AWARD NUMBER: W81XWH-18-1-0769

TITLE: A Fundamental Theory for Dexterous Surgical Skills Transfer to Medical Robots

PRINCIPAL INVESTIGATOR: Juan P Wachs

RECIPIENT: Purdue University, West Lafayette, IN

REPORT DATE: October 2020

TYPE OF REPORT: Annual Report

PREPARED FOR: U.S. Army Medical Research and Materiel Command
Fort Detrick, Maryland 21702-5012

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
October 2020	Annual	15Sep2019-14Sep2020

4. TITLE AND SUBTITLE

A Fundamental Theory for Dexterous Surgical Skills Transfer to Medical Robots

5a. CONTRACT NUMBER

5b. GRANT NUMBER
W81XWH-18-1-0769

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Juan P Wachs

E-Mail:

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Purdue University

West Lafayette, 47907, IN

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army Medical Research and Materiel Command
Fort Detrick, Maryland 21702-5012

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTE

14.ABSTRACT
Chronic diseases like cancers that evade the immune system continue to impact the lives of millions of patients worldwide causing high morbidity and mortality as well as significant economic burden. It is now evident that in melanoma tumors, persisting antigen results in the development of dysfunctional or exhausted anti-tumor T cells (Texs). Continuous TCR stimulation and multiple suppressive mechanisms limit effector T cell functions, including high expression and signaling of multiple inhibitory receptors (e.g., PD-1, CTLA- 4, Lag3, Tim3) including PSGL-1 (P-selectin glycoprotein ligand-1), which we identified. Therapeutic success targeting these inhibitory checkpoints (PD-1/PDL-1 and CTLA-4) using blocking antibodies can reinvigorate Texs and improve pathogen and tumor control in mice and patients. These drugs are now standard treatment for melanoma, but at present are only effective in a subset of patients, highlighting the need to identify additional inhibitory pathways that can be targeted to improve patient outcomes. Our purpose is to understand when PSGL-1 signaling is required to establish T cell exhaustion and in what cell types PSGL-1 contributes to inhibition of T cell function. We will also understand whether PSGL-1 can be targeted to reverse T cell exhaustion to promote tumor control.

15. SUBJECT TERMS

NONE LISTED

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			USAMRMC
U	U	U	UU	58	19b. TELEPHONE NUMBER *(include area code)*

TABLE OF CONTENTS

1. INTRODUCTION:

We research and study new approaches to transmit information (encoding/decoding) to maximize information content while reducing redundancy and common knowledge, under constrained communication settings. Based on the decoded and reconstructed information, we determine at what extent the machine needs to autonomously complete surgical procedures. The robot, in turn, will apply a sequence of maneuvers associated with the required surgical procedure. We theorize new approaches for transfer learning, where learnt patterns will be projected to different domains.

2. KEYWORDS:

Telerobotics, Transfer learning, telemedicine, deep neural networks, simulation, surgical training, co-presence, tele-operation, surgeme recognition.

3. ACCOMPLISHMENTS:

What were the major goals of the project?

Specific Aim 1: Data Collection and Sensory-motor Library Generation
Data collection – expanded dataset (30-Nov-2019 – 30-Sep-2020) 50%
Library Generation (15-Sept-2018 – 15-Dec-2018) 100%
Military provides feedback (15-Oct-2019 – 15-Dec-2019) 100%

Specific Aim 2: Semantics Rules Creation
Creation of terminal symbols (surgemes) (15-Jul-2020 – 10-Sept-2020) 100%
Development of grammar and rules (temporal models) (15-Nov-2019-15-Mar-2020) 10%
Use of vector representation (surgeme representation) 100%

Specific Aim 3: Learning with Imperfect and Randomly Delayed Information
Find Information theoretic bounds for optimal algorithms (15-Sept-2018 – 15-Mar-2019) 100%
Modification of standard algorithms, and analysis of error bounds (1-Feb -2020 – 1March-2020) 90%

