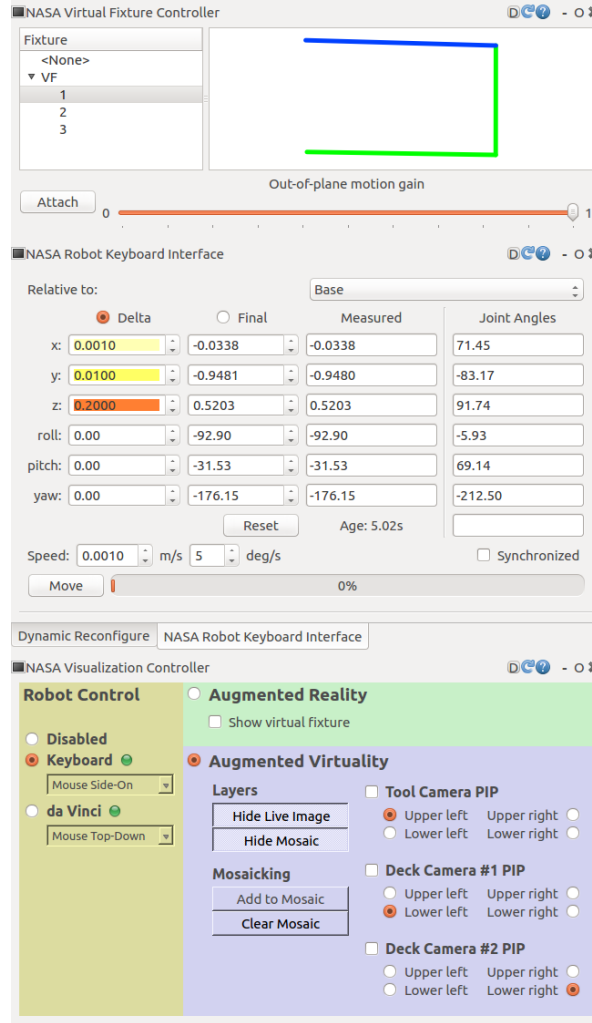


Figure 3.5: Conventional robot interface GUI

the expected result, and execute the command after confirming desired motion. In all configurations, the reference frame can be set to the robot's (stationary) base frame or the tool frame. When using the augmented virtuality interface, the task frame can also be chosen. The task frame is defined relative to the currently-selected segment of the desired cut path, with the x axis in the direction of the path and the y axis in the direction perpendicular to the hat's surface. The task frame is not available in the conventional interface because it

requires an accurate registration between the robot and the satellite, which is not normally available. The interface also displays the latest measured position of the robot, both in Cartesian and joint space.

As the operator adjusts the GUI, the visualization displays a preview of the commanded pose via the yellow ring described in Section 3.4.2. When the operator is satisfied with the command, the *Move* button initiates the motion. A progress bar shows completion percent and the *Move* button is replaced by an *Abort* button. During motion, the preview ring remains stationary, even if the tool frame is selected, allowing the operator to judge the progress towards the goal. Once a motion reaches its goal or is aborted, the preview ring resumes tracking the current value of the inputs.

The operator may also adjust the speed with which the remote robot will move to the goal, with independent control of linear and angular speed. A *Synchronized* checkbox, when checked, limits the faster of the two speeds so that the position and orientation goals are reached simultaneously.

3.4.4 dV: da Vinci Control Interface

The da Vinci interface allows the operator to use the master console of a da Vinci surgical robot (Fig. 3.2-center) to directly teleoperate the satellite servicing robot. The operator holds one of the two MTMs, looks into the stereo viewer, operates the six foot pedals, and optionally uses the space mouse. The da Vinci teleoperation interface extends on the previously reported system [57] with new features and improvements.

Previously, we used a foot pedal to activate teleoperation. In the current system, the operator must pinch the gripper sensor on the da Vinci MTM to

start moving the remote robot, freeing the foot pedal for other uses. The master manipulator begins directly controlling the remote robot after a delay of two seconds, which prevents motions caused by the pinch action from being conveyed to the robot.

We added the ability for the operator to lock the manipulator into rotation-only or translation-only mode using a foot pedal as a three-way toggle. When in one of these modes, the manipulator is still physically able to move in any direction or rotation, but the translation or rotation component of the motion, respectively, is discarded when computing the command velocity. The current state of the lock mode is displayed as an icon in the visualizer.

In addition to manipulating the virtual camera with the space mouse, which previously was the only available method, operators may reposition the virtual camera by moving the master manipulator. While holding the “Camera” foot pedal, the manipulator can be freely translated and rotated. As the operator moves the manipulator, the virtual camera moves to maintain the logical mapping between the manipulator pose and the apparent pose of the robot in the visualizer.

As with the traditional teleoperation interface, the desired cut path is displayed in the visualizer. Operators may opt to use it solely as a visual guide. However, the da Vinci interface also offers the ability to use the desired cut path as a virtual fixture, with non-isotropic gains and haptic feedback. The haptic feedback replaces the force gradient used in the previous system. Operators may *attach* to the currently-selected segment of the cut path, changing its color in the visualizer and activating the virtual fixture. When the virtual fixture is attached, a force or torque is applied in the translational and rotational

directions outside of the desired cut plane that gently pushes the manipulator back into the desired plane. The magnitude of the force is proportional to the displacement from the desired cut plane at a factor of 50 N m^{-1} for translation and 0.1 N m rad^{-1} for rotation. Additionally, a slider in the GUI allows the operator to scale down the velocity in the directions orthogonal to the virtual fixture plane, ranging from the default scale factor of 1 (no scaling) to 0 (disallow motion completely, i.e., a hard virtual fixture).

The traditional (KB) teleoperation interface allows operators to easily align the robot to the desired cut plane before motion by setting the relevant components of the desired pose to zero. We provide a similar ability in the da Vinci interface with an “auto-align” feature, enabled by pressing a foot pedal, which automatically moves the robot to align the end effector with the virtual fixture, first in rotation and then in translation.

3.5 Experiments

We evaluated three distinct system configurations to separately determine the effect of the augmented virtuality (AV) visualization and the direct telemanipulation (dV) interface. The following sections describe the subject population for the study, the experimental setup, and the three system configurations.

3.5.1 Study Subjects

The study used trained robotic teleoperators, who go through extensive training to learn the intricacies of remotely controlling on-orbit robot arms. The training process, which is modeled on that used for International Space Station (ISS) robot operators at Johnson Space Center, requires potential operators to become

familiar with robotic operations and robot kinematics. New teleoperators begin by observing ground based testing and acting as a safety operator. In addition to training on the robotic system, teleoperators also train for a given task by first using ground industrial robots before performing the task on the ground unit of the flight robot. This progression ensures that teleoperators for servicing missions are experts in both the robotic system they are operating as well as the task they are performing.

3.5.2 Experimental Setup

The experimental setup consists of a control station and a servicing platform, as shown in Fig. 3.2. The control station is either a keyboard and computer displays (KB) or the da Vinci master console (dV), as described in Section 3.4.

The mock satellite is constructed from 80/20 aluminum bars and panels wrapped in a layer of Mylar [56]. Not including its solar panel, the satellite is a box of size $24 \times 24 \times 36$ inches. MLI hats were manually assembled to replicate a space-grade hat at a reasonable cost. The blanket used for the hats is composed of 21 alternating layers of 0.5 mil (0.013 mm) polymer film (McMaster 8567K102) and fine tulle. The layers are then placed between two layers of 1 mil (0.025 mm) metalized PET (Mylar) film (CS Hyde 48-1F-1M). Kapton tape (McMaster 7648A34) is used to assemble the blanket and to fold the corners of the hats. The fabrication of each hat takes approximately 3 hours for a trained person. Because the MLI is folded multiple times at the corners, the cutting blade must cut through approximately 100 layers of film, tulle, and tape at the thickest point.

The initial preparation consists of first calibrating the camera intrinsics and

extrinsics, and then performing a robot-to-satellite registration, as described in [56]. Specifically, the robot acquires a set of 2D images from multiple poses, which are then registered to a 3D model of the rigid parts of the mock satellite. The resulting camera calibration and robot-to-satellite registration are used for all trials. Before each trial, a new hat is mounted in approximately the same location on the satellite. An image survey is performed, using predefined viewing angles, to manually reconstruct the hat’s geometry, as described in [57]. A desired cut path is defined in the same relative location on each reconstructed hat model. Each trial begins with the robot in the same position relative to the mock satellite.

During trials, operators sat out of visual range of the robot, relying only on the time-delayed camera feedback for visualization. In addition, all operators wore noise-canceling headphones, through which music or white noise was played, to prevent them from hearing real-time (i.e., undelayed) audio feedback, such as changes in the cutting motor sound. Figure 3.2-left shows an image from one trial. Operators completed a NASA TLX survey after each trial and a post-experiment survey after the last trial. The survey asked them to rate the difficulty of use for each system configuration and provided an opportunity for free-form feedback.

The order of trials was fixed to introduce no more than one new feature at a time. Each operator first performed the conventional (KB+CAM) trial, which emulates their familiar teleoperation interface, though with different hardware and software. Next, the augmented virtuality (AV) visualization was introduced, while keeping the familiar keyboard teleoperation interface (KB+AV).

Finally, the AV visualization was kept and the da Vinci teleoperation interface was introduced (dV+AV). Operators were allowed to practice with each configuration prior to beginning each trial.

3.5.3 Conventional Teleoperation (KB+CAM)

The first experimental condition used conventional teleoperation. The visualization was used in Augmented Reality mode, using the stereo display as a 2D monitor. The red “commanded” and yellow “preview” rings were overlaid on the tool camera image. Operators were free to enable picture-in-picture overlays of the deck cameras or to display the deck cameras on a second monitor. The robot was controlled using the traditional robot interface with the ability to control in task frame disabled, reflecting the fact that the conventional system does not provide a sufficiently accurate robot-to-satellite registration to define a task frame.

3.5.4 Conventional Teleoperation with Augmented Virtuality (KB+AV)

The second experimental condition was designed to measure the effect of the augmented virtuality visualization. The visualizer was used in Augmented Virtuality mode on the 3D monitor, as shown in Fig. 3.2-left. The “commanded” and “preview” rings and the desired cut path were displayed in the virtual environment. Operators were able to move the virtual camera with the space mouse or use buttons on the space mouse to cycle between predefined views. An additional GUI element allowed operators to select one of the three segments of the cut path to define the task frame. Operators used the conventional robot

interface, but with the task frame enabled.

3.5.5 da Vinci Teleoperation with Augmented Virtuality (dV+AV)

The final experimental condition was designed to measure the effect of the da Vinci teleoperation interface. The visualizer was used in Augmented Virtuality mode viewed through the da Vinci master console's stereo viewer. In addition to all elements from the previous experimental condition, the visualizer displayed icons to communicate the current internal state of the da Vinci robot interface. Operators were able to move the virtual camera using the space mouse or the da Vinci MTMs. On request from the operator, an experimenter operated GUI controls which are not currently exposed to the da Vinci interface, such as selecting and attaching to desired cut paths. Ultimately, these controls would either be implemented within the da Vinci interface or managed by a second robot operator.

3.6 Results

All five of our subjects completed the experiments with all three configurations. Table 3.1 shows the time that each operator spent operating each feature of the system during each trial. The Moving Robot category shows the time each operator spent sending commands to the remote robot. The Adjusting View category shows the time each operator spent manipulating the virtual camera in trials with augmented virtuality visualization. The Clutching category, only applicable to the da Vinci interface, shows the time spent repositioning the

Table 3.1: Time breakdown, seconds (Conv. configuration is KB+CAM)

		Moving Robot	Adjusting View	Clutching	Auto- Aligning	Total
Subject 1	Conv.	537.0	—	—	—	2358.5
	KB+AV	577.8	305.3	—	—	3139.4
	dV+AV	590.8	115.3	16.6	14.8	1067.2
Subject 2	Conv.	185.0	—	—	—	1088.1
	KB+AV	327.0	305.3	—	—	983.0
	dV+AV	509.8	316.7	15.1	0.0	776.5
Subject 3	Conv.	344.7	—	—	—	2088.5
	KB+AV	578.8	112.6	—	—	1639.0
	dV+AV	470.1	127.9	16.7	43.9	1008.7
Subject 4	Conv.	576.7	—	—	—	3154.8
	KB+AV	570.7	176.4	—	—	1912.8
	dV+AV	706.5	160.4	10.4	15.5	1162.5
Subject 5	Conv.	307.4	—	—	—	4620.2
	KB+AV	460.0	724.5	—	—	2846.6
	dV+AV	629.6	124.3	9.4	11.4	1298.0

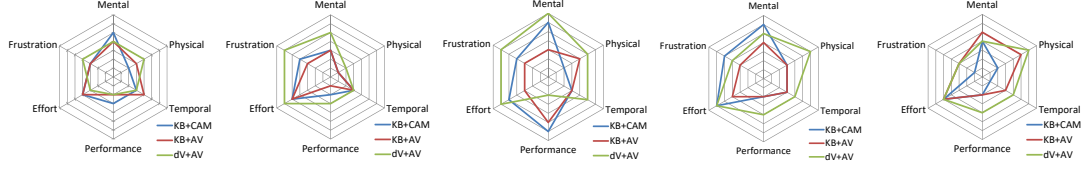


Figure 3.6: TLX survey results for Subjects 1-5 (left to right). Values range from 1 to 7, with 7 representing the greatest burden in that category.

MTM. The time was insignificant compared to the total experiment time, suggesting that subjects were not overly constrained by the limited workspace of the MTM. The Auto-Aligning category, also only applicable to the da Vinci interface, shows the time spent using the auto-align feature. The results show that all but one operator chose to use that feature. Finally, the total time to cut the two sides of the hat is shown. For four out of the five subjects, the conventional keyboard/GUI interface with the augmented virtuality visualization took less time than with the conventional visualization. However, all five subjects completed the task in the shortest time using the da Vinci interface. This is expected, as the da Vinci’s direct teleoperation does not allow the chance to preview the result of a command, eliminating the time spent confirming that the correct command will be issued. For this reason, the lower time does not necessarily indicate an improvement.

Figure 3.6 shows the results of the NASA TLX survey completed by each participant after each trial. It indicates that the augmented virtuality interface caused less stress (frustration) than the conventional visualization for three operators, a better self-assessed performance for three operators, and that all operators found it less or equally as difficult. These results are consistent with the faster completion times shown in Table 3.1 for the KB+AV configuration for four out of the five subjects. Three out of five operators found the da

Vinci teleoperation interface considerably more frustrating than either trial with the keyboard interface, and four operators thought it was more or equally as difficult. This is not surprising, given that it is an unfamiliar interface for these operators. This was also evident in the post-experiment survey, Table 3.2, where operators rated the difficulty of each system configuration on a scale from 1 (very easy) to 5 (very hard). All five operators selected the KB+AV configuration as the easiest or as one of the easiest and four operators rated the dV+AV configuration as the hardest.

Table 3.2: Post-Experiment survey results (1 = very easy, 5 = very hard)

Condition	Op-1	Op-2	Op-3	Op-4	Op-5	Mean
KB + CAM	4	3	3	3	3	3.2
KB + AV	3	2	2	2	3	2.4
dV + AV	3	4	5	4	4	4.0

In order to evaluate success of the cut, the total number of layers and successfully-cut layers were measured. Figure 3.7 shows the number of layers cut compared to the number of layers present. Note that the geometry of the hat construction causes a significant increase in the number of layers that must be cut at a corner. Table 3.3 shows the rate of success and the degree of failure in terms of the number of layers cut. We see that the KB+AV configuration led to the highest percentage of complete and acceptable cuts. The results also indicate that, despite the increased number of layers, the corners typically saw more success than the straight sides. We attribute this to the additional structural integrity of the hat, which restricts the layers from spreading apart as much as on the sides.

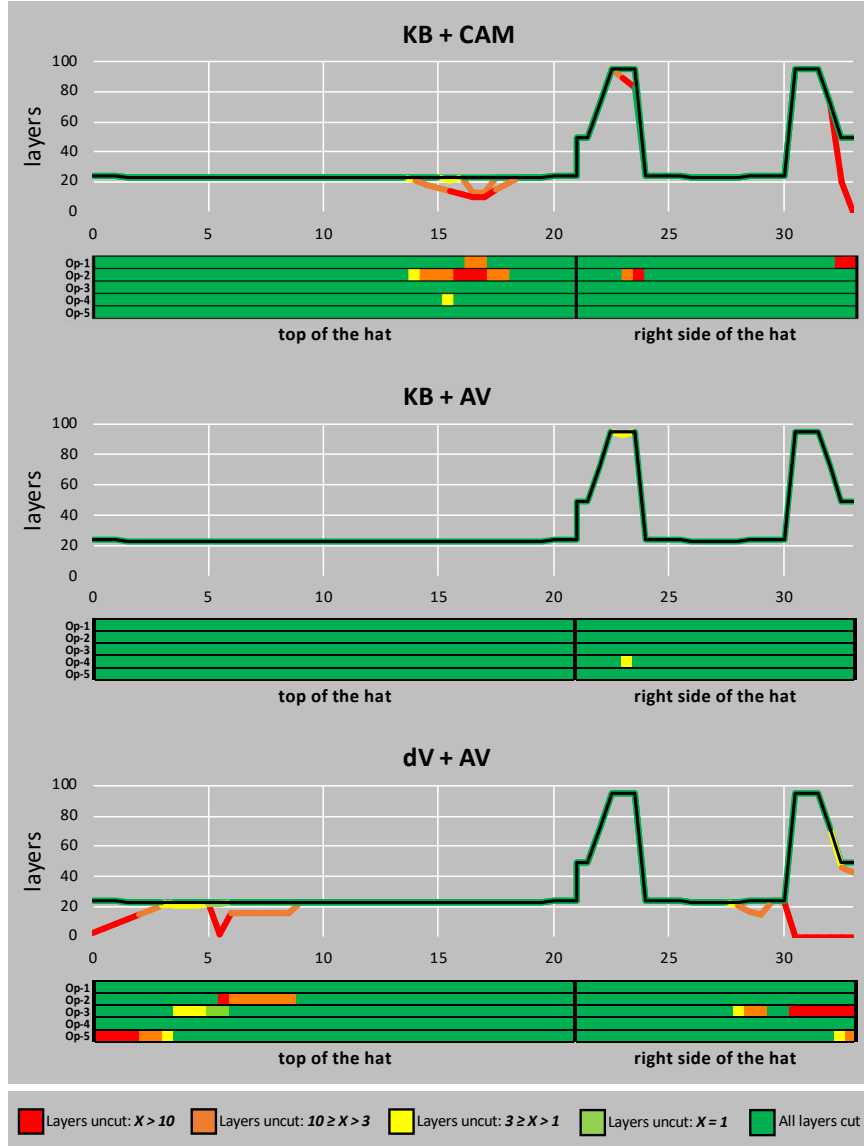


Figure 3.7: Visualization of the number of layers successfully cut in all MLI cutting trials. Horizontal axis represents cutting progress [cm], starting at the top of the hat then continuing on the right side. The thin black lines indicate the number of layers that need to be cut, and the thick colored lines show the number of successfully cut layers for each trial. A single sheet of MLI consists of 23 layers, but there are as many as 95 layers at the corners where the MLI is folded and taped multiple times. The colored horizontal bands under the charts show the number of layers cut for each trial. There are 5 bands for each task, representing the 5 operators.

