A Digital Twin Framework for Telesurgery in the Presence of Varying Network Quality of Service

Sophea Bonne[1]*, Will Panitch[1]*, Karthik Dharmarajan[1]*, Kishore Srinivas[1]*, Jerri-Lynn Kincade[1], Thomas Low[2], Bruce Knoth[2], Cregg Cowan[2], Danyal Fer[3], Brijen Thananjeyan[1], Justin Kerr[1], Jeffrey Ichnowski[1], Ken Goldberg[1]

***Abstract*— Remote telesurgery can enable expert surgeons to operate on patients in distant or underserved locales. However, network instability and delays hamper long-distance communication. To address this, we explore how a "digital twin," a 3D simulator that actively mirrors a real environment, can be applied to telesurgery. We focus on the Fundamentals of Laparoscopic Surgery peg transfer surgical training task. We present a framework that enables a teleoperator to perform this task over unstable or low-bandwidth communication channels using a digital twin. The surgeon remotely teleoperates the robot in our simulator, which abstracts their motions into commands and transmits them to the real robot for semi-autonomous execution. The system executes the transfer and then sends the real state of the pegboard back to the simulator. We present experiments that demonstrate that the operation of each portion of the framework in isolation maintains a high task success rate, and that the success rate of the digital twin framework is robust to network transmission instability and delays.**

I. Introduction

Robotic Surgical Assistants (RSAs), which allow an expert surgeon to teleoperate robot tools, are commonly used in operating rooms worldwide to perform a variety of complex procedures. These systems allow human surgeons to execute precise and intricate tasks with greater dexterity and visual acuity, enhancing the capabilities of our human operators. In doing so, RSAs enable more precise or less invasive surgical operations, which can reduce the risk of surgical complication and hospital readmission. Teleoperating these surgical robots over long distances in order to remotely perform procedures on patients, a practice often referred to as *telesurgery*, is a well-studied area of research and enables surgeons to care for patients who are in distant or inaccessible locations. This practice relies on sending data across a network and can require wireless transmission in remote or hostile environments, where the network quality of service (QoS) can experience a variety of breakdowns. These failures—such as packet losses and delays—can quickly result in catastrophic errors or surgeon fatigue during operations [1], [2].

One alternative to direct control is the use of a *digital twin*, which provides a real-time virtual simulation of the physical system. This digital representation models the system and workspace remotely, and is updated based on both a simulated dynamics model and real sensor observations [3], [4], [5], [6]. This can allow for significantly less data interchange over the network, reducing the number of possible points of failure. Additionally, as there are several different RSAs in use, a digital twin system allows the remote user to interact with a simulated RSA that is more familiar to them before translating and mapping these commands onto the physical RSA.

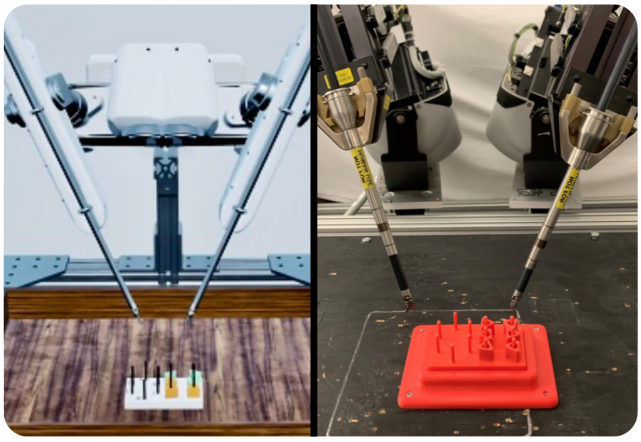

Fig. 1: **Digital twin framework. On left:** Simulated SRI Taurus Robotic Surgical Assistant (RSA) and pegboard. **On right:** Real dVRK RSA with 3D-printed pegboard.

We present a framework that uses a remote digital twin to perform the Fundamentals of Laparoscopic Surgery (FLS) peg transfer task [7]. This task involves transferring a set of six blocks from one set of pegs to another. Under our Digital Twin framework, the surgeon transfers blocks in a simulated VR teleoperation environment. Upon successful completion of a transfer, the system sends minimal information about the transfer via UDP or TCP to a client listener, which decodes the packets and sends the information to the robot to trigger a corresponding block transfer motion. The robot then autonomously executes the desired transfer, locally compensating for the cabling effects of the robot using a learned neural network dynamics model as in Hwang *et al.* [8]. The robot then sends the real world state of the pegboard back to the simulated environment.

Prior work uses a simulator for the peg transfer task. Gonzalez *et al.* [6] present a method that uses a simulator of the peg transfer task overlaid with images of the real

* equal contribution
[1] The AUTOLab at UC Berkeley (automation@berkeley.edu)
[2] SRI International
[3] UC San Francisco East Bay

world to encode and send surgeon commands over a delayed channel to an ABB YuMi robot to execute. The robot and camera then send back information regarding the state of the real world, which is used to update simulator state. We extend this work by considering delayed teleoperation of the peg transfer task with a cable-driven surgical robot, which introduces potential errors due to the effects of hysteresis and backlash [8], [9], [10], [11]. These additional errors can significantly increase the difficulty of the peg transfer task, as the precision required is under 1 millimeter [9]. Additionally, we consider a framework using two different RSAs: the SRI Taurus robot [12] in simulation and the da Vinci Research Kit (dVRK) [13] in the physical setup operating over channels that have substantial, widely-varying time delays and transmission errors due to wireless communication over ad-hoc networks.

