Accepted manuscript. Published in: Proceedings of the 18th International Conference on Social Robotics (ICSR+ART 2026), Lecture Notes in Artificial Intelligence, Springer, London, UK. The final authenticated version will be available online at: https://doi.org/[DOI to be assigned]

Supervisory Control via AR for Teleoperation under Communication Delays: An In-Space Assembly Use Case

Ioannis Marios Stavropoulos[1], Ayse Kucukyilmaz[2], Carol Martinez[3], Daniel J. Finnegan[1], and Juan David Hernández[1]

[1] School of Computer Science and Informatics, Cardiff University, United Kingdom
{StavropoulosI, FinneganD, HernandezVegaJ}@cardiff.ac.uk
[2] School of Computer Science, University of Nottingham, United Kingdom
Ayse.Kucukyilmaz@nottingham.ac.uk
[3] Space Robotics Research Group (SpaceR), Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg
carol.martinezluna@uni.lu

Abstract. Rigorous safety standards in space robotics require human-in-the-loop operation, thus limiting the level of robotic autonomy that can be deployed in practice. While shared control approaches are effective in low-latency settings, high communication delays force operators into inefficient *move-and-wait* strategies. Supervisory control has therefore emerged as a promising alternative for distant space missions like station keeping duties and assembly. However, existing solutions often rely on non-scalable task representations, such as Finite State Machines (FSMs) or provide immersive interfaces primarily designed for low-level robot control. As a result, they offer limited support for specifying complex tasks and managing execution failures in a structured and recoverable way. In this work, we present an AR-supported supervisory control system for in-space assembly tasks that allows operators to specify assembly goals directly within a 3D AR environment and executes them using behavior trees (BTs). BTs enable failure detection and recovery, while the AR interface provides visual feedback when failures occur. We evaluate the proposed approach in a user study comparing supervisory control with a shared control baseline under two communication latency conditions ($0s$ and $2s$ delay). Our results indicate that supervisory control significantly reduces task completion time, operator effort and workload, while improving robotic motion efficiency and usability. Furthermore, supervisory control remains robust under communication delay, whereas shared control performance degrades.

Keywords: Supervisory control · Shared Control · Space Robotics · Augmented Reality (AR) · Behavior Trees (BTs) · Failure Recovery

1 Introduction and Related Work

Industry trends highlight the growing demand for scalable infrastructure and sustained operations in space [22, 8, 16]. Modern space architecture now adopts

modular designs that are optimized for robotic manipulation, e.g., interchangeable parts, which are designed to be 3D printed in orbit or launched folded or disassembled that rely on snap-on mechanisms like magnetic latches. Thus, such modular designs are paving the way to more autonomous robotic operations for in-space servicing, assembly, and manufacturing (ISAM) [24].

However, strict safety standards challenge the use of fully autonomous systems in replacing human involvement [16], hence human-in-the-loop operation remains the preferred paradigm. While traditional bilateral control teleoperation prioritizes operator authority, long and variable signal latencies in space make direct control problematic. Such conditions can result in unstable bilateral feedback loops, which make operators adopt a *move-and-wait* strategy [19], thereby degrading performance and ultimately making teleoperation unreliable.

Consequently, shared autonomy approaches that involve higher levels of robot autonomy emerge as preferred alternatives. Such approaches offer a broad class of systems in which both humans and robots collaborate in task execution, and distribute control authority across a spectrum of autonomy levels. Within this range, *shared control* refers to an implementation that congruently blends human input with robotic autonomy [1, 12, 3], while *supervisory control* allows high-level human oversight over largely autonomous robotic execution [20].

In settings with low delays, i.e., when the task involves a remote system operating in the Earth's orbit, shared control may be preferred as it allows the operator to retain nearly full control. Through the use of guidance and forbidden-region virtual fixtures [2], and haptic feedback, shared control has been tested in satellite servicing tasks, such as thermal blanket cutting, and it has been shown to reduce completion time and eliminate significant manipulation errors, compared to unassisted teleoperation under delay [27, 28, 13, 23]. Hulin et al. [11] proposed the Model-Augmented Haptic Telemanipulation (MATM) framework, which consists of a local model employing a predictive haptic feedback method, and a remote model that enables shared autonomous functionality of the teleoperated robot. However, Louca et al. [14] discuss how greater responsibility must be delegated to the robot when long delays are present, and how features of bilateral control such as haptic feedback, which shared control approaches often heavily rely on, might degrade performance as latency increases (i.e., $\geq 540ms$).

Although shared control remains technically viable as delays increase to the order of $2 - 3s$, which is the case for Earth-Moon missions [4], supervisory control is increasingly preferred for its potential to scale effectively to even higher delays. The nature of tasks involved shifts to space station keeping duties, such as routine maintenance, cargo transfer, and assembly. Existing supervisory control systems often rely on action templates to specify tasks. However, such a design restricts operators to only sequentially issuing basic, object-afforded manipulation commands to the robot via a 2D interface (e.g., by connecting a data interface probe to a port) [18], thus not allowing to specify more abstract goals and forcing operators to wait in-between manually specifying subtasks. [9] addresses this limitation by pairing a task sequencing framework with affordance templates to chain subtasks together. However, such specification relies on rigid,

finite state machine (FSM)-like structures, which become brittle as task specifications get larger and highly composed, hindering dynamic failure recovery.