Table 3.3: MLI cutting success rate

Layers not cut	KB+CAM	KB+AV	dV+AV
All cut	95.29%	99.71%	91.18%
X=1	0.00%	0.00%	0.59%
$3 \geq X > 1$	0.59%	0.29%	1.76%
$10 \geq X > 3$	2.35%	0.00%	3.24%
X>10	1.76%	0.00%	3.24%

3.7 Discussion and Conclusions

We developed an augmented virtuality interface to support ground-based teleoperation of robots on orbit for satellite servicing tasks, subject to communication delays of several seconds and challenging visualization of the remote scene. The system was tested with trained robot operators and the results indicate that the augmented virtuality visualization can provide benefits, including reduced execution time and lower task load, compared to the conventional visualization. One likely explanation is that the augmented virtuality system provides teleoperators the ability to choose arbitrary views of the robot workspace, which can significantly improve their situational awareness. This reduces risk and makes teleoperated servicing tasks more efficient and reliable. Although satellite servicing resembles telesurgery, our experiments also revealed that the trained operators preferred the conventional keyboard/GUI interface over the da Vinci master console. This is likely due to their extensive training and familiarity with that interface, but also to the nature of the MLI cutting task, where most motions can be easily expressed within the task frame. The da Vinci master console might be more effective for other tasks, especially those involving complex motions that cannot be as easily commanded via a keyboard.

Chapter 4

Interactive Planning and Supervised Execution (IPSE)

In this chapter we describe a new teleoperation interface, Interactive Planning and Supervised Execution (IPSE), designed based on the results of the study described in Chapter 3. We describe the lessons learned from that study, review previous work in the area of visual robot programming, describe the implementation of the IPSE core and two user interfaces to IPSE, and present the results of a study comparing those interfaces to a conventional keyboard/GUI interface. This chapter is based on the work described in [39]. Balazs Vagvolgyi contributed the registration and reconstruction procedure described in Section 4.2.1.

4.1 Background and Motivation

NASA is developing new technologies around satellite servicing [36], with plans to use ground-based telerobotic control for proximity operations. These operations include cutting the multilayer insulation (MLI) that encases the satellite, cutting retaining wires, removing the cap on the fill/drain valve, attaching the

refueling hose, transferring fuel, and then reinstalling the cap and applying an MLI patch. The overall procedure requires both coarse motion to transit to/from the tool stowage area and fine motion to perform proximity operations.

The design of the interactive planning system was primarily motivated by the results of the study described in Chapter 3, in which we recruited five trained robot operators to evaluate several teleoperation interfaces for the task of cutting the multilayer insulation (MLI) “hat” that covers the satellite fill/drain valve. Specifically, we compared our recently-developed augmented reality (AR) visualization [57] to the conventional camera visualization and the da Vinci master console to the conventional keyboard/GUI control interface (Fig. 4.1). The results showed that the trained operators unanimously preferred the AR visualization, but only when paired with the conventional keyboard/GUI control interface. The da Vinci control interface was the least preferred; in fact, operators were more willing to sacrifice the AR visualization in order to retain the keyboard/GUI interface. This was surprising because the trained operators consistently performed the task faster with the da Vinci interface.

Post-experiment interviews with the trained operators revealed two primary reasons for their preferences. First, the task of cutting the MLI did not require the dexterity offered by the da Vinci master console, since the cutting operation primarily involved moving in a straight line. They hypothesized that other operations, such as large motions between the worksite and the tool stowage area, would be likely to benefit from the ability to command more complex motions. Second, operators opined that performing the task without error is more important than performing the task quickly. Although task failures could be easily remedied in our ground-based setup, the trained operators followed

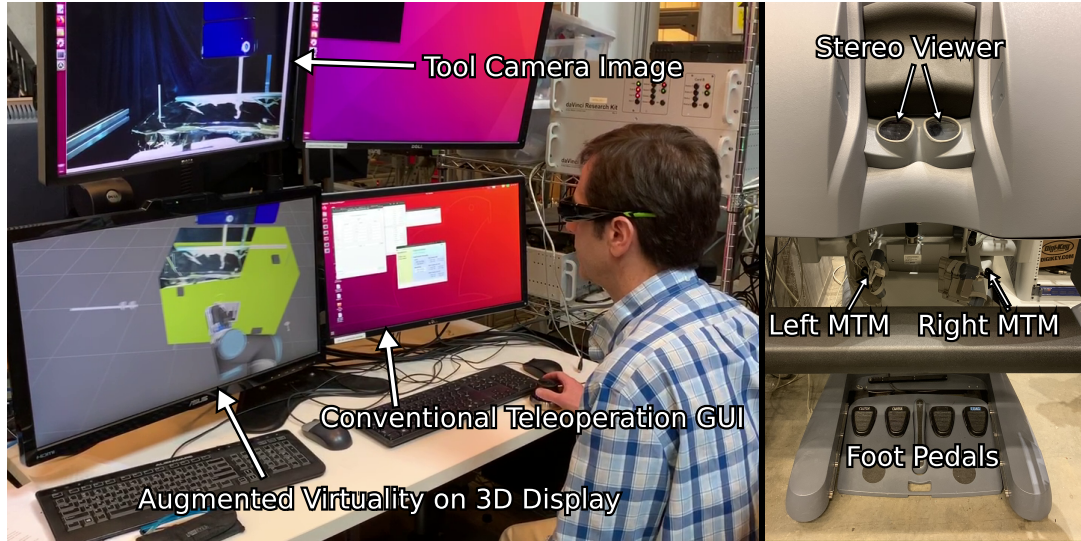


Figure 4.1: Teleoperation control interfaces: Keyboard/GUI (left) and da Vinci master console (right); left image also shows augmented virtuality (AV) visualization on 3D monitor (lower left).

their training to avoid failure. The keyboard/GUI interface provided a simple method to preview a motion before executing it; specifically, a yellow ring was overlayed on the camera images to show the position the cutter would reach if the operator pressed the “Move” button. In contrast, the da Vinci interface only allowed the operator to move the robot or to adjust the virtual camera (for the AV visualization) – there was no ability to preview motions. This revelation led to the development of an interactive planning capability, presented in this chapter, where the operator first plans in a virtual environment, previews the plan, makes adjustments if necessary, and then supervises execution of the motion with the ability to pause and replan/resume at any time.

The experiments reported in this chapter address the attachment of the refueling hose as a representative task for on-orbit telerobotic servicing of spacecraft. We assume that the robot has just removed the cap on the fill/drain valve and

must first move to the tool stowage area to acquire the hose manipulation tool, then move to grasp the hose and mate it with the fill/drain valve. We report the development of an interactive planning system, where the operator plans the motion in a virtual environment, created from real-time modeling of the actual environment [57], previews and adjusts the plan in the virtual environment, and then supervises its execution in the real environment.

This approach resembles offline robot programming, which has been used for decades [38]. However, offline programming is unable to respond to geometry which is unknown a priori, or to geometry which may change unpredictably during the operation. Another strategy which has been used to mitigate the effects of time delay is to have operators command the robot using high-level task goals, which generalize to variations in the environment, and the remote robot determines the exact details of the operation autonomously [25]. This relies on the autonomous system’s ability to execute a high-level goal without error, which creates unnecessary risks with high penalties in a satellite servicing environment. Likewise, systems which predict the response of the remote robot and environment to simulate real-time feedback, such as in [5, 54], are reliant on accurate simulation of the robot and environment.

More recent related work involves the use of mixed/augmented reality for visual programming of robot motions. In [42], a user can plan paths as a series of waypoints in an augmented reality (AR) environment, preview and edit the paths, and then execute them either autonomously or by allowing the user to control progress through the path. In [19], the user similarly builds a path out of primitives, visualizes the final path, and then executes it. [19] also evaluates the mixed reality interface against a 2D baseline interface among a

non-expert population, finding that the mixed reality interface was preferred. We implement a waypoint-based planning system with both a mixed reality and 2D interface and evaluate each on a high-risk task in the presence of time delay and with expert users.

4.2 System Description

An operation using our proposed system begins with a registration and reconstruction procedure, to locate and build the objects in the virtual (simulated) environment. The operator then creates a motion plan using the interactive planning system and executes the plan with supervised execution. The latter two steps are repeated until the task is complete. The system is implemented with a set of programs using the Robot Operating System (ROS) [43].

4.2.1 Registration and Reconstruction

The operator of our system views and manipulates the work-site via a computer simulation of the scene that contains the relevant objects, robots, and equipment. The simulation is based on design drawings, robot kinematics, camera images, and other sensor data. We did not include dynamics in our simulation because we assume that the scene contains rigid objects that do not collide with each other.

The simulation of most of the robotic servicing platform is reliable and accurate as it is based on well known manufacturing design and robot kinematics, however, the simulation of the object to be manipulated (in this case the satellite to be refueled) is in an unknown position and orientation with respect to

the servicing platform. The geometry of the object may also be just an approximation of the real object if the original manufacturing design is not entirely available.

We used our Workspace Registration Tool software [57] to perform object registration and to reconstruct the geometry of unknown but relevant features of the object. The registration and reconstruction process requires the robot operator to perform a visual survey of the object by taking several images of it from multiple positions using the tool camera mounted on the robot’s end effector. Visual landmarks which match reliably known features in the object’s CAD model are used for registration, while landmarks that are different from the CAD model, such as the MLI hat, are reconstructed using 3D triangulation and added to the model.

The result of the registration and reconstruction process is a 3D simulation of the scene that accurately represents the physical location of the relevant objects, robots, and equipment in the remote work-site.

4.2.2 Interactive Planning

The Interactive Planning component allows the operator to design a motion plan using one of multiple co-existent user interfaces. At any time, the operator can preview the resulting robot motion and edit the plan until the motion is satisfactory.

A motion plan consists of a series of waypoints, as shown on the left in Fig. 4.2. Each waypoint represents an intermediate destination in the motion plan. A motion planning engine, using the MoveIt! planning framework [50], plans a

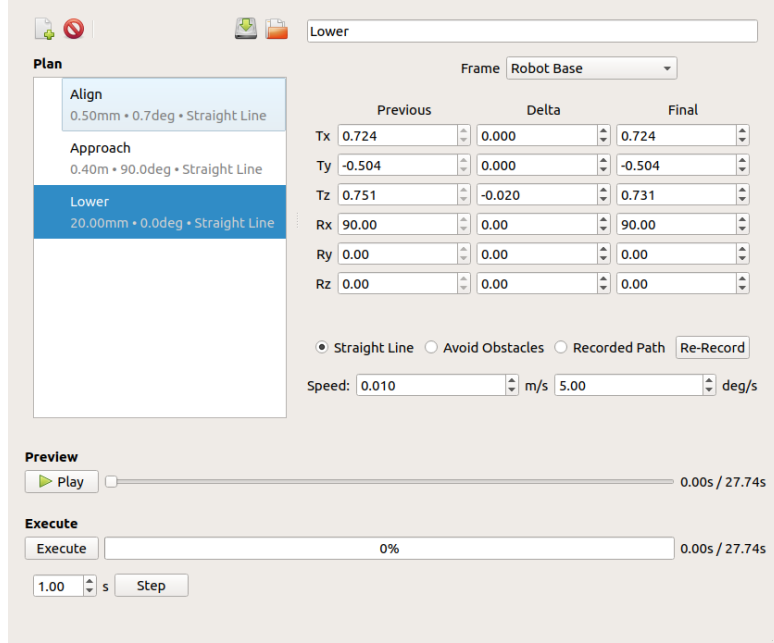


Figure 4.2: The custom user interface for the 2D planning interface allows the operator to view and edit the waypoints as delta (relative to the previous waypoint) and final (relative to the fixed frame) values.

trajectory to connect each waypoint’s destination pose with the final configuration of the previous waypoint’s trajectory, with the first waypoint connected to the robot’s current configuration. The resulting trajectories are collision-free when possible and marked as invalid when a collision cannot be avoided. Invalid trajectories cause execution to be disabled and the waypoint marker (Fig. 4.3, see Sec. 4.2.2.1) to be displayed in red.

The operator may configure each waypoint to use a straight-line path, which causes the end effector to follow a straight line in task space; to avoid obstacles, in which case the motion planner may select any collision-free path; or to follow the same task-space path that the operator followed to move the waypoint marker. Each waypoint also has an independent set of desired speeds, both linear and rotational. The trajectory is initially timed to obey pre-configured

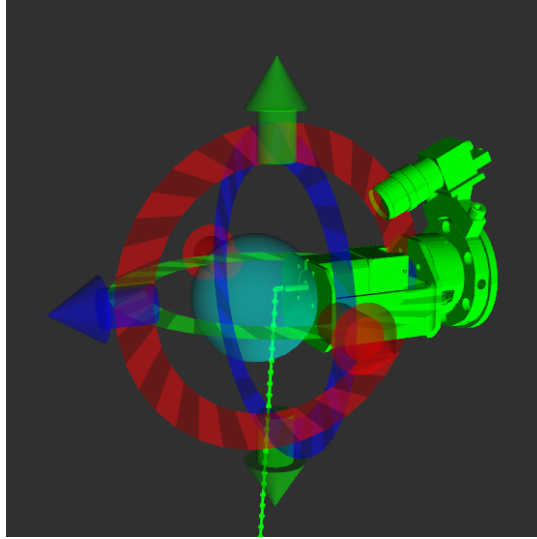


Figure 4.3: A waypoint marker displays as a colored end-effector with additional controls to enable moving it in all six degrees of freedom. A dotted line shows the path the end effector will take to reach this waypoint.

joint-level limits and then the duration of each trajectory segment is scaled up if necessary to ensure the segment obeys both limits on end-effector speed.

The planner is independent from any user interface. It communicates with any number of interfaces simultaneously using a ROS topic API, and changes made in any interface are immediately reflected in all connected interfaces. We implemented two user interfaces: a 2D mouse-and-keyboard interface composed of a custom GUI with RViz for visualizations and a 3D interface operated with the master console of a da Vinci surgical robot.

4.2.2.1 2D Interface

The 2D interface uses a custom configuration of RViz, the standard robot visualizer bundled with ROS, in combination with a custom Qt interface called the Planner GUI (Fig. 4.2). The Planner GUI displays the list of waypoints in

the current plan and allows the operator to edit the selected waypoint. In addition to naming each waypoint, the operator can input the goal pose as either a “delta” from the start pose or as a “final” absolute pose, and modifying either column will update the other. The operator can choose the reference frame in which to display these poses from a list, which can include any frame whose pose is known and static during an entire planning operation. If the pose of an object of interest to the task is known, the operator can choose that frame and align each degree of freedom independently by entering 0 in the appropriate input of the Final column.

As waypoints are added and edited, each one is represented in RViz as a colored end effector, as shown in Fig. 4.3. The end effector is green when the plan is valid, red when it is invalid, and white when the plan has not been computed. The latter case occurs briefly whenever an edit is made as well as whenever a previous waypoint was invalid, as each waypoint’s plan depends on the previous. Clicking each waypoint marker selects the corresponding waypoint. The selected end effector is surrounded by interactive markers which allow the waypoint to be translated along each axis (arrows), rotated about each axis (rings), or translated in the plane perpendicular to the virtual camera (sphere). When a trajectory has been computed between each pair of waypoints, it is shown as a dotted line tracing the path of the end effector tip.

The Planner GUI also allows for choosing between path types and setting the maximum speeds. Below the waypoint edit areas, the Play button and scrubber allow the operator to preview the current plan, either by playing it back in real-time or scrubbing through as in a typical media player. The “preview robot”, a partially-transparent version of the actual robot, animates through

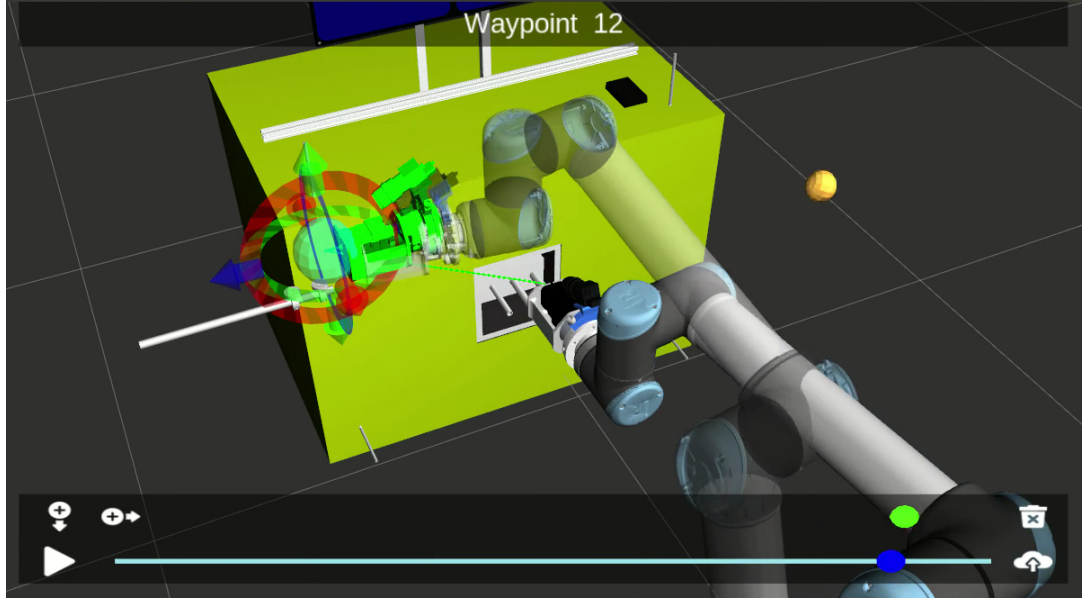


Figure 4.4: The 3D Interface displays the virtual world in 3D, with a GUI overlay to replicate the most common features of the Planner GUI.

the planned path. The preview and execute functions use the same stored trajectory, so the operator can be assured that the execution will follow the exact same configurations as the preview. The Planner GUI also includes the Supervised Execution controls, described in Sec. 4.2.3.

4.2.2.2 3D Interface

The planning system can also be controlled with a 3D interface, shown in Fig. 4.4, which uses the da Vinci master tool manipulators (MTMs) to control the same path. Each of these two MTMs is a 7DOF compliant robot arm which is used to track the operator's hands in space and also includes a gripper which may be pinched with the user's thumb and forefinger to command actuation of some tool. In our system, each of the MTMs controls a 3D cursor, represented in the virtual world as a sphere. Pinching the MTM's gripper acts as a click.

The operator looks into the da Vinci stereo viewer at the same virtual world as shown in RViz, here rendered in 3D. The 3D cursor can be used as a mouse to select and move waypoints in the same way as with the 2D interface, except that with a 6-DOF input device the sphere can be used to position and orient the waypoint anywhere in the workspace. The addition of 3D visualization and 6-DOF manipulation greatly enhances the operator’s ability to position the waypoints, which is especially important when recording paths.

The 3D interface includes an overlay GUI to give operators access to the most-used features of the Planner GUI without moving from the da Vinci console. When the 3D cursor moves behind one of the buttons visible at the bottom of Fig. 4.4, it transforms into a 2D cursor of the same color and allows clicking the buttons, which provides the ability to add a waypoint at the end of the plan, convert the current preview location to a new waypoint, delete the current waypoint, and preview and execute the plan. The 2D cursor also manipulates the preview scrubber, as illustrated by the lower interactive cursor in Fig. 4.4.