This paper contributes:

1) A digital twin environment for simulating the FLS peg transfer task and transmitting commands using the SRI Taurus RSA.
2) A digital twin framework to semi-autonomously perform this task on the Taurus and the da Vinci Research Kit (dVRK) RSA under varying network conditions.
3) Experimental results suggesting that the framework is robust to network delays and packet loss.

II. Related Work

A. Telesurgery and Latency

Remote teleoperation for surgery has a wide range of potential benefits [2], but it also faces significant challenges associated with network latency and instability [14]. Lum *et al.* [15] studied the effect of latency in the 150 ms-1000 ms range on the task of peg transfer, and found that the operator of a Raven surgical robot suffered considerable mental fatigue and a drastically lower success rate as latency increased. Even in tasks far less complex than surgery, moving from a low-latency to a high-latency regime reduced human performance and increased the severity of fatigue and stress responses [16], [17].

Meehan *et al.* [16] found that a change as subtle as moving from 50 ms to 90 ms latency resulted in a greater stress response and a decreased sense of presence. MacKenzie *et al.* [17] tested the effect of increasing computer mouse latency and observed 63% slower user task completion and an increasing error rate when going from a low-latency to a high-latency regime.

However, despite these negative effects, humans can still successfully perform tasks under moderate latency conditions in the range of 100–200 ms. Marescaux *et al.* [18] performed a 54 minute operation from 14,000 km away with a mean latency of 155 ms; surgeons described the operation as safe and reliable.

These findings motivate the development of alternate control schemes for telesurgery which provide a more responsive interface for human subjects. One such route is to incorporate stabilizing controllers and prediction, which allows the simulator to compensate directly for latency by predicting a few moments into the future. Ryu *et al.* [19] propose the Time Domain Passivity Approach (TDPA) to extend time-domain passive control to teleoperation and guarantee stable operation without exact state information. Panzirsch *et al.* [20] extend TDPA to teleoperation at Earth-to-Moon timescales, with up to 3 s round-trip delays. Grohmann *et al.* [21] also propose JAVRIS, an approach that uses deep learning predictions of robot state to give a zero-latency user experience. In contrast to these prior approaches, our proposed digital twin method gives the teleoperator a familiar interface to drive an autonomous process that can handle long delays (minutes) and communication disruptions by fully simulating the system.

B. Digital Twins for Telesurgery

Laaki *et al.* [4] created a digital twin system for controlling a UR3 arm in a surgical environment by streaming the robot's end effector pose over UDP, demonstrating that the real arm can match the end effector pose of the virtual arm with negligible delay except in certain cases. Hagmann *et al.* [22] used a digital twin for manipulating nylon as a pick-and-place task using Shared Control Templates to represent robot skills. Their results show that assistance by haptic feedback from the digital twin improved accuracy and ease of task completion compared to the case without feedback. Contrary to prior digital twins, our proposed method enables surgical peg transfer completion on a cable driven surgical robot in the presence of significant network delays. Furthermore, we allow our operator to work with the latest state of the pegboard without requiring high-bandwidth transmission.

C. Surgical Peg Transfer

The surgical peg transfer is one of five tasks in the Fundamentals of Laparoscopic Surgery surgeon training tasks [7]. The peg transfer task is commonly studied in surgical robotics literature, because it requires a high level of accuracy [6], [8], [9], [10], [23], [24], [25], [26]. The peg transfer task is challenging for surgical robots, as accurate open-loop control of these robots is difficult due to cable stretching effects such as hysteresis and backlash [8], [9], [10], [11], [27]. Peg transfer requires high precision, because the opening in each block only clears the peg by 1.15 mm on average [9].

Rosen *et al.* [25] first studied automation of the peg transfer task using a Raven surgical robot and a version of the task with three blocks. Hwang *et al.* [8] propose a calibration method that compensates for the cable-stretching properties of individual surgical instruments. With the calibration method, Hwang *et al.* [8] demonstrate a fully-autonomous success rate of over 96% on the peg transfer task. Paradis *et al.* [9] propose a different method, which uses closed-loop visual servoing to autonomously perform the peg transfer task with over 99% success rate. In this paper, we use the calibration method from Hwang *et al.* [8] to execute block transfers commanded by the remote surgeon.

The peg transfer task is also commonly used to study human-robot interfaces during teleoperation. Abiri *et al.* [28] use the peg transfer task to evaluate a novel method for haptic force feedback during surgeon teleoperation. Rivas-Blanco *et al.* [23] study how to automate surgeon camera movement while performing peg transfer. Conversely, Brown *et al.* [24] use contact forces and arm accelerations to autonomously rate the surgeon performance in the peg transfer task. The surgical peg transfer task is also used to study task segmentation algorithms that split and cluster human trajectories of the task into semantically-relevant subtasks [29], [30].

III. Problem Statement

We describe the assumptions, experimental setup, and peg transfer task in this section.

A. Assumptions

We consider a version of the FLS peg transfer task that involves transferring six red, 3D-printed blocks from the pegs on the left side of the board (Figure 1) to the right side of the board. Each starting peg p_s on the left side of the board has index $p_s \in \{0, 1, 2, 3, 4, 5\}$. Each target peg p_t on the right side of the board has index $p_t \in \{6, 7, 8, 9, 10, 11\}$ (Figure 2).

