DESERTS: DElay-tolerant SEmi-autonomous Robot Teleoperation for Surgery

Glebys Gonzalez[1], Mridul Agarwal[2], Mythra V. Balakuntala[3], Md Masudur Rahman[4], Upinder Kaur[3], Richard M. Voyles[3], Vaneet Aggarwal[1,2], Yexiang Xue[4], Juan Wachs[1]

***Abstract*— Telesurgery can be hindered by high-latency and low-bandwidth communication networks, often found in austere settings. Even delays of less than one second are known to negatively impact surgeries. To tackle the effects of connectivity associated with telerobotic surgeries, we propose the DESERTS framework. DESERTS provides a novel simulator interface where the surgeon can operate directly on a virtualized reality simulation and the activities are mirrored in a remote robot, almost simultaneously. Thus, the surgeon can perform the surgery uninterrupted, while high-level commands are extracted from his motions and are sent to a remote robotic agent. The simulated setup mirrors the remote environment, including an alpha-blended view of the remote scene. The framework abstracts the actions into atomic surgical maneuvers (surgemes) which eliminate the need to transmit compressed video information. This system uses a deep learning based architecture to perform live recognition of the surgemes executed by the operator. The robot then executes the received surgemes, thereby achieving semi-autonomy. The framework's performance was tested on a peg transfer task. We evaluated the accuracy of the recognition and execution module independently as well as during live execution. Furthermore, we assessed the framework's performance in the presence of increasing delays. Notably, the system maintained a task success rate of 87% from no-delays to 5 seconds of delay.**

I. INTRODUCTION

The use of medical teleoperation in austere environments, where the resources are limited, medical facilities lacking, and communication bandwidth is poor or nonexistent, is a top priority in military medicine. Critical injuries on a battlefield can get real-time remote medical assistance from robotic agents, reducing the risk of exposing medics to direct fire. Likewise, robotic assistants are reliable and expendable agents, easily reconfigurable for a variety of surgical and medical procedures. However, such austere environments are often limited by bandwidth, network connectivity, and latency issues, posing a challenge to teleoperation. This is critical where effective surgical intervention requires continuous high-resolution information relay, with minimal delay.

Previous research shows the fatal effect that delays have in teleoperated robotic surgery [1], [2], [3], [4], [5]. It has been shown that delays as short as 200 milliseconds can cause a significant effect on surgical performance [5]. The degradation of task performance can lead to increased mortality risks. Even with the fastest networks, delays are often unavoidable for long-distance communication [6], [7]. The problem is exacerbated due to jitters and interruptions introduced by low-quality connections in austere areas. These issues severely limit the practical applicability of telerobotic surgery. Hence, we leverage machine learning to overcome the challenges of real-time operation.

In this paper, we present a novel framework, referred to as - DESERTS - which uses a learning-enabled closed-loop system to overcome the challenges associated with telesurgery in austere conditions. The goal is to deploy a robotic agent on a forward operating basis, with a surgeon operating it from a safe location. The surgeon's movements are encoded into unit instructions (surgemes), which are then sent to a robot, thereby limiting the dependence on the quality and reliability of the network.

The proposed framework aims to tackle the problem of delays in teleoperated robotic surgery by using a virtual representation of the robot as a bridge between the surgeon and the physical robot. The surgeon operates using a simulator, instead of the live video stream. The simulated system at the server's side recognizes and delivers the surgical instructions to the remote robot, while the robot sends back pose information of objects and tools, thereby eliminating the need to encode video frames. This exchange of high-level commands is efficient and allows to semi-autonomously execute surgical maneuvers in a safe manner. The feedback module displays the location of the remote robot and relevant elements in the surgical environment. This information is shown alpha-blended in the simulator side to enable situational awareness. The complete pipeline is shown in Figure 1.

We tested the performance of this framework by executing a fundamental procedure of laparoscopic surgery: a peg transfer task, which represents a challenge for dual-arm coordination and robotic grasps involving obstacle avoidance. The peg transfer setup is colored red and submerged partially in artificial blood to simulate a realistic surgical scene. Using artificial blood poses additional challenges to computer vision algorithms and motor control.

We evaluated the accuracy of the recognition and execution modules, first independently, then we tested the performance of recognition, execution, and feedback modules, in-tandem. In particular, we assessed the framework's performance in the presence of delays. To test the response of the system under various delay setups, we first analyzed the performance under no delay, followed by delays of 1s and 5s to simulate extreme adversarial settings.