Specific Aim 4: Integrating predictions with semi-supervision
Find the prediction error and competitive guarantees (15-July-2020 – 15-October-2020) 100%
Coupling multiple expert trajectories with attendant visual cues (15-July-2019 – 15-Jan-2020) 100%

Specific Aim 5: Visual recognition and adaptive learning of primitives

Train a deep neural network (DNN) with artificial anatomical images (15-July-2019 – 15-Oct-2019) 100%

Specific Aim 6: Motor command generation in constrained and new settings

Development of the peg transfer model for the VREP environment (15-Oct-2019 – 15-Jul-2020) 100%

Experimental Design: Metrics to evaluate the technical specifications of the system

Experiment 2: Metrics to assess learning with Imperfect and Randomly Delayed Information (15-Mar-2020 – 15-Oct-2020) 20%

Experiment 3: Metrics to Assess the Accuracy of High-Level Surgical Descriptors (15-Oct-2019 – 15-Jan-2020) 100%

Experiment 4: Metrics to Assess Connectivity Dropout and Recovery with Multiple Expert (15-Oct-2019 – 15-Jan-2020) 20%

What was accomplished under these goals?

Task 1.1: Data collection and Sensory Motor Library Generation - Peg transfer using the Da Vinci robot

The robotic skill library was expanded to include the da Vinci Research Kit (dVRK)[1] surgical robot. This robot was teleoperated during a peg transfer task used the same protocol of the previous data collection. A total of eight subjects performed 8 peg transfer trials each. In every trial the subjects performed 3 peg transfers. The initial locations of the triangles and the picking order followed a completely randomized design, while the tilt of the pegboard (left-tilt, right-tilt, no-tilt) and the type of transfer (left-to-right or right-to-left) followed a repeated measures design [2]. Figure 1 depicts these variables.

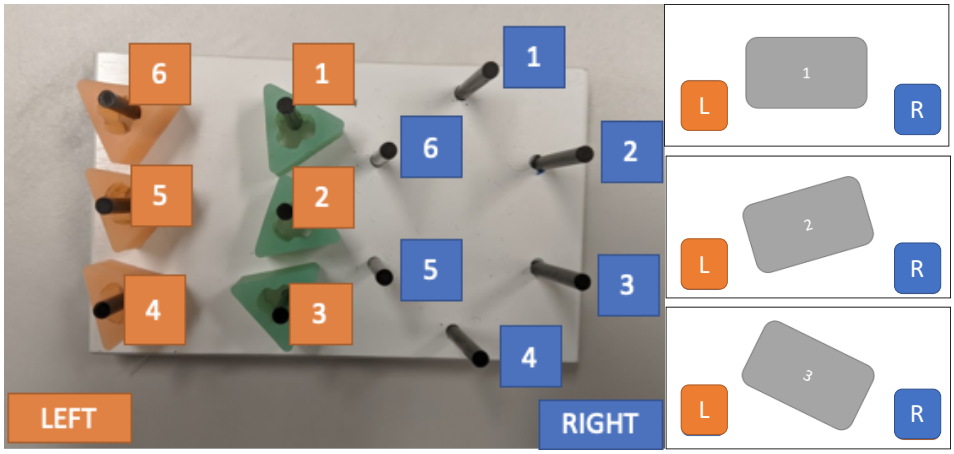

Figure 1. (Left) Possible triangle positions at the left and right sides of the pegboard. (Right) Possible tilts that the pegboard could be initialized with.

To exploit the flexibility and accuracy of the dVRK system, a professional laparoscopic pegboard was purchased. Thus, the task difficulty was increased to match the precision level offered by the da Vinci robot. Figure 2-Left shows the peg board used for the da Vinci setup.