1) Simulator Assumptions and Setup: Our digital twin framework utilizes a human-operable simulation of the SRI Taurus RSA robot performing the FLS peg transfer task. This simulator generally reflects the dynamics of the physical system, modeling the robot kinematics as well as the states of the blocks and pegboard. In certain rare cases, however, the physics of the simulator differ markedly from the real RSA. Occasionally, collision calculation errors allow two blocks to overlap on the same peg, or cause the pegboard to fly off the screen after contact with the robot arm. Similarly, friction mismatches sometimes cause the blocks to stick to or drop from the end effector despite no surgeon error. Nevertheless, these issues are infrequent and present minimal obstruction to our use. We assume that the real robot and simulator are controlled by different computers connected via an IP network. The simulator used in this work is based on the SRI Taurus robot [12], which has two 7 DOF arms and an actuated overhead camera with a single DOF. The environment (Figure 2) is designed in Unreal Engine and displayed using an Oculus Rift VR headset. The simulator presents the remote surgeon with a console displaying the view from the Taurus' overhead camera, and conveys the performed actions to the robot-side computer over our simulated network.

The framework determines p_s and p_t by analyzing the user's actions in the simulator; each peg has a bounding box that detects contact at the bottom of the board. p_s is reported as the peg corresponding to the bounding box the block is lifted out of, and p_t is the peg corresponding to the bounding box the block is placed in.

2) Real Workspace Assumptions and Setup: The robot setup consists of a da Vinci Research Kit (dVRK) with two 7-DOF patient side manipulator (PSM) arms equipped with large needle driver instruments [13]. The robot is equipped with a system to perform peg transfers autonomously given an input starting and ending peg tuple (p_s, p_t) from the simulator. The physical board configuration is modeled on the setup devised in Derossis *et al.* [7] to ensure comparability to other results. As in prior work [6], [8], [9], [10] we consider a visually-challenging setup where both the board and blocks are red (Figure 1, right). This setup ensures the robustness of our method under even low-contrast regimes. We also assume access to an overhead depth camera; our setup utilizes an inclined Zivid OnePlus S camera, which captures RGBD images at 1920x1200 resolution, as its primary input signal. Our method requires that all pegs are positioned within the dexterous workspace of both the simulated and real RSAs, and are in the field of view of both cameras; however, our method requires no further assumptions regarding the kinematic match between the simulated and real robots.

Fig. 2: **Sim Interface.** Four views of our simulator environment. **Top left:** The surgeon operates the simulator using an Oculus Rift VR headset and two handheld Oculus Touch controllers. **Top right:** The digital twin provides a full simulation of the robot and peg transfer environment. **Bottom left:** The peg transfer task from the surgeon's viewpoint in VR. **Bottom right:** The digital twin simulates the physics of the peg transfer task, providing a realistic digital analog.

B. Problem Definition

We define a **trial** of the peg transfer task as an attempt to move all six red blocks from the left pegs to the right pegs. As in prior work [9], we decompose each trial into several subtasks. A **pick** involves grasping a block and lifting it off the peg. A **place** is when a block is dropped on a target peg. A **transfer** is the act of picking a block from a peg and placing it on a target peg. We represent a transfer using a tuple (p_s, p_t) for the starting and ending pegs, as defined in Section III-A. A transfer is considered successful if transfer (p_s, p_t) is executed in simulation, and the same transfer (p_s, p_t) is executed successfully on the physical robot.

A robot trial starts with all six blocks on the left side of the board in both the simulator and real setup. In the simulator setup, the orientation of each block is fixed, such that each trial begins with the blocks in the same orientation, but in the

real setup, each block can be rotated around its peg by an arbitrary amount. The surgeon executes a sequence of transfers in simulation. The simulation then extracts information about the surgeon's actions and transmits packets to the robot. The robot receives this information and executes the surgeon's commands to transfer the blocks in the real setup. Once the real transfer is complete, the system sends the real state of the pegboard back to the simulator. The trial ends when the sixth transfer attempt concludes or when the simulator or robot enter an irrecoverable state. An irrecoverable state is a state from which subsequent transfers are dynamically infeasible due to kinematic or system limitations of the arms, either in simulation or real. One example of an irrecoverable state is if a block is dropped in a location that impedes future block placements. The transfer attempt responsible for reaching an irrecoverable state is considered a failure.

C. Failure Modes

A trial can experience the following errors during execution:

1) **Block Drop, Simulation (Block):** The block is irrecoverably dropped out of reach in simulation.
2) **Pegboard Inaccessible, Simulation (Board):** The simulation pegboard is pushed out of reach of the arms. This is an irrecoverable state, requiring the current trial to be scrapped.
3) **Surgeon Failure, Simulation (Surg.):** The surgeon moves the block such that it enters the bounding box of an incorrect peg, causing an incorrect message to be sent to the robot. For instance, if the surgeon intends to perform the transfer $(4, 10)$ but accidentally enters the bounding box of peg 7, the command sent to the robot is $(4, 7)$ instead of $(4, 10)$; this is registered as a surgeon failure. The robot places the block on peg 7 in real and the surgeon eventually places the block on peg 10 in simulation. The surgeon then correctly performs a subsequent transfer $(1, 7)$ in simulation, but the robot detects a block already present on peg 7 and refuses to execute the transfer in real. This is not registered as a failure for the second transfer, since the robot behaved as expected given the new state of the pegboard.
4) **Perception Failure, Real (Perc.):** The real robot incorrectly identifies the start peg as empty or the end peg as occupied, and fails to execute the transfer.
5) **Pick Failure, Real (Pick):** The real robot attempts but fails to pick the block.
6) **Place Failure, Real (Place):** The physical robot attempts but fails to place the block on the target peg, either placing on the wrong peg or failing to place on any peg at all.