Other works [17, 26] diverge from 2D interfaces, and propose virtual reality (VR) supervisory interfaces in an attempt to reduce cognitive load and enhance situational awareness. Nevertheless, these approaches are still focused on low-level robot control, and the underlying challenge of inflexible task representations still remains. To address this issue, an alternative is to use behavior trees (BTs) [16], which can break down tasks into smaller, reusable components (behaviors) that can be composed into complex objectives. Furthermore, BTs can also encode conditional logic and fallback behaviors to handle unexpected changes in the environment dynamically without re-planning. Their composable structure further allows for their expansion to accommodate new subtask behaviors, which can be particularly useful for failure resolution. While the framework proposed in [16] solves execution brittleness via modular BTs, its reliance on a 2D interface for visual programming and supervision misses out on the benefits of 3D visualization and interaction modalities enabled by immersive interfaces, which have been proven to be effective in the field.

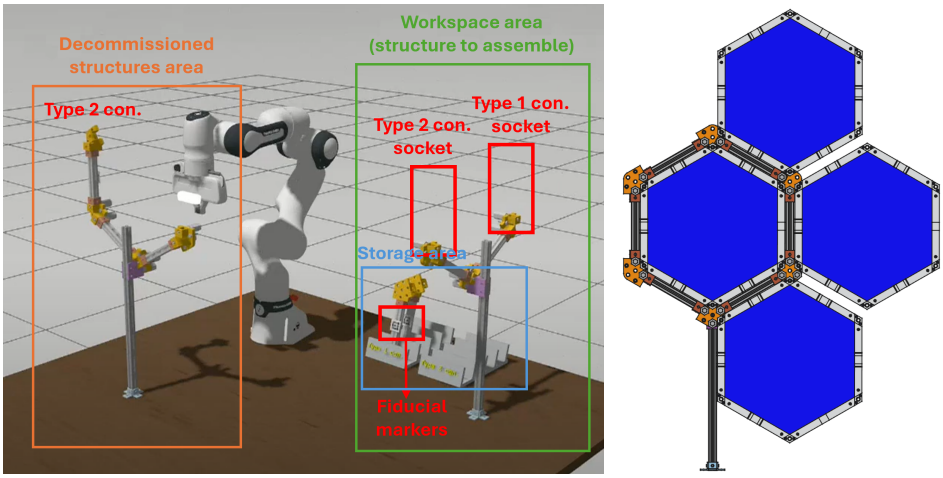
To address the critical gaps in teleoperation under delay–namely, high-level task specification, failure communication to the operator, and recovery–we propose a supervisory control system for in-space assembly that can handle complex tasks involving many sequential perception and manipulation operations. Our approach combines BTs with an immersive augmented reality (AR) supervisory interface to enable intuitive high-level task specification and failure recovery. Using an AR Head-Mounted Display (HMD), operators can virtually assemble a structure, thus specifying a high-level goal for a robotic manipulator. The poses of the placed parts are used as post-conditions in an automatically generated BT. The system further supports user intervention by visualizing failed behaviors in AR and dynamically replacing them with corrective subtrees in the BT. We demonstrate our system in a solar-panel structure assembly task highlighting a specific error visualization use-case. We evaluate the proposed approach through a user study, comparing supervisory control with an AR-supported shared control baseline under two communication delay conditions ($0s$ and $2s$). The results indicate that our supervisory control system improves task performance, robot motion efficiency, and usability, reduces operator effort and workload, and remains robust under communication delay compared to shared control.

2 In-Space Assembly Task and Use Case

To evaluate our supervisory control system, we selected a structured ISAM use case, where a robotic manipulator is teleoperated to assemble a solar panel structure in a remote space environment subject to communication delays.

2.1 Task Description and Environment

The manipulator is placed in the center of an environment divided into three distinct zones: 1) a workspace area, where the main structure under assembly is



(a) (b)

Fig. 1: (a) Task environment. (b) Example of a fully assembled structure.

located; 2) a storage area that contains spare parts grouped by type; 3) and a decommissioned structure area (See Fig. 1a).

The assembly process prioritizes retrieving various types of connectors and solar panels from their designated containers in the storage area. The environment also contains decommissioned structures that still have working, compatible parts attached to them available for disassembly and repurpose.

2.2 Failures and Operator Oversight

Executing an assembly task semi-autonomously in space is prone to unpredictable disruptions. During task execution, the system may encounter that the robot cannot find a part in the expected storage area. For example, the onboard perception system might fail to detect a specified connector because the storage area might be unexpectedly empty.

In such failure states, the robot cannot safely proceed on its own because it operates in a high-stakes environment. The operator must maintain a high-level oversight and have the final say before the robot executes any major operations, such as electing to disassemble a part from a decommissioned structure to use as a replacement.

2.3 Requirements for Task Representation and Failure Communication

This use case requires a flexible task representation capable of fully describing the entire assembly operation. It must accommodate the sequencing of heteroge-

neous robotic operations, including perception routines (e.g., scanning for fiducial markers on the parts), pick-and-place operations, and constrained assembly operations resembling a peg-in-hole insertion. In addition, the task representation must allow the task structure to be dynamically updated when failures happen, thus enabling recovery during execution.