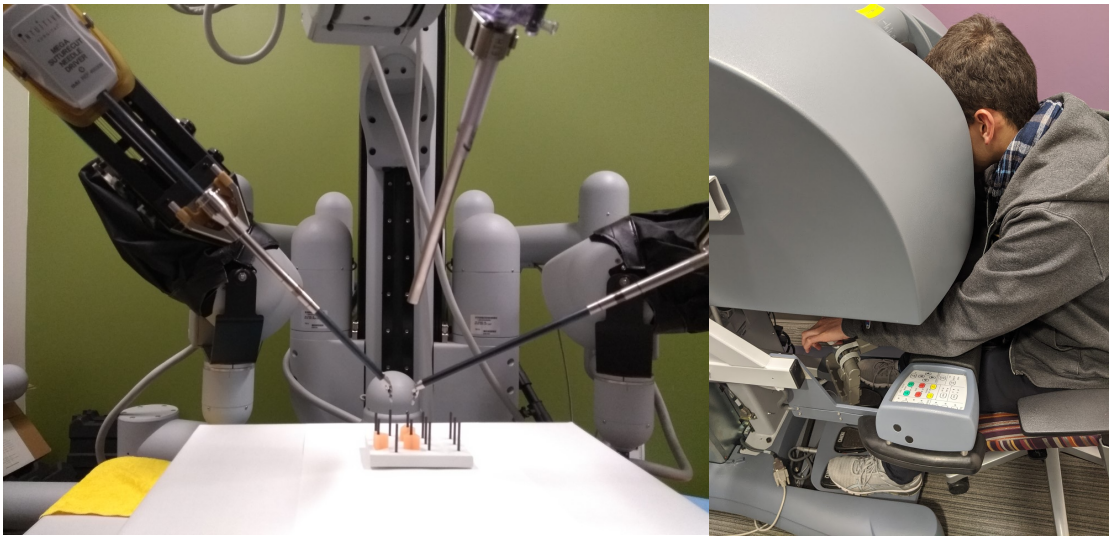

Figure 2. (Left) dVRK peg transfer setup. (Right) dVRK teleoperation setup for data collection.

A total of 144 peg transfers were collected from eight subjects. Each subject performed six trials. The subjects had a 15 minute practice session to familiarize themselves with the da Vinci console and controls (See Figure 2-Right) before the data recording. Every trial required three transfers from one side of the peg board to the opposite, starting from a random position. Both RGB video and kinematic information were collected during this phase. Table 1 explains the collected features in detail.

Table 1. Features recorded during data collection

Feature	**Description**
Endoscopic recording	RGB video recorded from the da Vinci stereoscopic endoscope. Since the RGB output is stereoscopic, two files are saved, the left-side video and the right-side video. The timestamp of each frame was also recorded for image-processing and annotation purposes.
Joint angles	The six joint angles that determine the position and orientation of the da Vinci arm. The angles are expressed in radians. The joint angles are recorded at each timestamp for both arms.
Gripper pose	The pose (orientation) of the gripper in quaternion format. This feature is recorded at teach timestamp for both arms.
Gripper position	The x,y,z position of the gripper with respect to the da Vinci base. This feature is recorded at each timestamp for both arms.
Gripper state	The aperture angle of the left and right grippers in radians. This feature is also recorded at each timestamp.

Each trial was annotated for surgeme segmentation and classification using the tool developed at the first stages of this project. A total of 4 annotators were familiarized with the concept of surgemes and had a practice annotation session before they annotated the real dataset. The annotation files show the initial and end frame of each observed surgeme, which surgeme was performed and if the execution was a success or not. Table 2. Shows the class distribution for the da Vinci dataset.

Table 2. Surgeme class distribution for the dVRK surgical robot library

Label	**Surgeme**	**N. of samples**
S1	Approach triangle	159
S2	Align and grasp triangle	168
S3	Lift	158
S4	Transfer – Get together (Approach opposite gripper)	153
S5	Transfer – Exchange triangle	153
S6	Approach pole	150

S7	Align and drop peg	150

1.1.1 Data collection of Operator data in Simulation:

To reduce latency, the operator in our framework performs the surgical task in a simulator that resembles the real scenario. The actions performed by the user are recognized and sent to the remote robot, where each instruction is performed autonomously. Thus, the framework requires automatic action recognition on the simulator system. To this end, we collected data for peg transfer task in the developed simulator for the ABB Yumi Robot (See Figure 3).