IV. Digital Twin Framework

The digital twin framework consists of three stages for each individual peg transfer: **simulator command extraction** (corresponding to $\boldsymbol{B_i}$ and $\boldsymbol{C_i}$ in Figure 3), **command transmission**, **physical robot execution** (corresponding to $\boldsymbol{D_i}$ and $\boldsymbol{E_i}$), **real pegboard state identification** (corresponding to $\boldsymbol{F_i}$), **pegboard state transmission**, and **simulator state update** (corresponding to $\boldsymbol{G_i}$) (Figure 3).

A. Simulator Command Extraction

The surgeon performs a peg transfer in the simulator by teleoperating the simulated robot using the handheld Oculus Touch controllers. The surgeon picks a block from a starting peg and places it on a target peg.

B. Command Transmission

After the surgeon successfully executes a transfer in simulation, the simulator compiles the starting peg p_s and target peg p_t associated with the transfer into a network packet, where $p_s \in \{0, 1, 2, 3, 4, 5\}$ and $p_t \in \{6, 7, 8, 9, 10, 11\}$. The packet is sent to the physical robot's computer. If successfully transmitted, the received messages are decoded and p_s and p_t are sent to the robot to execute a physical transfer.

C. Physical Robot Execution

The robot receives a commanded peg transfer (p_s, p_t) via TCP (in the case of a lossy network) or UDP (all other cases), then decodes and autonomously executes it on the physical peg board. The robot uses the peg transfer algorithm from Hwang *et al.* [8] to perceive the block state using depth sensing and the iterative closest point (ICP) algorithm. The robot also uses the system identification and control algorithm from Hwang *et al.* [8] to provide accurate control by compensating for the cabling properties of the arm.

D. Real Pegboard State Identification

After the robot attempts a peg transfer, the robot takes a picture of the pegboard and determines which pegs are occupied by blocks. This step catches any errors made in the physical robot execution stage. The result is a six-tuple with one entry per block, listing the peg that each block is on.

E. Pegboard State Transmission

The robot transmits the six-tuple representing the peg position of each of the six blocks over the network to the the simulator.

F. Simulator State Update

Once the simulator receives the six-tuple, it sets the positions of each block to the corresponding peg number from the six-tuple.

V. Experiments

Each full trial consists of 6 block transfers, moving each block from the left side of the pegboard to the right side. For all experiment types (sim-only, real-only, and teleop) we execute transfers the following order, listed here in sim coordinates: $(4, 10), (1, 7), (5, 11), (2, 8), (6, 12), (3, 9)$. In all trials, one co-author (K. Srinivas) served as the human operator, after approximately 10 hours of experience using the VR interface.