Because the remote system operates under supervisory control, it must also support communication with the operator. In particular, failures occurring during task execution must be communicated to the operator together with sufficient contextual information to enable informed intervention. The underlying task representation, therefore, must encapsulate failure information to be visualized through the operator's interface, subsequently allowing the operator to expand the failing sequence with a corrective subtask based on their input.

These requirements motivate the use of BTs as the underlying task representation for our supervisory control system. BTs provide a modular representation of complex tasks in which failures can be detected and handled locally, enabling flexible recovery strategies without restructuring the entire task.

3 Supervisory Control System

The proposed supervisory control system (See Fig. 2) consists of the remote robot and the local AR supervisory interface. The remote system handles robot functionality for the task, including perception and motion planning. It also processes intermittent input, and integrates our BT approach, which handles errors by defining a control flow logic, with planning and interfacing with low-level robot functionality capabilities. The AR user interface enables the operator to specify task goals and is responsible for input validation, goal and error visualization, generating visual cues, reporting subtask status and displaying a model-based visualization of the remote environment.

3.1 AR Supervisory Interface

The AR interface allows the operator to specify assembly goals and monitor task execution. Goals are represented as green semi-transparent instances of target objects, which the operator can place directly onto the structure to virtually define desired assembly configurations (See Fig. 2). The published goal poses are used to specify the task, which is represented as a BT in the remote system. The interface also performs input validation, ensuring that goal parts are placed at valid attachment locations on the structure. Acceptable positions are visualized as yellowed semi-transparent instances of the part, which appear on the structure as the operator moves the part around the scene. Error visualization and correction manifest as a pause during task execution to show the step at which the robot has encountered an irregularity and requires clarification. Predefined error codes can be received from the robotic side describing the issue. Such error codes contain contextual information that is used to visualize different types of

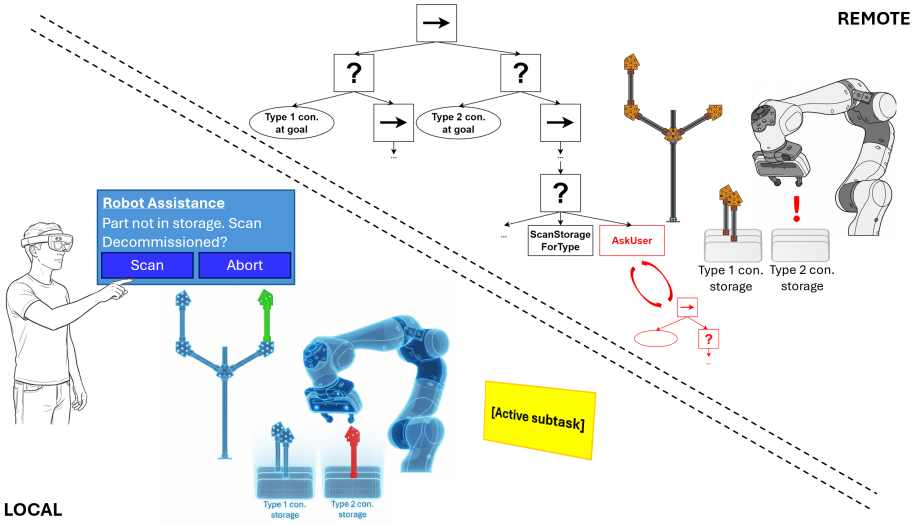


Fig. 2: Supervisory control diagram. A BT enables semi-autonomous operation of the remote robot. A model-based visualization of the remote environment is reconstructed around the operator in AR, mirroring the remote robot's configuration and interactions with objects. The operator has specified a task by placing green "goal" parts on the structure. The robot has not detected the part in its expected position, and pauses awaiting user response. A red part has appeared in the empty position and a prompt for robot assistance is shown to the operator. The BT is then expanded with a subtree and the task can resume.

failures and suggest prompts for resolution. A "wait" response from the AR application is sent to hold the execution of the underlying BT on the robotic side, before sending a recovery code response to extend the BT, thus replacing the recovery node with a new subtree accordingly. Status updates are also received and reported in the progress feedback area, which describes the robot's current subtask.

3.2 Remote Robotic System Implementation

As a supervisory control system does not rely on continuous input, a BT architecture backbone is required to enhance the decision-making capabilities of the robot, allowing it to fill in the low-level details that are not specified by the operator. BTs are a reactive and modular control architecture representing the execution of actions based on conditions and changes in the environment caused by external agents [6].

A BT consists of control-flow and execution nodes. Nodes can encapsulate various levels of functionality; the design choice can vary for different use cases and system capabilities. As the BT can be ticked at a fixed rate or control cycle, the flow of ticks enable it to retry, skip, or re-enter previous behaviors without

explicit re-planning, thus enabling reactivity, and its hierarchical composition of reusable subtree behaviors make it a modular and scalable approach. We use BehaviorTree.CPP[4], a C++ library for the construction and creation of BTs, and BehaviorTree.ROS2[5], a library for BT-Robot Operating System 2 (ROS2) integration.