[1]School of Industrial Engineering, Purdue University, West Lafayette, IN, 47907, USA gonza337, jpwachs@purdue.edu

[2]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, 47907, USA agarw180, vaneet@purdue.edu

[3]School of Engineering Technology, Purdue University, West Lafayette, IN, 47907, USA mbalakun, kauru, rvoyles@purdue.edu

[4]Department of Computer Science, Purdue University, West Lafayette, IN, 47907, USA rahman64, yexiang@purdue.edu

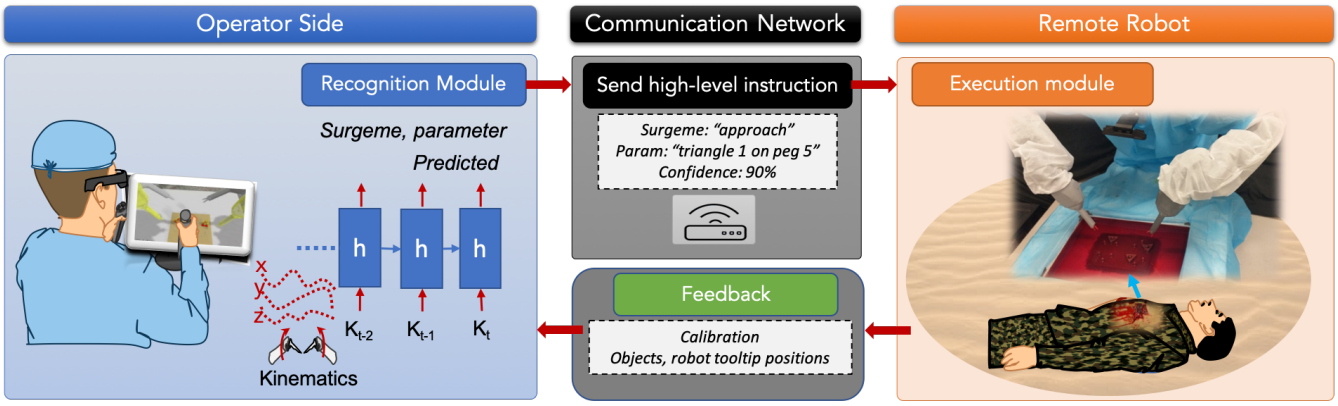

Fig. 1. An overview of the DESERTS framework.

The contributions of this paper are:

- A closed-loop telesurgery framework that is robust to delays and bandwidth limitations.
- A deep-learning architecture for frame-wise real-time recognition of atomic surgical maneuvers (surgemes).
- A detailed analysis of the systems' performance as well as the sub-modules under various adversarial settings.

The paper is organized as follows: The paper is organized as follows: II discusses the previous work in this area, followed by the methods used in the DESERTS framework in Section III. Section IV presents the experiments performed for validating the effectiveness of this framework, along with the results. Section V discusses the conclusion of this work, along with the future scope for the extension of these methods.

II. Background and Related Work

Teleoperation can be employed to control a robot from a distance, where the operator performs a task using the live feedback coming from the remote robot. Teleoperation includes factors such as task performance, context, and autonomy [8], [9], [10]. Teleoperated settings can range from direct teleoperation, where the robot mimics the surgeon's movement, to complete-autonomy, where a robot entirely replaces the surgeon's role [11], [12], [13], [14], [15], [16]. and operate alone. To introduce autonomy during teleoperation, one approach focuses on sending high level commands to the robot, instead of low-level kinematic data [17], [18], [19]. We propose a system were these high-level commands are not only automatically executed, but automatically recognized, in a simulated environment that does not rely on uninterrupted feedback. In particular, the framework facilitates the feedback to the surgeon by sending information of the tools and objects from the remote site and showing this environment information in alpha-blended displays [20], [21].

In surgery, a well-established concept for high-level commands can be found in Surgemes. Surgemes are a fundamental unit of surgical activity [22], [23], [24], which can help quantifying clinical skills among surgeons and even surgical proficiency [23]. Additionally, by decomposing a surgical procedure into parameterized surgemes, one can recognize patterns to create semi- autonomous robotic systems [25], [26], [27], [28], [29].

The DESERTS framework uses automatic surgeme recognition and execution in a closed-loop feedback setting, where an operator controls a robot using continuous feedback received from the remote site [30], [31]. The feedback information can be in the form of a video or kinematic data [32], [33], [34], [35], [36], [37]. Communication delays and intermittent connectivity [1], [2], [3], [4], [38], [39] severely real-time task performance. The following section describes in detail the implementation of the framework.

III. Methodology - DESERTS Framework

Our framework uses a simulator-based interface instead of live-streamed video to reduce the amount of information exchange. While the user operates the simulated robot, they can also see the pose of the actual robot and other objects related to the task, presented as an alpha-blended view in the simulator. At the remote site, an object detector identifies the objects of interest in the robot's field of view and sends them to the simulator for reconstruction. The simulator works on atomic units of surgical maneuvers referred to as surgemes. Using surgemes we obtain a high-level abstraction of the surgical procedure, hence reducing the size of the content to be exchanged. Our framework consists of four primary components: vision, recognition, communication, and execution modules, which are described in detail, as follows.

A. Vision Module

The vision module first identifies the objects and estimates their position in the real world (using semantic segmentation). The simulator then uses this information to update the scene. We first describe the object recognition unit running at the robot. Then we describe the simulator's display unit, which reproduces the remote robot scene in the simulator.