4.2.3 Supervised Execution

When the operator is satisfied with the planned trajectory, they may execute it on the remote robot. The Execute button in the Planner GUI (see Fig. 4.2) transmits the current trajectory to the remote robot immediately, but due to the communication delay, the robot does not appear to move until one round-trip time (RTT). Once feedback is available, the progress bar fills to indicate progress through the planned trajectory and the opaque robot shown in the virtual world follows the robot telemetry feedback.

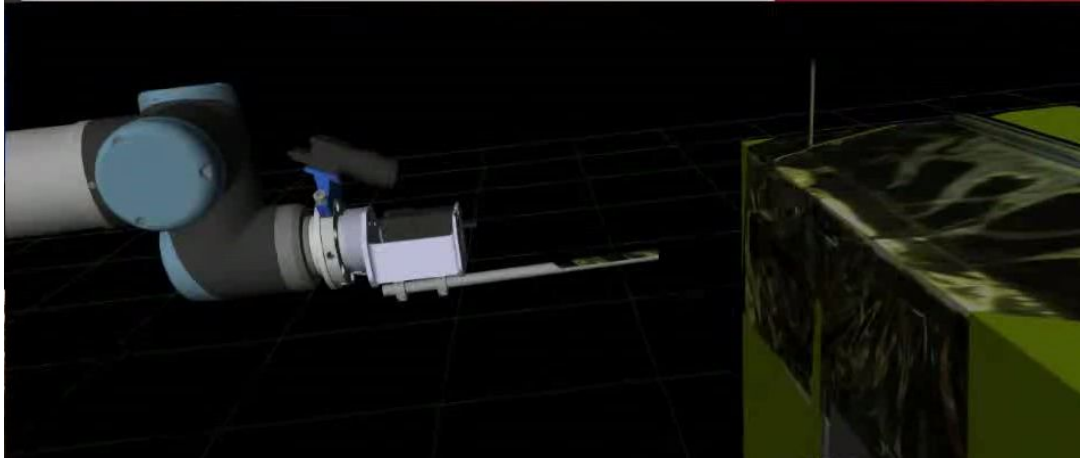


Figure 4.5: Augmented Reality (AR) visualization: the tool camera image is projected onto the virtual environment (in this case, both the robot tool and satellite).

During execution, the operator may watch the robot’s progress in an Augmented Reality (AR) visualization environment (Fig. 4.5). This visualization displays the robot and environment as in RViz, but also augments the virtual models with a projection of the image from the robot’s tool camera. The registration procedure in Sec. 4.2.1 is sufficiently accurate that each area of the camera image is projected onto the virtual model of the object it belongs to with minimal error. The projection improves the operators’ situational awareness and ability to judge the completion of the task by transforming the 2D image into 3D textured objects; furthermore, it helps operators recognize inaccuracies in the simulation which informs them when more caution is required. In principle, this visualization and the visualization used during planning could be implemented in the same software tool, but in our implementation they are separate due to implementation limitations.

If the operator judges that the trajectory is inaccurate, they may pause the trajectory from the Planner GUI. However, due to the time delay, the robot

will continue moving 1 RTT past its apparent position before the pause command is received. For cases where this delay could damage the mission, the operator can instead use the Step functionality to execute only a small portion of the trajectory at a time, allowing the operator to reassess the trajectory's success between each step. The step function truncates the trajectory to the specified time and sends only the truncated portion to the robot. The robot's progress through the trajectory is retained through multiple uses of the Step and Execute buttons so the trajectory may be resumed from the robot's current position. Before sending a trajectory to the robot, whether by step or by full execution, the portion to be sent is re-timed (including post-processing to satisfy maximum end effector speed) to ensure smooth acceleration and, if the trajectory is not paused, deceleration. This re-timing may lengthen the actual execution duration, although in practice the difference is insignificant at the speeds typical during a satellite servicing operation.

When the trajectory requires correction, the operator may choose to start a new motion plan, at which point the trajectory progress is reset. If the difference is small, the operator may also choose to use a space mouse as a joystick to adjust the robot's pose, which is also subject to communications delay. The operator can then re-compute the trajectory for the remaining portion of the motion plan using the new start position and resume normal execution.

For technical reasons, Supervised Execution was only implemented in the 2D interface.

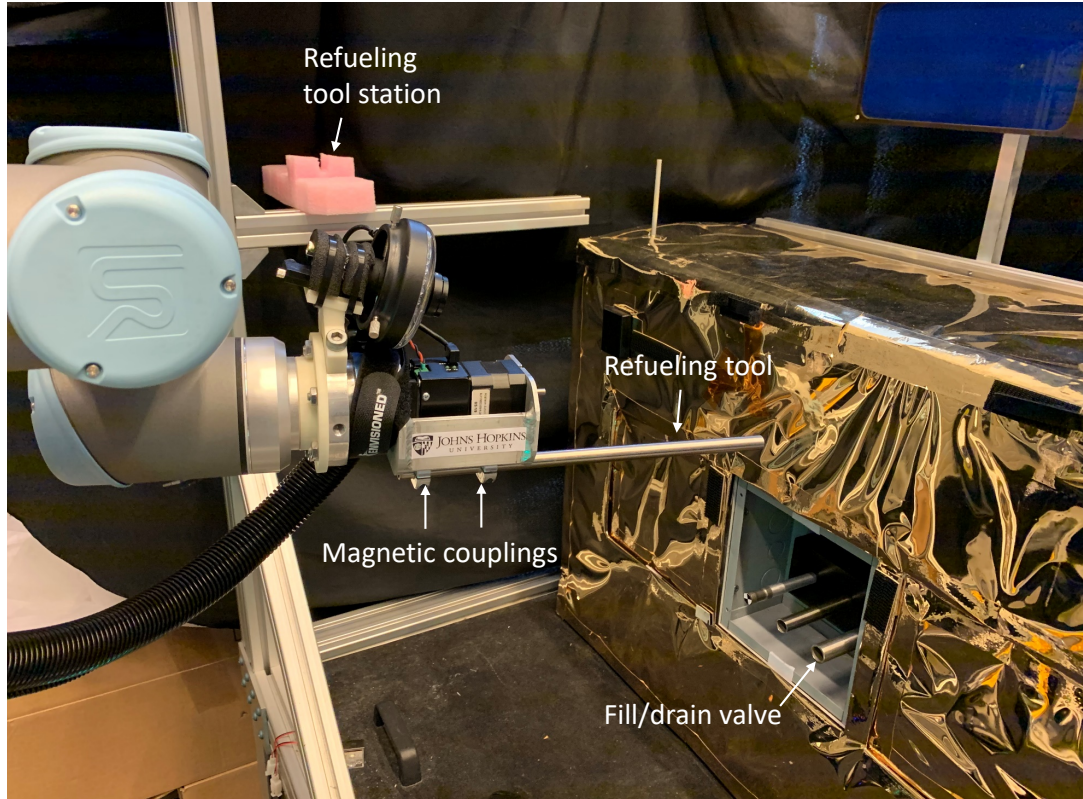


Figure 4.6: The space-side setup includes a mock satellite (right), mock servicing robot (front left), and refueling tool station (back left).

4.3 Experiments

4.3.1 Experimental Setup

The experimental setup is based on that described in Section 3.5.2, using the same robot, cameras, and mock satellite. A set of three pipes of varying diameters is added, fixed within a cavity in one side of the satellite; one of these pipes is selected to represent the satellite fill/drain valve. A tool stowage area, visible behind the mock servicing robot in Fig. 4.6, is attached to the same structure as the mock satellite. The motor in the tool mounting assembly is not used in this experiment, and instead a set of magnets to which passive tools can be

attached is added. The refueling tool, a section of pipe affixed to magnets, is shown attached to the tool mount. The outer diameter of the refueling tool fits snugly into the smallest pipe on the mock satellite. At the base of the servicing robot, not visible, are two deck cameras with views of the pipe cavity on the mock satellite.

While operating the robot, the operator sits at an operator station which includes several monitors, a keyboard and (2D) mouse, a (6D) space mouse, and a da Vinci master console. A 3D monitor, used with active shutter 3D glasses, displays the Augmented Virtuality Visualizer described in Sec. 4.2.3. A standard 2D monitor displays the Planner GUI and RViz, described in Sec. 4.2.2.1, and an additional 2D monitor displays the unaltered tool camera images.

4.3.2 Experimental Task

An experiment begins with the mock satellite already registered using the Vision Assistant software tool described in Sec. 4.2.1. The robot begins in a standard location not near the mock satellite or tool stowage area and the refueling tool begins in the tool stowage area. During the experiment, the operator must first command the robot to the tool stowage area and lower it onto the tool to engage the magnetic attachment. Once at least one magnet attaches, an experimenter records whether the tool was aligned correctly. The operator must then command the robot to move the tip of the refueling tool inside one of the pipes on the mock satellite (which represents the fill/drain valve). The experiment is complete when the operator believes the tool tip is inserted at least 3 cm inside the pipe, or failed if the operator believes it is not possible to insert the tool tip into the pipe (for example, if the tool is knocked off the

magnetic mount).

4.3.3 Conditions

Each operator performs the experimental task three times. In all three conditions, the operator looks at the Augmented Virtuality environment on the 3D monitor and the unaltered tool camera image for feedback. Under the first condition, “Conventional teleoperation”, the operator may use a joystick to move the robot or enter a pose command into a simple GUI. The commanded pose is indicated in the Augmented Virtuality environment with a small marker.

Under the second condition, the operator uses the 2D Interactive Planning with Supervised Execution (Sec. 4.2) interface and may also use the joystick. In the final condition, the operator uses the 3D Interactive Planning interface to plan, but the 2D interface for Supervised Execution, as the 3D interface does not provide any way to view the camera image.

Although the Interactive Planning was designed to enable the operator to switch between the 2D and 3D interfaces at will, we decided to evaluate them in separate trials for two reasons: (1) to compare their effectiveness in performing the task, and (2) to ensure that each interface was actually used (otherwise an operator could choose to only use one of them). Note that because the 3D interface did not fully support all functionality, operators were allowed to use the 2D interface in cases where the 3D interface did not provide the necessary functionality (e.g., to modify the motion speed or to view the Augmented Virtuality visualization during execution).

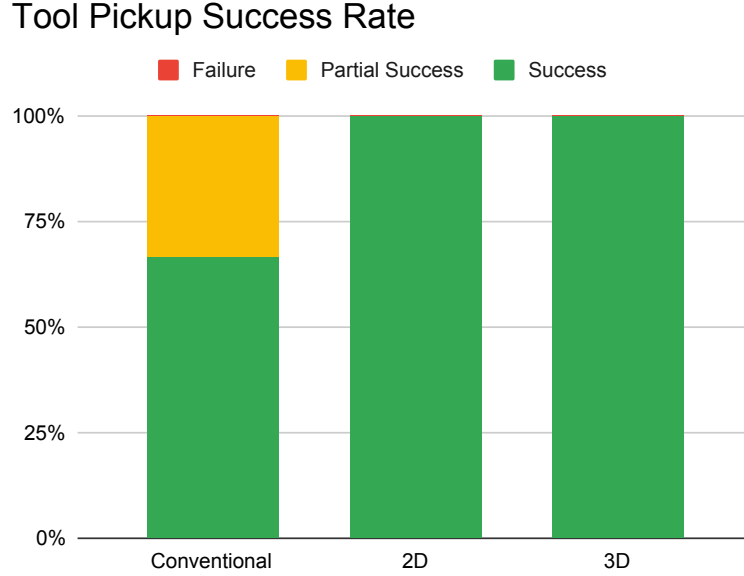


Figure 4.7: In the tool pickup task, no operator failed in any condition and both the 2D and 3D Interactive Planning cases saw complete success.

4.4 Results

Six operators participated in the user study (JHU HIRB 00000701). Operators were recruited from a population familiar with using the da Vinci surgical system for teleoperation, to reflect the fact that outside of laboratory conditions this task would be performed by trained operators. Three of the operators rated themselves as “Experienced” and two as “Familiar” with remote teleoperation systems. All operators performed all three tasks in the same order: Conventional teleoperation, 2D Interactive Planning, then 3D Interactive Planning.

In a high-risk task such as an on-orbit operation, task success is the most relevant metric. We categorized the results for the two tasks, tool pickup and refueling, into three categories: Full success, partial success, and failure. Partial success was defined as an attached but improperly aligned tool in the tool pickup

Table 4.1: Pickup Task Success

	Conv.	2D	3D
Full Success	66.7%	100%	100%
Partial Success	33.3%	0%	0%
Failure	0%	0%	0%
Position Error (cm)	2.3	0.6	0.9
Orientation Error (deg)	4.3	1.8	4.2

task, and as a tool inserted less than the desired 3 cm in the refueling task. For both tasks, failure was categorized as when the operator believed it was no longer possible to complete the task. We also measured the pose difference between the final tool pose at the end of each task and the nominal tool pose. It should be noted that this nominal tool pose is subject to registration error, and so some small amount of deviation from the nominal pose may indicate a more accurate tool placement. Large deviations, however, indicate misplacement. Position and orientation error are reported as average values over all successful trials.

Table 4.2: Refueling Task Success

	Conv.	2D	3D
Full Success	50%	83.3%	50%
Partial Success	16.7%	16.7%	0%
Failure	33.3%	0%	50%
Tool Misalignment	50%	16.7%	33.3%
Orientation Error (deg)	2.4	1.9	1.2

Success metrics for the pickup task are shown in Table 4.1 and Figure 4.7. This task had no failures under any of the three conditions, which we believe reflects the fact that the magnetic mount is sufficiently strong to attach even across a fairly large distance. However, attaching at a distance increases the probability

Refueling Success Rate

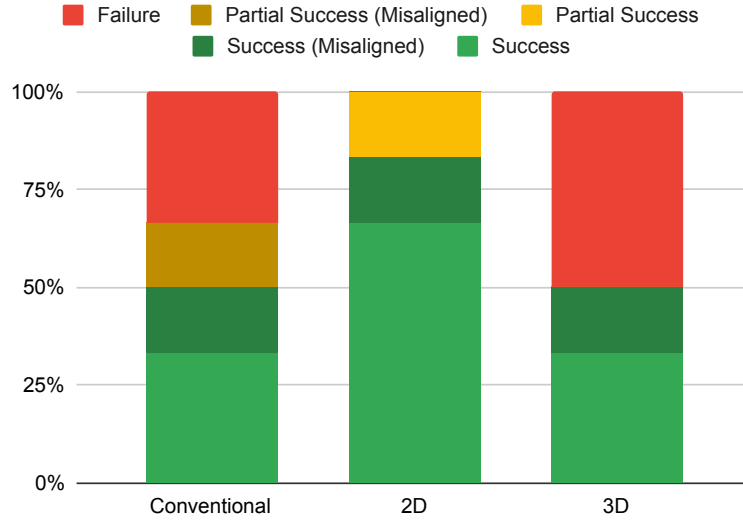


Figure 4.8: In the refueling task, only the 2D planning interface saw zero failures. In this task it was possible for operators to misalign the tool and still complete the task, shown here in darker color.

of a misaligned tool, which was evident in the partial success rate. Using conventional teleoperation, two of the operators misaligned the tool. The displacement metrics indicate that operators were able to position the tool much more accurately to the nominal pickup position using Interactive Planning, which may be due to the ability to preview the commanded end effector pose using a model of the end effector itself, rather than the simple ring indicator used in Chap. 3. The orientation error is similar between the conventional and 3D interfaces, but lower with the 2D interface, which may indicate the influence of being able to place a waypoint with respect to the nominal tool pickup pose using the 2D interface (i.e., by setting the **Frame** in Fig. 4.2 to the tool pickup pose).

For the refueling task, the environment places a very tight constraint on the tool position. The position deviation from the nominal value is likely more

representative of registration error than operator error, and so is not reported. However, this task affords the opportunity to dislodge the magnetically attached tool without knocking it off entirely, and the rate of such misalignment is also reported in Table 4.2. Tool misalignments are reported as the percentage of (fully or partially) *successful* tasks for which the tool was misaligned.

Success rates for the refueling task (Table 4.2 and Figure 4.8) were much lower, demonstrating the significantly higher requirement for precision in this task. Of the five failures across all conditions, four were due to the operator knocking the tool off the mount with some obstacle in the real world which was not modeled in the virtual environment. Of these, three were caused by contacting the refueling tool holder, which was visible when the operators were introduced to the task but was not included in the environment. In addition, two of the four instances of operators knocking the tool out of alignment occurred on the same object. For this reason, the promise of the virtual environment and/or collision detection may have been detrimental to overall performance because operators expected that every collision would be visible in the virtual environment or detected by the Interactive Planning system.

In another example of operators failing the task due to placing trust in the virtual environment, the operator who achieved a partial success in the refueling task using the 2D interface had correctly entered the command to insert the tool by exactly 3 cm. However, due to a 1.4 mm registration error, the refueling tool only entered 2.86 cm into the fuel pipe. This failure mode was not readily available in the 3D interface, in which no operators achieved partial success.

The combined success rate for the 3D interface (50%) was the lowest of the

three conditions, followed by the conventional interface (67%). Only the 2D Interactive Planning interface had a 0% failure rate, and it also had the highest full success rate of the three. While the 2D Interactive Planning interface improved task performance, the 3D interface led to worse results than the conventional interface. Multiple operators reported that they did not use the camera view, or used it much less, while using the 3D interface because of the effort of switching between the 3D planning and 2D execution interfaces. Other operators reported that the preview feature was more difficult to activate using the 3D interface, leading them to use it less. It was also noted that the 3D interface did not offer the advantage of specifying precise movements with respect to a defined frame in the environment, such as the tool mount frame or the refueling pipe frame, which are known as a result of the 3D scene model constructed from the camera survey. It is likely that a fully-featured 3D interface, which does not require exiting to the 2D interface to access some functionality, would produce different results (see Chapter 5).

Orientation error appears to once again be lower using Interactive Planning, although in this task the 3D Interactive Planning interface had the lowest average error. However, since these numbers are the average of only trials that were completed successfully with no misalignment, the conventional and 3D cases are each derived from only two data points. The 2D case, which is derived from five data points, shows a similar alignment error to the pickup task.

We also measured the workload for each interface using a version of the NASA Task Load Index (TLX) which reports values on a scale from 1 to 7. The results correlate with the performance measures: The conventional interface was rated an average of 3.7, the 2D Interactive Planning interface imposed the lowest

workload with 2.2, and the 3D Interactive Planning interface was significantly more difficult, with an average rating of 4.5. We also asked the participants to rate the difficulty on a scale of 1 to 5 (where higher numbers indicate greater difficulty) with similar results: 3.5, 1.7, and 4.5 respectively.

Table 4.3: Average Task Duration (minutes:seconds)

	Conv.	2D	3D
Tool Pickup	11:31	6:34	10:59
Refueling	17:35	12:18	30:22

Although execution time is significantly less important than success rate, we also recorded the average time to successful completion of each task, where applicable, which is shown in Table 4.3. The results for the conventional and 2D cases show that the 2D Interactive Planning interface allowed operators to complete the task faster, which we attribute to the lower difficulty and the operators’ increased confidence in their ability to safely execute longer motions. The results for the 3D interface, however, indicate that in the less-constrained tool pickup task the 3D interface was not worse than the conventional interface. It was in the severely constrained refueling task that the execution time in the 3D interface was much longer than in both the other interfaces. At least some of this time difference was likely due to the need to frequently switch between the 3D planning and 2D execution, as well as the difficulty of using features such as the path preview with the da Vinci manipulators.