BT Templates: To validate our AR supervisory control interface, we have chosen an assembly use case with a controlled task flow. We have also created predefined subtree templates, which can achieve tasks that are limited to this particular use case. A main sequence node accommodates a `goal` tree for each individual goal part specified by the operator. A `goal` tree is structured according to the Postcondition-Precondition-Action (PPA) format [7], which aims to satisfy a series of preconditions and postconditions of actions to detect, pick, and place an object. An `AskUser` subtree, typically re-runs an `AskUser` node, previously referred to as a "recovery" node, until a response is received from the operator. An action node that could fail, such as `ScanForObjectType` that attempts to locate any object of a specific type at a known location, is placed inside a `Fallback`, and followed by an `AskUser` node that can recover from not achieving the expected effect of the action node. In our case, we have created an `AskUser` node specifically for object detection failures. It is already parameterized with the specific error type that will be used by the AR interface to create a visualization of the error to the operator. Finally, we have subtrees for scanning the decommissioned structure, and setting the robot to predefined configurations. Action nodes may also be parameterized by the variables used within the node; the variable names will be refined when the template is implemented. For instance, a `Pick` node is parameterized by `object_id_goal#` to reference the specific ID of the identified object for the object type specified in goal #, so when the `goal` tree template is implemented the goal number in the variable name will be refined.

Creation and Execution of BTs: A ROS2 node listens for high-level goals from the AR interface, which consists of an array of object types and poses. BTs are drafted using Extensible Markup Language (XML) elements programmatically. For every goal in the goal array, the `goal` and `AskUser` subtrees are refined and appended to the `main` tree. In addition, a ROS2 action client is created for every action leaf node in the tree, such that when an action node is ticked, a request is made to the corresponding ROS2 action server including the parameters of the node. Once the BT has been fully constructed in XML, it is converted to an executable BT which is ticked repeatedly. A software layer bridges logic-level action nodes to robot actuation via ROS2.

Modification of BTs: After each tick the system checks whether a user instruction (sent from the task monitoring AR module) has been received for resolving a possible failure. If this is the case, the same routine of creating the tree is followed, and the relevant `AskUser` node is replaced by the subtree that corresponds to the operator's new request, and execution resumes from there.

[4] `https://github.com/BehaviorTree/BehaviorTree.CPP.git`
[5] `https://github.com/BehaviorTree/BehaviorTree.ROS2.git`

3.3 Implementation Details

The AR user interface is built in Unity and runs on the Magic Leap 2 (ML2) AR HMD. The headset supports 6-DoF tracking using a hand controller, enabling the operator to navigate the 3D environment and inspect the scene from multiple perspectives. The AR application connects to the ROS2 server via the Unity Robotics ROS-TCP-Connector package[6], which creates a layer of abstraction for subscribing and publishing to ROS2 topics.

The remote robotic system is implemented as a Gazebo Ignition simulation using the Franka Emika FR3 manipulator [10]. Motion planning and manipulation are performed using MoveIt!2 [5]. The Unity ROS-TCP-Endpoint[7] ROS2 package is used to complete the communication setup with the AR application. A camera is mounted on the robot's end effector (EE) in an eye-in-hand-configuration to enable Apriltag detection. Detected markers are used to detect the structures, estimate object poses, update the collision scene, compute grasp poses and automate picking in supervisory control.

The local model is a model-based visualization of the remote environment. This avoids the need to show the operator direct camera feeds from unintuitive eye-in-hand perspectives, which lack stereo visualization, and eliminates the effects of narrow field of view [21]. We assume every model in the environment and its pose is known. The pose data of static and tracked structure part models is extracted directly from the simulation in the ROS2 side, which is a provisional setup replaceable by a fully implemented perception pipeline. The configuration of the robot is also continuously visualized in AR by sourcing coordinate frame transformation data from the `/tf` topic.

4 Shared Control Baseline

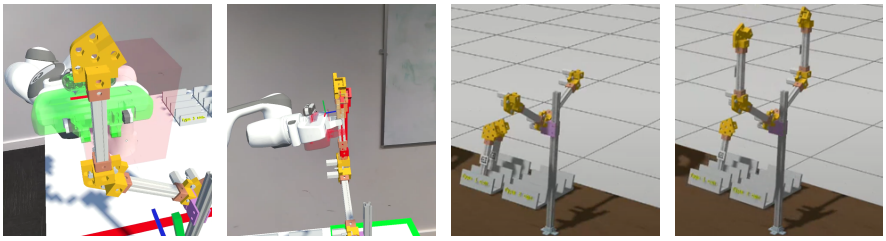
To evaluate the benefits of the supervisory control approach, we implemented an AR-based shared control system that allows the operator to directly guide the robot end-effector.

4.1 AR Shared Control Interface

The operator controls a virtual end-effector goal pose in AR (See Fig. 3a), which is continuously published to the robot at 20 Hz. Gripper commands (open/close) are issued on demand. When a structure part is grasped, a preview of the grasped object is attached to the goal end-effector to assist with placement. Kinematic constraints are enforced to keep the goal EE pose within workspace limits.