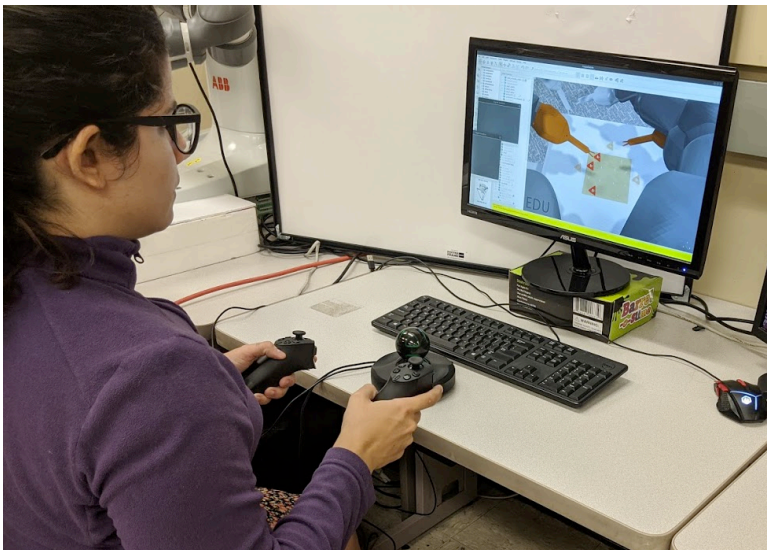

Figure 3. Data collection in the Simulator

Table 3. Data collection for the Yumi Simulator

Robot System	**No. of Subjects**	**No. of peg transfers**	**No. of surgical actions**
Yumi Simulator	3	108	Approach (object): 125 Align and grasp: 135 Lift (object): 124 Get together: 111 Exchange: 112 Approach (peg): 116 Align and place (peg): 115

The surgical actions (referred to as surgemes) for the Yumi simulation dataset were recorded from three subjects. Each subject was trained for 5+ hours to achieve a good level of dexterity for the task. The training sessions consisted on moving the arms through straight paths, followed by pick and place operations and finally performing peg transfer for 5 sessions. The data was recorded after the training, with a total of 108 peg transfers. The recording setup was the same as the real Yumi setup in DESK dataset. Each subject recorded 12 trials with 3 peg

transfers per trial[1]. Alternating with objects initialized on the left or the right. At the beginning of each session the initial position of the objects was randomized across the 6 pegs on the chosen side of the board. Table 3 summarizes the collected data from the Yumi simulator.

The new data collected was annotated with respect to the surgical actions using annotation tool developed in previous iterations.

Task 1.3 Policy for imperfect and delayed observations

We tested multiple models with the DESK dataset to predict current surgeme using past surgeme in case of delays.

We created a model using LSTM to use the temporal nature of the data. We used stateful LSTM to work with long sequences which can also have variable length. The input at each time consisted of the last available surgeme and the current kinematics of the robot. We found that using DESK data set we get only 25% accuracy using LSTMs. The performance doesn't improve much after hyperparameter tuning. Since the dataset size is small for a LSTM, we will attempt to use this model in an artificially augmented dataset in future iterations.

We tested the sensitivity of current surgeme on past surgemes or predictability of the data using a fully connected network by varying the input size to include previous h surgeme history. We found that the performance of the fully connected neural network does not significantly increase on increasing h. Detailed results of the experiments are provided in Figure 4.