4.5 Discussion and Conclusions

We developed a teleoperation system to allow operators to accurately command satellite servicing robots from the ground in the presence of multi-second communication delays and non-ideal camera views. The system includes Interactive Planning, which allows operators to define a motion plan as a series of waypoints, and Supervised Execution, in which operators can observe the plan’s execution and respond if failure is imminent. We implemented two interfaces to Interactive Planning, one using RViz and a custom GUI in 2D, and one using the da Vinci to allow planning in 3D.

The system was evaluated with six operators, most of whom rated themselves as Experienced or Familiar with teleoperation systems. We found that operators were more successful with the 2D Interactive Planning and Supervised Execution system than with a conventional teleoperation interface, and rated the 2D interface as easier to use. We also found that the 3D interface was, in most measures, equal to or worse than the conventional interface in both success metrics and difficulty. Some operators indicated specific limitations of the 3D interface that influenced their success, such as having to leave the da Vinci console to view camera feedback or the inability to specify precise motions with respect to certain defined task frames. We hypothesize that the 3D interface may have been effective for some parts of the task, but not for the entire task. In Chapter 5, we will explore an alternative 3D interface which alleviates or eliminates these limitations.

Chapter 5

A VR HMD Interface to IPSE

This chapter presents the development and evaluation of an alternative 3D interface to IPSE using a Virtual Reality (VR) Head-Mounted Display (HMD). We describe the motivation for choosing HMDs and background on previous work in VR/AR planning using HMDs, describe our design and implementation choices in the translation of the IPSE interface to the HMD, and present an evaluation of the HMD interface to IPSE against the previous best, 2D, interface. This chapter is based on work published in [41]. Liam J. Wang assisted with the design and implementation of the HMD interface and in conducting the evaluation.

5.1 Introduction

The conventional operator interface for teleoperating robots in space (and in other extreme environments) relies on traditional input devices, such as a keyboard and mouse, and multiple displays for visualizing remote camera views, robot telemetry, and simulated robot motion. While this approach has worked for decades, the recent proliferation of mixed reality hardware, in particular

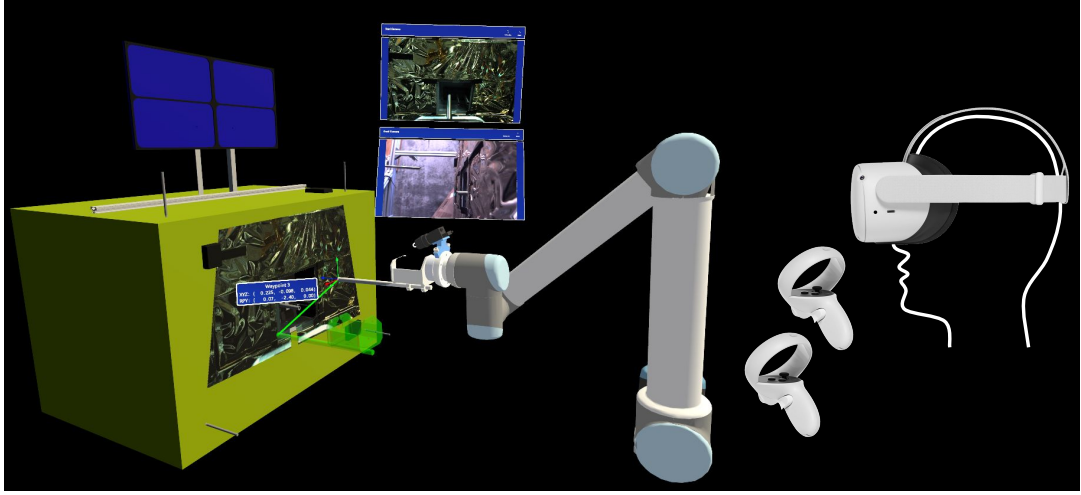


Figure 5.1: Virtual reality planning environment using Meta Quest 2 headset.

head-mounted displays (HMDs), offers the possibility for a more compact, intuitive and immersive environment.

In this chapter, we present a mixed reality interface (Fig. 5.1) that facilitates interactive planning as a mechanism for teleoperation of technically challenging and high-risk robotic operations, such as satellite servicing. Our goals are to alleviate the burden of these procedures for the ground-based operator and to decrease the risk of failure and errors. As an illustrative task, we focus on a critical step of refueling of a spacecraft where the remote manipulator combines large motions to reach its tool staging area with fine motions to insert the nozzle into the filler neck.

Our first contribution is the development of a mixed reality implementation, using a virtual reality (VR) headset with six degrees of freedom (DOF) controllers, of our Interactive Planning and Supervised Execution (IPSE) method [39]. IPSE previously relied on mouse and keyboard inputs to manipulate interactive markers in an augmented virtual environment and visualize the results

on multiple monitors (an alternate 3D implementation, using the stereo display and 3D input devices of a da Vinci surgical console, was less preferred). Our second contribution is a multi-user study that compares the mixed reality implementation of IPSE to the prior system.

5.2 Related Work

Advances in robotics hardware have enabled robots to execute increasingly complicated tasks in a broad variety of environments requiring equally complex programming. The benefits of using mixed reality in robotics and, in particular, for interactive robot programming have been researched for several years but the recent development of immersive and affordable HMDs has increased the amount of research in this area [8, 67]. Yet, few of these immersive robot programming technologies have been designed for robots operating in non-engineered remote environments where feedback is subject to latencies of several seconds.

A large portion of the research in this area is aimed at industrial applications, with the objective of making robot programming accessible to operators without experience [10]. In [42], a user can plan paths as a series of waypoints in an augmented reality (AR) environment, preview and edit the paths, and then execute them either autonomously or by allowing the user to control progress through the path. Likewise, path-planning within an AR environment is presented in [7] where the user is able to visualize the free space and select configurations by using interactive markers. In [37], instead of moving a virtual robot, input devices are used to directly define paths in an AR workcell. In [14], dynamic constraints were also included to improve the preview of the trajectory

of the robot. As is often the case for industrial applications, the aforementioned methods all assume knowledge of the robot’s operating environment. This is primarily achieved by accurate CAD models and careful calibration or by the operator physically sharing the workcell with the robot. As the operator is physically present near the robot in these applications, the operator can view the physical robot to monitor execution and compensate for any inaccuracies in the motion plan.

Other research focuses on enabling HMD-based teleoperation in more dynamic environments, using deep learning pose estimation to localize objects. The system described in [65] maps the operator’s VR hand controller pose directly to the end effector. In [61], operators plan and preview robot paths by placing a series of waypoints, but the interface lacks methods for precise waypoint positioning. Neither system was designed for high-latency teleoperation.

Drone applications have a natural need for immersive environments for First-Person View (FPV) drone flying [28, 47], building 3D immersive environments [3] or monitoring and surveillance [51, 73]. Similar to the industrial applications, the proximity between the drones and the operators makes the sensed data readily available to the operator. Additionally, few drone applications require precise close-proximity movements, so slight inaccuracies in virtual environments are inconsequential.

Our application requires an interface that allows precise teleoperation of a remote robot where direct observation is not possible. Erroneous motion must be avoided, as operator reaction time is limited due to the communication time delay of several seconds. We aim to improve upon these prior systems by integrating methods to visualize the remote environment with the available

limited sensor data, to tolerate inaccuracies in the CAD model of the operating environment, and to allow confident execution in situations with large round-trip time delay.

5.3 Background and Motivation

Satellite servicing will require robots to perform multiple tasks, with different tools designed for these tasks. It is expected that multiple tool changes will be necessary during the servicing task, which necessitates navigation of the robot arm between the worksite on the target satellite and the tool stowage area on the servicing spacecraft. This must be accomplished while avoiding collisions between the robot arm and structures on either spacecraft. Thus, a typical satellite servicing task will require both coarse motion to transit to/from the tool stowage area and fine motion to perform proximity operations.

In Chapters 2 and 3, we proposed methods for modeling the remote environment to overcome the visualization challenges due to the limited camera views. Specifically, the servicing robot first performs an image survey by moving its tool-mounted camera through a number of positions around the target satellite. Features on these 2D images are manually identified and used to either register to a known model (for example, the satellite frame) or to reconstruct an unknown or imprecisely known object, such as soft structures attached to the satellite. The resulting 3D model enables a VR visualization of the remote environment. However, the approach proposed in our prior work [56, 57] went beyond VR by projecting the received camera images onto the 3D models. This

projection of real content onto the virtual environment is referred to as augmented virtuality (AV). A user study with trained robot operators (Chapter 3) revealed that the AV visualization led to improved task performance and higher operator satisfaction when compared to the traditional visualization interface (i.e., camera views that are unmodified, or with minimal augmented reality overlays).

In Chapter 4 we developed a system for Interactive Planning and Supervised Execution that leverages the constructed 3D models to provide an intuitive environment to specify a plan involving both coarse and fine motion, verify its accuracy to the operator’s satisfaction, and execute it with as much human-in-the-loop supervision as can be afforded in the satellite servicing scenario. We evaluated this system using two user interfaces: one with keyboard-and-mouse interaction on a traditional 2D computer monitor (with occasional use of a 3D monitor), and one using the da Vinci surgical robot control console for 3D interaction. The results of this study clearly indicated that the 2D user interface led to better outcomes, however, users indicated that their lack of success with the 3D interface may be attributable to implementation details rather than inherent properties of the interaction. Specifically, the 3D interface lacked some key features, which included the ability to (1) numerically specify precise motions with respect to a selected coordinate frame in the model, and (2) view the augmented virtuality visualization during the supervised execution phase. Thus, in this work we implement an alternative 3D interface to the same system using a VR headset and evaluate its success compared to the previously-indicated best interface.

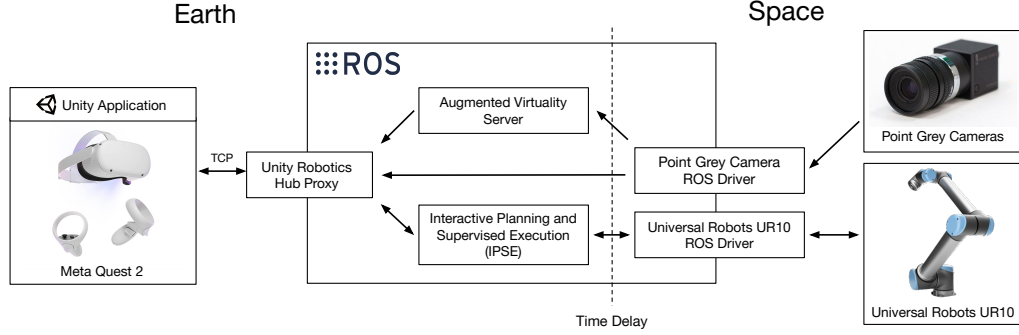


Figure 5.2: Overview of virtual reality planning interface architecture. The core IPSE planning system and augmented virtuality server is implemented using ROS. The ROS network is distributed across ground-based computers and space-side computers. The Unity Robotics Hub TCP proxy is used for bidirectional communication between ROS nodes and the VR Unity application.

5.4 System Description

The system we present in this chapter builds upon the Interactive Planning and Supervised Execution (IPSE) system presented in Chapter 4, which was implemented using the Robot Operating System (ROS) [43]. The VR interface is implemented using the Unity 3D game engine, which is the predominant development platform for mixed reality. We used the Unity Robotics Hub [55] ROS-TCP-Connector package to interface the Unity application with IPSE and the AV server using ROS topics over a local Wi-Fi network (Fig. 5.2). Because Unity supports a number of different hardware platforms, including most (if not all) commercially-available AR and VR headsets, it also provides the advantage of portability to different hardware. In this chapter, we report the development of a VR interface using the Quest 2 (Meta, Menlo Park, CA), a standalone wireless headset with 6-DOF tracked hand controllers, with the following section presenting the rationale for this choice.

5.4.1 Selection of Mixed Reality HMD

Our development effort initially targeted the Microsoft HoloLens 2 (HL2) augmented reality headset due to our prior experience with this platform for other projects. However, we concluded that the HL2 hand tracking capability was not sufficient for precise motion specification and therefore integrated a Xence-labs Quick Keys handheld scroll wheel controller for reduced-DOF waypoint adjustment. Later in development we compared the HL2 to the Meta Quest 2 and we found that the improved field of view and visual clarity of the Quest 2 enabled a more immersive operating experience. Additionally, the Quest 2 hand controllers offered greater precision than the built-in hand tracking of the HoloLens 2 and were more ergonomic than the Quick Keys controller. Since the VR interface was developed with Unity, minimal code changes were required to deploy to Quest 2.

5.4.2 Interactive Planning: Camera Visualization

The virtual world includes the AV camera feed overlays (Fig. 5.3) which augment the virtual models with tool camera images. This is implemented using the previously-developed OpenGL-based software [40], running on an external PC, that projects video frames captured by the tool camera onto texture maps covering the 3D model of the satellite. Texture maps are calculated real-time using ray casting to account for occlusions, then published by the renderer to compressed ROS image topics. The HMD receives the images via the ROS-TCP-Connector and updates the texture maps on the rendered 3D models. In addition, operators can view multiple live camera feeds with virtual 2D windows

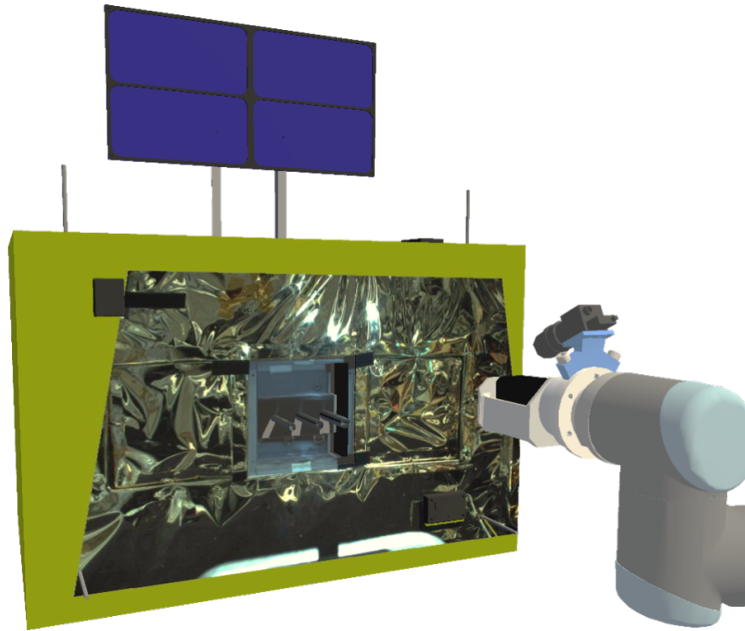


Figure 5.3: Augmented Virtuality texture overlays are displayed in the virtual reality planning environment.

which can be arbitrarily resized and positioned, as can be seen in Fig. 5.4.

5.4.3 Interactive Planning: Virtual Reality Interface

Using the headset, operators can view the virtual world (Fig. 5.4) in 3D and create IPSE motion plans, which consist of a series of waypoints. The built-in 6-DOF inside-out tracking on the Meta Quest 2 allows operators to look around the virtual world by moving in physical space. The 6-DOF tracked hand controllers allow operators to interact with IPSE and the virtual world. The hand controllers have joysticks, triggers, and buttons that are mapped to various actions within the planning environment. Using the hand controllers, operators can move, rotate, and scale the virtual world to inspect areas of interest.

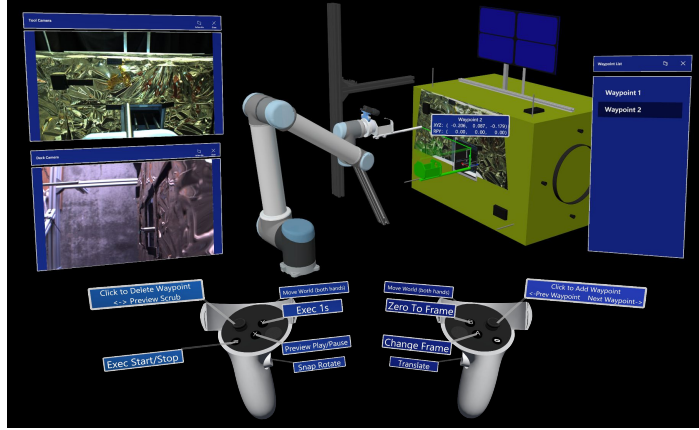


Figure 5.4: A screenshot of the operator's perspective in the Meta Quest 2 headset. The 3D virtual planning scene (top center) shows the pose of the space-side robot relative to the satellite and refueling tool station. Operators can place green waypoint markers to create a motion plan using the Meta Quest 2 6-DOF hand controllers, which are displayed virtually with labeled buttons (bottom). Virtual camera windows (top left) display live 2D video from space-side cameras. The waypoint list (top right) displays the waypoints in the motion plan and highlights the currently selected waypoint.

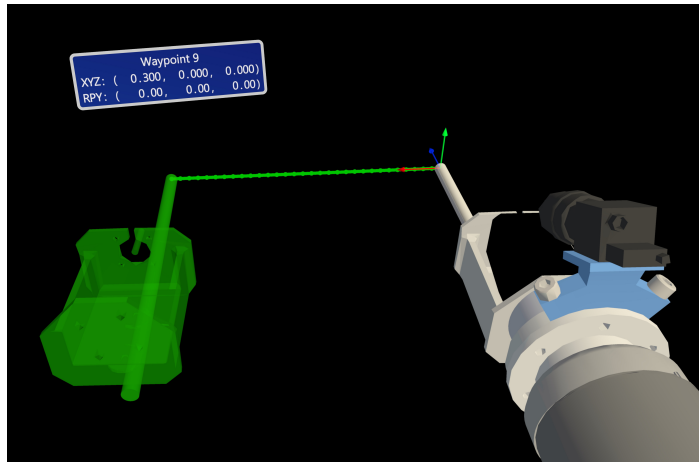


Figure 5.5: Motion plans are constructed by placing green waypoint markers (left) in 3D space. A dotted line shows the path of the end effector when moving from the initial pose to the destination pose.

Buttons on the controllers allow operators to add and delete waypoints (Fig. 5.5), change waypoint reference frames, and reset waypoints to reference frame origins. Operators can grab waypoints with the controllers and move them in all six degrees of freedom. Using an alternate grab button, operators can translate waypoints along a single axis at a time relative to the selected reference frame. For precise translational adjustment of waypoint position, operators can scale the world to “zoom in” before grabbing waypoints, which allows large physical motion of the hand controllers to be mapped to an arbitrarily small virtual movement of waypoints. Operators can view the position and rotation of waypoints using the virtual tooltip which appears above the selected waypoint. The sequence of waypoints in the motion plan can be viewed in the waypoint list window. The ability, described above, to precisely move each waypoint with respect to a selected reference frame solves the first limitation of our prior implementation on the da Vinci console (Section 5.3).