Collision checking is performed in the AR environment to ensure that commanded poses remain collision-free. If a virtual collision is detected, the goals turn red, and the goal pose is temporarily restricted until the robot reaches a

[6] `https://github.com/Unity-Technologies/ROS-TCP-Connector.git`
[7] `https://github.com/Unity-Technologies/ROS-TCP-Endpoint.git`



(a) (b) (c) (d)

Fig. 3: Shared control interface visualisations and context: (a) The operator controls the goal end-effector (EE) to scan the decommissioned structure for a Type 2 connector. After detection, the part appears in AR and the robot moves to match the commanded pose.(b) Collision handling during shared control: when the grasped object collides with the structure, the system visualizes the error by highlighting the object, and disables input until the robot returns to the last collision-free pose. (c) Initial state of structure. (d) Target assembly configuration with Type 1 and Type 2 connectors attached.

valid configuration (See Fig. 3b). Once the grasped part is close to an attachment location on the structure, the system evaluates the position and orientation offset from a perfect placement pose and allows opening the gripper only if the error is below a certain threshold[8].

While a part is in-hand, yellow indicators are shown at each compatible attachment location on the structure, providing visual cues for placement. While the goal is being moved, the indicators are dimmed such that they do not occlude the real and goal parts during placement. Task monitoring checks whether motion planning has failed or whether there are no valid poses to fall back to, in which case the robot resets to a safe collision-free configuration and the operator is required to continue the task from there.

4.2 Remote Robotic System Implementation

The commanded goal poses are forwarded to the motion planner, which generates feasible trajectories using MoveIt!2. If reachability cannot be verified, the goals are ignored. Motion planning is used instead of direct inverse kinematics control to ensure kinematic continuity and avoid unsafe configurations near singularities.

5 Methodology

We conducted a user study with 20 participants (16 males, 4 females, age = 29.5 ± 9.7) to evaluate the effectiveness of supervisory control versus shared

[8] Threshold values are empirically selected in our experiments as $\tau_l = 0.0065$ m and $\tau_h = 0.996$, respectively for position and orientation.

control in a space assembly task. The majority of participants (60%) had no experience with robotics or AR systems.

The study included two communication conditions: no delay and a $2s$ round-trip delay. Prior work suggests that delays around $2s$ correspond to the upper limit at which shared control approaches, such as model-augmented telemanipulation, can be used, and the minimum at which a supervisory control approach would be used [11]. Multiple delay levels were not considered in our experimental design, as the aim of the study was to discover the effects of the presence of delay and its interaction with the control method (shared vs supervisory) on performance, workload, and usability.

Ethical approval was obtained from Cardiff University's Ethics Committee (SREC reference: COMSC/Ethics/2025/003) prior to commencing the study. Participants signed an informed consent form after reviewing the hardware risk assessment and the participant information sheet detailing privacy and safety information.

5.1 Experimental Task

Participants undertook a subset of the assembly scenario described in Section 2. The task required them to attach two connectors ('Type 1' and 'Type 2') onto the workspace structure by commanding a Franka Emika robot arm. They were instructed to place a Type 1 connector on the left side and a Type 2 connector on the right side of the target structure (See Fig. 3d).

The storage area did not contain all the necessary components required to complete the assembly. As a result, participants needed to explore a nearby decommissioned structure to retrieve compatible connectors when the storage area was insufficient.

To assist assembly, the AR system highlighted valid attachment locations on the assembly structure. Connectors were equipped with Apriltag fiducial markers, and the robot's gripper camera was used to detect them. Participants were informed that markers needed to be visible to the camera in order for the parts to appear in AR environment.

5.2 Experimental Procedure

We used a 2×2 factorial within-subjects design with two independent variables: control method (shared control vs. supervisory control) and delay (no delay vs. $2s$ delay), comprising four conditions. To control for ordering effects, the four conditions were counterbalanced across participants using a Latin square design.

At the start of the experiment, the experimenter explained the task goals and requirements, demonstrated the workspace layout, and showed demonstration videos illustrating how the task is performed under each control method. Participants completed an interactive familiarization session, which lasted up to 10 minutes, before using a control method for the first time. This training allowed them to practice using the AR interface and understand the interaction

workflow prior to the experimental trials. To avoid biasing participants toward the experimental task, simplified training exercises were used: In supervisory control training, participants practiced creating and publishing a goal pose to place an object in a target container. In shared control training, they carried out a peg-in-hole task that mirrored the experimental workflow: moving the end-effector to capture a fiducial marker with the gripper camera, grasping it, and placing it into a container. These exercises familiarized participants with system behavior during object detection and collision handling.

After completing all four experimental conditions, participants completed a questionnaire assessing perceived workload and usability.

5.3 Evaluation Metrics

We collected both objective and subjective data. Objective data were logged automatically from the simulation and participants' AR interactions. The following metrics were selected to capture complementary aspects of system performance, including task completion time, operator effort, and robot motion efficiency, as well as subjective perceptions of workload and usability.

- **Task completion time**, reflecting the overall efficiency of task execution.
- **Controller distance**, computed as the total distance traveled by the participant's handheld controller. This metric provides insight into the amount of human input required to complete the task and serves as an indicator of operator effort.
- **End-effector distance**, computed as the total distance traveled by the robot end-effector. This metric quantifies the amount of robot motion required to accomplish the task and reflects robot motion efficiency.

Subjective workload and usability were collected through a post-experiment questionnaire. Workload was measured using individual NASA-TLX subscales on a 7-point Likert scale, following established precedents [15]. System usability was measured using the System Usability Scale (SUS) with a 5-point Likert scale. Both questionnaires were completed after participants had experienced all experimental conditions.

