Analog Twin Framework for Human and AI Supervisory Control and Teleoperation of Robots

Nazish Tahir and Ramviyas Parasuraman, *Senior Member, IEEE*

***Abstract*—Resource-constrained mobile robots that lack the capability to be completely autonomous can rely on a human or artificial intelligence (AI) supervisor acting at a remote site (e.g., control station or cloud) for their control. Such a supervised autonomy or cloud-based control of a robot poses high networking and computing capabilities requirements at both sites, which are not easy to achieve. This article introduces and analyzes a new analog twin (AT) framework by synchronizing mobility between two mobile robots, where one robot acts as an AT to the other robot. We devise a novel priority-based supervised bilateral teleoperation strategy for goal navigation tasks to validate the proposed framework. The practical implementation of a supervised control strategy on this framework entails a mobile robot system divided into a Master–Client scheme over a communication channel where the Client robot resides on the site of operation guided by the Master robot through an agent (human or AI) from a remote location. The Master robot controls the Client robot with its autonomous navigation algorithm, which reacts to the predictive force received from the Client robot. We analyze the proposed strategy in terms of network performance (throughput and delay), task performance (tracking error and goal reach accuracy), and computing efficiency (memory and CPU utilization). Extensive simulations and real-world experiments demonstrate the method's novelty, flexibility, and versatility in realizing reactive planning applications with remote computational offloading capabilities compared to conventional offloading schemes.**

***Index Terms*—Analog twin (AT), cloud robotics, mobile robots, networked systems, supervised control, teleoperation.**

I. Introduction

CYBER–PHYSICAL systems (CPSs) encompass robots, autonomous vehicles, smart grids, and other such complex systems. Recent developments in Industry 4.0 Technology and the future Industry 5.0 innovations are driven through the advancements in CPS and robotic systems, with the concept of digital twin (DT) playing a pivotal role in it [1]. DT includes two components: 1) a physical device and 2) a digital replica. The digital module is the closest resemblance to the physical entity in a software/simulator environment. Here, both modules exchange information collected via network links, potentially including a cloud computing module. Besides being used as a prototype in digital space representing a physical entity in the real world, DT also allows for commanding control and remote supervision of tasks.

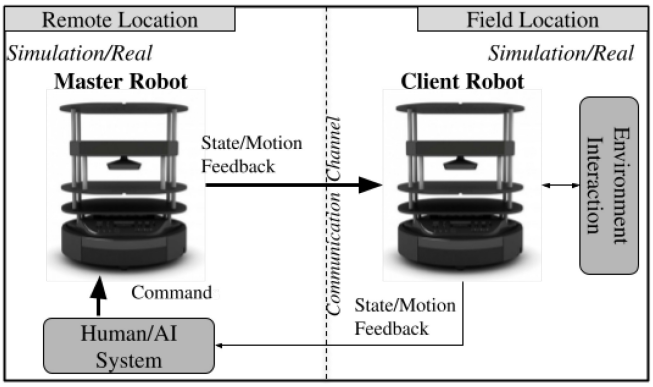


Fig. 1. Overview of the AT framework with a Master–Client architecture for mobile robot teleoperation.

In robotics research and development, DT provides the flexibility of many teams working independently and simultaneously without access to the physical robot, with the added benefit of safety for experimenting in a digital environment. Although DT demands high fidelity (increasing the requirement for data storage and transmission), it creates an opportunity to visualize the active state of the robot and test the process or models in dedicated offline training phases before deploying the models on real robots. However, DT is currently only used for design, simulations, modeling, or verification before deploying the models onto real robots. Currently, DT is limited in evaluating algorithms and models in synchronization with the actual robot operations [2].

We address this gap by proposing an analog twin (AT) framework, focusing on a mobile robot system. Our AT framework provides a unique way to verify robot algorithms and create new algorithms that can enable task or computation collaboration between the real robot and its twin robot. In contrast to the DT framework, the robots (whether physical or simulated) are synchronized in their mobility and actions in the AT, while digitally twin robots are typically asynchronous. The proposed AT framework can accommodate both simulated and real robots on both sides of the synchronized system. Fig. 1 depicts a supervised teleoperation application of a mobile robot operating in a real-world environment (field location) in collaboration with a remotely located analog twinned robot using the AT framework.

Manuscript received 11 August 2022; accepted 4 October 2022. Date of publication 3 November 2022; date of current version 17 April 2023. This article was recommended by Associate Editor C.-T. Lin. *(Corresponding author: Ramviyas Parasuraman.)*

The authors are with the Heterogeneous Robotics Research Lab, School of Computing, University of Georgia, Athens, GA 30602 USA (e-mail: nazish.tahir@uga.edu; ramviyas@uga.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TSMC.2022.3216206.

Digital Object Identifier 10.1109/TSMC.2022.3216206

Teleoperation and navigation are essential control tasks for mobile robots in real-world applications, including urban search and rescue scenarios [3], [4] and cooperative search and transportation [5], [6]. In most cases, a simple processor on the robot's most basic setup can only handle the sensor data from the cameras, laser scanners, actuators, etc. On the other hand, the real-time applications of robotics, such as artificial vision and object recognition and tracking, are resource-intensive with stringent requirements on the processing, modeling, and training of a constant stream of video data about the robot's surroundings. Their resources may become insufficient to do such resource-intensive operations, resulting in reduced computational performance. Such applications can be significantly benefited by cloud robotics [7], [8], by allowing cooperative computation with robots and sharing resources among themselves or offloading calculations to a resource-rich remote server/cloud. The processed outputs are delivered back to the robots for incorporation into the overall computation since the expensive computations are conducted either by robot–robot or robot–cloud collaborative computing [9].

This article applies an AT framework to a mobile robot system, focusing on a supervised teleoperation and navigation task. Our objective is to enhance robot resource efficiency, connectivity, and usage by offloading computationally heavy operations to remote computers/servers. We aim to improve the following key performance metrics that are relevant to cloud-connected robots [10]: computing workload, network latency, and data throughput without affecting the control task performance.

The main contributions of this article are as follows.

1) We propose a new framework termed AT, exploring the novel idea of synchronized (supervised) control of mobile robots through a networked system. The new framework enables us to utilize the capabilities of a fully autonomous robot to operate another robot for algorithmic collaboration and verification remotely.
2) We implement a new priority-based bilateral teleoperation strategy on the AT framework to share the networking and computing resources between two robots for realizing supervised control and navigation.
3) We construct a collaborative robotics scenario in which the less computational field robot offloads its intensive computational tasks to a remote (or cloud) server that handles the calculations, thereby reducing the field robot's computational overhead cost.
4) We validate the proposed system in various situations with increasing complexities through extensive simulation experiments with multiple obstacle settings and demonstrate the approach with real-world robot (hardware) demonstrations. We look at each instance's navigation, latency, communication, and computation performances.
5) We analyze the performances compared to conventional onboard computation and remote offloading schemes.

The proposed strategy enhances remotely offloaded robot navigation tasks by achieving improved computing and network performances without losing the bilateral system's tracking (control) performance. The proposed contributions are critical to enhancing the CPS and robotics research in the field and industrial applications by creating a new class of AT-based telerobotics and CPSs.

II. Related Work

We first analyze the literature from the perspective of robot teleoperation before highlighting how we depart from the related work. Then, we look at the critical challenges associated with achieving a reliable mobile robot teleoperation system with the AT framework from a wireless network standpoint.

Teleoperation of a robot refers to controlling a robot remotely via a controller device (e.g., a haptic controller for a manipulator or a gamepad controller for a mobile robot). Bilateral teleoperation is a well-researched topic, which refers to the architecture of two-way (simultaneous or synchronous) control between a Master and a Client robot [11]. Several bilateral teleoperation control laws and protocols have been proposed to address the challenges of control stability, time-delayed networks, etc. [12]. For instance, Franken et al. [13] resolved the time-varying destabilization issue of the bilateral system caused by the relaxed grasp of the user, hard contacts, stiff control settings, or communication delays by splitting the control architecture into two layers. The top layer is responsible for implementing a strategy to achieve desired transparency, and the bottom layer ensures that no virtual energy is generated. Separate communication channels connect the layers at the Client and the Master side. On the same lines, a recent work [14] introduced a control framework for bilateral teleoperation for a manipulator remote robot to ensure position tracking of the end effector while carrying out subtask control such as obstacle avoidance.