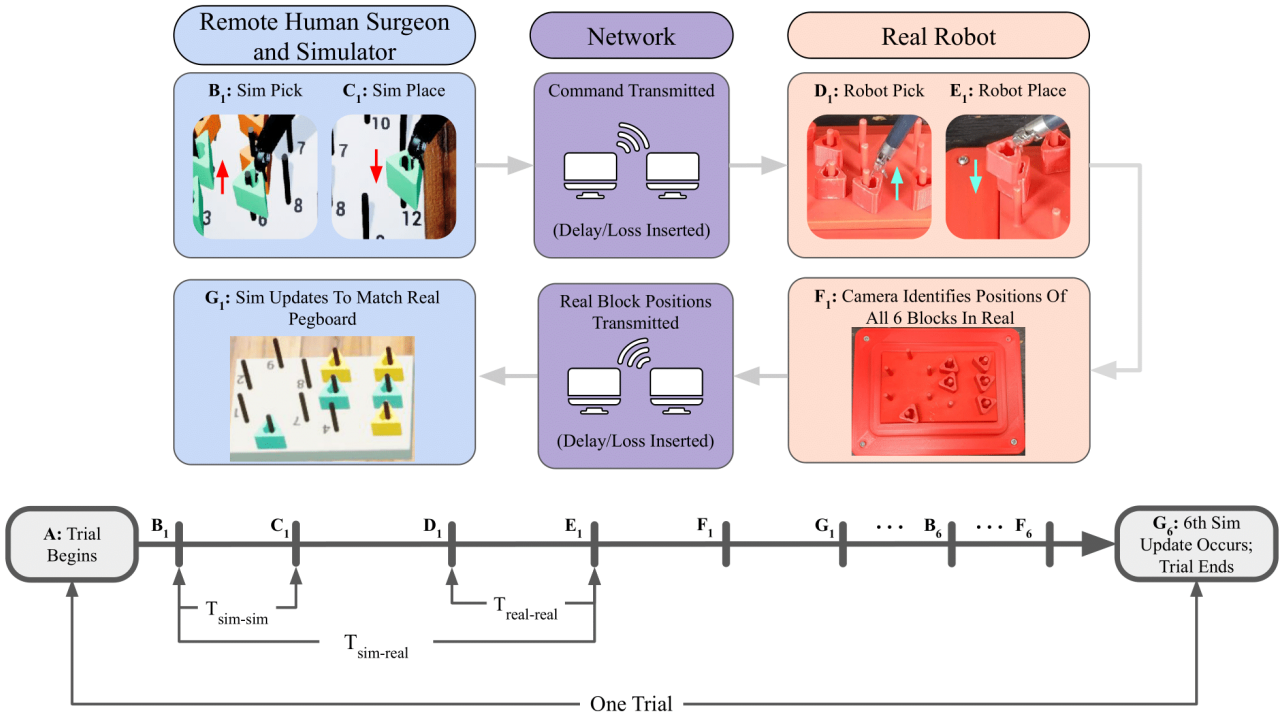

Fig. 3: **Method and data collection overview.** During each transfer run, the operator begins moving the simulated robot ($\boldsymbol{B_1}$ and $\boldsymbol{C_1}$), attempting to move all 6 blocks from one peg to another. Upon successful completion of each move in the simulator, a high-level command is sent over the network to the robot, subject to delays and instability. The robot attempts to duplicate each move made by the operator ($\boldsymbol{D_1}$ and $\boldsymbol{E_1}$). Once the physical transfer is complete, the robot takes an image of the pegboard to determine positions of all of the real blocks ($\boldsymbol{F_1}$). The robot then sends the positions of all six blocks back to the simulator over a network subject to delays and instability. After receiving all six block positions, the simulator updates to match the real state ($\boldsymbol{G_1}$). We measure 3 durations for each trial: $T_{sim-sim}$: start of simulation transfer to end of simulation transfer. This is measured from when the arm leaves the bounding box of the start peg in sim to when it enters the bounding box of the end peg in sim. $T_{real-real}$: start of real transfer to end of real transfer. This is measured from when the robot commands the arm to perform a transfer in real to when the arm places the block on the end peg in real. $T_{sim-real}$: start of simulation transfer to end of real transfer. This is measured from when the arm leaves the bounding box of the start peg in sim to when the arm places the block on the end peg in real.

A. Evaluation Metrics

We evaluate each attempted transfer individually and categorize failures as described in Section III-C. We record whether each attempted peg transfer is successful and report the trial's success rate. Each trial may contain at most 6 transfers, and can have fewer if an irrecoverable state was reached in the middle of a trial (Section III-C).

B. Simulation-Only Experiment

We benchmark the simulation quality by executing fully-teleoperated peg transfer trials without communication with the physical robot. For each simulation-only trial, we record 1 type of duration:

- $T_{sim-sim}$**:** start of sim transfer to end of sim transfer. This is measured from when the block leaves the bounding box of the start peg in simulation to when it enters the bounding box of the end peg in sim.

We evaluate failures as described in Section III-C, abbreviating failure modes as *Block* (block drop failures) or *Board* (pegboard inaccessible failures).

C. Simulation-Only Results

We perform 10 trials of the sim-only case with the dVRK, and report results in Table I. Across all 10 trials, we performed 55 total attempts at moving a block from the left side of the board to the right side, each of which took 7.65 seconds on average. A "perfect" trial with no failures involves 6 total attempts (left to right). We encountered 2 Pegboard Inaccessible failures, which prevented us from carrying out the remaining transfers in the respective trial. The success rate for a single transfer attempt is 52/55, or 94.54%.

$T_{sim-sim}$	Success / Attempts	Success Rate (%)	Failure Mode		
			(1)	(2)	(3)
7.65	52/55	94.54	1	2	0

TABLE I: **Simulation Results:** Success rate and transfer time (in seconds) for moving blocks from left side of the peg board to the right side of the peg board in the simulation. We categorize failures for the simulation into 3 modes: (1) block drop, (2) inaccessible pegboard, and (3) surgeon failure.

D. Real Robot-Only Experiment

To benchmark the quality of the autonomous physical robot peg transfers, we run real robot-only trials autonomously without the human teleoperator. We initialize a trial in real by randomly placing six blocks on to the six pegs on the left side of the peg board, producing variation

in the pose of each block. For each real robot-only trial, we record 1 type of duration:

- $T_{real-real}$**:** start of real transfer to end of real transfer. This is measured from when the robot commands the arm to perform a transfer in real to when the arm places the block on the end peg in real.

We evaluate failures as described in Section III-C, abbreviating failure modes as *Pick* (pick failures), *Place* (place failures), or *Perception* (perception failures).

E. Real Robot-Only Results

We also provide results for a robot-only case, in which the robot performs the aforementioned peg transfer task with no human input. We run 20 trials of 6 transfers each, relying entirely on the robot's own perception. As the system does not have to wait for the human operator in this case, the average length of a full trial is shorter, coming to an average of 5.859 seconds per transfer. Over these 120 transfers, the success rate of block transfer attempts was 100% (Table II). Across all 120 transfers, there was 1 partial perception failure, which the robot was able to recover from, and no pick or place failures.

$T_{real-real}$	Success / Attempts	Success Rate (%)	Failure Modes		
			(4)	(5)	(6)
5.859	120/120	100.00	0	0	0

TABLE II: **Real Robot Results:** Success rate and mean transfer time (in seconds) for moving blocks from left side of the peg board to the right side of the peg board in real. We categorize failures for the real robot into 3 modes: (4) perception failure, (5) pick failure, and (6) place failure.

F. Teleoperation Experiment

We perform a transfer in simulation using an Oculus Rift S headset with hand-held controllers and then send the tuple of (start peg, end peg) to the robot over the network. Upon receiving this tuple, the robot begins attempting the commanded transfer. For each teleop trial, we record 3 types of duration:

- $T_{sim-sim}$**:** start of simulation transfer to end of simulation transfer. This is measured from when the block leaves the bounding box of the start peg in sim to when it enters the bounding box of the end peg in sim.
- $T_{real-real}$**:** start of real transfer to end of real transfer. This is measured from when the robot commands the arm to perform a transfer in real to when the arm places the block on the end peg in real.
- $T_{sim-real}$**:** start of simulation transfer to end of real transfer. This is measured from when the arm leaves the bounding box of the start peg in sim to when the arm places the block on the end peg in real.