6 Results

This section reports the effects of control method and communication delay on task performance, operator effort, motion efficiency, workload, and perceived usability. Objective metrics were analyzed using aligned rank transform (ART) ANOVA [25], using the ARTool package in R, due to violations of normality in several conditions. Subjective measures (NASA-TLX and SUS) were analyzed using the same approach to maintain consistency across analyses.

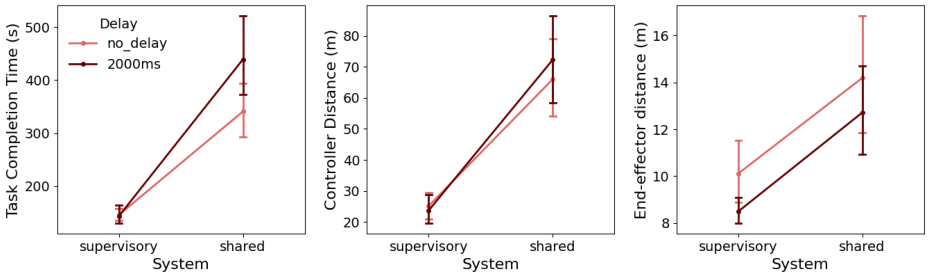


Fig. 4: Task completion time, controller distance, and end-effector distance across conditions. Error bars indicate 95% confidence intervals.

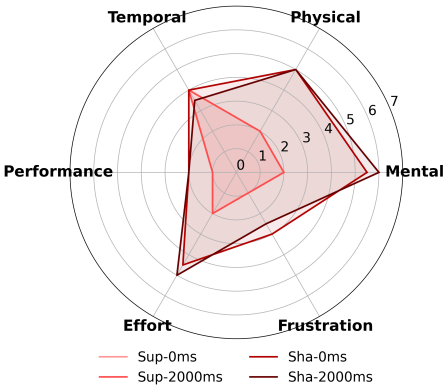


Fig. 5: NASA-TLX Subscales Across Conditions (medians)

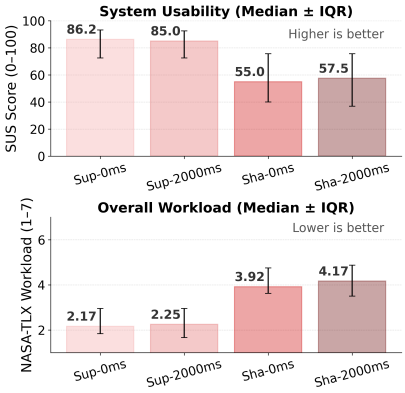


Fig. 6: SUS & NASA-TLX Summary Across Conditions

6.1 Task Performance

Task completion time was analyzed to evaluate overall task efficiency. As shown in in Fig. 4(left), participants required substantially more time under shared control (439.2 ± 38.6 s with delay, 340.9 ± 26.0 s without delay) compared to supervisory control (144.1 ± 9.2 s with delay, 146.3 ± 5.9 s without delay). ART ANOVA revealed a significant main effect of control method ($F(1, 57) = 194.56$, $p < 0.001$, $\eta_p^2 = 0.77$), indicating that participants completed the task significantly faster under supervisory control. A significant main effect of delay was also observed ($F(1, 57) = 5.80$, $p = 0.019$, $\eta_p^2 = 0.09$). In addition, a significant interaction between control method and delay was found ($F(1, 57) = 6.93$, $p = 0.011$, $\eta_p^2 = 0.11$), suggesting that the presence of communication delay had a stronger impact on task completion time under shared control than supervisory control. This interaction can also be observed in Fig. 4(left), where completion times increase under delay for shared control, while supervisory control remains comparatively stable across delay conditions.

6.2 Operator Effort and Motion Efficiency

Controller distance was analyzed as an indicator of operator effort (See Fig. 4 (middle)). Participants moved the controller substantially more when using shared control (72.2 ± 7.8 m with delay, 65.9 ± 6.5 m without delay) compared to supervisory control (23.7 ± 2.4 m with delay, 25.2 ± 2.2 m without delay). A significant main effect of control method was observed on controller distance ($F(1, 57) = 232.71$, $p < 0.001$, $\eta_p^2 = 0.80$). No significant effect of communication delay ($F(1, 57) = 0.24$, $p = 0.623$, $\eta_p^2 = 0.004$) or interaction between control method and delay ($F(1, 57) = 1.81$, $p = 0.184$, $\eta_p^2 = 0.03$) was observed. These results indicate that shared control required substantially more operator input than supervisory control.

Robot motion efficiency was evaluated using the total distance traveled by the robot end-effector. The robot traveled longer distances under shared control (12.7 ± 1.0 m with delay, 14.2 ± 1.3 m without delay) compared to supervisory control (8.5 ± 0.3 m with delay, 10.1 ± 0.7 m without delay). Statistical analyses revealed a significant main effect of control method ($F(1, 57) = 27.35$, $p < 0.001$, $\eta_p^2 = 0.32$), indicating that the robot traveled longer distances when tasks were performed under shared control. No significant effect of delay ($F(1, 57) = 1.66$, $p = 0.203$, $\eta_p^2 = 0.03$) or interaction effect ($F(1, 57) = 0.04$, $p = 0.849$, $\eta_p^2 = 0.001$) was observed. These findings suggest that supervisory control enables more efficient robot motion during task execution.