5.4.4 Interactive Planning: Motion Preview and Execution

After creating an IPSE motion plan, operators can use buttons and joysticks on the hand controllers to preview the robot motion and execute the motion plan on the remote robot. The VR environment always displays the robot model at its actual location, based on (time-delayed) telemetry feedback. The preview option moves a second robot model (the preview robot) through each waypoint in the motion plan, enabling operators to determine whether the motion plan is acceptable or whether the plan must be revised. The interface also includes a “scrubber” bar, enabling operators to scroll through the plan at any speed.

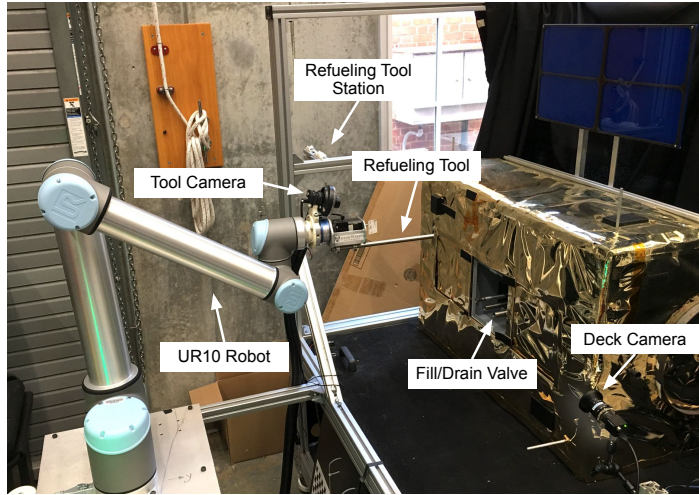


Figure 5.6: The space-side setup includes a mock satellite (right), mock servicing robot (left), and refueling tool station (center).

Plans are calculated and stored in robot joint space, so operators can be confident that the actual robot motion will match the previewed motion.

Once satisfied with the plans, operators can choose the execution option to send the motion plan to the remote robot. In this case, the robot model will follow the motion plan, subject to the telemetry delay. A heads-up display in the virtual world contains a progress bar to indicate execution progress. During execution, the camera views will be shown as AV projections on the models, as in Figs. 5.3 and 5.4, and optionally on separate 2D virtual windows, as in Fig. 5.4. The availability of the AV visualization in the VR environment solves the second limitation of our prior implementation on the da Vinci console (Section 5.3), which lacked this capability and therefore required operators to leave the console and view the visualization on a 3D monitor with shutter glasses.

5.5 Experiments

5.5.1 Experimental Setup

The space side of the experimental setup, including the servicing robot, tools, and patient satellite, is the same as described in Sec. 4.3.1 except for an improved design of the refueling tool station.

The operator station is located in a different room, on a different floor of the building, to reproduce the remote teleoperation experience and to eliminate the need for noise-canceling headphones, used in Chapter 4. The conventional teleoperation interface consists of four monitors (one of which is a 3D monitor), a keyboard, standard mouse, and a 3D mouse, as in Sec. 4.3.1. It includes one new feature, which leverages the software described in Section 5.4.2 to incorporate AV visualization in rviz, in addition to displaying it on the 3D monitor as in our prior implementation. The experimental teleoperation interface consists of the Meta Quest 2 and hand controllers, as described in Section 5.4. A software-created 5 second delay is added to emulate the multi-second delay expected between the ground station and on-orbit robot.

5.5.2 Experimental Task

An experiment begins with the mock satellite already registered using the Vision Assistant software tool described in Ch. 4. The robot begins in a standard location not near the mock satellite or tool stowage area and the refueling tool begins in the tool stowage area. During the experiment, the operator must first command the robot to the tool stowage area and lower it onto the tool to engage the magnetic attachment. Once attached, the operator must then

command the robot to move the tip of the refueling tool inside one of the pipes on the mock satellite (which represents the fill/drain valve). The experiment is complete when the operator believes the tool tip is inserted at least 3 cm inside the pipe, or failed if the operator believes it is not possible to insert the tool tip into the pipe (for example, if the tool is knocked off the magnetic mount).

5.5.3 Conditions

Each operator performs the experimental task twice, once with each experimental condition. The order of the conditions was randomly selected for each participant.

Both experimental conditions use the same virtual environment, planning and execution engine, and satellite registration. The registration error was estimated to be about 3-5 mm, which is large enough to require operators to visually confirm (on the camera images) alignment of the refueling tool with the pipe, rather than relying on the VR environment. Tool pickup was not adversely affected because the magnetic attachment tolerated that amount of registration error.

In the test condition, the subject interacts with the planning and execution engine using the VR interface described in Section 5.4. In the control condition, the subject uses the 2D interface described in Ch. 4.

5.6 Results

We evaluated the system with an IRB-approved user study (HIRB00000701) consisting of nine operators all of whom self-reported as “experienced” or “familiar” with robotic teleoperation. This reflects the intended training level of

the operators of this system in a real-life environment.

The most important metric for a teleoperation system in a high-risk environment with a high cost of failure is task success. All users successfully completed both portions of the experimental task under both interfaces. Two users of the proposed VR interface and one user of the baseline 2D interface misaligned the tool after contact with the environment, but all successfully performed the task using the misaligned tool. All three users who misaligned the tool did so on their first trial, suggesting that additional practice mitigated this outcome.

We also recorded the results of a NASA TLX survey administered after each experimental condition. TLX results are reported in Fig. 5.7. The TLX showed a preference for the proposed VR interface ($p=0.070$), with four users rating it as more favorable by at least one full TLX point. Those who did not favor the VR interface rated it either equal (one user) or only slightly worse than the baseline 2D interface. The average TLX score was 3.21 for the 2D interface and 2.39 for the VR interface, with standard deviations 0.87 and 0.58 respectively.

Additionally, users were asked to rate the difficulty of using each interface on a scale from 1 (Very Easy) to 5 (Very Hard). Users reported an average of 1.56 with the VR interface compared to 2.56 with the 2D interface. This also shows a preference for the proposed VR interface ($p=0.081$).

Task duration, shown in Table 5.1, for the two conditions also favored the VR interface. While task duration is much less important than task success for the application we consider, increased operating time can contribute to the operators' mental and physical load. The decrease in task time by 22% and 25% respectively for the two segments of the task may be partially responsible for the improved TLX score for the VR interface, although this is a weaker

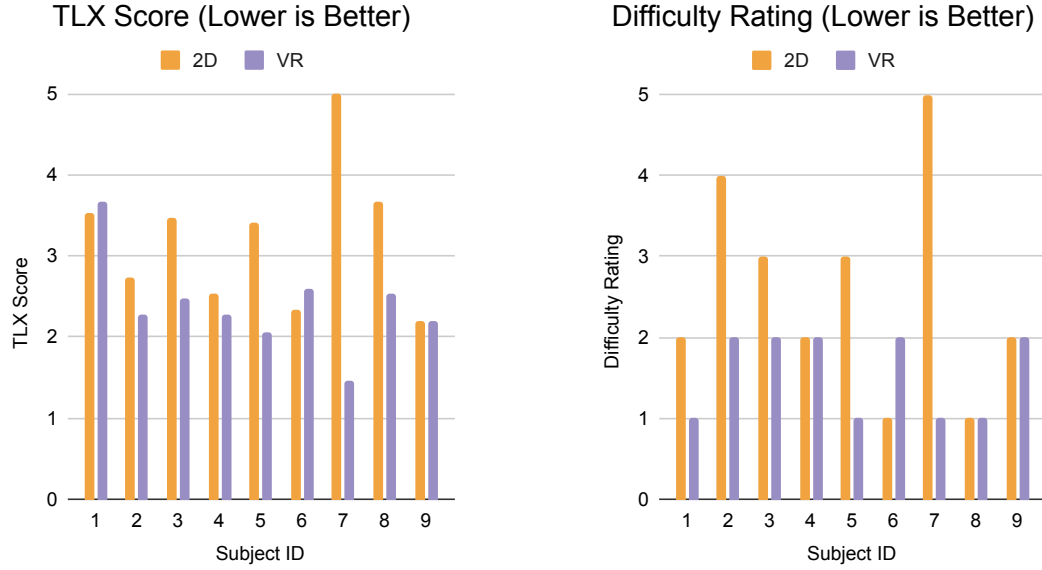


Figure 5.7: NASA TLX results (left) and operator difficulty ratings (right) show that most operators preferred the VR interface, and those who did not only had a slight preference against it.

result ($p=0.242$ and 0.214 respectively). Additionally, decreased time for operations without corresponding increases in task failure rate improves the overall productivity.

Table 5.1: Task Duration (seconds)

	2D Mean	2D Std.Dev.	VR Mean	VR Std.Dev.
Tool Pickup	250	158	198	101
Refueling	740	460	575	346

To further evaluate successful execution of the experimental task, we recorded the robot’s position and orientation at the end of each segment of the task. Due to imperfect registration between the robot and the satellite, the absolute position and orientation are not meaningful. However, the variance reflects the consistency operators were able to achieve with each interface. These results,

Table 5.2: Variance in Position (mm^2) and Orientation (deg^2)

	2D		VR	
	Pos.	Ori.	Pos.	Ori.
Tool Pickup	0.84	1.53	0.70	1.00
Refueling	0.66	0.29	0.60	0.25

shown in Table 5.2, indicate no significant difference between the two interfaces. Cases where the tool became misaligned are excluded from the calculation as the true tool position is not known.

5.7 Discussion and Conclusions

We developed a virtual reality interface for the Interactive Planning and Supervised Execution (IPSE) robotic planning system as an alternative to the keyboard and mouse interface presented in Ch. 4. The virtual reality interface was implemented for the Meta Quest 2 standalone headset and 6-DOF hand controllers. While using the headset, operators can create, preview, and execute motion plans on satellite servicing robots while monitoring execution progress with Augmented Virtuality overlays and 2D camera feeds.

The system was evaluated against the baseline keyboard and mouse 2D interface through a user study with nine operators, all of whom rated themselves as “experienced” or “familiar” with teleoperation systems. We found significant indication that the VR interface led to a lower operator workload as compared to the 2D interface in Ch. 4, without any indication that the VR interface lowers performance. This improves over the VR interface presented in Ch. 4 (there termed “3D interface”), which was shown to have worse outcomes and higher

workload. In addition, the VR interface we implemented requires only a commercial off-the-shelf virtual reality headset and a standard personal computer, whereas the previous 3D interface used a highly specialized da Vinci master console. The 2D interface used previously and in this work contains four monitors, one of which is a stereo display. Thus, the VR interface also offers portability and economy of space. These advantages were not offset by any loss in performance and were accompanied by an improvement in operator workload. Future work may include experiments with alternative headsets and input devices.

Chapter 6

Extension to 7-DOF Teleoperation

In previous chapters we addressed the challenge of commanding a 6-DOF robot in Cartesian space for applications in an ISAM task. However, ISAM applications frequently prefer 7-DOF manipulators, such as the Space Station Remote Manipulator System (“Canadarm2”) and two arms of the Special Purpose Dexterous Manipulator (“Dextre”) on the International Space Station [17] and multiple arms on the Tiangong space station [24]. 7-DOF manipulators provide redundancy which can be utilized for increased flexibility in manipulation, but which must be considered in the arm’s control system. In this chapter we present an extension of the IPSE system to allow full teleoperation of a 7-DOF robot with minimal cognitive burden.

6.1 Background

Unlike a 6-DOF robot, where each Cartesian pose can be reached by a small finite number of configurations which cannot be traversed with self motions, a 7-DOF robot can reach most poses in its workspace with an infinite number of

different configurations. Therefore, control of a 7-DOF robot must, implicitly or explicitly, decide which of the infinite feasible configurations to select. Most commonly, an optimization criterion is defined and various methods are used to compute or approximate the optimal configuration according to this criterion [27, 52, 60, 70]. When used with teleoperated robots, these methods impose no burden on operators, since no additional input is required to select a configuration. However, these methods restrict the operator’s ability to control the robot, and if the optimal solution is not the operator’s desired solution there is no recourse. For manipulators which are sufficiently similar to the human arm, a mapping can be used to allow the operator to command the robot simply by moving their arm [20, 71]. While this has very low cognitive overload, it has the downside of limiting the robot’s precision to that of the human operator’s arm and of transmitting accidental motions, similar disadvantages to the method explored in Ch. 2.

Another, simpler strategy is to simply “lock” one joint of the 7-DOF arm, fixing it to its current position, which effectively transforms the robot into a 6-DOF arm which does not require infinite redundancy resolution [1]. As with the optimization criterion strategy, this limits the operator’s ability to take advantage of the redundancy provided by the 7-DOF arm. Some control of redundancy can be offered by allowing operators to choose which joint is locked for a given motion, but the relationship between this choice and the resulting change in motion profile is unintuitive [1]. Alternatively, the redundancy can be utilized in a separate control mode which only allows self-motion, such as the Canadarm’s Pitch Plane Change mode [1]. This allows more intuitive control, as the user can specify the rotation of some single joint and the self-motion

constraint uniquely determines the position of the remaining joints. However, implementing this strategy as a separate mode prevents the redundancy from being used continuously during an end-effector motion to navigate precise paths. The operator may be allowed to specify joint trajectories directly, but this imposes a high cognitive burden if a specific tool pose is desired, as the operator is now required to perform complex inverse kinematics calculations mentally.

For the commonly used Spherical-Revolute-Spherical (SRS) type 7-DOF arm, such as the Canadarm2 [1], the redundant degree of freedom controls the location of the elbow joint. Specifying this location is a logical way to allow the operator to control the redundancy. There is no single joint of the SRS robot which directly controls the elbow joint position, but it is possible to define a parameter which encodes this location and then derive an analytical solution to the inverse kinematics problem given a desired end effector pose and value of this parameter. Many variants of this parameterization exist under many names, including “redundancy angle” [9], “arm angle” [32, 46, 72], “swivel angle” [44, 71], and “Shoulder-Elbow-Wrist (SEW) angle” [13]. In this work we prefer the term “SEW angle”. Previous studies have explored the automatic selection of SEW angle to maximize some criterion, such as distance from joint limits [9, 46]. We explore the use of this parameterization as an easy-to-understand way for operators to control the redundancy of the SRS arm during teleoperation.

In this work we use the SEW angle to enable a visualization to assist operators in choosing the best resolution of the redundancy. There is previous work exploring methods to visualize constraints within the 3D environment [68, 69], but we are unaware of any research into using visualization over time to aid in trajectory design.

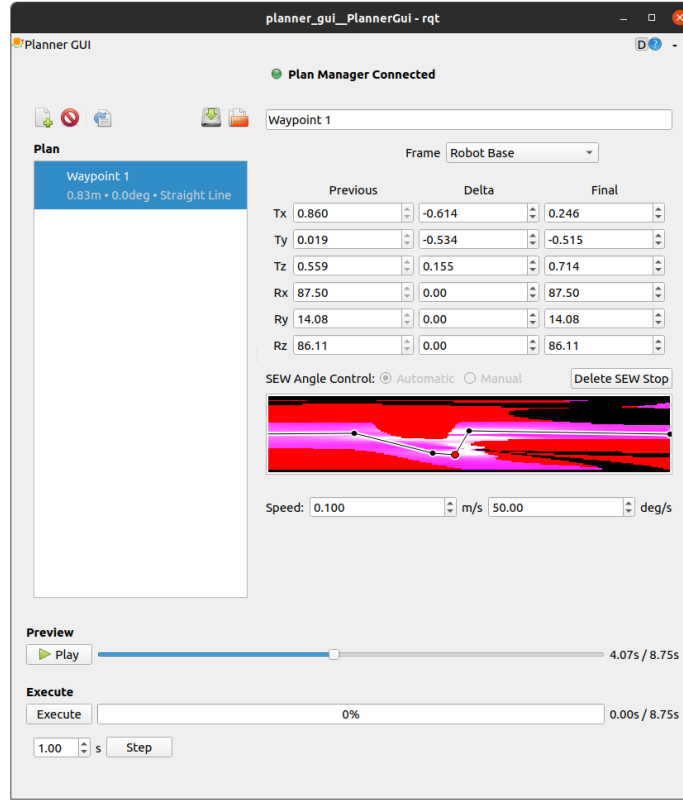


Figure 6.1: The updated 2D interface keeps most of the controls the same, but adds a visualization of the redundancy space available for the selected waypoint's motion.

6.2 Method

Our method extends the IPSE planning system described in previous chapters, and its 2D interface, to support planning for an SRS 7-DOF robot. Extending these methods to the 3D VR interface is left as future work. Though the majority of the UI is unchanged from the 6-DOF case, significant changes to the planning engine are required to enable the new redundancy resolution capabilities.

6.2.1 2D Redundancy Resolution Interface

The updated IPSE interface adds redundancy resolution as an augmentation to the existing method of specifying poses, so the majority of the interface is still in place. The user still builds a plan as a series of waypoints, where the primary datum that defines a waypoint is a Cartesian pose. The tools for specifying this pose — an interactive marker in RViz and an x-y-z-roll-pitch-yaw numerical representation in the custom Planner GUI program — are still in place. However, the previously-offered obstacle avoidance mode, where MoveIt! [50] was used to compute a collision-free trajectory, is no longer available due to changes in the planner. Recorded-path mode, where the user could record an exact path by moving the end-effector marker, is similarly unavailable. Both of these modes are possible to replicate in the updated planner, but we noted that operators never elected to use these features in our previous trials and so did not prioritize their reimplementations. This leaves a straight-line path between waypoints as the only option.

In the previous 6-DOF case, most desired poses in the robot’s workspace have a small finite number of inverse kinematics solutions and the robot cannot navigate from one solution to another using only self-motion. (The exceptions where a pose has an infinite number of inverse kinematics solutions occur at singularity, which robot teleoperators typically avoid because of their potential to amplify slow task-space motions into dangerously fast joint-space motions.) This means that there is little utility in using these multiple solutions to increase maneuverability or avoid obstacles, and allowing MoveIt! to automatically select the appropriate configuration was sufficient. However, with the 7-DOF robot,

we want to provide the operator full control over the redundancy. To achieve this, we use a combination of the SEW angle, a method to parameterize the infinite family of solutions which are mutually reachable with a self-motion, and the kinematic branch, a categorization of the finitely many families of solutions defined by their mutual reachability with a self-motion.

6.2.1.1 SEW Angle

As described in Sec. 6.1, the SEW angle is a parameter which encodes the location of the elbow joint of the SRS manipulator for a given end-effector pose using a single number. The SEW angle is defined as the angle between two planes: the shoulder-elbow-wrist plane, which is defined as the plane which passes through the shoulder, elbow, and wrist points; and the reference plane, which is defined as the shoulder-elbow-wrist plane of the configuration that reaches the desired pose with the axes of joints 2 and 4 parallel (typically when $\theta_3 = 0$) [46]. This is a particularly easy parameterization to visualize, since a zero SEW angle occurs with its elbow at either the highest or lowest position with respect to the joint 1 axis (except at the shoulder singularity, when all SEW angles place the elbow at equal height). A SEW angle of π or $-\pi$ places the elbow at the opposite extreme, and intermediate SEW values behave intuitively.