While most of the work in bilateral teleoperation focuses on manipulators [15], researchers have also looked at applying this concept to mobile robots [16], [17], [18], [19]. In [20], a new feedback force rendering scheme for the teleoperation of mobile robots is proposed by adaptively tuning it based on range-based distance measured from the obstacles and time derivatives of those distances. The proposed methodology of variable force feedback gain performs significantly better than the conventional feedback schemes proposed in the literature. Kawai et al. [21] proposed a control law for achieving bilateral teleoperation on a mobile robot, using a virtual Master robot that drives the real Client robot in a remote environment. The stability of the system is achieved by utilizing the scattering theory, and passivity-based control scheme [16]. In another work [22], researchers used a haptic teleoperation control framework for multiple unmanned aerial vehicles (UAVs) to demonstrate how a single remote human user can stably teleoperate multiple distributed UAVs with some helpful haptic feedback over the Internet with varying delay, packet loss, and multiple layered abstractions for control. Bu et al. [23] implemented a similar bilateral teleoperation technique by using the online generation of virtual fixtures for bilateral teleoperation based on intention recognition. Similar to [13] involving a single-Master–single-Client system, [24] puts forward a novel scheme for multi-Master–multi-Client

teleoperation by decomposing the system into Master–Client pairs and implementing the standard duo; coordination strategy and transparency layer on them all the while using only two shared energy tasks between the Masters and the Clients. Similarly, [25] proposed bilateral teleoperation of multirobots by a single Master over a delayed network by scheduling communication. At each sampling instant, only one Client can transmit the information over the communication channel via scheduling protocols like Round Robin.

As we can see, the great bulk of research is devoted to ensuring the stability of closed-loop systems with a human in the loop. These stability issues are tackled by either time-domain passivity control or scattering theory. Sometimes the emphasis is placed on maintaining stability with model mismatches that comes into play. Contrary to these works, we differ by providing a new perspective to bilateral teleoperation [26] and propose a novel priority-based switching (multiplexed) control for bilateral teleoperation achieving an open-loop synchronization between a Master and a Client robot embedded into our AT framework that can accommodate both human and an artificial intelligence (AI) agent in the control loop.

We use an open-loop control method to reduce stability concerns and eliminate the requirement for model mismatch because the Master device is a mobile robot [21]. Second, the literature is keen primarily on devising alternate methods of force feedback calculation. The feedback imposed on a Master device has to be a haptic or tactile force that creates a sense of transparency for the human operator. In our work, we use predictive force (indicative of the obstacles ahead) to provide the Master robot with the essential cue to reroute its path for safer supervised navigation. Further, our framework is aided by the recent developments in remote computation technology. This enables the capabilities of a real-time algorithm adaptation and supervised control system using AI or a human operator.

The key challenge here is to balance networking, computing, and control performances, especially since network connectivity is crucial and should be rapid and seamless for realizing a stable, high-performance teleoperation system. However, we cannot send all the sensor data to the remote device every time, and realistic wireless channels exhibit intermittent, low bandwidth, and unreliable transmission, mainly due to the mobility of the robot [27]. While the computational performance such as CPU/Memory utilization is optimized through the use of remote computation [28], we must consider the network-related parameters, such as the delay and throughput as they will have a significant impact on the control performance of the proposed system [17], [29]. Also, the need for predictive feedback that is not designed to be applied on a joystick is particularly challenging because much of the work in the literature focuses on remote teleoperation. Still, only a few have delved into the autonomous (with AI in the loop) bilateral teleoperation domain. Therefore, this article aims to improve the supervised teleoperation task through remote offloading and achieve higher computation and network performances at the field robot compared to the conventional schemes of remote offloading and teleoperation.

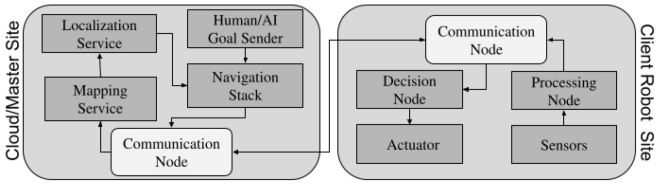


Fig. 2. Architecture of a remotely offloaded mobile robot navigation task example.

III. Background and Preliminaries

Our approach uses remote bidirectional teleoperation to allow a Master robot to help a Client robot move to a destination in an unknown environment using map-based path planning. The Master robot receives the predictive feedback force from the Client to determine the field environment.

Specifically, through the proposed AT framework, we envision a teleoperation system that relies on an AI agent (or a human operator), such as an autonomous robot (Master) located at a cloud/remote resource, to remotely control another computationally less intelligent robot (Client) through a network. The Client follows the Master's path from the remote location while delivering raw sensory data back to the AI at the Master. The AI uses this sensor data to control the Master, which is reflected at the Client, allowing it to execute complicated tasks like obstacle avoidance and navigation at the remote location while simultaneously offloading computing loads from the Client. Preliminaries to the main aspects of our proposed scheme are discussed as follows.

A. Navigation and Path Planning

To navigate across an unknown environment while avoiding collisions, a robot has to typically solve the simultaneous localization and mapping (SLAM) problem [30]. SLAM provides an estimate of the map of the robot's environment while localizing the robot in relation to that map, resulting in a joint estimation problem. Different probabilistic formulations are used in approximating the pose of the robot and map, e.g., extended Kalman filters (EKFs) [31], Markov [32], Monte Carlo particle filter (MCL) [33], and Rao–Blackwellization technique [34]. Appendix A in the supplementary material provides further details on path planning using the SLAM map, specifically the dynamic window approach (DWA) [35] planner used in this study.

B. Remote Computation

The idea of remote computing and cloud robotics has sparked a lot of curiosity in the recent decade. Localization, navigation, perception, and mapping-related tasks can be offloaded to a remote workstation or a cloud because of the limited computing capability of the mobile robots [36]. The basic theme of a remote computation setup is shown in Fig. 2. For instance, Salmerón-García et al. [28] described the viability of using the cloud for offloading low-level and intensive computing tasks such as the vision-based navigation assistance of a mobile service robot. Benavidez et al. [37] proposed offloading the SLAM processing to clouds with cloud-based

deployment. Another work in [38] suggests a cloud-based collaborative Visual SLAM system for transferring vast amounts of data.

Arumugam et al. [39] used a cloud service running the FastSLAM algorithm, demonstrating a significant improvement in execution time compared to running the SLAM on the robot. Another paper [40] proposes a cloud-based (i.e., remote computing) architecture for large-scale autonomous robots that utilize three subsystems, including the middleware subsystem, background task subsystem, and control subsystem. These jobs may need immediate or deferred processing.

In our work, we use remote computation by outsourcing the tasks of mapping and localization to the remote mobile robot, Master, with better computational capability to perform these tasks based on the sensor data received from the less capable on-field robot (Client). The Client robot on the field navigates to the desired goal position through this process.

C. Force Feedback

In classical bilateral mobile robot teleoperation, the human operator exerts a forward force on the Master device (manipulator or a controller) that causes displacement transmitted to the Client robot to mimic the same movement. If the Client robot has force or torque sensors, it sends a reflective force back to the Master. This reaction force comes from its interaction with its environment, which enters into the input torque of the Master, and the Client is said to be *bilaterally* controlled.

With the use of force feedback, the operator's control over the robot's motion can be significantly improved [19]. For the feedback force, haptic or visual devices are typically introduced into the system at the Master to enhance the human operator's sense of telepresence by presenting the interaction force with the environment acting on the Client robot. These devices offer a new mode of perception and transparency of the remote environment by enhancing a user's depth perception as well as environmental awareness with obstacle detection.

In a predictive force feedback system, a system of controllers is used based on the mobile robot's distance to the obstacles in the environment such that the force is perceived (predicted) before actual contact with the environment [41]. With the help of this information, the operator avoids navigating into the proximity of the obstacles and enhances the telepresence of the remote user. As a robot's distance from nearby obstacles decreases, a force reciprocal to the distance from the obstacle is computed and delivered through the haptic device, which the user perceives as a force and steers the robot toward a smooth and obstacle-free route. Such a system is highly effective in search and rescue mission scenarios [42]. We introduce this predictive force into our teleoperation system to create the bilateral loop, where the feedback augmentation reflects the proximity of obstacles in the steering direction.

IV. Proposed Methodology

In our work, a teleoperation scheme for a single-Master–single-Client system is set up based on the proposed AT framework, where both the Master and Client robots are coupled through a bilateral control strategy. The method addresses the problem of situation transparency by implementing reflective commands on the Client and enhances better navigation by prioritizing path planning and teleoperation. We evaluate computational offloading on the remote agent that functions as an AI controller via a wireless network.

We introduce prioritized bilateral teleoperation through a Master robot situated on the remote server, which provides map-based path planning to the Client. The Client, in turn, provides reactive feedback for corrective navigation achieving overall goal point precision and obstacle avoidance over a wireless network. Unlike conventional approaches, this strictly predictive feedback force is provided to an autonomous Master robot, based on expected contact with the environment, rather than an impact force traditionally applied to a haptic device operated by a human. While executing autonomous map-based navigation directed by a path planner, the Master robot assures rerouting of the Client around obstructions by leveraging feedback.