Due to transmission delays, $T_{sim-sim}$ and $T_{real-real}$ for a particular transfer do not necessarily add up to $T_{sim-real}$ for that transfer.

1) Teleop with Random Start State: We study the performance of the digital twin framework on the peg transfer task with a random initial state and no additional network conditions. In this version of the task, every block is assigned a random number between 1-12 without replacement to indicate the peg it starts on. Furthermore, there is no defined transfer order; the objective is to have a final state where all of the six blocks are on the right side.

2) Teleop with Varying Delay: We study the effects of network delays by delaying the simulation pick and place messages received by the dVRK. Delay is implemented using `netem`, which inserts a delay after every packet sent from sim, with the amount of delay sampled from a Gaussian distribution with specified mean and variance. We performed 5 trials each of 3 normal distributions:

- $\mathcal{N}(1 \text{ sec}, 1 \text{ sec})$
- $\mathcal{N}(10 \text{ sec}, 1 \text{ sec})$
- $\mathcal{N}(100 \text{ sec}, 1 \text{ sec})$

3) Teleop with Varying Packet Loss: We also study the effects of packet loss by intermittently dropping packets containing the pick and place messages sent by the sim. Packet loss is also implemented using `netem`, which drops each packet with a specified probability. We performed 5 trials each with 2 different probabilities: 10% and 25%.

G. Teleoperation Results

We report results in Table III for the various network cases. For the standard network case, we perform 11 trials, and have a success rate of 90.91% across all transfers, with a total of 6 failed transfers. 1 failure was a perception issue, where the robot either incorrectly identified the start peg as empty or the end peg as occupied and consequently refused to execute the commanded transfer. 5 failures were place failures in real, where the robot attempted to place the block on a target peg and either placed it on the wrong peg or failed to place it on any peg.

1) Teleop with Random Start State: For the random start state with standard network case, we perform 10 trials, and obtain a success rate of 86.20% across all transfers, with a total of 4 failed transfers. 2 failures were perception failures, and 2 were place failures in real. We report results in Table III.

2) Teleop with Varying Delay: For the delayed network case, the human operator performs 5 trials under each network distribution. We report results in Table III.

The 5 trials using the distribution with mean 1 sec have a success rate of 93.33% across all transfers, with a total of 2 failed transfers. Both were place failures in real, where the robot attempted to place the block on a target peg and either placed it on the wrong peg or failed to place it on any peg.

The 5 trials using the distribution with mean 10 sec have a success rate of 88.89% across all transfers, with a total of 4 failed transfers. 3 of these failures were place failures in real and 1 was a pick failure in real.

The 5 trials using the distribution with mean 100 sec have a success rate of 93.10% across all transfers, with a total of

Network Condition	$T_{sim-sim}$ (s)	$T_{real-real}$ (s)	$T_{sim-real}$ (s)	Success / Attempts	Success Rate (%)	Failure Mode: (1)	(2)	(3)	(4)	(5)	(6)
No delay, $p_{\text{loss}} = 0$	5.29	5.90	11.19	60/66	90.91	0	0	0	1	0	5
No delay, random start, $p_{\text{loss}} = 0$	7.45	5.96	13.43	25/29	86.20	0	0	0	2	0	2
$\mathcal{N}(1sec, 1sec)$ delay, $p_{\text{loss}} = 0$	8.09	6.06	15.25	28/30	93.33	0	0	0	0	0	2
$\mathcal{N}(10sec, 1sec)$ delay, $p_{\text{loss}} = 0$	5.33	6.04	23.17	32/36	88.89	0	0	0	0	1	3
$\mathcal{N}(100sec, 1sec)$ delay, $p_{\text{loss}} = 0$	4.20	5.56	113.29	27/29	93.10	0	0	0	0	1	1
No delay, $p_{\text{loss}} = 0.10$	4.42	5.91	10.37	28/30	93.33	0	0	0	1	0	1
No delay, $p_{\text{loss}} = 0.25$	4.57	5.91	11.29	26/30	86.67	0	0	0	1	1	2

TABLE III: **Teleop Results in the Presence of Varying Network Conditions:** Success rate and mean transfer time (in seconds) for transfers with varying QoS. We combine the failure cases from sim-only and real-only experiments for a total of 6 failure modes: (1) block drop, (2) inaccessible pegboard, (3) surgeon failure, (4) perception, (5) pick, and (6) place failures.

2 failed transfers. 1 failure was a pick failure in real, and 1 failure was a place failure in real.

The $T_{sim-sim}$ and $T_{real-real}$ times remain fairly consistent across all distributions but the $T_{sim-real}$ times increase with the mean of the distribution. This is because the delay is introduced between sim and real, so neither the operator's ability to operate in sim nor the robot's ability to operate in real is affected by this delay. The delay only adds to the time between the operator starting a transfer in sim and the robot completing that transfer in real, explaining the positive correlation between the mean of the distribution and the recorded $T_{sim-real}$ times.

3) Teleop with Varying Packet Loss: We perform 5 trials with 10% chance of packet loss and 5 trials with 25% chance of packet loss. We report results in Table III.