Object Recognition Unit. A 3D camera (Intel Realsense) is assembled on top of a remote ABB YuMi teleoperated robot. The camera streams color (RGB) and Depth image frames. This image stream is used to understand the environment state. This module extracts the 3D object poses

and robot tool-tip poses using two neural network-based architectures, Darknet (YoloV3) [40], and Mask-RCNN [41].

The YoloV3 (Darknet) network detects the 2D object bounding boxes in the RGB image frame. Since the YoloV3 network can only identify objects as regions of interest, an object tracker is added to track objects of the same class (for example, three triangular objects). We extend the tracking algorithm, SORT [42], which can additionally address noise, occlusion, and failed detection.

The tracker uses a filter to smooth the objects' motion per frame. It uses the object's velocity and a Kalman filter to estimate objects' position. The estimates are used to keep track of objects' position. The tracking system accumulates the object detection information of the last ten frames and estimates the confidence of the objects' presence, which reduces false positives.

Display Unit. This unit allows the operator to see what is occurring at the remote site by reconstructing the scene from the objects recognized by the semantics identification unit. The simulator shows the remote robot environment in an alpha-blended layout (objects, remote robot), as shown in Figure 4. The simulator updates the alpha-blended objects at regular time steps.

As the simulated robot may run independently, the position of the simulated robot and the alpha-blended real robot may diverge. To control this discrepancy, the operator can reset the simulated robot to synchronize with the alpha-blended version. This allows the operator to perform error recovery when the remote robot fails. This offers a safety measure in case the remote robot performs fails during surgeme execution.

B. Recognition Module

The operator works with a simulation that corresponds to the remote robot's environment. A recognition module automatically identifies high-level surgical actions from the trajectories and video scenes. The recognized high-level surgical actions are sent to the remote robot through a communication network continuously.

To leverage the history of activity data, the recognition module predicts surgemes using an LSTM arquitecture [43]. A notation for the LSTM is defined as follows. Let each surgeme instance be a sequence of kinematic and video frames of length T, with true labels $\{y_t\}_{t=1}^{T}$ where $y_t \in \Delta^K$ and Δ^K is a probability simplex in K dimensions for K classes. The LSTM predicts class probability $\hat{y}_t \in \Delta^K$ at time t. As our framework consists of 7 surgeme classes, we have $K = 7$. We use a cross entropy loss defined defined in Equation (1) to train the LSTM network.

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{t=1}^{T} \sum_{k=1}^{7} \boldsymbol{y}_{t,i} \log(\hat{\boldsymbol{y}}_{t,i}) \tag{1}$$

As compared to the earlier methods that use a transfer learning framework [27], or use fixed-length sequences [26], using an LSTM to predict the surgeme at every time step allows identifying the surgeme class accurately before the operator completes the action. Further, this early surgeme prediction allows DESERTS to mitigate the impact of delays in the system.

C. Communication Module

The communication module is responsible for the data transfer between the operator and the robot side by facilitating message passing, including high-level surgeme commands and feedback information. The communication module has two components: (1) Operator to Robot: The identified surgeme in simulator is sent to the remote robot. In case of any disruption (delay or interruption), the robot needs to gets the missing surgeme information from the operator, without data loss, when the connection re-establishes. To ensure this, surgemes are communicated using a TCP protocol. (2) Robot to Operator: The operator has to be updated with the latest feedback as early as possible. In case of disruption (delay or interruption), the most recent feedback has to be sent to the operator when the connection re-establishes. The proirity here is to send the latest data rather than to avoid packet drop. Thus, a UDP protocol is adopted for this path to alleviate the costly handshakes and to update the simulator with the latest feedback messages [44].

D. Execution Module

After receiving the surgeme from the simulator via the communication module, the robot executes it in the real environment. The robot receives the surgeme label, prediction confidence and the surgeme parameters from the recognition module. The robot executes the surgemes only if there is sufficient confidence in the classification. The threshold for surgeme execution is set empirically. The recognition module can classify the surgeme with high confidence at an early stage of the execution, then the remote robot starts performing the surgeme before the operator finishes thereby reducing the lag between the operator and robot execution.

Surgeme Execution Unit. This unit performs a model-based execution of the surgemes to complete the peg transfer task. It requires two inputs from the recognition module; surgeme label and parameter; to execute a surgeme. Additionally, the model-based execution unit needs to identify the surgeme parameters, for example, the object grasp points. To recognize the surgeme parameters we perform instance segmentation and object shape detection using a Mask-RCNN network [45]. The surgeme parameters such as grasp points can be obtained from the object masks.

The execution of the surgemes on the peg transfer task is particularly challenging in a blood-occluded setting [46]. Because the foreground and background in the environment are uniform, and the liquid surface produces reflections, the depth image may be incorrect. To mitigate this issue, the object mask is dilated to fill any holes in the depth image.

Following is the description of the model-based surgeme execution,

S1. *Approach. Parameter- Target object triangle*. It recognizes the object of interest and extracts a region of interest

(ROI) from the object recognition unit. Based on the orientation of the object a approach point is estimated. The tooltip is then moved to the approach point.