Suppose there is a human in the loop (e.g., to provide emergency interventions), the Master robot functions as a physical surrogate of the remote agent on the control site. The predictive force applied to the Master comes from the Client and serves for improved awareness for the human operator.

In a nutshell, the proposed methodology comprises a *priority-based bilateral control* to teleoperate/navigate a Client robot toward a specified goal through an autonomous Master robot remotely over a wireless network. The Client robot subscribes to the Master's state to replicate the Master's motion at the field site. However, the Client robot comes equipped with an added layer of safe navigation for reactive obstacle avoidance provided through a fast, low-intensive *predictive force feedback* computation conveyed to the Master robot through a high-priority channel. We employ a velocity multiplexer (MUX) on both sides for switching control between different control velocities based on priority. This MUX-based architecture generates open-loop control, which ensures system stability. An overview of our idea is shown in Fig. 3. Below, we describe the key components of our method.

A. Predictive Force Computation

Consider the wheeled robot, with control inputs consisting of linear velocity v_m and the angular velocity ω_m of the Master robot are derived from its position (x_m, y_m) and rotation θ_m. The kinematics model of a wheeled robot is given in

$$\begin{bmatrix} x_m \\ y_m \\ \theta_m \end{bmatrix} = \begin{bmatrix} \cos(\theta)_m & 0 \\ \sin(\theta)_m & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_m \\ \omega_m \end{bmatrix}. \tag{1}$$

The control of the Client robot is based on the Master's linear and angular velocities, as shown in

$$\begin{pmatrix} V_c \\ \omega_c \end{pmatrix} = \begin{pmatrix} k_V & 0 \\ 0 & k_\omega \end{pmatrix} \begin{pmatrix} V_m \\ \omega_m \end{pmatrix} \tag{2}$$

where k_V and k_ω are scaling constants to account for dissimilar kinematics or scale (robot dimensions). Our proposed predictive force delivered via the Client robot is based on

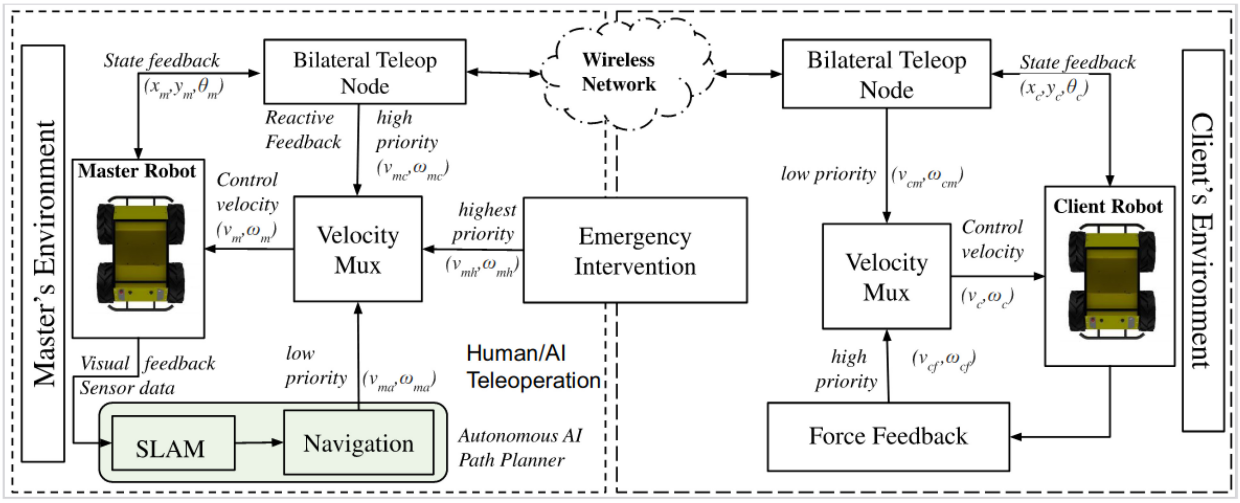


Fig. 3. Proposed priority-based bilateral teleoperation mechanism to enable Master–Client coupling for AI-driven navigation.

obstacle range information. This force is calculated based on the measured distances of the mobile robot to the obstacles coming in through the range values from the LIDAR data, demonstrated through

$$\Delta = R_o - R = (\delta_1 \quad \delta_i, \ldots, \delta_n) \tag{3}$$

where $i = 1, \ldots, n$ and n is a number of ranges (or directions) in the sensor data. R is a polar vector that represents the distance of the robot to the obstacles r_i. While R_o is a vector of distance from the obstacle r_{oi} which generates the predictive force. If the distance from the closest obstacle from the robot at any given instance of time becomes less than r_{oi}, it generates the force reflected back to the Master robot via feedback force. Δ defines the difference between R_o and R. In short, the following basic law governs the principle of predictive force applied:

$$f_i = \begin{cases} \frac{k_i}{\delta_i}, & r_i < r_{oi} \\ 1, & r_i = r_{oi} \\ 0, & r_i \geq r_{oi} \end{cases} \tag{4}$$

$$F = (f_i \quad f_2, \ldots, f_n) \tag{5}$$

$$F \leftarrow f_i. \tag{6}$$

Here, f_i is a predictive force calculated inversely proportional to the obstacle distance δ_i measured via ith sensor (or ith direction) with a multiplication gain (weight) k_i for that direction such that $f_i \geq 1 \forall r_i \geq r_{oi}$. k_i is usually kept constant, but can also be a variable depending on the scenario.

B. Priority-Based Control Strategy

Similar robots, either real or simulated, are considered on the Master and Client sides. The Master robot is controlled by AI input in a low-priority channel, while the Client robot is controlled using (2) on a low-priority channel. Both are physically and logically separate entities connected via a communication channel with time-varying delay characteristics. The Master acts as an enabler for the resource-constrained Client by allowing it to "offload" computational and storage-intensive activities like mapping, localization, and navigation to the Master.

The proposed priority-based Master–Client coupling performs concurrent bidirectional teleoperation on a low-priority queue through a velocity MUX on both sides. A MUX arbitrates incoming commands velocity messages from several topics, allowing only one channel (topic) at a time to publish, based on priorities. The bilateral coupling works by first enabling the Master robot to subscribe to the velocity commands from the Human/AI on a low-priority channel while, in parallel, the Client subscribes to the Master's velocity on a low-priority channel. Second, both agents subscribe to the reactive feedback force on a high-priority channel. This force coupling with the Master, based on the predictive force in (6), aids the Client in navigating its local environment while avoiding obstacles.

Specifically, we apply the below reactive force feedback (V_r, ω_r) on the high-priority MUX channels of both the Client and the Master robots. This feedback is designed in a way to turn the robot around the nearest obstacle direction

$$V_r = f_{\text{safe}}(d)$$

$$\omega_r = f(F) = \begin{cases} \omega_{\text{turn}}, & \max(f_i) \geq f_{th} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$\begin{pmatrix} V_m \\ \omega_m \end{pmatrix} = \begin{pmatrix} V_c \\ \omega_c \end{pmatrix} = \begin{pmatrix} k_V & 0 \\ 0 & k_\omega \end{pmatrix} \begin{pmatrix} V_r \\ \omega_r \end{pmatrix}. \tag{8}$$

Here, ω_{turn} is a fixed angular velocity, $f_{\text{safe}}(d)$ is a function to apply safe linear velocity, and f_{th} is a force threshold.

The above force feedback mechanism (7) is only taken as an example to show the effectiveness of the AT framework. However, it is possible to apply complex controllers to this framework to achieve AI verification-related objectives, e.g., we can design functions that can provide disturbances in the system to simulate uncertainty in the sensor data or the environment to verify the robustness of an AI controller (at the Master). Further, we can consider the Master robot as the AI teleoperator employing SLAM-based navigation to steer itself and the Client robot to a specified target in the Master's frame

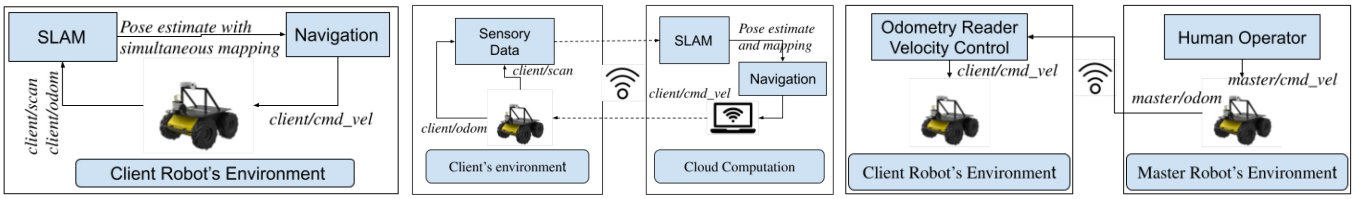


Fig. 4. Thematic representation of the experiment cases analyzed in this experimental study. Here, "odom," "scan," and "cmd_vel" are the ROS topics for odometry, LIDAR scan, and command velocity (control) information. (a) Case 0 (onboard computation). (b) Case 1 (remote computation). (c) Case 2 (manual teleoperation).

of reference. The readjustment of the route after applying force feedback on the Master robot is necessary for safer navigation around a highly complicated environment on the Client's side.