Although there is some value in using a static, but operator-selectable, SEW angle for an entire motion [1], some constrained motions require the SEW angle to change throughout the motion to maintain clearance from some nearby obstacle, a nearby joint limit, or both. For this reason, we allow the operator to define a piecewise linear path for the SEW angle to follow over the duration

of each waypoint, as shown in Fig. 6.1. Users specify this line by manipulating control points on a 2D representation of the waypoint’s redundancy space. A graph is displayed, with the x-axis being progress through the waypoint’s straight-line path and the y-axis being SEW angle. A SEW angle path is represented by a straight line across the width of this graph, where the y-position of the line at each point determines the SEW angle at the corresponding point in the trajectory. By default the SEW angle remains at whatever value it had at the previous waypoint (or the robot’s current SEW angle for the first waypoint). To add a control point, the operator clicks on the line and drags the cursor to the desired position. A new control point is created and placed where the operator indicated. The operator can move existing control points by clicking them and dragging, and can delete the selected control point with a button.

Behind the SEW path, the SEW graph is colored to indicate which combinations of pose and SEW angle are disallowed or undesirable due to joint limits, collision, or singularity. In this graph, black areas violate one or more joint limits; red areas obey joint limits but place the robot in collision; and purple areas are feasible non-colliding configurations which are close to singularity. Computation of this graph is discussed in Sec. 6.3.

As the user manipulates the control points, the preview robot in RViz updates to show the corresponding configuration (see Fig. 6.2). The user can inspect the configuration of the control points to ensure they meet any requirements, or can intentionally move a control point into a black, red, or purple area to visualize the corresponding limit, collision, or singularity. Additionally, a visualization of the redundancy circle at the preview robot’s configuration is displayed in the virtual environment. The redundancy circle uses the same

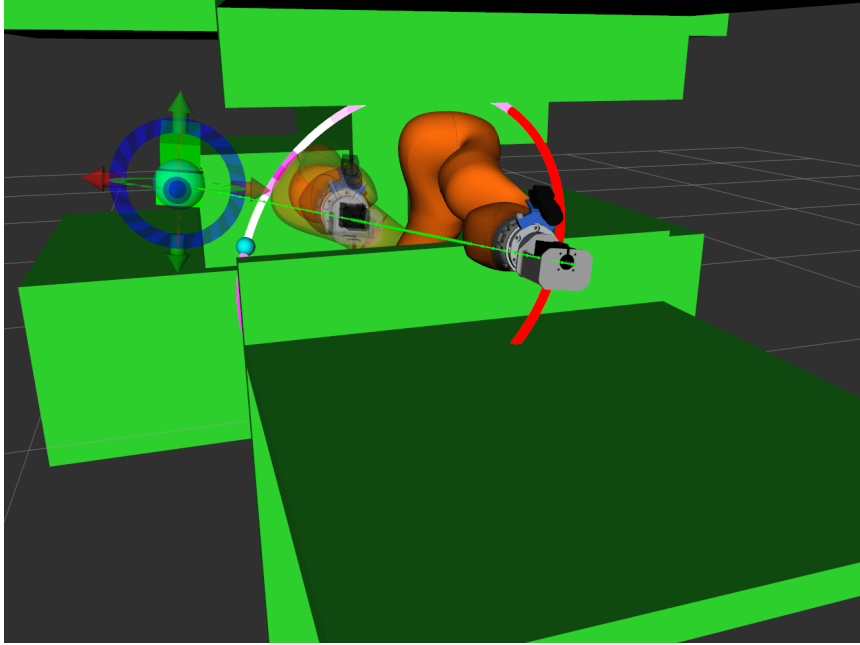


Figure 6.2: The RViz visualisation during a planning operation in the pilot study environment, showing the addition of the redundancy hoop.

color coding as the SEW graph to indicate which SEW angles are disallowed or dangerous. A ball rides along the circle, allowing the user to change the desired SEW angle directly in the virtual environment. If a SEW control point is selected, moving this ball moves the selected point along the y-axis in the SEW graph. If no control point is selected, the ball appears partially transparent and moving it creates a new control point. The x coordinate of the new control point is determined by the current preview position.

6.2.1.2 Kinematic Branches

The combination of pose and SEW angle is, in most cases, not enough to uniquely specify a single configuration. The SRS arm has three additional redundancies: For any configuration $\theta = \{\theta_1, \dots, \theta_7\}$, a configuration θ' can be

defined such that

$$\boldsymbol{\theta}' = \{\theta_1 \pm \pi, -\theta_2, \theta_3 \pm \pi, \theta_4, \theta_5, \theta_6, \theta_7\}$$

where $\theta_n \pm \pi$ is chosen so the result is in the interval $[-\pi, \pi)$. The resulting $\boldsymbol{\theta}'$ has the same end effector pose and same SEW angle, but links 1 and 2 are rotated by π about the corresponding joint's axes. The same is true of the configuration with θ_4 negated and $\theta_{3,5}$ offset,

$$\boldsymbol{\theta}' = \{\theta_1, \theta_2, \theta_3 \pm \pi, -\theta_4, \theta_5 \pm \pi, \theta_6, \theta_7\}$$

and the configuration with θ_6 negated and $\theta_{5,7}$ offset.

$$\boldsymbol{\theta}' = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5 \pm \pi, -\theta_6, \theta_7 \pm \pi\}$$

These transformations may be applied in any combination, yielding 8 configurations for each pose and SEW angle.

Unlike SEW angles, kinematic branches typically cannot be traversed without moving the end effector (i.e., without self motion). For this reason, the operator is not allowed to choose an arbitrary branch for each waypoint. Instead, the interface for changing branches is conceptualized as consisting of two features: First, when a waypoint ends with any of joints 2, 4, or 6 at zero, the next waypoint offers the operator a choice of branches for those particular joints. Either branch is reachable from the zero position, as $-0 = 0$, and so this branch choice can occur without causing the end effector to deviate from the selected trajectory. Second, to assist in initiating a branch switch, a waypoint can be configured to lock any of joints 2, 4, or 6 to zero. This would enable the user

to select the most advantageous location to make the potentially large motions involved in a branch switch. This branch switching concept is not implemented due to time constraints.

6.2.2 Planner Implementation

The implementation of the features described in Sec. 6.2.1 requires SEW- and branch-aware inverse kinematics and planning. This section describes our implementation of a custom SEW- and branch-aware motion planner. The planner computes each waypoint’s plan in turn, and uses the final end effector pose, SEW angle, and branch as the starting values for the next waypoint. The starting values for the first waypoint are derived from the robot’s current configuration.

6.2.2.1 Workspace projection

While not strictly related to 7-DOF planning, one limitation of the previous IPSE implementation was its inability to display a preview configuration for poses outside the robot’s workspace. Such a preview configuration would not be able to reach the desired goal, but displaying a best-attempt configuration helps operators understand why their commanded pose is unattainable and how to fix it. For this reason, the first step in computing a waypoint’s plan is to project the desired end effector pose into the workspace. The optimization-based 7-DOF SEW-aware inverse kinematics implementation provided in [13] is used to compute a best-effort inverse kinematics solution for the desired pose and SEW angle. This software returns a set of configurations and a value indicating whether the solution reaches the goal position. If this value indicates that the solution does not reach the goal position, we select an arbitrary returned

solution and use forward kinematics to get the closest feasible pose to the goal position. Otherwise the desired pose can be used directly.

6.2.2.2 Interpolation

Once the projected pose is computed, we perform a coordinated interpolation between translation, rotation, and SEW angle change. The interpolation density depends on the amount of change in translation, rotation, and SEW angle. Of those three, only SEW angle changes at a varying rate. Because of this, the segments between each pair of SEW control handles are interpolated separately.

Each SEW control handle is a (Φ, s) pair where $\Phi \in [-\pi, \pi)$ is the desired SEW angle and $s \in [0, 1]$ is the point along the end-effector path where the desired SEW angle should be reached. There is an implicit handle at $s = 0$ where Φ is equal to the SEW angle at the end of the previous waypoint, or the robot's current SEW angle if this is the first waypoint. If there is no explicit handle at $s = 1$, an implicit handle is added at $s = 1$ where Φ is equal to the Φ of the (possibly implicit) previous handle. These handles separate the waypoint into segments. We refer to the handle at the beginning of the segment as $(\Phi_{start}, s_{start})$ and the handle at the end of the segment as (Φ_{end}, s_{end}) .

For each segment, the start and end pose are interpolated

$$\mathbf{p}_{start} = \mathbf{p}_{n-1} + s_{start}(\mathbf{p}_n - \mathbf{p}_{n-1})$$

$$\boldsymbol{\theta}_{start} = \text{slerp}(\boldsymbol{\theta}_{n-1}, \boldsymbol{\theta}_n; s_{start})$$

$$\mathbf{p}_{end} = \mathbf{p}_{n-1} + s_{end}(\mathbf{p}_n - \mathbf{p}_{n-1})$$

$$\boldsymbol{\theta}_{end} = \text{slerp}(\boldsymbol{\theta}_{n-1}, \boldsymbol{\theta}_n; s_{end})$$

where \mathbf{p}_n and $\boldsymbol{\theta}_n$ are the desired end effector position and orientation at way-point n , or the current pose if $n = 0$. The number of intermediate points T is computed as:

$$T = \max \left(2, \left\lceil \frac{\|\mathbf{p}_{end} - \mathbf{p}_{start}\|}{D_p} \right\rceil, \left\lceil \frac{\|\boldsymbol{\theta}_{end} - \boldsymbol{\theta}_{start}\|}{D_\theta} \right\rceil, \left\lceil \frac{|\Phi_{end} - \Phi_{start}|}{D_\Phi} \right\rceil \right)$$

where D_p , D_θ , and D_Φ are constants representing the maximum linear, angular, and SEW angle distance between points. At each integer t from 0 to T , the end effector pose \mathbf{p}_t , $\boldsymbol{\theta}_t$ and SEW angle Φ_t are computed

$$\mathbf{p}_t = \mathbf{p}_{start} + s_t (\mathbf{p}_{end} - \mathbf{p}_{start})$$

$$\boldsymbol{\theta}_t = \text{slerp}(\boldsymbol{\theta}_{start}, \boldsymbol{\theta}_{end}; s_t)$$

$$\Phi_t = \Phi_{start} + s_t (\Phi_{end} - \Phi_{start})$$

where $s_t = \frac{t}{N-1}$. The result is visible in Fig. 6.2: The density of points along the green trajectory line changes in concert with the slope of the line segments of the SEW graph in Fig. 6.1.

6.2.2.3 Parametric Inverse Kinematics

In [46], Shimizu et al. describe the computation of a parameterized inverse kinematics, where expressions for each joint value as a function of the SEW angle ψ are derived for an SRS robot defined using DH parameters. We use a KUKA LBR iiwa 14, whose parameters are not defined using the DH convention. For this robot, the orientation of link 3 relative to the base link ${}^0\mathbf{R}_3$ is given by

$${}^0\mathbf{R}_3 = \begin{bmatrix} * & * & \cos \theta_1 \sin \theta_2 \\ * & * & \sin \theta_1 \sin \theta_2 \\ -\sin \theta_2 \cos \theta_3 & \sin \theta_2 \sin \theta_3 & \cos \theta_2 \end{bmatrix}$$

and the expressions for joints 1-3 are

$$\begin{aligned}\tan \theta_1 &= \frac{a_{s23} \sin \psi + b_{s23} \cos \psi + c_{s23}}{a_{s13} \sin \psi + b_{s13} \cos \psi + c_{s13}} \\ \cos \theta_2 &= a_{s33} \sin \psi + b_{s33} \cos \psi + c_{s33} \\ \tan \theta_3 &= \frac{a_{s32} \sin \psi + b_{s32} \cos \psi + c_{s32}}{-a_{s31} \sin \psi - b_{s31} \cos \psi - c_{s31}}\end{aligned}$$

where, as in [46],

$$\begin{aligned}\mathbf{A}_s &= \left[{}^0\mathbf{u}_{sw} \times \right] {}^0\mathbf{R}_3^o \\ \mathbf{B}_s &= - \left[{}^0\mathbf{u}_{sw} \times \right]^2 {}^0\mathbf{R}_3^o \\ \mathbf{C}_s &= \left[{}^0\mathbf{u}_{sw} {}^0\mathbf{u}_{sw}^T \right] {}^0\mathbf{R}_3^o\end{aligned}$$

and ${}^0\mathbf{u}_{sw}$ is the unit vector of the vector from the shoulder to the wrist and $[\mathbf{v} \times]$ denotes the skew-symmetric matrix of the vector \mathbf{v} . a_{sij} , b_{sij} , and c_{sij} denote the (i, j) th component of \mathbf{A}_s , \mathbf{B}_s , and \mathbf{C}_s respectively.

Similarly, the value of ${}^4\mathbf{R}_7$ is

$${}^4\mathbf{R}_7 = \begin{bmatrix} * & * & \cos \theta_5 \sin \theta_6 \\ -\sin \theta_6 \cos \theta_7 & \sin \theta_6 \sin \theta_7 & \cos \theta_6 \\ * & * & -\sin \theta_5 \sin \theta_6 \end{bmatrix}$$

and the expressions for joints 5-7 are

$$\begin{aligned}\tan \theta_5 &= \frac{-a_{w33} \sin \psi - b_{w33} \cos \psi - c_{w33}}{a_{w13} \sin \psi + b_{w13} \cos \psi + c_{w13}} \\ \cos \theta_6 &= a_{w23} \sin \psi + b_{w23} \cos \psi + c_{w23} \\ \tan \theta_7 &= \frac{a_{w22} \sin \psi + b_{w22} \cos \psi + c_{w22}}{-a_{w21} \sin \psi - b_{w21} \cos \psi - c_{w21}}\end{aligned}$$

where

$$\mathbf{A}_w = {}^3\mathbf{R}_4^T \mathbf{A}_s^T {}^0\mathbf{R}_7^d$$

$$\mathbf{B}_w = {}^3\mathbf{R}_4^T \mathbf{B}_s^T {}^0\mathbf{R}_7^d$$

$$\mathbf{C}_w = {}^3\mathbf{R}_4^T \mathbf{C}_s^T {}^0\mathbf{R}_7^d$$

and ${}^3\mathbf{R}_4$ is the orientation of link 4 in the frame of link 3 and ${}^0\mathbf{R}_7^d$ is the desired tip orientation.

As in [46], θ_4 is determined independently of the SEW angle as

$$\cos \theta_4 = \frac{\|{}^0\mathbf{x}_{sw}\|^2 - d_{se}^2 - d_{ew}^2}{2d_{se}d_{ew}}$$

where d_{se} is the distance from the shoulder to the elbow and d_{ew} is the distance from the elbow to the wrist.

We store these values both in the given parametric form, to facilitate computing the SEW graph (see Sec. 6.3) and as joint angles, using the interpolated SEW angle for Ψ and the selected kinematic branch to resolve the inverse cosine ambiguity — that is, to select whether to use the $\theta_n \leq 0$ solution or the $\theta_n \geq 0$ solution for joints 2, 4, and 6.

The planner is designed to be tolerant at all steps so that users can still preview an invalid plan in as much detail as possible to understand why the plan is invalid. The inverse kinematics may result in configurations where some joints are out of limits, and instead of returning failure, the trajectory is marked as invalid but computation continues. In some cases, the trajectory may pass outside the workspace during a trajectory, such as when the interpolated pose

passes too close to the base of the robot, and there is no real-valued solution (in this case, the value of $\cos \theta_4$ is outside the range $[-1, 1]$). The planner leaves the configuration blank but still stores the interpolated pose and SEW angle for visualization. The execute functionality checks that a trajectory is not marked as invalid before allowing its execution.

6.2.2.4 Trajectory Timing

The prior steps compute a path for each waypoint, and the final step is to add timing information to turn the path into a trajectory. The time between each pair of interpolated points is the greater of the linear distance divided by desired linear speed, angular distance divided by desired angular speed, and each joint distance divided by that joint's velocity limit. Acceleration limits are not currently implemented.

To support error tolerance, interpolation points for which no configuration could be computed are still assigned a duration. The linear speed and angular speed limits are used, and the joint speed limits are not. As with Sec. 6.2.2.1, this offers an improvement over the previous implementation by allowing more robust preview of invalid trajectories.

6.3 Computing the SEW Graph

When a new trajectory is planned, a SEW graph update is queued. Currently, the colors of the SEW graph are computed by sampling each pixel. The configuration corresponding to that pixel is computed using the parametric configuration described in Sec. 6.2.2.3, and the resulting joint configuration is checked for violations of joint angles, collision constraints, or singularity requirements

using MoveIt!. Although the computation is multi-threaded, it is still computationally intensive, taking up to 3 seconds on a 6-core Intel® Core™ i7-8700 CPU for a 250×60 sample SEW graph. Future work could include improving the actual computation time by precomputing reachability maps, using closed-form expressions for joint limits in terms of the SEW angle as described in [46], or improving collision check batching. Perceived computation time could be decreased by using similar progressive loading techniques to those used in image file formats, which allow the viewer to begin perceiving detail before the image finishes loading. More sophisticated image upscaling could reduce the appearance of pixelation.

To avoid repeated work, the colorization of the SEW hoop is derived by taking a vertical slice of the already-computed SEW graph at the appropriate x coordinate. Best results are achieved when the vertical resolution of the SEW graph is an integer multiple of the number of segments used to render the hoop.

6.4 Experiments

We conducted a pilot study using a simulated KUKA LBR iiwa 14 collaborative robot. This is a 7-DOF SRS robot where each joint has limits smaller than $[-\pi, \pi]$, so use of the redundancy to manage joint limits is necessary in some situations. We designed an “obstacle course” environment with 6 goals of varying difficulty, shown in Fig. 6.3. Goals 1 and 2 can be accessed with minimal difficulty, goals 3 and 4 require navigating the robot through a narrow gap, and goals 5 and 6 require the robot to be operated near the edge of its workspace. Gazebo is used for dynamic simulation of the robot, but the

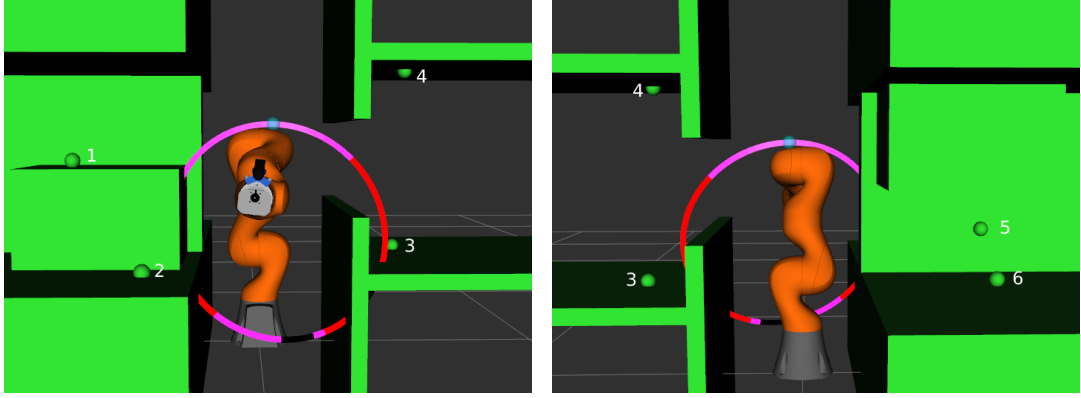


Figure 6.3: The experimental environment, with the six spherical touch targets labeled 1-6.

obstacles are only present in the collision-checking environment and the RViz visualization. Therefore, collisions are not simulated, but are monitored. The task is considered a failure if the robot collides with the environment. Due to the limitations of the simulation environment, Augmented Virtuality (AV) is not used in this experiment.