S2. *Align and grasp. Parameter- Target object triangle*. The Point of Interest module estimates the feasible grasp points from the masked image. First the object mask centroid is computed. Next, the closest points to the center that lie on the mask contour are selected as possible grasp points (see Figure 2). Finally, the grasp point that is closest to the gripper is used to pick up the object .

Fig. 2. Object masks obtained from the image segmentation. The proposed method first obtains the contour of the mask (in blue). It then finds the closest point from the centroid on each edge of the contour (in red). The grasp point is the contact point with minimum distance (in green).

S3. *Lift. Parameter- None*. The dominant arm moves upward while holding the target object.

S4. *Transfer - get together. Parameter - None*. It brings together both arms in the middle of the robot's workplace.

S5. *Transfer-exchange. Parameter - None*. The initial hand releases the object, and the non-dominant hand grasps it.

S6. *Approach target. Parameter - Target pole*. It detects the center point of the pole and horizontally moves the arm with the objects close to the point. It ensures the robot hand remains above the top of the pole.

S7. *Align and place. Parameter - Target pole*. It moves down toward the top of the pole and finally releases the object on the target pole.

The complete pipeline is also provided in Figure 3 for a quick overview.

IV. Experiments and Results

A. Experiment Setup

We set up a peg transfer task and modified the environment to uniformly red to simulate a real surgical scene which is similar to [46]. The pegs and poles were partially submerged on artificial blood to reproduce the challenges during surgery. In particular, the artificial blood adds scattering, and reflections to the vision system.

On the remote robot side, the recognition system relies on two different networks to continuously identify and segment the objects in the environment. The first network detects the object's bounding boxes from the RGB image. Then, the second network segments the object in the bounding box. The vision networks were trained using image frames from the peg transfer environment.

The training data was further augmented by flipping, and adding variation in illumination by adding gaussian noise and changing the image gamma and saturation. These modifications increased the reflection variations in the artificial blood setup, helping mitigate object detection errors. The

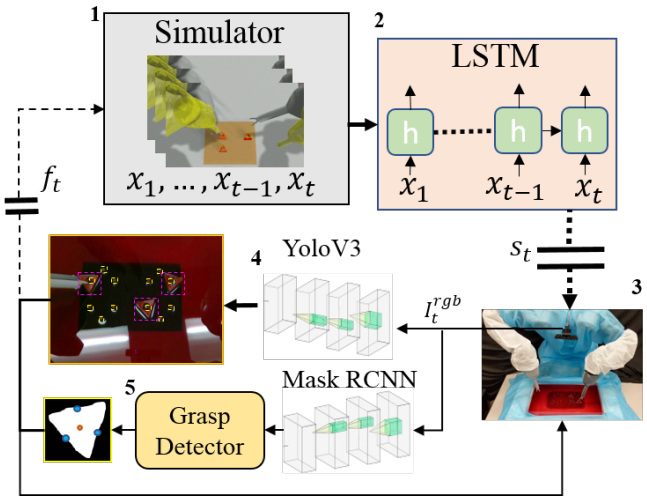

Fig. 3. The network architecture of DESERTS. At time t, the operator moves the simulated robot generating the kinematics $\{x_\tau\}_{\tau=1}^{t}$ of the simulated robot in the simulator (1). From $\{x_\tau\}_{\tau}^{t}$ the LSTM (2) predicts the surgeme. It is then sent to the robot (3). The robot obtains execution parameters from the YoloV3 (4) and the Mask RCNN (5). The feedback f_t from the YoloV3 is also sent back to simulator from the YoloV3.

scattering and reflections due to blood also affected the depth perception of the camera. Thus, we performed a depth value outlier rejection and median filtering to stabilize the image.

B. Live system control

We implemented an interface using the Razer Hydra sensing controller. In the simulator, the position and orientation of the controller were matched to the robot's end-effector. In addition, we used a pedal system to activate the robot's motion. The robot followed the controllers' motion only when the pedal remained pressed, allowing the user to reconfigure the hand to a comfortable place when the pedal is released (see Figure 4).

C. Results

The goal of the proposed framework is to mitigate the effect of delays over remote teleoperation. First, this section describes how each module on the framework was accessed. Then it outlines the task performance using different types of delays.

Vision Accuracy.

The vision system described in Section III-A has two components: a detection-tracker and an in- stance segmentation network. The detection-tracker network was trained weights using 100 images, which were manually annotated. Then, this model was tested using frames from a video sample of the robot interacting with the artificial blood setup. A total of 50,000 images were collected for testing. The object labels were generated through a semi-autonomous system that tracked the initial human annotations through entire image sequences. These annotations for the test images were manually verified at a later stage. The object detector produced an accuracy of 97.5% with a false positive rate of 0.01%, making it a reliable source of object locations during the live simulation feedback.