6.3 Workload

Perceived workload was measured using NASA-TLX scores (See Fig. 5. Participants reported higher workload under shared control (4.17, IQR = 1.38 with delay; 3.92, IQR = 1.12 without delay) compared to supervisory control (2.25, IQR = 1.29 with delay; 2.17, IQR = 1.13 without delay). ART ANOVA revealed a significant main effect of control method ($F(1, 57) = 61.56$, $p < 0.001$, $\eta_p^2 = 0.52$), indicating that supervisory control significantly reduced perceived workload compared to shared control. No significant effect of delay ($F(1, 57) = 0.06$, $p = 0.806$, $\eta_p^2 = 0.001$) or interaction between control method and delay ($F(1, 57) = 0.004$, $p = 0.948$, $\eta_p^2 < 0.001$) was observed. Overall, supervisory control significantly reduced perceived operator workload while communication delay did not produce a measurable change in workload ratings.

6.4 Usability

System usability was evaluated using the System Usability Scale (SUS). Participants rated the usability of the supervisory control mode higher than that of shared control. Median SUS scores were 86.25 (IQR = 20.62) and 85.00 (IQR = 20.00) for supervisory control under no-delay and 2s delay conditions respectively, compared to 55.00 (IQR = 35.62) and 57.50 (IQR = 38.75) for shared control. A significant main effect of control method was observed ($F(1, 57) = 42.90$, $p < 0.001$, $\eta_p^2 = 0.43$), indicating that supervisory control was perceived as

significantly more usable than shared control. No significant effect of delay ($F(1, 57) = 0.56$, $p = 0.459$, $\eta_p^2 = 0.01$) or interaction effect ($F(1, 57) = 0.03$, $p = 0.858$, $\eta_p^2 = 0.001$) was revealed.

7 Discussion

The results demonstrate clear advantages of supervisory control over shared control for the evaluated assembly task. Across both objective and subjective metrics, supervisory control enabled faster task completion, required less operator input, increased robotic motion efficiency, reduced perceived workload, and was rated as more usable.

As predicted by Sheridan [19], participants resorted to the move-and-wait strategy under a $2s$ communication delay, which reflected in the increase in task completion time (+28.8%). In contrast, supervisory control remained largely unaffected by latency, with similar results across the two delay conditions.

In addition, NASA-TLX and SUS results suggest that supervisory control could provide a more sustainable interaction model for long delay space missions, with workload ratings nearly 50% lower overall. While communication delay significantly affected task performance under shared control but did not produce a measurable change in perceived workload or usability.

Taken together, these findings highlight the potential of AR-based supervisory control as a practical alternative to continuous teleoperation for robotic assembly tasks in environments with unavoidable communication latency.

8 Conclusion and Future Work

This paper presented an AR-supported supervisory control paradigm on a remote robotic assembly task. The proposed system combines AR-based task specification with behavior-tree-based execution, enabling operators to supervise complex assembly tasks while delegating low-level manipulation to the robot.

A user study with 20 participants was conducted to evaluate the performance of the proposed supervised control method against a shared control approach under communication delay. The two control paradigms were compared across objective performance metrics and subjective measures of workload and usability.

The results showed that supervisory control enabled significantly faster task completion, required less operator input, and resulted in lower workload and higher usability ratings compared to shared control. Furthermore, task performance under supervisory control remained comparatively stable in the presence of communication delay, while shared control performance degraded.

This study has some limitations: Our experiments focused on a structured assembly task with a controlled sequence of subtasks. Consequently, the study did not fully explore the reactive capabilities of BTs when responding to disturbances caused by dynamic changes in the environment. Nevertheless, the supervisory control system successfully demonstrated the flexibility of BTs in representing

complex tasks and supporting failure recovery, highlighting the advantages of integrating them with our AR interface. The perception was simplified through the use of Apriltags and evaluation was done in a simulated environment.

Future work will integrate a perception pipeline that does not rely on Apriltags and extend the BT planner to support dynamic environments. We will also evaluate the approach with physical robotic systems and explore applications on different domains that share similar communication constraints, such as nuclear and underwater robotics, as well as investigate recovery action suggestion mechanisms following generative AI advancements. Our findings suggest that supervisory control combined with AR-based task specification can provide a robust alternative to continuous teleoperation for robotic assembly tasks in environments where communication latency is unavoidable. Such approaches may help enable more efficient human–robot collaboration in future space assembly and maintenance operations. Future work will explore more complex assembly scenarios and investigate system performance under a wider range of communication delays.