The 5 trials with 10% chance of packet loss have a success rate of 93.33% across all transfers, with 2 failed transfers. 1 failure was a perception failure in real, and 1 was a place failure in real. The 5 trials with 25% chance of packet loss have a success rate of 86.67%, with 4 failed transfers. 1 failure was a perception failure in real, 1 was a pick failure in real, and 2 were place failures in real.

Note that in both the varying delay and varying packet loss cases, none of the failures are caused directly by the delay or packet loss, but rather due to errors contained entirely within the robot. 9/14 failures are robot place failures, where the robot received the correct command but executed it incorrectly, and 3/14 failures are robot pick failures, where the robot failed to pick up the instructed block. The remaining 2/14 failures are robot perception failures, so all 14 failures are entirely contained within the robot.

For trials with added delay, the time to task completion increases roughly by the amount of the delay. For the trials involving 10% and 25% packet loss, the time to task completion did not increase compared with the no-delay trials. One notable exception to this is that the $T_{sim-real}$ time for the case with 10% chance of packet loss is lower than the $T_{sim-real}$ time for the standard network case by 0.82 s. This comes from the fact that the $T_{sim-sim}$ time for the 10% packet loss case is 0.87 s lower than the $T_{sim-sim}$ time for the standard network case, so even with the additional time taken to re-send dropped packets, the $T_{sim-real}$ time is lower than the standard network case.

VI. Discussion

Remote telesurgery has the potential to enable expert surgeons to provide medical assistance to patients who are in dangerous or difficult-to-reach environments. However, delays and network instability can make long-distance communication infeasible. This paper presents a novel Digital Twin framework for performing surgical peg transfer over delayed and lossy communication channels. The framework addresses these instability issues by maintaining a corresponding simulated environment for the human operator to interact with, and transmitting only the operator actions in this twin to the real RSA system for execution. The operator performs an action in this digital twin, which is then encoded into a high-level maneuver by the simulator and sent over the lossy network to the robot. The robot then receives this representation and autonomously performs the requested action. Results show that the success rate is comparable under varying network conditions, and that the failures we do encounter are not due to the state of the network but rather the performance of the real robot itself or uncorrelated operator failure in sim; however, total completion time was highly correlated with network delays. In future work, we hope to investigate the performance of the method under more extreme network conditions, and improve the feedback mechanism between the real robot and simulator to detect and recover from these failure cases.

Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, the CITRIS "People and Robots" (CPAR) Initiative, the Real-Time Intelligent Secure Execution (RISE) Lab and UC Berkeley's Center for Automation and Learning for Medical Robotics (Cal-MR). This work is supported in part by the Technology & Advanced Telemedicine Research Center (TATRC) project W81XWH-19-C-0096 under a medical Telerobotic Operative Network (TRON) project led by SRI International and donations from Intuitive Surgical, Google, Siemens, Autodesk, Bosch, Toyota Research Institute, Honda, Intel, Hewlett-Packard and by equipment grants from PhotoNeo and NVidia. The da Vinci Research Kit is supported by the National Science Foundation, via the National Robotics Initiative (NRI), as part of the collaborative research project "Software Framework for Research in Semi-Autonomous Teleoperation" between The Johns Hopkins University (IIS 1637789), Worcester Polytechnic Institute (IIS 1637759), and the University of Washington (IIS 1637444).

References

[1] G. H. Ballantyne, in *Robotic Surgery, Telerobotic Surgery, Telepresence, and Telementoring. Review of Early Clinical Results*, vol. 16, Surgical Endoscopy Interventional Techn, 2002.

[2] P. J. Choi, R. J. Oskouian, and R. S. Tubbs, "Telesurgery: Past, present, and future," *Cureus*, vol. 10, no. 5, 2018.

[3] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE access*, vol. 8, pp. 108 952–108 971, 2020.

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

[5] A. Bilberg and A. A. Malik, "Digital twin driven human–robot collaborative assembly," *CIRP annals*, vol. 68, no. 1, pp. 499–502, 2019.

[6] G. Gonzalez, M. Agarwal, M. V. Balakuntala, M. M. Rahman, U. Kaur, R. M. Voyles, V. Aggarwal, Y. Xue, and J. Wachs, "Deserts: Delay-tolerant semi-autonomous robot teleoperation for surgery," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 12 693–12 700.

[7] A. M. Derossis, G. M. Fried, M. Abrahamowicz, H. H. Sigman, J. S. Barkun, and J. L. Meakins, in *Development of a Model for Training and Evaluation of Laparoscopic Skills*, vol. 175, The American Journal of Surgery, 1998.

[8] M. Hwang, B. Thananjeyan, S. Paradis, D. Seita, J. Ichnowski, D. Fer, T. Low, and K. Goldberg, in *Efficiently Calibrating Cable-Driven Surgical Robots With RGBD Fiducial Sensing and Recurrent Neural Networks*, IEEE Robotics and Automation Letters (RA-L), 2020.

[9] S. Paradis, M. Hwang, B. Thananjeyan, J. Ichnowski, D. Seita, D. Fer, T. Low, J. E. Gonzalez, and K. Goldberg, in *Intermittent Visual Servoing: Efficiently Learning Policies Robust to Instrument Changes for High-precision Surgical Manipulation*, IEEE International Conference on Robotics and Automation (ICRA), 2021.

[10] M. Hwang, D. Seita, B. Thananjeyan, J. Ichnowski, S. Paradis, D. Fer, T. Low, and K. Goldberg, in *Applying Depth-Sensing to Automated Surgical Manipulation with a da Vinci Robot*, International Symposium on Medical Robotics (ISMR), 2020.