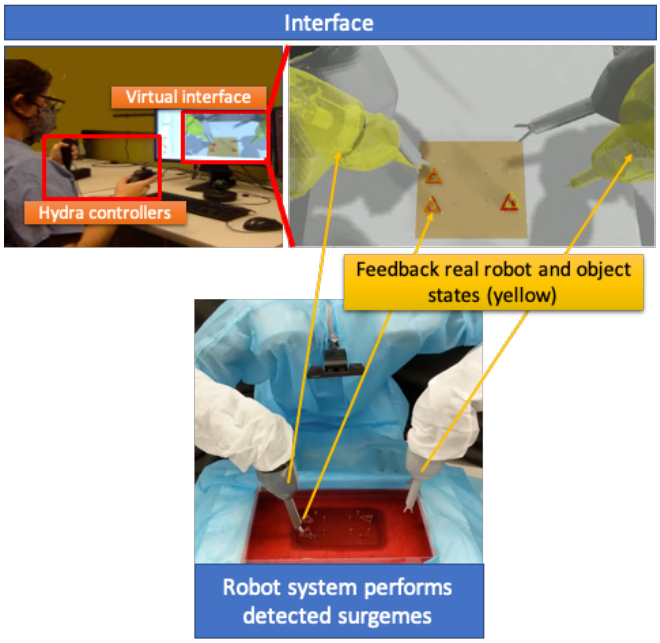

Fig. 4. Live simulator interface. Simulator screen showing the simulated robot in grey, the simulated pegs (triangles) in red, and the real pegs and the real robot as alpha blended in yellow. Real robot performing the task being completed in the simulator.

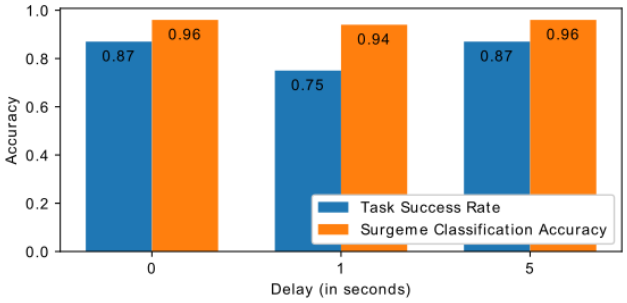

Fig. 5. Surgeme detection accuracy and task completion rate under delays. Since surgemes are recognized from the operators simulated robot state, the recognition accuracy does not suffer degradation.

Surgeme Recognition. To access the surgeme recognition network, we obtained the recognition accuracy of the LSTM network while the user executed the peg transfers in the simulated environment. During data collection, the simulator received live feedback from the remote system and used the communication network to send the surgeme recognized at each frame. This setup had a natural delay that was similar to the testing environment.

We collected and annotated 24 peg-transfer trials from two subjects (12 each). Then, the LSTM surgeme recognition network was trained and tested using a 80-20 data distribution model. The LSTM model was implemented in PyTorch [47] using a hidden layer size of 32. The input data consisted of kinematic information (similar to [25], [26], [27]) from the simulated robot. The model output was the surgeme labels (from seven surgemes). The recognition model achieved a testing accuracy of 85% when using the entire surgeme history.

Live surgeme recognition. In this section we discuss the surgeme recognition performance during the live peg transfer trials. As described in Section III-D, the execution module deter- mines when to execute a surgeme based on the recognition confidence, which is sent by the recognition module, along each label. The thresholds for each surgeme were empirically determined during a tuning phase. Once the remote robot receives a surgeme label with a high confidence (i.e. 90% confidence), the robot starts executing the corresponding surgeme. Since the surgemes can be recognized with a high confidence using partial history from the operator's performance, the robot can start the current action even before the operator finishes the surgeme in the simulator.

TABLE I

RECOGNITION PERFORMANCE AT THRESHOLDS (%)

Item	S1	S2	S3	S4	S5	S6	S7	Avg
Confidence Threshold	90	40	40	80	80	70	70	-
Fraction of surgeme	54	71	23	24	53	68	51	49
Accuracy	88	25	63	81	100	50	81	70

Table I shows the partial surgeme recognition results for peg transfer. It can be seen that the model achieves the thresholds on average using only half (i.e., 49%) of the surgeme historical data. Early surgeme detection reduces the robot's waiting time after completing a surgeme, thus decreasing the overall task completion time.

DESERTS Framework performace. Performace. To assess the effi- cacy of the developed framework, we evaluated the system under three delay configurations: 1) No delay, 2) One second of delay, and 3) Five seconds of delay. Previous research has shown that the chosen delays severely hinder surgical tele- operation, making it effectively unusable for the surgeon [4], [5]. Thus, we measured the surgeme classification accuracy, the completion time, and task success rate to evaluate the effects of delays on the DESERTS system.

First, we measured the recognition accuracy for the surgemes performed by the remote robot during execution. In a successful trial with no mistakes, the robot should recognize the seven peg-transfer surgemes for a successful trial. The results for remote recognition accuracy using different delays are shown in Figure 5. These results indicate that the recognition performance was not affected by delays. In particular, the no-delay setup obtained the same recognition accuracy as the 5 second-delay setup at 96%.