The experiment begins with the robot in a standard position, with the end effector between goals 3 and 4. The operator must touch all six goal spheres with the end effector in any order. Each operator performs the experimental task twice: once using the proposed 7-DOF extension of IPSE, and once using a simplified version where the SEW graph is not visible, the redundancy circle is not colored, and the operator is restricted to commanding a single goal SEW angle at each waypoint. This is treated by the planner as a single linear segment spanning the entire waypoint, with the single control point positioned at the end of the motion.

The interfaces are evaluated on task success rate, total time taken to touch all six targets, and the results of a NASA TLX survey and difficulty rating to

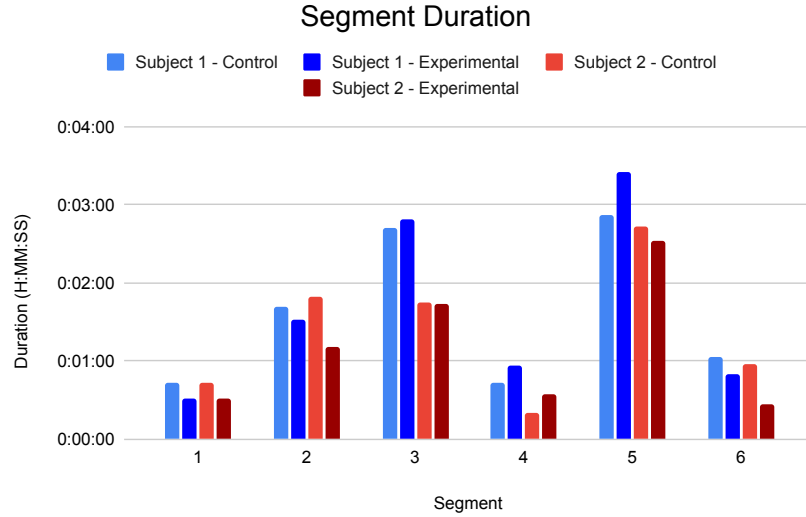


Figure 6.4: Duration of each task segment from both operators under both experimental conditions. Some segments took longer under the control case, while others were longer in the experimental case. Segments are numbered in the order the operators attempted them, which was the same for both operators, not by which target is approached.

judge operator workload. To minimize the impact of learning effects between trials, the order of the conditions is randomized.

6.5 Results

We evaluated the system with an IRB-approved pilot study (HIRB00000701) consisting of two operators, both of whom self-reported as “experienced” with robotic teleoperation. This experience level reflects the intended training level of the operators of this system in a real-world environment.

On the most important metric, task success rate, the two experimental conditions were equal. Both subjects were able to touch all targets successfully. Task duration also shows no conclusive difference between the two interfaces,

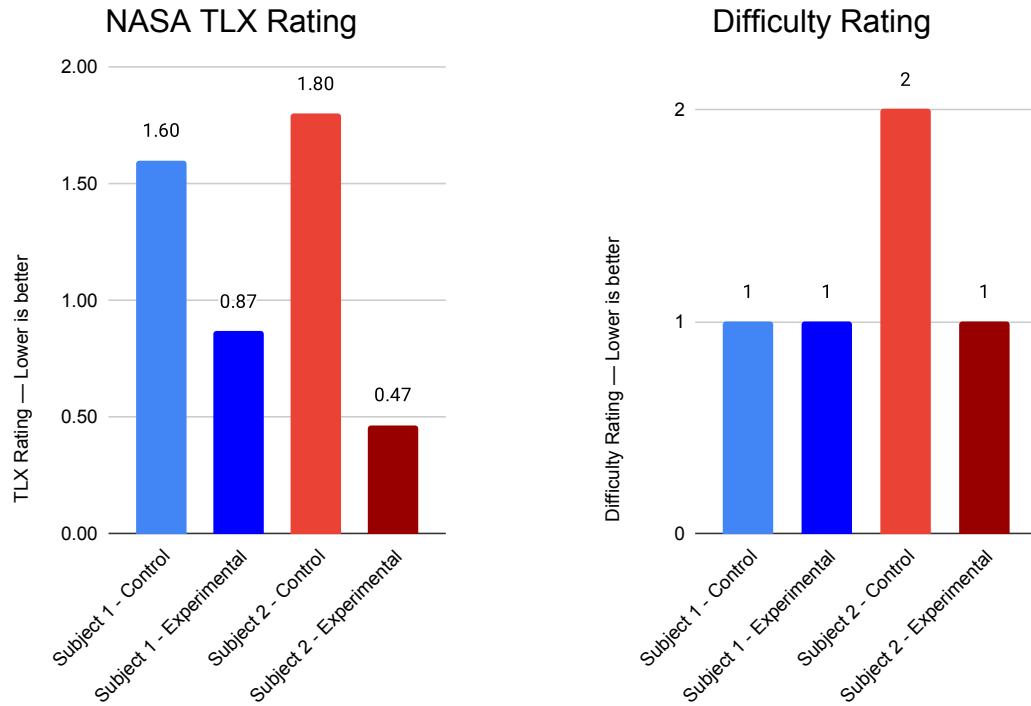


Figure 6.5: NASA TLX (left) and difficulty rating (right). The TLX result shows a clear preference for the experimental condition, while the difficulty rating is less conclusive.

with one operator completing the task 15% faster using the proposed method and the other 3% slower. In both cases, the subject's second trial was faster than the first, suggesting the time difference is in part caused by a learning effect. Although task order was not specified, the operators chose to approach the goals in the same order in all trials, allowing direct comparison of task segments. Fig. 6.4 shows that task segment duration has no strong correlation with experimental condition.

Unlike the objective metrics, the NASA TLX survey results show a correlation. The TLX rating for the experimental case is lower (more favored) than for the control case, by 0.73 for Subject 1 and 1.33 for Subject 2. While the

numerical difference is large, with only two subjects this result does not reach statistical significance (with $p = 0.18$), but it is sufficient to justify a larger trial.

For the difficulty rating, subjects are asked to rate the difficulty on a scale of 1 to 5. For all trials except one, the reported difficulty was the lowest allowed value, and the remaining trial was only one step above that. The low resolution of this rating and small sample size mean no conclusion can be drawn from these results.

6.6 Discussion and Conclusions

We presented an extension of the Interactive Planning and Supervised Execution (IPSE) teleoperation method to 7-DOF SRS manipulators, which are commonly used in ISAM applications. We use the SEW angle parameterization of redundancy, along with the concept of kinematic branches, to provide a human-friendly method of resolving the redundancy of the 7-DOF robot while still allowing full use of the robot’s workspace. A 2D display helps operators define a piecewise linear path for the SEW angle to follow over the course of a single waypoint’s execution, enabling easy traversal of narrow gaps where precise SEW angle control is required to avoid obstacles and joint limits. We evaluated the proposed interface with a pilot study in simulation and found justification to perform a larger study.

Future work includes implementation of the proposed kinematic branch switching functionality, improvements to the computation speed and resolution of the SEW graph, extension of the 7-DOF teleoperation functionality into the VR interface, and evaluation with a physical robot.

Chapter 7

Discussion and Conclusion

This thesis explores teleoperation methods for the purpose of In-Space Servicing, Assembly, and Manufacturing (ISAM), particularly focusing on the challenges of communications delay, non-ideal camera views, and the very high cost of failure.

7.1 Summary

Chapter 1 outlines the opportunities offered by ISAM, explains the need for ground-based teleoperation, and details the challenges posed by this unique environment. In Chapter 2 we consider a direct teleoperation approach, while also incorporating a previously-developed Augmented Reality (AR) visualization tool. We aim to imitate the direct teleoperation methods which are successful in surgical robotics but which cannot be used in the presence of multi-second time delay. A pilot study found that the combination of AR visualization and direct teleoperation led to significant improvement in task success rate, but we did not quantify how much of the improvement was due to the teleoperation method and how much was attributable to the AR visualization.

In Chapter 3 we conduct a detailed evaluation with trained experts in teleoperation for ISAM. From consultation with the experts, we develop a new conventional interface which mimics the one the experts are trained to use. We conduct an evaluation with three experimental conditions: conventional visualization and teleoperation, AV visualization with conventional teleoperation, and AV visualization with direct teleoperation. Results show that AV visualization improves performance and decreases operator workload, but direct teleoperation is detrimental to both performance and workload.

Based on the results detailed in Chapter 3, we develop the Interactive Planning and Supervised Execution (IPSE) system for ISAM teleoperation. Chapter 4 describes the initial development of IPSE. Rather than attempting to imitate methods from surgical robotics, we design a tool which allows for slow, deliberate planning, extensive verification, and execution with as much supervision as the time delay allows. The result is a visual planning environment where the operator builds a plan from waypoints, may visualize the plan’s execution in a virtual replica of the remote environment, and may modify the plan until satisfied with its safety (Interactive Planning). The operator can then execute the plan, in whole or in part, while monitoring its progress in the virtual environment and with camera views (Supervised Execution). IPSE supports multiple user interfaces, and we present a 2D interface using a standard display, mouse, and keyboard, and a 3D interface using the da Vinci surgical robot for visualization and user input. Our evaluation indicates that the 2D interface to IPSE significantly improves performance and operator workload, but the 3D interface performs worse than the conventional interface. We attribute this decrease in

performance to the operators' lack of familiarity with the da Vinci Master Console hardware and to key features which were present in the 2D interface but not the 3D interface.

In Chapter 5 we present an alternative 3D interface to IPSE which uses a Head-Mounted Display (HMD) and commercial handheld controllers in place of the da Vinci. We describe how the features of IPSE are adapted for use in the HMD, and conduct an evaluation of the new 3D HMD interface compared to the previous best 2D interface. We find that the new interface is comparable or better than the 2D interface in task performance and operator workload.

Finally, in Chapter 6 we extend IPSE with features to control the redundancy of a 7 degree of freedom (DOF) manipulator. We use the SEW angle parameterization to allow operator control of the redundant DOF with minimal workload overhead. We propose an interface for providing piecewise linear control of the SEW angle over the course of a waypoint's execution and a novel visualization of joint limit, collision, and singularity constraints in terms of the SEW angle path. A pilot study justifies further investigation into 7-DOF IPSE.

7.2 Contributions

The contributions of this thesis are the development of IPSE and several conclusions about effective teleoperation interfaces for ISAM applications. We determined that conveying human input directly to robot action is ill-suited for this task because of the likelihood of conveying accidental actions. With IPSE we showed that a system that allows detailed planning and verification before execution leads to higher task success and lower cognitive burden. In our user

studies we found that defining waypoints by manipulating objects in a 3D environment was easier for tasks with low precision requirements, such as gross motions, but in tasks which require finer positioning users prefer additional tools such as numerical inputs or a “snap-to-frame” feature with frames that are relevant to the task. We observed the importance of providing an intuitive method to move and scale the user’s view when precisely interacting with the 3D scene, since ideal viewpoint positioning helps users provide precise input; this was most visible in the comparison between the 3D da Vinci and 3D HMD interfaces to IPSE. Finally, we found that the SEW angle parameterization of redundancy was a useful tool in human-in-the-loop command of a 7-DOF robot, and that a 2D visualization of the redundancy space with marked regions of joint limits, collisions, and singularities assisted the user in designing feasible and safe trajectories for the redundant robot.

7.3 Future Work

There are many more opportunities in the area of ground-based teleoperation for ISAM. On the ground side, automated planning can be combined with IPSE to assist the user in developing a plan while still allowing full human oversight over what is executed. On a single-waypoint scale, the SEW graph provides a 2D environment with obstacles and disfavored areas which is well-suited for a low-dimensional path planner such as A*. On a full-plan scale, a path planner could automatically place the minimal number of waypoints necessary to reach the specified goal pose while avoiding joint limits, collision, and regions near singularity.

Another potential ground-based direction of research is in unifying the 2D and 3D interfaces. Currently, each has an advantage: the precision and familiarity of a mouse and keyboard make the 2D interface to IPSE easier to interact with for most tasks. However, the 3D interface offers depth perception, easier manipulation of the view into the 3D scene, and direct 6-DOF positioning of waypoints using hand controls. An interface which married the two may be able to offer the benefits of both.

In space, the area with the greatest potential is automated error detection. Currently, Interactive Planning comprises most of the implementation of IPSE. This is by necessity, as the communications delay means that timely reaction to environmental feedback on the ground is impossible. However, if a component of IPSE were able to run on the space side of the communications barrier, much more is possible. One especially useful ability for space-side processing is error detection. Each object in the environment could be tracked, and expectations of which objects will move in which way can be encoded into the plan which is sent to the space-side robot. During execution, the space-side processing could monitor the motion of each object and trigger an error condition if the motion of any object deviates from the expected. In cases where reaction time is critical, such as when an important object is accidentally freed from whatever mechanism keeps it in place and begins moving away, automated error recovery could also be beneficial.

Space-side processing could also open the door to more complex interactions between the robot and the world. For example, a cutting task may benefit from hybrid position-force control, and IPSE could be used to specify the position component of motion, the desired force or torque to be applied, and how to

resolve the redundancy for motions along the force-controlled axis. Similarly, many fine manipulation tasks may benefit from visual servoing, and IPSE could be used to indicate the target and to give safety bounds which the robot must stay within during the visual servoing task.

Appendix A

Installing and Running IPSE

To install the Interactive Planning and Supervised Execution (IPSE) system, visit https://git.lcsr.jhu.edu/teleop/satellite_surgery (access permissions and Johns Hopkins University user account required). To install the 6-DOF version using the UR10, stay on the default `master` branch. To install the 7-DOF version using the KUKA, switch to the `seven-dof` branch. Follow the instructions in `INSTALL.md` to install.

To run the system, first activate the ROS workspace with

```
source devel/setup.bash
```

or the equivalent setup file for your shell, within the workspace directory. If you installed the 6-DOF version, run IPSE using the command

```
roslaunch satellite_surgery sim_satellite_surgery.launch
```

If you installed the 7-DOF version, run IPSE using the command

```
roslaunch satellite_surgery sim_satellite_surgery.launch \
  space_robot:=kuka
```

References

- [1] V. Abbasi, B. Azria, E. Tabarah, V. Menon, E. Phillips, and M. Bedirian. “Improved 7-DOF Control of ISS Robotic Manipulators”. In: *Space OPS 2004 Conference* (2004).
- [2] T. Abuhamdia and J. Rosen. “Constant visual and haptic time delays in simulated bilateral teleoperation: Quantifying the human operator performance”. In: *Presence: Teleoperators and Virtual Environments* 22.4 (2013), pp. 271–290.
- [3] Z. Ai, M. A. Livingston, and I. S. Moskowitz. “Real-time unmanned aerial vehicle 3D environment exploration in a mixed reality environment”. In: *International Conference on Unmanned Aircraft Systems (ICUAS)*. 2016, pp. 664–670. DOI: 10.1109/ICUAS.2016.7502588.
- [4] A. K. Bejczy, W. S. Kim, and S. C. Venema. “The phantom robot: predictive displays for teleoperation with time delay”. English. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 1990, pp. 546–551. DOI: 10.1109/ROBOT.1990.126037.
- [5] A. K. Bejczy, W. S. Kim, and S. C. Venema. “The phantom robot: predictive displays for teleoperation with time delay”. In: *Proceedings., IEEE International Conference on Robotics and Automation*. 1990, 546–551 vol.1. DOI: 10.1109/ROBOT.1990.126037.
- [6] A. Bejczy and W. Kim. “Predictive displays and shared compliance control for time-delayed telemanipulation”. In: *IEEE Intl. Workshop on Intelligent Robots and Systems (IROS)*. 1990, pp. 407–412.
- [7] J. Chong, S. Ong, A. Nee, and K. Youcef-Youmi. “Robot programming using augmented reality: An interactive method for planning collision-free paths”. In: *Robotics and Computer-Integrated Manufacturing* 25.3 (2009), pp. 689–701. ISSN: 0736-5845. DOI: <https://doi.org/10.1016/j.rcim.2008.05.002>.

- [8] G. d. M. Costa, M. R. Petry, and A. P. Moreira. “Augmented Reality for Human-Robot Collaboration and Cooperation in Industrial Applications: A Systematic Literature Review”. In: *Sensors* 22.7 (2022). ISSN: 1424-8220. DOI: [10.3390/s22072725](https://doi.org/10.3390/s22072725).
- [9] P. Dahm and F. Joublin. “Closed form solution for the inverse kinematics of a redundant robot arm”. In: Institut für Neuroinformatik Ruhr-Universität Bochum Internal Report 97-08 (1997).
- [10] F. De Pace, F. Manuri, A. Sanna, and C. Fornaro. “A systematic review of Augmented Reality interfaces for collaborative industrial robots”. In: *Computers & Industrial Engineering* 149 (2020), p. 106806. ISSN: 0360-8352. DOI: <https://doi.org/10.1016/j.cie.2020.106806>.
- [11] A. Deguet, R. Kumar, R. Taylor, and P. Kazanzides. “The *cisst* libraries for computer assisted intervention systems”. In: *Midas Journal: MIC-CAI Workshop on Systems and Arch. for Computer Assisted Interventions* (2008). URL: <http://hdl.handle.net/10380/1465>.
- [12] M. Draelos, B. Keller, C. Toth, A. Kuo, K. Hauser, and J. Izatt. “Teleoperating Robots from Arbitrary Viewpoints in Surgical Contexts”. In: *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*. Vancouver, Canada, 2017, pp. 2549–2555.
- [13] A. J. Elias and J. T. Wen. *Redundancy parameterization and inverse kinematics of 7-DOF revolute manipulators*. 2023. URL: <https://arxiv.org/abs/2307.13122>.
- [14] H. Fang, S. Ong, and A. Nee. “Interactive robot trajectory planning and simulation using Augmented Reality”. In: *Robotics and Computer-Integrated Manufacturing* 28.2 (2012), pp. 227–237. ISSN: 0736-5845. DOI: <https://doi.org/10.1016/j.rcim.2011.09.003>.
- [15] W. R. Ferrell. “Delayed force feedback”. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 8.5 (1966), pp. 449–455.
- [16] M. M. Finckenor and D. Dooling. *Multilayer Insulation Material Guidelines*. Tech. rep. NASA Marshall Space Flight Center, Huntsville, AL United States, 1999.
- [17] A. Flores-Abad, O. Ma, K. Pham, and S. Ulrich. “A review of space robotics technologies for on-orbit servicing”. In: *Progress in aerospace sciences* 68 (2014), pp. 1–26.