1) System Stability: In a classical closed-loop bilateral teleoperation system, the Master maintains constant control over the Client by position and velocity control inputs, followed by feedback via corrective position. This may render the overall system to become unstable under time-delayed information flow, necessitating the use of additional control schemes to ensure system stability [29].

However, our proposed control in (2) (low priority) and (8) (high priority) creates a reliable and modular system. The velocity MUX and the channel priorities will ensure that the system is in an open loop at a given time ensuring the stability of the system at both the robots, as shown in Fig. 3. At any instance of time, the Client only subscribes to the Master's velocity in an open-loop manner (2). When an obstruction is nearby, the Client and Master robots switch to (8) to avert a potential collision. Once the possibility of a collision is averted, the regular navigation controller takes over, thereby exhibiting a switching-based control.

This strategy also keeps the AI planner aware of the change in state and responds accordingly. Specifically, because the AI is reactive in our navigation task, it does not require a high-precision control loop, which allows us to accommodate the AT framework ensuring stability and safety. For instance, research in [43] found that open-loop architectures are more preferred as they avoid stability issues, transmitting motion commands while allowing any force feedback only via sensory substitution. Therefore, this finding is valid for both human and AI operators controlling a robot.

2) Advantages of AT Framework: The proposed AT framework has the following advantages over conventional DT framework.

1) AT can be added to an already established DT through the proposed priority-based synchronization.
2) AT provides reliable and stable synchronization between the physical and remote environments.
3) AT provides additional storage and computational power through remote offloading for the physical system with reduced network-induced latency.
4) AT provides an avenue in closing the simulation-to-real gap in the design and verification of robot algorithms.

V. Experimental Analysis

In our work, we implement the methods on the popular Robot Operating Systems (ROS) software framework [44], providing open-source software tools and libraries for both simulation and real-world hardware experiments. We utilize the *move_base* navigation package in ROS (library), which uses the DWA [35] for the local (base) planner and the A^* [45] for the global planner to navigate toward the goal. To learn grid maps from the robot's laser range scanner and wheel encoder sensors, we use the Gmapping SLAM package [46], which is based on the Rao–Blackwellization technique with a particle filter in our simulation experiments. We use the Hector SLAM [3], which is based on an EKF approach, in our hardware experiments. This is due to the compatibility with the Husky mobile robot platform used in ROS Gazebo simulations and the Turtlebot robot platform used in hardware experiment trials. In both simulations and actual hardware experiments, we implement the following experimental cases (strategies) to compare the performances.

1) *Case 0 [Baseline: Onboard Computing]—Standalone Client Robot With SLAM Running Onboard):* We test our system against the conventional scheme of an AI path planner driving a standalone mobile robot [see Fig. 4(a)]. The robot creates the map of the environment using SLAM while using it to navigate toward the goal location. The performance of this setup is used as a baseline for comparison with the proposed strategy.
2) *Case 1 [Baseline: Remote Computing]—Client Robot With Conventional Remote Computation):* We implement a typical "remote-brained robot" scenario where physically and logically, the robot's body and its "brain" are separated, and remote intelligence is applied to the robot as shown in Fig. 4(b). The goal is given to the Client robot through the output of the SLAM mapper and path planner over the network. No Master robot is employed, but a remote computer/laptop is used on the Master side. This case enables the Client robot to remotely offload its computationally extensive tasks like SLAM, path planning, and navigation to a cloud-based server or agent.
3) *Case 2 [No Feedback]—Master–Client Coupled With Human Teleoperation):* In this case, we experiment with a Master robot that is teleoperated in its environment and the Client robot subscribes to the odometry data from the Master in an open-loop manner as shown in Fig. 4(c). The Master robot is steered by a human user through a joystick toward the desired target, with visibility to only the Master's environment while the Client side is obscured to the user. As the Master moves toward

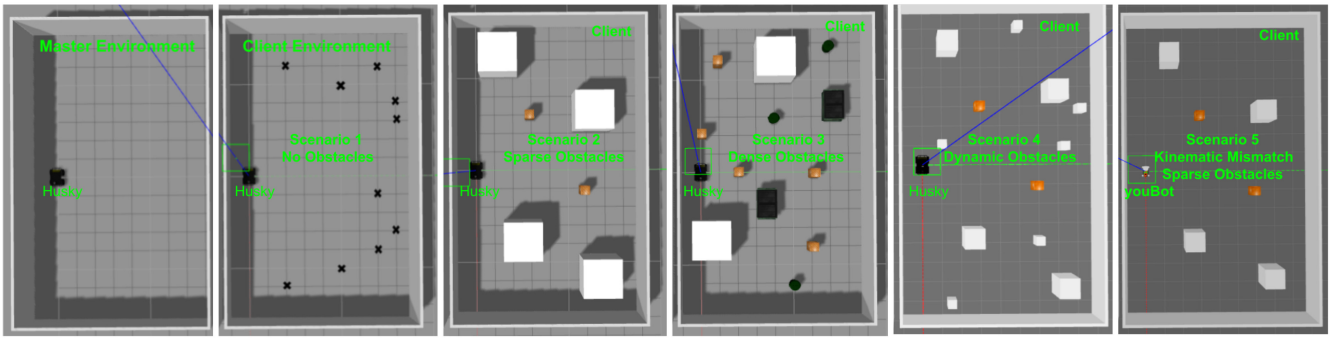


Fig. 5. Experimental setup of the simulations in ROS Gazebo. The Master environment always has no obstacles, while the client side has no obstacles (Scenario 1), sparse obstacles (Scenarios 2, 4, and 5), dense obstacles (Scenario 3), and additional dynamic obstacles (Scenario 4). The goal locations are shown with a cross in Scenario 1, which are the same for all scenarios.

the goal point, the Client robot remotely follows the Master robot through the velocity commands read over the network and replicates them in its own environment. Force feedback (or bilateral coupling) is not applied in this case and no SLAM mapping is used on either side. This provides an ideal scenario for comprehending the location tracking inaccuracy between the Master and Client over a network.

4) *Case 3 [Proposed AT]—Master–Client Coupling With Prioritized Force Feedback):* We implement our proposed robot–robot synchronization methodology (Fig. 3) to test the accuracy, delay, and time elapsed of the navigation task. According to our proposed approach, the Client robot offloads the computational overhead of mapping and navigation to the Master side and accepts the velocity commands from the Master robot on a low-priority MUX channel, and navigates its environment through the motion commands received from the Master. The mapping and navigation are performed by the Master of its environment, while the Client implements the velocity computed by the Master robot. The Master tries to arrive at the goal position based on the information from the map, while the Client simply follows the Master's odometry commands remotely and gets to the goal position. The Client reacts to the obstacles in the environment and sends predictive force feedback reacting to the contacts from the environment. Based on the feedback, the Master performs a corrective action on its trajectory to arrive at the goal position.

VI. Simulation Experiments

For testing the effectiveness of the proposed method, we perform experiments using ROS-based Gazebo 3-D physics-engine robot simulations with a Clearpath Husky robot (differential drive) model as both a Master and a Client in two identically similar experiment rooms of size 17 m × 8 m. The robots are simulated in two laptops connected via a real Wi-Fi network, which is shown to have an average of 10-ms delay in the network. We achieve overriding of velocity commands through a velocity MUX in the ROS package, *twist_mux*, both on the Master and Client sides. The MUX on Husky is configured via a *.yaml* file by prioritizing the velocity topics that reload at run time. The Client can detect nearby obstacles within the $r_{oi} = 0.5$ m range. The parameters $f_{th} = 1$ and $\omega_{\text{turn}} = 0.5(rad/s)$ set in the simulations.

A. Experiment Scenarios

We test these cases in three different scenarios, as shown in Fig. 5. In all these scenarios, the Master robot's environment is kept constant (empty room), meanwhile varying the Client's environment to be an empty room, a room with sparse obstacles, and a room with dense obstacles for scenarios 1–3, respectively. For each scenario and every case, we conduct at least five trials, and the results show the variations across the trials for every performance metric measured. For each trial, a different goal location was set, ranging from 6 to 10 m in the distance from the origin (robot's start) (see Fig. 5). These goal locations are the same across the cases and scenarios.