We also evaluated the performance of DESERTS to complete the peg transfer task. The task completion rate is defined as the number of successfully completed trials by the number of trials. The completion rate at different delays is presented in Figure 5.

In addition, we compared the task completion time in the presence of delays. We defined the user completion time as

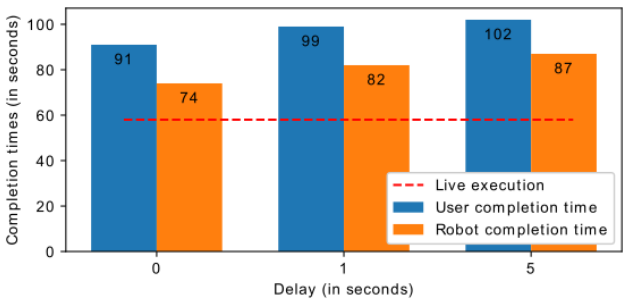

Fig. 6. Completion time under delays. Because the framework uses a simulator based interface, the operator can continue completing the task without increasing the completion time.

the duration of a user trial in simulation. The trial starts when the user moves the robot for the first time and ends when the last peg transfer is completed. Then, we measured the task completion time. The robot completion time is defined as the elapsed duration between the robot finishing execution of the last surgeme and the robot beginning the execution of the next surgeme. The task duration results are presented in Figure 6. We also obtained the completion time for direct teleoperation using the same robot under no delays (shown as a red line baseline in Figure 6).

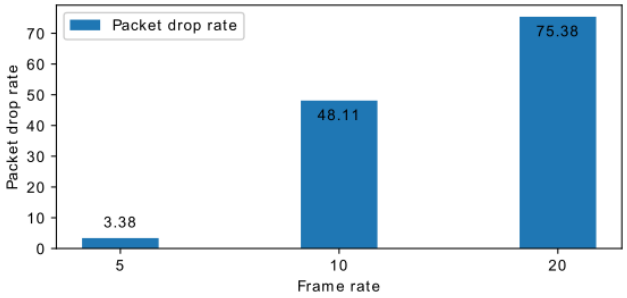

Fig. 7. Packet drop rate for different camera frame rates.

We analyzed the natural transmission delay while sending surgical instructions in absence of artificial delays. For the surgeme transfer from the simulator to the robot, the average delay was 0.057 seconds and a jitter of 0.048 seconds. Additionally, we observed that the simulator takes approximately 200 ms to update every frame and the surgeme messages are generated at a frequency of 5 Hz. The resulting bandwidth requirement for sending surgemes is only 1 KBps.

Finally, we evaluated the packet drop rate of the feedback module running on the UDP protocol (described in Section III-C) for different frame rates in the camera (see Figure 7). We noted that although the delays were 4 times lesser for a frame rate of 20 fps compared to the frame rate of 5 fps, the packet drop rate is 25 times higher. For a 30 fps camera video rate, without the presence of any artificial network delays, the natural delay of our system is bounded by 0.36 seconds with 99% probability. In contrast, the commercial live streaming services provide the lowest delay guarantee of 8 seconds [48]. We also compared the bandwidth requirement for the feedback module of DESERTS. Since a color image from a camera is a tensor of size $640 \times 480 \times 3$ bytes for 8-bit encoding, At 5 fps, the camera generates a stream of about 6 MBps. However, our proposed framework only generates 10 KBps, resulting in a total bandwidth requirement of 11 KBps (a reduction of 99.99% percent).

V. Conclusions

In dangerous austere environments, teleoperated robots can provide medical assistance for critical injuries, without risking the lives of medics in the field. However, network constraints and connectivity issues have restricted the use of remote teleoperation. We address the limited bandwidth and latency problem and their impact in teleoperation by introducing a novel closed-loop telesurgical framework, DESERTS. DESERTS works by abstracting low-level image data into high-level command data. First, the objects in the scene are recognized, which are tracked, and in turn, displayed at the simulator. At the operator side, only high level instructions (surgemes) are recognized and sent to the remote robot. Representing surgical maneuvers with high-level abstraction, in the form of surgemes, not only reduce the bandwidth requirements, but also allow the robot to operate semi-autonomously for short segments during the procedure, thereby making the whole system robust to delays. The validation of this framework includes the execution of a peg transfer task, a fundamental laparoscopic surgery skill, within a realistic surgical environment. We evaluated the overall and individual modules' performance on this task for a variety of delay settings. Notably, the framework achieves a task success rate of 87% with delays up to 5 seconds. These results show the effectiveness of the framework even in extreme adversarial settings. In the future, this framework can be extended to incorporate increasingly complex surgical procedures such as debridement, suturing, and incisions to be more useful to the field of telesurgery.

Acknowledgements

This work was supported by the following funding agencies: The Office of the Assistant Secretary of Defense for Health Affairs under Award No. W81XWH-18-1-0769, the NSF Center for Robots and Sensors for the Human Well-Being under Award No. CNS-1439717, the NSF FMitF program under award No. CCF-1918327 and the CR-II program under Award No. IIS-1850243. The Computational infrastructure was partially supported by Microsoft AI for Earth program. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by these institutions.