- [18] J. Funda, T. S. Lindsay, and R. P. Paul. “Teleprogramming: Toward delay-invariant remote manipulation”. In: *Presence: Teleoperators & Virtual Environments* 1.1 (1992), pp. 29–44.
- [19] S. Y. Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex, and G. Konidaris. “End-User Robot Programming Using Mixed Reality”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2019, pp. 2707–2713.
- [20] J. Grasshoff, L. Hansen, I. Kuhlemann, and K. Ehlers. “7DoF Hand and Arm Tracking for Teleoperation of Anthropomorphic Robots”. In: *Proceedings of ISR 2016: 47st International Symposium on Robotics*. 2016, pp. 1–8.
- [21] G. S. Guthart and K. Salisbury. “The IntuitiveTM telesurgery system: overview and application”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2000, pp. 618–621.
- [22] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl. “Sensor-based space robotics-ROTEX and its telerobotic features”. In: *IEEE Trans. on Robotics and Automation* 9.5 (1993), pp. 649–663.
- [23] I. Ince, K. Bryant, and T. Brooks. “Virtuality and reality: a video/graphics environment for teleoperation”. In: *IEEE Intl. Conf. on Systems, Man, and Cybernetics (SMC)*. 1991, pp. 1083–1089.
- [24] Z. Jiang, X. Cao, X. Huang, H. Li, and M. Ceccarelli. “Progress and Development Trend of Space Intelligent Robot Technology”. In: *Space: Science & Technology 2022* (2022).
- [25] M. D. Johnston and K. J. Rabe. “Integrated planning for telepresence with time delays”. In: *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT’06)*. 2006. DOI: 10.1109/SMC-IT.2006.39.
- [26] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio. “An open-source research kit for the da Vinci[®] Surgical System”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2014, pp. 6434–6439.
- [27] Z. Kemeny. “Redundancy resolution in robots using parameterization through null space”. In: *IEEE Transactions on Industrial Electronics* 50.4 (2003), pp. 777–783.
- [28] D.-H. Kim, Y.-G. Go, and S.-M. Choi. “An Aerial Mixed-Reality Environment for First-Person-View Drone Flying”. In: *Applied Sciences* 10.16 (2020). ISSN: 2076-3417. DOI: 10.3390/app10165436.

- [29] D. Koppel, Y.-F. Wang, and H. Lee. “Image-based rendering and modeling in video-endoscopy”. In: *IEEE Intl. Symp. on Biomedical Imaging: Nano to Macro*. 2004, pp. 269–272.
- [30] Y. Koreeda, S. Obata, Y. Nishio, S. Miura, Y. Kobayashi, K. Kawamura, R. Souzaki, S. Ieiri, M. Hashizume, and M. G. Fujie. “Development and testing of an endoscopic pseudo-viewpoint alternating system”. In: *International Journal of Computer Assisted Radiology and Surgery* 10.5 (2015), pp. 619–628.
- [31] J. C. Lane, C. R. Carignan, and D. L. Akin. “Advanced Operator Interface Design for Complex Space Telerobots”. In: *Autonomous Robots* 11.1 (2001), pp. 49–58.
- [32] B. Ma, Z. Xie, Z. Jiang, Y. Liu, and Z. Wang. “An Efficient Inverse Kinematic Strategy for the 7-DOF Offset Space Manipulator with Arm Angle Parameterization”. In: *IEEE International Conference on Mechatronics and Automation* (2022), pp. 1732–1737.
- [33] M. A. Menchaca-Brandan, A. M. Liu, C. M. Oman, and A. Natapoff. “Influence of perspective-taking and mental rotation abilities in space teleoperation”. In: *ACM/IEEE Intl. Conf. on Human-Robot Interaction (HRI)*. 2007, pp. 271–278.
- [34] P. Milgram and F. Kishino. “A taxonomy of mixed reality visual displays”. In: *IEICE Transactions on Information and Systems* 77.12 (1994), pp. 1321–1329.
- [35] P. Mitra and G. Niemeyer. “Model-Mediated Telemanipulation”. In: *Intl. Journal of Robotics Research* 27.2 (2008), pp. 253–262.
- [36] NASA GSFC. *Restore-L Robotic Servicing Mission*. URL: <https://sspd.gsfc.nasa.gov/restore-L.html> (visited on 09/11/2017).
- [37] S. Ong, A. Yew, N. Thanigaivel, and A. Nee. “Augmented reality-assisted robot programming system for industrial applications”. In: *Robotics and Computer-Integrated Manufacturing* 61 (2020), p. 101820. ISSN: 0736-5845. DOI: <https://doi.org/10.1016/j.rcim.2019.101820>.
- [38] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish. “Recent progress on programming methods for industrial robots”. In: *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*. VDE. 2010, pp. 1–8.

- [39] W. Pryor, B. Vagvolgyi, A. Deguet, S. Leonard, L. Whitcomb, and P. Kazanzides. “Interactive Planning and Supervised Execution for High-Risk, High-Latency Teleoperation”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2020, pp. 1857–1864. DOI: 10.1109/IROS45743.2020.9340800.
- [40] W. Pryor, B. P. Vagvolgyi, W. J. Gallagher, A. Deguet, S. Leonard, L. L. Whitcomb, and P. Kazanzides. “Experimental Evaluation of Teleoperation Interfaces for Cutting of Satellite Insulation”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2019, pp. 4775–4781.
- [41] W. Pryor, L. J. Wang, A. Chatterjee, B. P. Vagvolgyi, A. Deguet, S. Leonard, L. L. Whitcomb, and P. Kazanzides. “A Virtual Reality Planning Environment for High-Risk, High-Latency Teleoperation”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 11619–11625. DOI: 10.1109/ICRA48891.2023.10161029.
- [42] C. P. Quintero, S. Li, M. K. Pan, W. P. Chan, H. M. Van der Loos, and E. Croft. “Robot Programming Through Augmented Trajectories in Augmented Reality”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2018, pp. 1838–1844.
- [43] *ROS.org / Powering the world’s robots*. URL: <http://www.ros.org/> (visited on 08/05/2014).
- [44] J. Sandoval, H. Su, P. Vieyres, G. Poisson, G. Ferrigno, and E. D. Momi. “Collaborative framework for robot-assisted minimally invasive surgery using a 7-DoF anthropomorphic robot”. In: *Robotics and autonomous systems* 106 (2018), pp. 95–106.
- [45] T. Sheridan. “Space teleoperation through time delay: review and prognosis”. In: *IEEE Trans. on Robotics and Automation* 9.5 (1993), pp. 592–606.
- [46] M. Shimizu, H. Kakuya, W. -. Yoon, K. Kitagaki, and K. Kosuge. “Analytical Inverse Kinematic Computation for 7-DOF Redundant Manipulators With Joint Limits and Its Application to Redundancy Resolution”. In: *IEEE Transactions on Robotics* 24.5 (2008), pp. 1131–1142.
- [47] N. Smolyanskiy and M. Gonzalez-Franco. “Stereoscopic First Person View System for Drone Navigation”. In: *Frontiers in Robotics and AI* 4 (2017). ISSN: 2296-9144. DOI: 10.3389/frobt.2017.00011.

- [48] E. H. Spain. “Stereoscopic versus orthogonal view displays for performance of a remote manipulation task”. In: *SPIE, Stereoscopic Displays and Applications III*. 1991, pp. 103–110.
- [49] M. Stein, R. Paul, P. Schenker, and E. Paljug. “A cross-country teleprogramming experiment”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 1995, pp. 21–26.
- [50] I. A. Sucas and S. Chitta. *MoveIt*. Version ROS Melodic. 1, 2020. URL: <http://moveit.ros.org>.
- [51] M. Sun, N. Dong, C. Jiang, X. Ren, and L. Liu. “Real-Time MUAV Video Augmentation with Geo-information for Remote Monitoring”. In: *Fifth International Conference on Geo-Information Technologies for Natural Disaster Management*. 2013, pp. 114–118. DOI: 10.1109/GIT4NDM.2013.15.
- [52] X. Tian, Q. Xu, and Q. Zhan. “An analytical inverse kinematics solution with joint limits avoidance of 7-DOF anthropomorphic manipulators without offset”. In: *Journal of the Franklin Institute* 358.2 (2021), pp. 1252–1272.
- [53] R. Y. Tsai and R. K. Lenz. “Real time versatile robotics hand/eye calibration using 3D machine vision”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE. 1988, pp. 554–561.
- [54] Y. Tsumaki and M. Uchiyama. “Predictive display of virtual beam for space teleoperation”. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*. Vol. 3. 1996, 1544–1549 vol.3. DOI: 10.1109/IROS.1996.569018.
- [55] Unity Technologies. *Unity Robotics Hub*. URL: <https://github.com/Unity-Technologies/Unity-Robotics-Hub>.
- [56] B. Vagvolgyi, W. Niu, Z. Chen, P. Wilkening, and P. Kazanzides. “Augmented Virtuality for Model-Based Teleoperation”. In: *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*. Vancouver, Canada, 2017.
- [57] B. Vagvolgyi, W. Pryor, R. Reedy, W. Niu, A. Deguet, L. Whitcomb, S. Leonard, and P. Kazanzides. “Scene Modeling and Augmented Virtuality Interface for Telerobotic Satellite Servicing”. In: *IEEE Robotics & Automation Letters* 3.4 (2018), pp. 4241–4248.

- [58] S. Vozar, Z. Chen, P. Kazanzides, and L. L. Whitcomb. “Preliminary study of virtual nonholonomic constraints for time-delayed teleoperation”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2015, pp. 4244–4250.
- [59] S. Vozar, S. Leonard, P. Kazanzides, and L. L. Whitcomb. “Experimental evaluation of force control for virtual-fixture-assisted teleoperation for on-orbit manipulation of satellite thermal blanket insulation”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2015, pp. 4424–4431.
- [60] S. Wang, Z. Liu, Z. Ma, H. Chang, P. Huang, and Z. Lu. “A closed-form solution for inverse kinematics of redundant space manipulator with multiple joint offsets”. In: *Advances in space research* 72.5 (2023), pp. 1844–1860.
- [61] M. Wonsick, T. Keleştemur, S. Alt, and T. Padır. “Telemanipulation via Virtual Reality Interfaces with Enhanced Environment Models”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 2999–3004. DOI: 10.1109/IROS51168.2021.9636005.
- [62] T. Xia, S. Léonard, A. Deguet, L. Whitcomb, and P. Kazanzides. “Augmented reality environment with virtual fixtures for robotic telemanipulation in space”. In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2012, pp. 5059–5064. DOI: 10.1109/IROS.2012.6386169.
- [63] T. Xia, S. Léonard, I. Kandaswamy, A. Blank, L. Whitcomb, and P. Kazanzides. “Model-based telerobotic control with virtual fixtures for satellite servicing tasks”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2013, pp. 1479–1484. DOI: 10.1109/ICRA.2013.6630766.
- [64] E. Yang and M. Dorneich. “The Emotional, Cognitive, Physiological, and Performance Effects of Variable Time Delay in Robotic Teleoperation”. English. In: *Intl. J. of Social Robotics* 9.4 (2017), pp. 491–508. DOI: 10.1007/s12369-017-0407-x.
- [65] L. S. Yim, Q. T. Vo, C.-I. Huang, C.-R. Wang, W. McQueary, H.-C. Wang, H. Huang, and L.-F. Yu. “WFH-VR: Teleoperating a Robot Arm to set a Dining Table across the Globe via Virtual Reality”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 4927–4934. DOI: 10.1109/IROS47612.2022.9981729.

- [66] W.-K. Yoon, T. Goshozono, H. Kawabe, M. Kinami, Y. Tsumaki, M. Uchiyama, M. Oda, and T. Doi. “Model-based space robot teleoperation of ETS-VII manipulator”. In: *IEEE Trans. on Robotics and Automation* 20.3 (2004), pp. 602–612. DOI: 10.1109/TRA.2004.824700.
- [67] J. Yu, T. Wang, Y. Shi, and L. Yang. “MR Meets Robotics: A Review of Mixed Reality Technology in Robotics”. In: *6th International Conference on Robotics, Control and Automation (ICRCA)*. 2022, pp. 11–17. DOI: 10.1109/ICRCA55033.2022.9828895.
- [68] F. Zacharias, C. Borst, and G. Hirzinger. “Capturing robot workspace structure: representing robot capabilities”. In: *IEEE International Workshop on Intelligent Robots and Systems (IROS)* (2007).
- [69] D. Zhang, F. Cursi, and G.-Z. Yang. “WSRender: A Workspace Analysis and Visualization Toolbox for Robotic Manipulator Design and Verification”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3836–3843.
- [70] Y. Zhang, Y. Liu, B. Cao, Y. Liu, B. Ma, and Z. Xie. “Joint Limit Optimal Inverse Kinematics of the 7-DoF Manipulator with Link Offset based on Semi-analytical Solution”. In: *IEEE International Conference on Robotics and Biomimetics* (2021), pp. 483–489.
- [71] L. Zhao, Y. Liu, K. Wang, P. Liang, and R. Li. “An intuitive human robot interface for tele-operation”. In: *IEEE International Conference on Real-time Computing and Robotics (RCAR)* (2016), pp. 454–459.
- [72] S. Zhou, H. Liu, C. Jiang, H. Du, Y. Gan, and Z. Chu. “Research on Kinematics Solution of 7-axis Redundant Robot Based on Self-motion”. In: *Chinese Automation Congress (CAC)* (2020), pp. 2622–2627.
- [73] S. Zollmann, C. Hoppe, T. Langlotz, and G. Reitmayr. “FlyAR: Augmented Reality Supported Micro Aerial Vehicle Navigation”. In: *IEEE Transactions on Visualization and Computer Graphics* 20.4 (2014), pp. 560–568. DOI: 10.1109/TVCG.2014.24.

Will Pryor

(full name: John William Rogers Pryor)

☎ +1 818 438 8260
✉ willpryor@jhu.edu
🌐 will.pryor.phd
🐙 github.com/beiju

EDUCATION

- Johns Hopkins University**, Baltimore, MD 2017–2023
PhD in Computer Science
Thesis: *Teleoperation Methods for High-Risk, High-Latency Environments*
Advisors: Peter Kazanzides and Simon Leonard
- Worcester Polytechnic Institute**, Worcester, MA 2016–2017
Master of Science in Robotics Engineering
Advisor: Dmitry Berenson
- Worcester Polytechnic Institute**, Worcester, MA 2012–2016
Bachelor of Science, Double Major in Robotics Engineering and Computer Science

WORK EXPERIENCE

- Robotics and Controls Intern — Verb Surgical** JUNE 2019–AUG 2019; MAY 2018–AUG 2018
- 2019: Designed and implemented a new control mode for Verb’s robotic platform.
 - 2018: Implemented dynamic compensation on a medical robot. Work included signal conditioning, MATLAB and C++ data analysis and software development, physical testing, and performance analysis.
- Web Developer — Little Weaver Web Collective** OCTOBER 2016–AUGUST 2017
- Web Developer — Giantsource Inc.** JUNE 2012–JANUARY 2016
- Designed and built client web applications using PHP and JavaScript, both independently and collaboratively.

RESEARCH EXPERIENCE

- Research Assistant — Johns Hopkins University** AUGUST 2017 – SEPTEMBER 2023
- Researched visualization and planning technologies to support telerobotic satellite servicing in Prof. Peter Kazanzides’ Sensing, Manipulation, and Real-Time Systems lab, leading to multiple publications.
 - Developed and evaluated full-featured human-in-the-loop motion planning system with multiple user interfaces, written in C++ and ROS.
 - Designed and implemented hybrid deep learning and classical-method needle localization system for magnetic needle control, leading to 2021 publication.
 - Engineered a multi-threaded, multi-process robot controller in Python, enabling simultaneous high-frequency control (over 1kHz), computationally intensive localization, and real-time monitoring.
- Researcher — Society for Internet Baseball Research** AUGUST 2017 – SEPTEMBER 2023
- Analyzed pitch-by-pitch event data from the internet baseball simulator Blaseball to characterize new game mechanics and emergent phenomena, enabling player optimization.
 - Leveraged analytical tools such as Support Vector Classification to reverse-engineer significant portions of the game code and enable third-party implementations. Collaborated using Discord and GitHub.
- Research Assistant — Worcester Polytechnic Institute** MAY – AUGUST 2016
- Researched incremental perception and its integration with motion planning in Prof. Dmitry Berenson’s lab, culminating in 2016 publication.

TEACHING EXPERIENCE

Head TA in Algorithms for Sensor-Based Robotics — Johns Hopkins University

SPRING 2019

- Held twice-weekly office hours, hours by appointment, and exam review sessions, graded homeworks, and coordinated other TAs' office hours and grading schedules.

TA in Operating Systems — Worcester Polytechnic Institute

SPRING 2016

- Held twice- or thrice-weekly office hours, answered student questions online.


SKILLS

Robotics Teleoperation, Inverse Kinematics, Motion Planning, Visualization, Localization
Programming C, C++, Python, Javascript, Typescript, Rust, MATLAB, Ruby, Java
Software & Platforms ROS, Linux, PyTorch, OpenRAVE, MATLAB, OpenCV, PCL, Arduino
Web Javascript, Typescript, JSX, HTML, CSS, SCSS, XSLT, React, Angular, SQL
Data Analysis Excel, numpy, pandas, scipy, scikit-learn, matplotlib, Jupyter


OPEN SOURCE SOFTWARE

Agent Annotator:  <https://github.com/beiju/agent-annotator>

A tool for efficient labeling of video frames with locations of known geometries for machine learning applications. Written in Rust (Rocket) and JavaScript (React).


Blaseball Analysis:  https://github.com/beiju/blaseball_analysis


A collection of scripts, commands, and queries for analyzing Blaseball data. Demonstrates knowledge of SQL, Python, numpy, matplotlib, and API programming.


Resim:  <https://github.com/xSke/resim>


Reverse-engineered portions of Blaseball from archived data. My contributions include data processing, multi-core support, and methods to derive equations from data, such as Support Vector Classification.


PUBLICATIONS


Will Pryor, Liam Wang, Arko Chatterjee, Balazs Vagvolgyi, Anton Deguet, Simon Leonard, Louis Whitcomb, Peter Kazanzides,  [*A Virtual Reality Planning Environment for High-Risk, High-Latency Teleoperation*](#), IEEE Int. Conf. on Robotics and Automation, London, United Kingdom, 2023.

Will Pryor, Yotam Barnoy, Suraj Raval, Xiaolong Liu, Lamar Mair, Daniel Lerner, Onder Erin, Gregory D Hager, Yancy Diaz-Mercado, Axel Krieger,  [*Localization and Control of Magnetic Suture Needles in Cluttered Surgical Site with Blood and Tissue*](#), IEEE Int. Conf. on Robotics and Automation, Prague, Czech Republic, 2021.

Will Pryor, Balazs Vagvolgyi, Anton Deguet, Simon Leonard, Louis Whitcomb, Peter Kazanzides,  [*Interactive Planning and Supervised Execution for High-Risk, High-Latency Teleoperation*](#), IEEE Int. Conf. on Intelligent Robots and Systems, Las Vegas, NV, 2020.

Will Pryor, Balazs Vagvolgyi, William Gallagher, Anton Deguet, Simon Leonard, Louis Whitcomb, Peter Kazanzides,  [*Experimental Evaluation of Teleoperation Interfaces for Cutting of Satellite Insulation*](#), IEEE Int. Conf. on Robotics and Automation, Montreal, Canada, 2019.

Balazs Vagvolgyi, **Will Pryor**, Ryan Reedy, Wenlong Niu, Anton Deguet, Louis Whitcomb, Simon Leonard, Peter Kazanzides,  [*Scene Modeling and Augmented Virtuality Interface for Telerobotic Satellite Servicing*](#), IEEE Robotics and Automation Letters Vol. 3, No. 4, October 2018, Presented at IROS, Madrid, Spain, 2018.

Will Pryor, Yu-Chi Lin, Dmitry Berenson,  [*Integrated affordance detection and humanoid locomotion planning*](#), IEEE Int. Conf. on Humanoid Robots, Cancún, Mexico, 2016.