Furthermore, we add two more scenarios based on the second scenario (sparse obstacles) to verify the robustness of the approach in the presence of dynamic obstacles (Scenario 4), and the flexibility in the case that the Master and Client robots have dissimilar kinematics (Scenario 5), where the Master is a Husky robot with differential drive kinematics and the Client is a youBot robot with omnidirectional kinematics. More details on these two scenarios are given in Appendix B in the supplementary material.

B. Performance Metrics

This study covers the assessment of the below measures to evaluate the system's performance.

1) *Navigation Task Performance:*
   a) *Goal Accuracy (m):* Absolute difference between the final robot position and the desired goal position.
   b) *Tracking Error (m):* Average of the absolute difference between the instantaneous positions of the Master and the Client robots.
   c) *Efficiency (s):* The time elapsed to reach a goal point by the Client robot.
2) *Network Performance:*
   a) *Network Throughput (Mbps):* The average data rate at which the Master–Client machines communicate.

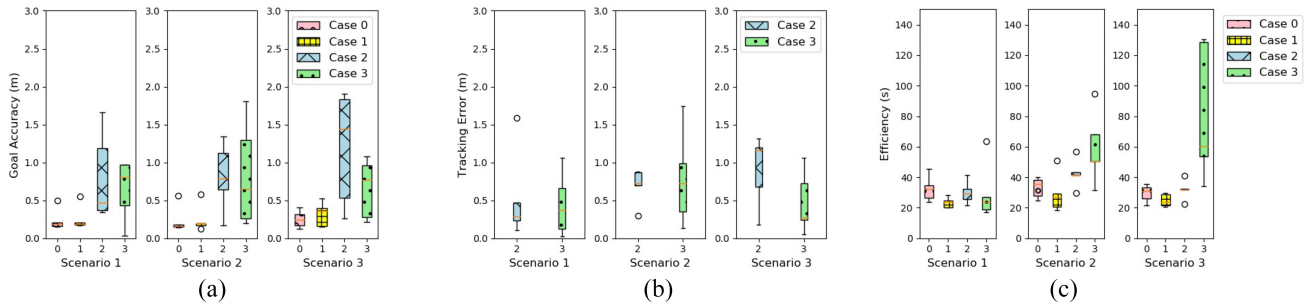


Fig. 6. Navigation task performance—end goal accuracy, tracking error, and efficiency in simulation experiments. (a) End goal accuracy. (b) Master–Client tracking error. (c) Task efficiency.

b) *Network Latency (s):* The average delay with which the Master and Client machines communicate.

3) *Computing Performance:*
   a) *CPU Utilization (%):* The average CPU load (measured every 5 min) at the Client robot/machine during the experiments.
   b) *Memory Utilization (Mb):* The average RAM memory used by the Client robot during the experiments.

As the study focuses on remote computation, we are more interested in the wireless network quality and performance for remote teleoperation. The majority of data from the Client site is delivered to the Master site, which is in charge of navigation. For instance, Case 1 transmits full raw sensor data to the Master, while Cases 2 and 3 exchange only partial information. Since sustaining communication is critical to our proposed approach, network latency must be continuously monitored. The Master robot sends the data, and the Client receives and resends new information obtained from the change in its environment. In addition, we analyze how network performance impacts computational offloading and if there is a significant lag or performance degradation while the data is being offloaded or had been offloaded.

We use our *ROS Network Analysis* package, hosted in a public GitHub repository[1] for measuring network performance metrics through ROS nodes. For instance, the ping action implemented through ROS actionlib is used to calculate the latency or network delay. At the same time, we extract the network throughput reported by the Wi-Fi device driver.

C. Results and Discussion

We analyze the results from the perspective of each metric and then discuss the flexibility and robustness.

1) End Goal Accuracy: The results of the experiments performed across different cases are summarized in Fig. 6(a). We test our strategy against the baseline scheme, Case 0, in which the robot is a standalone robot that maps and navigates its own environment without the assistance of any Master or autonomous agent. The findings obtained in an open-world environment (scenario 1) reveal that our proposed strategy (Case 3) has a significantly controlled goal error, amounting to no more than 1-m error, better than the teleoperation tracking (Case 2). Because of autonomous navigation, Cases 0 and 1 show the best results in task performance metrics.

For scenario 2, the mean value of end goal error for Case 3 is slightly lower than that of Case 2. But, the accuracy of Case 3 is significantly better than Case 2 in scenario 3, where the obstacles get denser. Navigation without feedback becomes difficult due to obstacles blinded in the Master environment, as clearly seen by the performance of Case 2 getting worse over successive scenarios. However, our proposed strategy manages to keep the end goal accuracy under control because of the force feedback to the AI planner.

2) Tracking Error: The tracking performance between the Master and the Client robot is shown in Fig. 6(b) for Cases 2 and 3 since only these two cases involve having a Master and a Client robot. The performance of Case 2 in scenario 1 is the baseline for this analysis as that provides us the best case tracking possible with a velocity subscriber at the Client. Tracking is challenged by the number of obstacles, as seen by the deviation between trials in Case 2 results. However, the feedback mechanism applied in Case 3 helps maintain consistent performance across the scenarios and significantly improves the tracking performance compared to Case 2.

3) Task Efficiency: According to the data obtained shown in Fig. 6(c), the baseline cases show consistent efficiency across scenarios as the AI planner uses the SLAM map. It is interesting to note that Case 1 (remote computation) efficiency is better than Case 0 (onboard computation), demonstrating CPU load's impact on task performance. In scenario 1, the proposed method performs at par with the baseline cases of autonomous navigation and remote offloading because of no impact on the obstacles. However, as expected, the efficiency of the proposed method is significantly challenged by the presence of obstacles. This is because dense obstacles necessitate a stronger and more persistent reaction to the force feedback loop in Case 3. An improved feedback function in (7) could be able to help overcome this issue.

4) Network Throughput: Performance comparison with respect to the throughput experienced by all the cases except the baseline (Case 0, which does not use a wireless network) is summarized in Fig. 7(a). It determines the number of network packets exchanged between the Master and the Client robots at any given time. Throughput is averaged through the length of the trial. As expected, the throughput calculated for Case 1

[1]https://github.com/herolab-uga/ros-network-analysis

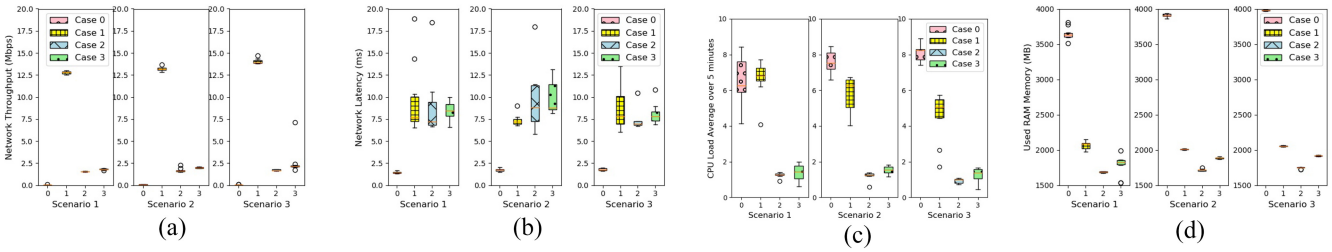


Fig. 7. Network and computing performance results in simulation experiments. (a) Network throughput. (b) Network latency. (c) CPU utilization. (d) Memory usage.

would always be greater than the other cases, as it involves full data offloading to a remote workstation. As we can see, the proposed method provides at least a 5× reduction in network throughput than Case 1 while still performing core SLAM computation remotely, thanks to the feedback and synchronization strategy. Case 3's throughput is only marginally greater than Case 2, where the major data being shared is the Master robot states. This explains that the addition of reactive force in an open-loop manner does not increase data requirement considerably, significantly helping the system with better task performance while ensuring system stability.

5) Network Latency: The network latency experienced by the system in all scenarios is shown in Fig. 7(b). As the baseline case does not involve using a network, being a standalone robot, it experienced no network latency. Latency is relatively consistent and comparable in most cases, as the wireless network connecting the Master and Client machine is within a lab environment, ensuring the stability of the network and the control loop. The variations in network latency can be suppressed by sending fewer data via the network, as demonstrated by the results of scenario 3 comparing Case 1 (sharing all data) and Case 3 (sharing partial data), indicating the usefulness of the suggested technique.