References

[1] T. Kim, P. Zimmerman, M. Wade, and C. Weiss, "The effect of delayed visual feedback on telerobotic surgery," *Surgical Endoscopy and Other Interventional Techniques*, vol. 19, no. 5, pp. 683–686, 2005.

[2] M. D. FABRLZIO, B. R. Lee, D. Y. Chan, D. Stoianovici, T. W. Jarrett, C. Yang, and L. R. Kavoussi, "Effect of time delay on surgical performance during telesurgical manipulation," *Journal of endourology*, vol. 14, no. 2, pp. 133–138, 2000.

[3] M. Anvari, T. Broderick, H. Stein, T. Chapman, M. Ghodoussi, D. W. Birch, C. Mckinley, P. Trudeau, S. Dutta, and C. H. Goldsmith, "The impact of latency on surgical precision and task completion during robotic-assisted remote telepresence surgery," *Computer Aided Surgery*, vol. 10, no. 2, pp. 93–99, 2005.

[4] S. Xu, M. Perez, K. Yang, C. Perrenot, J. Felblinger, and J. Hubert, "Determination of the latency effects on surgical performance and the acceptable latency levels in telesurgery using the dv-trainer® simulator," *Surgical endoscopy*, vol. 28, no. 9, pp. 2569–2576, 2014.

[5] A. Kumcu, L. Vermeulen, S. A. Elprama, P. Duysburgh, L. Platiša, Y. Van Nieuwenhove, N. Van De Winkel, A. Jacobs, J. Van Looy, and W. Philips, "Effect of video lag on laparoscopic surgery: correlation between performance and usability at low latencies," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 13, no. 2, p. e1758, 2017.

[6] S. Kassing, D. Bhattacherjee, A. B. Águas, J. E. Saethre, and A. Singla, "Exploring the "internet from space" with hypatia," in *Proceedings of the ACM Internet Measurement Conference*, ser. IMC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 214–229. [Online]. Available: https://doi.org/10.1145/3419394.3423635

[7] M. Handley, "Using ground relays for low-latency wide-area routing in megaconstellations," in *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, 2019, pp. 125–132.

[8] G. Adamides, G. Christou, C. Katsanos, M. Xenos, and T. Hadzilacos, "Usability guidelines for the design of robot teleoperation: A taxonomy," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 2, pp. 256–262, 2014.

[9] S. Lichiardopol, "A survey on teleoperation," *Technische Universitat Eindhoven, DCT report*, 2007.

[10] P. G. De Barros and R. W. Linderman, "A survey of user interfaces for robot teleoperation," 2009.

[11] M. Yip and N. Das, "Robot autonomy for surgery," *arXiv preprint arXiv:1707.03080*, p. 1, 2017.

[12] S. A. Pedram, P. Ferguson, J. Ma, E. Dutson, and J. Rosen, "Autonomous suturing via surgical robot: An algorithm for optimal selection of needle diameter, shape, and path," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2391–2398.

[13] B. Kehoe, G. Kahn, J. Mahler, J. Kim, A. Lee, A. Lee, K. Nakagawa, S. Patil, W. D. Boyd, P. Abbeel, and K. Goldberg, "Autonomous multilateral debridement with the Raven surgical robot," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1432–1439.

[14] J. Rosen, B. Hannaford, and R. M. Satava, *Surgical robotics: systems applications and visions*. Springer Science & Business Media, 2011.

[15] G. S. Guthart and J. K. Salisbury, "The intuitive/sup tm/telesurgery system: overview and application," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1. IEEE, 2000, pp. 618–621.

[16] R. Taylor, P. Jensen, L. Whitcomb, A. Barnes, R. Kumar, D. Stoianovici, P. Gupta, Z. Wang, E. Dejuan, and L. Kavoussi, "A steady-hand robotic system for microsurgical augmentation," *The International Journal of Robotics Research*, vol. 18, no. 12, pp. 1201–1210, 1999.

[17] D. Hu, Y. Gong, B. Hannaford, and E. J. Seibel, "Semi-autonomous simulated brain tumor ablation with ravenii surgical robot using behavior tree," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3868–3875.

[18] J. Kofman, Xianghai Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 5, pp. 1206–1219, 2005.

[19] G. De Rossi, M. Minelli, A. Sozzi, N. Piccinelli, F. Ferraguti, F. Setti, M. Bonfé, C. Secchi, and R. Muradore, "Cognitive robotic architecture for semi-autonomous execution of manipulation tasks in a surgical environment," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7827–7833.

[20] H. Hedayati, M. Walker, and D. Szafir, "Improving collocated robot teleoperation with augmented reality," in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 78–86.

[21] J. Bohren, C. Paxton, R. Howarth, G. D. Hager, and L. L. Whitcomb, "Semi-autonomous telerobotic assembly over high-latency networks," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2016, pp. 149–156.