[11] A. Wilcox, J. Kerr, B. Thananjeyan, J. Ichnowski, M. Hwang, S. Paradis, D. Fer, and K. Goldberg, "Learning to localize, grasp, and hand over unmodified surgical needles," *arXiv preprint arXiv:2112.04071*, 2021.

[12] *Taurus: This small robot is reaching new heights and solving once-thought impossible challenges*, `https://medium.com/dish/taurus-this-small-robot-is-reaching-new-heights-and-solving-once-thought-impossible-challenges-e858fdbbb4ab`.

[13] P. Kazanzides, Z. Chen, A. Deguet, G. Fischer, R. Taylor, and S. DiMaio, in *An Open-Source Research Kit for the da Vinci Surgical System*, IEEE International Conference on Robotics and Automation (ICRA), 2014.

[14] T. Haidegger, J. Sándor, and Z. Benyó, "Surgery in space: The future of robotic telesurgery," *Surgical endoscopy*, vol. 25, no. 3, pp. 681–690, 2011.

[15] M. J. Lum, J. Rosen, T. S. Lendvay, A. S. Wright, M. N. Sinanan, and B. Hannaford, in *TeleRobotic Fundamentals of Laparoscopic Surgery (FLS): Effects of Time Delay-Pilot Study*, International Conference of the IEEE Engineering in Medicine and Biology Society, 2008.

[16] M. Meehan, S. Razzaque, M. C. Whitton, and F. P. Brooks, "Effect of latency on presence in stressful virtual environments," in *IEEE Virtual Reality, 2003. Proceedings.*, IEEE, 2003, pp. 141–148.

[17] I. S. MacKenzie and C. Ware, "Lag as a determinant of human performance in interactive systems," in *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, 1993, pp. 488–493.

[18] J. Marescaux, J. Leroy, F. Rubino, M. Smith, M. Vix, M. Simone, and D. Mutter, "Transcontinental robot-assisted remote telesurgery: Feasibility and potential applications," *Annals of surgery*, vol. 235, no. 4, p. 487, 2002.

[19] J.-H. Ryu, D.-S. Kwon, and B. Hannaford, "Stable teleoperation with time-domain passivity control," *IEEE Transactions on robotics and automation*, vol. 20, no. 2, pp. 365–373, 2004.

[20] M. Panzirsch, H. Singh, T. Krüger, C. Ott, and A. Albu-Schäffer, "Safe interactions and kinesthetic feedback in high performance earth-to-moon teleoperation," in *2020 IEEE Aerospace Conference*, IEEE, 2020, pp. 1–10.

[21] A. I. Grohmann, J. V. Busch, A. Bretschneider-Perez, C. Lehmann, and F. H. Fitzek, "JAVRIS: Joint artificial visual prediction and control for remote-(robot) interaction systems," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2022, pp. 835–840.

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

[23] I. Rivas-Blanco, C. J. Perez-del-Pulgar, C. López-Casado, E. Bauzano, and V. F. Munoz, in *Transferring Know-How for an Autonomous Camera Robotic Assistant*, vol. 8, Electronics: Cognitive Robotics and Control, 2019.

[24] J. D. Brown, C. E. O'Brien, S. C. Leung, K. R. Dumon, D. I. Lee, and K. Kuchenbecker, in *Using Contact Forces and Robot Arm Accelerations to Automatically Rate Surgeon Skill at Peg Transfer*, IEEE Transactions on Biomedical Engineering, 2017.

[25] J. Rosen and J. Ma, in *Autonomous Operation in Surgical Robotics*, vol. 137, Mechanical Engineering, 2015.

[26] M. Hwang, B. Thananjeyan, D. Seita, J. Ichnowski, S. Paradis, D. Fer, T. Low, and K. Goldberg, "Superhuman surgical peg transfer using depth-sensing and deep recurrent neural networks," *arXiv preprint arXiv:2012.12844*, 2020.

[27] H. Kim, M. Hwang, J. Kim, J. M. You, C.-S. Lim, and D.-S. Kwon, in *Effect of Backlash Hysteresis of Surgical Tool Bending Joints on Task Performance in Teleoperated Flexible Endoscopic Robot*, vol. 16, The International Journal of Medical Robotics and Computer Assisted Surgery, 2020.

[28] A. Abiri, J. Pensa, A. Tao, J. Ma, Y.-Y. Juo, S. J. Askari, J. Bisley, J. Rosen, E. P. Dutson, and W. S. Grundfest, in *Multi-Modal Haptic Feedback for Grip Force Reduction in Robotic Surgery*, vol. 9, Scientific Reports, 2019, p. 5016.

[29] M. M. Rahman, N. Sanchez-Tamayo, G. Gonzalez, M. Agarwal, V. Aggarwal, R. M. Voyles, Y. Xue, and J. Wachs, "Transferring Dexterous Surgical Skill Knowledge between Robots for Semi-autonomous Teleoperation," in *IEEE International Conference on Robot and Human Interactive Communication (Ro-Man)*, 2019.

[30] N. Madapana, M. M. Rahman, N. Sanchez-Tamayo, M. V. Balakuntala, G. Gonzalez, J. P. Bindu, L. Venkatesh, X. Zhang, J. B. Noguera, T. Low, *et al.*, in *DESK: A Robotic Activity Dataset for Dexterous Surgical Skills Transfer to Medical Robots*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019.