6) CPU and Memory Utilization: The Client robot's computing performance (CPU load and memory use) across all cases in different scenarios is compared in Figs. 7(c) and 7(d), respectively. As expected, Case 0 has the highest CPU load, and memory use since all computations are done locally on the Client robot. Again, in Case 1, because the CPU is tasked with continuous network transmission to offload the data to the Master side, while the memory is unused for computations, the Client robot shows a high CPU load but low memory use. Case 2 has the lowest CPU load and memory usage because the Client robot is only tasked with replicating the Master robot, requiring the least computing resources. The proposed Case 3 provides comparable performance to Case 2 since it optimally balances both the computing load and remote computation. We can observe up to 3× better CPU load and more than 2× better RAM use in Case 3 compared to the baseline cases.

a) Flexibility and robustness: To analyze the robustness of the approach (Case 3) with dynamic obstacles in Scenario 4, we look at Fig. 8. Here, the goal accuracy was not affected by the introduction of moving obstacles, and a reasonable synchronization performance between Master–Client was maintained throughout all goal points tested. Also, the task

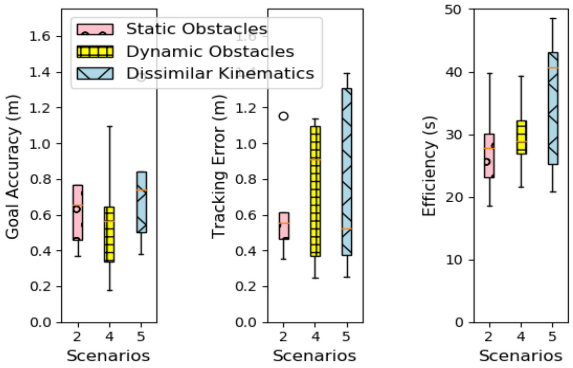
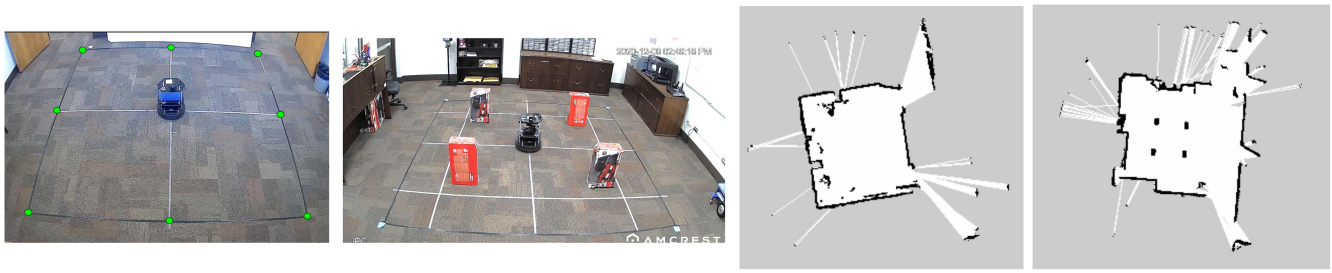


Fig. 8. Comparison of Case 3's task performances in the presence of static obstacles (Scenario 2), dynamic obstacles (Scenario 4), and dissimilar kinematics (Scenario 5).

efficiency did not suffer in Scenario 4 compared to Scenario 2. Overall, similar task efficiency was observed regardless of the spontaneity of the obstacles encountered. As expected, through the indicated results, we can derive that satisfied task performance can be obtained under the proposed framework in a highly dynamic setting.

To analyze the flexibility of the approach with dissimilar kinematics at the Master and Client sides, we compare Scenario 5 in Fig. 5 to the baseline Scenario 2, where the kinematics are the same under the same obstacle environment. Our results shown in Fig. 8 indicate that our proposed AT-aided feedback produced comparable task performance outcomes in terms of end-goal accuracy, Master–Client tracking, and efficiency in kinematically mismatched Master–Client pairs. While the end goal and tracking error remained well within the average of ±0.75 m for all goal points tested, the task efficiency of the dissimilar Husky–youBot pair experienced a slight decline from the similar robots Husky–Husky setup. We believe this is mostly due to imperfect synchronization amongst robot drivers with varied reaction times. The results indicate that the proposed scheme of remote collaboration between dissimilar Master–Client mobile robots is viable.

b) Summary and limitations: The proposed solution has achieved a balance between task performance, networking, and computing requirements, but at a cost of reducing task efficiency, especially when the environment is complex or unpredictable. Therefore, this aspect needs to be further investigated. Our system has some flexibility and resilience due to the use of variable feedback open-loop control, which proved useful even in a complicated setting with no human intervention. Although there is a possibility of congestion and an



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

Fig. 9. Experimental setup for the hardware experiments with the SLAM maps generated by the Client and Master robots. (a) Master room showing the goal locations in green dots. (b) Client room showing the obstacle (red boxes) placements. (c) Master room SLAM map. (d) Client room SLAM map.

increased load on the Master robot, this mechanism is designed to keep the Master's load at the minimum by running the odometry reader node on the Client robot and by the use of a MUX that prevents deadlock and prioritizes the incoming velocity commands. It is worth noting that the above analyses are exclusively from the Client robot's perspective, as it is the target of interest. We assume the Master robot is in a secure location without restriction to computing resources, enabling integration with DT.

VII. Real-World Robot Hardware Experiments

To evaluate the real-time expediency of the proposed scheme, we test the same four experimental cases as in simulations (see Section V). We use two identical mobile robots, Turtlebot 2e's, in our laboratory environment. Both the robots are connected via a Wi-Fi network (same as in simulations) and are located approx 20 m apart in two separate experiment rooms, mimicking the Master and Client sides. Each room is marked with a 3 × 3 m subsquare with rectangular obstacles. Eight trials are conducted for each case, with goals ranging within a $(\pm 1.5, \pm 1.5)$ offset from the origin (center). The hardware trials consist of one scenario (environment with four obstacles) as shown in Fig. 9. Here, the robot always starts from the center, and the eight goal locations are located at the corners of the 3 × 3 m area, as shown in Fig. 9(a).

The Master robot has a 2D-LIDAR LDS-01 laser scanner, and the Client robot is equipped with a Hokuyo URG-04LX-UG01 2D-LIDAR laser range scanner as sensors used to create a SLAM map of the environment. Both robots are connected to computers running Ubuntu 18.04 with ROS melodic.

Performance Metrics: In addition to the metrics outlined in Section VI-B, we include another metric called *throughput loss*, which is the difference between the throughput obtained between the Master and the Client side. It is indicative of the packet loss in the network to estimate how much of the offloaded data was lost in the network between the one-way (Cases 1 and 2) and two-way exchange (Case 3) of the data between the machines.

Obstacle Placement: The Client environment has obstacles laid as in Fig. 9(b). The SLAM map outputs are shown for the Master and Client rooms in Fig. 9(c) and (d), respectively. The Master room has no obstacles for all cases, except Case 2, where it is intentionally kept this way to mimic the obstacles set up in the Master room for guiding human teleoperation. This way a reasonable route to goal points can be manually realized around the obstacles (without an AI planner).

A. Results and Discussion

We discuss the results obtained in the hardware experiments below and compare them to the simulation results.

1) End Goal Accuracy: The end goal accuracy result is shown in Fig. 10(a). The proposed approach performed at par with Case 2, showing a minimal influence in reaching the end goal location due to the force feedback. It is worth noting that the performance of remote computation (Case 1) is better than that of the onboard computation (Case 0), possibly due to the higher computational resources spent on the robot.

2) Tracking Error: Fig. 10(b) shows the comparison of tracking error between Master and Client robots with and without force feedback. Through bilateral teleoperation of the mobile robots, we aim for the Client robot to follow a similar trajectory as planned by the Master robot for its environment, with feedback cues indicative of the Client environment to move the robot to the goal. Case 3 shows significantly less error in mimicking the Master side, demonstrating the advantages posed by the proposed priority-based feedback strategy.

3) Task Efficiency: We observe through the data collected in Fig. 10(c) that it takes more time on average to arrive at the goal position when using a bilateral teleoperation system with predictive force as expected. Nevertheless, the proposed system efficiency is comparable to the remote offloading scenario without transmitting all sensor data.