[22] H. C. Lin, I. Shafran, D. Yuh, and G. D. Hager, "Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions," *Computer Aided Surgery*, vol. 11, no. 5, pp. 220–230, 2006.

[23] C. E. Reiley and G. D. Hager, "Task versus Subtask Surgical Skill Evaluation of Robotic Minimally Invasive Surgery," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2009*, ser. Lecture Notes in Computer Science, G.-Z. Yang, D. Hawkes, D. Rueckert, A. Noble, and C. Taylor, Eds. Springer Berlin Heidelberg, 2009, pp. 435–442.

[24] ——, "Decomposition of robotic surgical tasks: an analysis of subtasks and their correlation to skill," in *M2CAI workshop. MICCAI, London*, 2009.

[25] M. M. Rahman, M. V. Balakuntala, M. Agarwal, U. Kaur, L. N. V. Venkatesh, G. Gonzalez, N. Sanchez-Tamayo, Y. Xue, R. M. Voyles, V. Aggarwal, and J. Wachs, "Sartres: A semi-autonomous robot teleoperation environment for surgery," in *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization Journal - AECAI2020 Special Issue*, 2020.

[26] 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 *28th IEEE International Conference on Robot and Human Interactive Communication (Ro-Man-2019)*, 2019.

[27] N. Madapana, M. M. Rahman, N. Sanchez-Tamayo, M. V. Balakuntala, G. Gonzalez, J. P. Bindu, L. N. V. Venkatesh, X. Zhang, J. B. Noguera, T. Low, R. Voyles, Y. Xue, and J. Wachs, "Desk: A robotic activity dataset for dexterous surgical skills transfer to medical robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2019)*, 2019.

[28] S. Sen, A. Garg, D. V. Gealy, S. McKinley, Y. Jen, and K. Goldberg, "Automating multi-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4178–4185.

[29] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Unsupervised surgical task segmentation with milestone learning," in *Proc. Intl Symp. on Robotics Research (ISRR)*, 2015.

[30] 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.

[31] J. Kofman, X. Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE transactions on industrial electronics*, vol. 52, no. 5, pp. 1206–1219, 2005.

[32] A. J. Hung, J. Chen, A. Shah, and I. S. Gill, "Telementoring and telesurgery for minimally invasive procedures," *The Journal of urology*, vol. 199, no. 2, pp. 355–369, 2018.

[33] S. Treter, N. Perrier, J. A. Sosa, and S. Roman, "Telementoring: a multi-institutional experience with the introduction of a novel surgical approach for adrenalectomy," *Annals of surgical oncology*, vol. 20, no. 8, pp. 2754–2758, 2013.

[34] D. H. Shin, L. Dalag, R. A. Azhar, M. Santomauro, R. Satkunasivam, C. Metcalfe, M. Dunn, A. Berger, H. Djaladat, M. Nguyen, *et al.*, "A novel interface for the telementoring of robotic surgery," *BJU international*, vol. 116, no. 2, pp. 302–308, 2015.

[35] F. Ferland, F. Pomerleau, C. T. Le Dinh, and F. Michaud, "Egocentric and exocentric teleoperation interface using real-time, 3d video projection," in *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, ser. HRI '09. New York, NY, USA: ACM, 2009, pp. 37–44. [Online]. Available: http://doi.acm.org/10.1145/1514095.1514105

[36] P. Gambadauro and A. Magos, "Nest (network enhanced surgical training): a pc-based system for telementoring in gynaecological surgery," *European Journal of Obstetrics & Gynecology and Reproductive Biology*, vol. 139, no. 2, pp. 222–225, 2008.

[37] H. Sebajang, P. Trudeau, A. Dougall, S. Hegge, C. McKinley, and M. Anvari, "Telementoring: an important enabling tool for the community surgeon," *Surgical innovation*, vol. 12, no. 4, pp. 327–331, 2005.

[38] J. S. Kay and C. E. Thorpe, "Operator interface design issues in a low-bandwidth and high-latency vehicle teleoperation system," *SAE transactions*, pp. 487–493, 1995.

[39] L. H. Frank, J. G. Casali, and W. W. Wierwille, "Effects of visual display and motion system delays on operator performance and uneasiness in a driving simulator," *Human Factors*, vol. 30, no. 2, pp. 201–217, 1988.
[40] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.
[41] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.
[42] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464–3468.
[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[44] M. Schwartz, *Telecommunication networks: protocols, modeling and analysis*. Addison-Wesley Reading, MA, 1987, vol. 7.
[45] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," 2017.
[46] M. Hwang, D. Seita, B. Thananjeyan, J. Ichnowski, S. Paradis, D. Fer, T. Low, and K. Goldberg, "Applying Depth-Sensing to Automated Surgical Manipulation with a da Vinci Robot," in *International Symposium on Medical Robotics (ISMR)*, 2020.
[47] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.
[48] "Live event low latency settings," https://docs.microsoft.com/en-us/azure/media-services/latest/live-event-latency, accessed: 2020-10-30.