References

1. Abbink, D.A., et al.: A topology of shared control systems—finding common ground in diversity. IEEE THMS **48**(5), 509–525 (2018)
2. Abbott, J.J., Marayong, P., Okamura, A.M.: Haptic virtual fixtures for robot-assisted manipulation. In: Robotics Research: Results of the 12th International Symposium ISRR. pp. 49–64. Springer (2007)
3. Al-Saadi, Z., et al.: Resolving conflicts during human-robot co-manipulation. In: 2023 HRI. pp. 243–251 (2023)
4. Allan, M., et al.: Planetary rover simulation for lunar exploration missions. In: 2019 IEEE Aerospace Conference. pp. 1–19 (2019). https://doi.org/10.1109/AERO.2019.8741780
5. Coleman, D., Sucan, I., Chitta, S., Correll, N.: Reducing the barrier to entry of complex robotic software: a moveit! case study. arXiv preprint arXiv:1404.3785 (2014)
6. Colledanchise, M., Almeida, D., Ögren, P.: Towards blended reactive planning and acting using behavior trees. In: 2019 ICRA. pp. 8839–8845 (2019). https://doi.org/10.1109/ICRA.2019.8794128
7. Colledanchise, M., Ögren, P.: Behavior trees in robotics and AI: An introduction. CRC Press (2018)
8. Estable, S., et al.: Outcomes of the period project on in-space manufacturing, assembly and refuelling technologies. Journal of Physics: Conference Series (2023). https://doi.org/10.1088/1742-6596/2526/1/012121
9. Farrell, L., Strawser, P., Hambuchen, K., Baker, W., Badger, J.: Supervisory control of a humanoid robot in microgravity for manipulation tasks. In: 2017 IROS. pp. 3797–3802 (2017). https://doi.org/10.1109/IROS.2017.8206229
10. Haddadin, S., et al.: The franka emika robot: A reference platform for robotics research and education. IEEE Robotics and Automation Magazine **29**(2), 46–64 (2022). https://doi.org/10.1109/MRA.2021.3138382
11. Hulin, T., et al.: Model-augmented haptic telemanipulation: Concept, retrospective overview, and current use cases. Frontiers in Robotics and AI **8** (2021). https://doi.org/10.3389/frobt.2021.611251

12. Kucukyilmaz, A., Demiris, Y.: Learning shared control by demonstration for personalized wheelchair assistance. IEEE ToH **11**(3), 431–442 (2018)
13. Li, X., Kazanzides, P.: Task frame estimation during model-based teleoperation for satellite servicing. In: 2016 ICRA. pp. 2834–2839 (2016). https://doi.org/10.1109/ICRA.2016.7487446
14. Louca, J., Vrublevskis, J., Eder, K., Tzemanaki, A.: Elicitation of trustworthiness requirements for highly dexterous teleoperation systems with signal latency. Frontiers in Neurorobotics **17** (2023). https://doi.org/10.3389/fnbot.2023.1187264
15. Miyauchi, G., Lopes, Y.K., Groß, R.: Sharing the control of robot swarms among multiple human operators: A user study. In: 2023 IROS. pp. 8847–8853 (2023). https://doi.org/10.1109/IROS55552.2023.10342457
16. Moll, M.: A General Framework for Remote Command and Control of Robots in Space (2023). https://doi.org/10.2514/6.2023-1069
17. Pryor, W., et al.: A virtual reality planning environment for high-risk, high-latency teleoperation. In: 2023 ICRA. pp. 11619–11625 (2023). https://doi.org/10.1109/ICRA48891.2023.10161029
18. Schmaus, P., Leidner, D., Krüger, T., Bayer, R., Pleintinger, B., Schiele, A., Lii, N.Y.: Knowledge driven orbit-to-ground teleoperation of a robot coworker. RA-L **5**(1), 143–150 (2020). https://doi.org/10.1109/LRA.2019.2948128
19. Sheridan, T., Ferrell, W.: Remote manipulative control with transmission delay. IEEE Transactions on Human Factors in Electronics **HFE-4**(1), 25–29 (1963). https://doi.org/10.1109/THFE.1963.231283
20. Sheridan, T.B.: Telerobotics, automation, and human supervisory control. MIT press (1992)
21. Vagvolgyi, B., Niu, W., Chen, Z., Wilkening, P., Kazanzides, P.: Augmented virtuality for model-based teleoperation. In: 2017 IROS. pp. 3826–3833 (2017)
22. Verma, V., et al.: Autonomous robotics is driving perseverance rover's progress on mars. Science Robotics **8**(80), eadi3099 (2023). https://doi.org/10.1126/scirobotics.adi3099
23. Vozar, S., Léonard, S., Kazanzides, P., Whitcomb, L.L.: Experimental evaluation of force control for virtual-fixture-assisted teleoperation for on-orbit manipulation of satellite thermal blanket insulation. In: 2015 ICRA. pp. 4424–4431 (2015). https://doi.org/10.1109/ICRA.2015.7139811
24. Wang, Z., Wang, P., Duan, J., Tian, W.: Review of on-orbit assembly technology with space robots. Aerospace **12**(5) (2025). https://doi.org/10.3390/aerospace12050375
25. Wobbrock, J.O., Findlater, L., Gergle, D., Higgins, J.J.: The aligned rank transform for nonparametric factorial analyses using only anova procedures. In: Proc. CHI. pp. 143–146 (2011)
26. Wonsick, M., Padir, T.: Human-humanoid robot interaction through virtual reality interfaces. In: 2021 IEEE Aerospace Conference (2021)
27. Xia, T., Léonard, S., Deguet, A., Whitcomb, L., Kazanzides, P.: Augmented reality environment with virtual fixtures for robotic telemanipulation in space. In: 2012 IROS. pp. 5059–5064 (2012). https://doi.org/10.1109/IROS.2012.6386169
28. Xia, T., et al.: Model-based telerobotic control with virtual fixtures for satellite servicing tasks. In: 2013 ICRA. pp. 1479–1484 (2013). https://doi.org/10.1109/ICRA.2013.6630766