4) Throughput Loss: Throughput loss data presented in Fig. 10(d) shows the results for Cases 1–3 where there is a data exchange between the Master and the Client. Although this metric is not a direct measure of packet loss, it indicates how packets are retransmitted and handled at the two machines for the same data communication. Since Case 1 involves heavy data offloading from the Client, the throughput loss of this case is the highest among all, as anticipated. Also, we expect that the throughput loss at the Master is generally higher than the loss at the Client since the Master sends its state (position) to the Client continuously at a constant rate and receives sensor or feedback data from the Client. This is seen in the outcome of all the cases. Case 2 has the least amount of data to transmit, showing the lowest loss. The proposed strategy (Case 3) shows

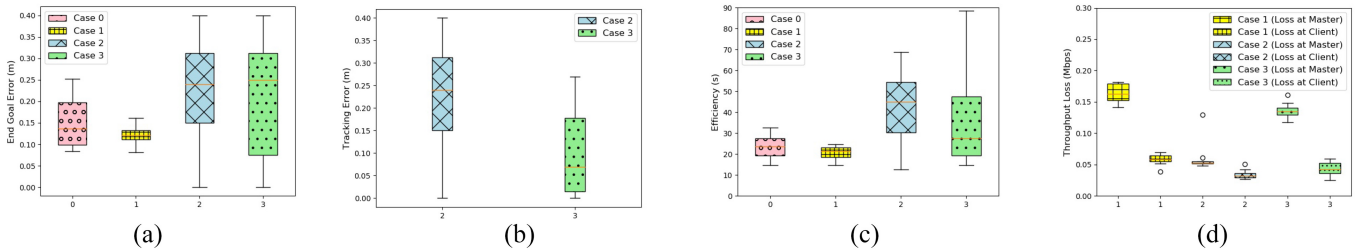


Fig. 10. Task performance (end goal error, tracking error, and efficiency) and throughput loss in hardware experiments. (a) End goal accuracy. (b) Master–Client tracking error. (c) Task efficiency. (d) Throughput loss.

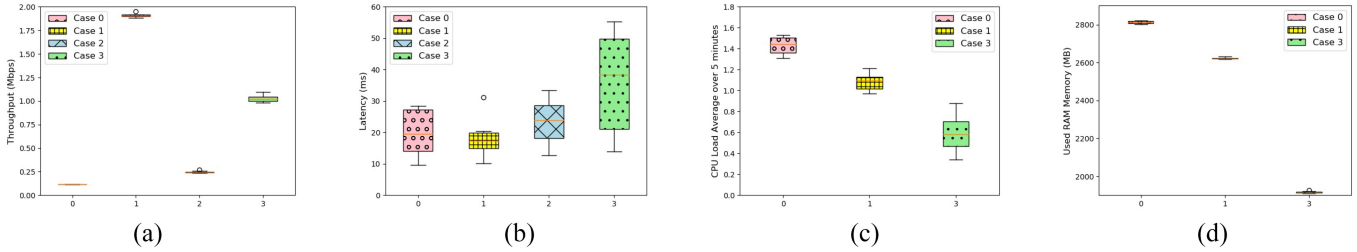


Fig. 11. Network and computing performance parameters in the real-world robot (hardware) experiments. (a) Network throughput. (b) Network latency. (c) CPU utilization. (d) Memory usage.

a balanced loss compared to Cases 1 and 2. This demonstrates the optimum and effective use of the network.

5) Network Throughput: The throughput utilized by all the cases in hardware experiments is summarized in Fig. 11(a). As expected, the throughput of Case 1 is the highest due to offloading computationally expensive tasks like mapping and navigation and getting data back to the Client. Between Cases 2 and 3, data exchange is lowest for Case 2 as teleoperating the Master robot drives the Client robot without exchanging any information from the Client, while Case 3 does entail two-way communication owing to feedback requiring larger yet balanced throughput. Throughput in Case 3 is between 1 and 1.15 Mb/s due to two-way data transferred and received, unlike Case 1. Case 3 shows around a 2× reduction in network throughput requirement, allowing higher scalability of operations using the proposed strategy.

6) Network Latency: We observe more network latency for Case 3, as seen in Fig. 11(b). Network latency can lead to a broader tracking error, although with the exception of a few spikes for our testing trials, this latency is under control. Note that for all cases, we run the *roscore* ROS node (which is responsible for establishing communication between the machines) at the Master machine, regardless of whether sensor/state data is exchanged (including the baseline Case 0). This is to obtain a realistic baseline for the minimum network latency for handshake communication with a remote machine in the same network. We observe an average latency of around 40 ms for Case 3, about 2× higher than the baseline (Case 0). Nevertheless, no significant performance degradation or tracking error is observed due to this delay.

7) CPU and Memory Utilization: Compared to the baseline case, the CPU utilization drops by around 57% for Case 3 compared to Case 0, as depicted in Fig. 11(c). The proposed strategy effectively balances between performing all the computation itself (Case 0) and offloading all the computational tasks to the remote machine (Case 1). Memory utilization data is shown in Fig. 11(d) follows a similar trend as in the CPU load metric. We observed a significant drop in Case 3's RAM usage compared to the baseline case. This is due to the offloading of the computationally hungry tasks which becomes the liability of the Master robot in a bilateral and collaborative teleoperation system. Note that we omitted presenting Case 2's data in these figures, as there was another process running on the Client robot during Case 2 trials that compromised our computing metrics measurement.

The computing metrics prove that collaborative offloading leads to low computation requirements at the Client robot by leveraging on the communications network and the computational resources offered by the remote robot. Moreover, computation offloading prevents processing deadlocks at the Client side and improves performance and efficiency. However, the reliance on onboard computation decreases but leads to more trust in the network conditions, requiring a robust, reliable, and efficient communication network.

VIII. Conclusion

We proposed an AT framework using a Master–Client collaboration architecture for synchronizing robots. The proposed approach can help simulate new algorithms and innovations for collaborative AI and real-time verification of algorithms. To illustrate the proposed framework, a novel priority-based bilateral teleoperation strategy is presented and tested for a human-teleoperated and AI-supervised navigation task. This strategy uses computational offloading for resource-constrained mobile robots that lack computational capacity.

Our strategy is verified for effectiveness and flexibility through extensive simulations and real-world robot experiments. We analyzed the performance metrics from the perspective of task (tracking and goal accuracy and efficiency), networking (throughput and delay), and computation (memory and CPU usage). Our strategy outperformed conventional strategies like onboard computation, remote computation, and manual teleoperation schemes. The results demonstrate reductions in network and computing requirements without compromising task performance, signifying reliable AT collaboration.

In general, similar trends are observed in both hardware and simulated cases for efficiency, throughput, and network latency. However, a significant difference in the size of the robot's workspace created certain dissimilarities between the results from simulation and hardware studies. Moreover, in simulations, computing resources also consider the processing of the graphical visualization of the simulated robots. In our future work, we will conduct experiments on continuous navigation tasks that could shed more light on the benefits and limitations of the proposed AT-aided collaboration framework.

We believe that the results of the proposed idea of bilateral teleoperation through a computationally efficient AT will aid in the research and development of novel human or AI in the loop robotics and CPSs.

REFERENCES

[1] N. Kousi, C. Gkournelos, S. Aivaliotis, C. Giannoulis, G. Michalos, and S. Makris, "Digital twin for adaptation of robots' behavior in flexible robotic assembly lines," *Procedia Manuf.*, vol. 28, pp. 121–126, Jan. 2019.

[2] P. Staczek, J. Pizoń, W. Danilczuk, and A. Gola, "A digital twin approach for the improvement of an autonomous mobile robots (AMR's) operating environment—A case study," *Sensors*, vol. 21, no. 23, p. 7830, 2021.

[3] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. von Stryk, "Hector open source modules for autonomous mapping and navigation with rescue robots," in *Robot Soccer World Cup*. Berlin, Germany: Springer, 2013, pp. 624–631.

[4] I. R. Nourbakhsh, K. Sycara, M. Koes, M. Yong, M. Lewis, and S. Burion, "Human-robot interaction: Human-robot teaming for search and rescue," *IEEE Pervasive Comput.*, vol. 4, no. 1, pp. 72–79, Jan.-Mar. 2005.

[5] S. Luo, J. Kim, and B.-C. Min, "Asymptotic boundary shrink control with Multirobot systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 1, pp. 591–605, Jan. 2022.

[6] T. B. Sheridan, "Teleoperation, telerobotics and telepresence: A progress report," *Control Eng. Pract.*, vol. 3, no. 2, pp. 205–214, 1995.

[7] S. Chinchali et al., "Network offloading policies for cloud robotics: A learning-based approach," *Auton. Robots*, vol. 45, no. 7, pp. 997–1012, 2021.

[8] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 398–409, Apr. 2015.

[9] A. Manzi, L. Fiorini, R. Limosani, P. Sincak, P. Dario, and F. Cavallo, "Use case evaluation of a cloud robotics teleoperation system (short paper)," in *Proc. 5th IEEE Int. Conf. Cloud Netw. (Cloudnet)*, 2016, pp. 208–211.

[10] W. Chen, Y. Yaguchi, K. Naruse, Y. Watanobe, K. Nakamura, and J. Ogawa, "A study of robotic cooperation in cloud robotics: Architecture and challenges," *IEEE Access*, vol. 6, pp. 36662–36682, 2018.

[11] M. Ferre, M. Buss, R. Aracil, C. Melchiorri, and C. Balaguer, *Advances in Telerobotics*, vol. 31. Berlin, Germany: Springer, 2007.

[12] R. Wirz, R. Marín, M. Ferre, J. Barrio, J. M. Claver, and J. Ortego, "Bidirectional transport protocol for teleoperated robots," *IEEE Trans. Ind. Electron.*, vol. 56, no. 9, pp. 3772–3781, Sep. 2009.

[13] M. Franken, S. Stramigioli, S. Misra, C. Secchi, and A. MacChelli, "Bilateral telemanipulation with time delays: A two-layer approach combining passivity and transparency," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 741–756, Aug. 2011.

[14] A. Zakerimanesh, F. Hashemzadeh, A. Torabi, and M. Tavakoli, "Controlled Synchronization of nonlinear Teleoperation in task-space with time-varying delays," *Int. J. Control Autom. Syst.*, vol. 17, no. 8, pp. 1875–1883, 2019.

[15] B. Davies, "Robotic surgery–A personal view of the past, present and future," *Int. J. Adv. Robot. Syst.*, vol. 12, pp. 1–6, 2015.

[16] J. Li et al., "Dual-master/single-slave haptic teleoperation system for semiautonomous bilateral control of hexapod robot subject to deformable rough terrain," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 4, pp. 2435–2449, Apr. 2022.

[17] A. Shahzad and H. Roth, "Bilateral telecontrol of AutoMerlin mobile robot," in *Proc. Int. Conf. Open Source Syst. Technol. (ICOSST)*, 2016, pp. 1–6.

[18] J. A. Vences-Jimenez, A. Rodriguez-Angeles, and J. Alvarez-Gallegos, "Bilateral time delay compensation in bilateral master slave teleoperation of differential mobile robots using IMC," in *Proc. 13th IEEE Conf. Ind. Electron. Appl. (ICIEA)*, 2018, pp. 1302–1307.

[19] X. Hou and R. Mahony, "Dynamic kinesthetic boundary for haptic teleoperation of VTOL aerial robots in complex environments," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 5, pp. 694–705, May 2016.

[20] I. Farkhatdinov and J.-H. Ryu, "Improving mobile robot bilateral teleoperation by introducing variable force feedback gain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2010, pp. 5812–5817.

[21] Y. Kawai, T. Namerikawa, and M. Fujita, "Bilateral teleoperation of wheeled mobile robot with time delay using virtual image robot," in *Proc. IEEE Int. Conf. Control Appl.*, vol. 2, 2010, pp. 77–82.

[22] D. Lee, A. Franchi, P. R. Giordano, H. I. Son, and H. H. Bülthoff, "Haptic teleoperation of multiple unmanned aerial vehicles over the Internet," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 1341–1347.

[23] W. Bu, G. Liu, and C. Liu, "Online generation of virtual fixture for bilateral teleoperation based on intention recognition," in *Proc. Int. Conf. Adv. Robot. Mechatron. (ICARM)*, 2016, pp. 122–126.

[24] M. Minelli, F. Ferraguti, N. Piccinelli, R. Muradore, and C. Secchi, "An energy-shared two-layer approach for multi-master-multi-slave bilateral teleoperation systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 423–429.

[25] Y. Li, K. Liu, W. He, Y. Yin, R. Johansson, and K. Zhang, "Bilateral Teleoperation of multiple robots under scheduling communication," *IEEE Trans. Control Syst. Technol.*, vol. 28, no. 5, pp. 1770–1784, Sep. 2020.

[26] N. Tahir and R. Parasuraman, "Robot controlling robots-a new perspective to bilateral Teleoperation in mobile robots," in *Proc. Robot. Sci. System (RSS) Workshop Reacting Contact Enabling Transp. Interact. Through Intell. Sens. Actuation*, 2020, pp. 1–3.

[27] P. Pandey and R. Parasuraman, "Empirical analysis of bi-directional wi-Fi network performance on mobile robots and connected vehicles," 2021, *arXiv:2110.03011*.

[28] J. Salmerón-García, P. Íñigo Blasco, F. Díaz-del Río, and D. Cagigas-Muñiz, "A tradeoff analysis of a cloud-based robot navigation assistant using stereo image processing," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 444–454, Apr. 2015.

[29] W. Li, L. Ding, H. Gao, and M. Tavakoli, "Haptic Tele-driving of wheeled mobile robots under Nonideal wheel rolling, kinematic control and communication time delay," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 1, pp. 336–347, Jan. 2020.

[30] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*, I. J. Cox and G. T. Wilfong, Eds. New York, NY, USA: Springer, 1990. [Online]. Available: https://doi.org/10.1007/978-1-4613-8997-2_14

[31] P. Jensfelt and S. Kristensen, "Active global localization for a mobile robot using multiple hypothesis tracking," *IEEE Trans. Robot. Autom.*, vol. 17, no. 5, pp. 748–760, Oct. 2001.

[32] W. Burgard, D. Fox, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *J. Artif. Intell. Res.*, vol. 11, pp. 391–427, Dec. 1999.

[33] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artif. Intell.*, vol. 128, nos. 1–2, pp. 99–141, 2001.

[34] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. AAAI*, vol. 18, 2002, pp. 593–598.

[35] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.
[36] P. Galambos, "Cloud, fog, and mist computing: Advanced robot applications," *IEEE Syst., Man, Cybern. Mag.*, vol. 6, no. 1, pp. 41–45, Jan. 2020.
[37] P. Benavidez, M. Muppidi, P. Rad, J. J. Prevost, M. Jamshidi, and L. Brown, "Cloud-based realtime robotic visual SLAM," in *Proc. Annu. IEEE Syst. Conf.*, 2015, pp. 773–777.
[38] P. Zhang, H. Wang, B. Ding, and S. Shang, "Cloud-based framework for scalable and real-time multi-robot slam," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2018, pp. 147–154.
[39] R. Arumugam et al., "DAvinCi: A cloud computing framework for service robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 3084–3089.
[40] S. A. Miratabzadeh et al., "Cloud robotics: A software architecture: For heterogeneous large-scale autonomous robots," in *Proc. World Autom. Congr. (WAC)*, 2016, pp. 1–6.
[41] G.-P. Liu, "Predictive control of networked multiagent systems via cloud computing," *IEEE Trans. Cybern.*, vol. 47, no. 8, pp. 1852–1859, Aug. 2017.
[42] O. Linda and M. Manic, "Fuzzy force-feedback augmentation for manual control of multirobot system," *IEEE Trans. Ind. Electron.*, vol. 58, no. 8, pp. 3213–3220, Aug. 2011.
[43] P. Shull and G. Niemeyer, "Open-loop bilateral teleoperation for stable force tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 5121–5126.
[44] M. Quigley et al., "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, vol. 3. Kobe, Japan, 2009, p. 5.
[45] A. V. Le, V. Prabakaran, V. Sivanantham, and R. E. Mohan, "Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor," *Sensors*, vol. 18, no. 8, p. 2585, 2018.
[46] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

Nazish Tahir received the M.S. degree in information technology from the National University of Science and Technology, Islamabad, Pakistan, in 2016. She is currently pursuing the Ph.D. degree in computer science with the School of Computing, University of Georgia, Athens, GA, USA.

Her research interests include collaborative control, remote computing, and edge/cloud robotics.

Ramviyas Parasuraman (Senior Member, IEEE) received the Ph.D. degree in robotics and automation from TU Madrid, Madrid, Spain, in 2014.

He is an Assistant Professor with the School of Computing, University of Georgia, Athens, GA, USA. He directs the Heterogeneous Robotics Research Lab, which conducts cutting-edge research in heterogeneous multirobot systems, networked autonomous vehicles, human–robot interfaces, search and rescue robotics, and machine learning applications to robotics and sensing. He conducted doctoral research with the European Organization for Nuclear Research (CERN), Meyrin, Switzerland. Before joining the University of Georgia as an Assistant Professor in 2018, he was a Postdoctoral Researcher with Purdue University, West Lafayette, IN, USA, and the KTH Royal Institute of Technology, Stockholm, Sweden. He has published over 40 research papers in top-tier journals and conferences, including IEEE TRANSACTIONS ON CYBERNETICS, *Robotics and Autonomous Systems*, *NPG Asia Materials*, and IEEE SENSORS.

Dr. Parasuraman was a recipient of the prestigious Marie Skłodowska Curie ESR Fellowship at CERN. He has served as the Guest Editor for *Technologies* and an Associate Editor for the International Conference on Robotics and Automation 2019.