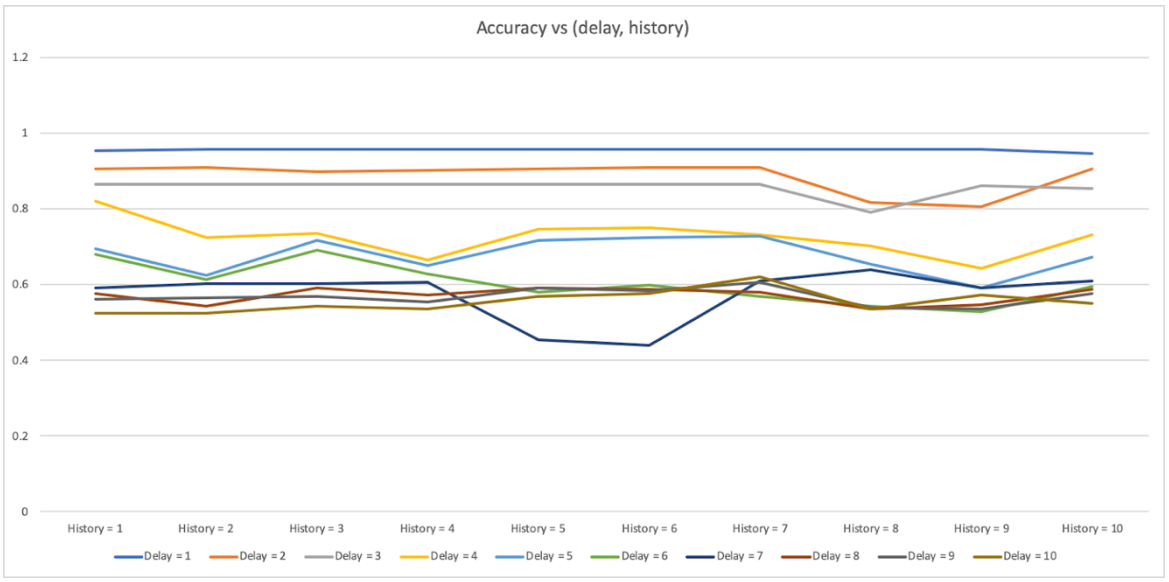

Figure 4: Chart showing the performance of fully connected neural network with varying history length

Table 4: Achieved accuracy of prediction for the current surgeme in presence of delays

Delay (in s)	Accuracy(in %)
0.000	100.00
0.167	95.28
0.333	90.50
0.500	71.00
0.667	83.00
0.833	68.50
1.000	65.43
1.167	62.40
1.333	57.37

We also present the achieved results in presence of varying delays using our fully connected neural network model. We note that the model can learn correctly as the case with no delays (delay = 0 s) gives 100% accuracy. This was not observed in LSTMs. Further, the accuracy of the model remains high (> 90%) with added delays. The accuracy of the model decreases as the delay increases from practical delay measurements (up to 500 ms). Table 4 shows the prediction accuracy for the NN model for different delays.

Task 2.1 Integrating predictions with semi-supervision

2.1.1 Automatic surgeme execution for artificial data generation

A script for automatic execution of the peg transfer task was developed in the Vrep simulation environment. The script autonomously executes the 7 surgemes of the peg transfer task while creating the annotation file that encompasses the kinematics variables of the robot and the environment variables at each frame and the corresponding surgeme. Table 5 shows the data transferred between the simulated sytem and the execution system.

Table 5. Data vector used in the communication between the simulated system and the real robot

Column	Recorded data
0	If the trial is valid
1	Recording Session
2	Count trial
3	Frame
4	Time stamp
5	Count trial
6	Surgeme value
7	Source pole id (from 1-12)
8	Target pole id (from 1-12)
9	Side pole transfer start (right or left)
10-16	Left joint angles
17-23	Left pose (position 17-19, Orientation 20-22, Gripper state 23)
24-30	Right joint angles
31-37	Right pose (position 31-33, Orientation 34-36, Gripper state 37)

38-43	Triangle 1 pose (position 38-40, Orientation 41-43)
44-49	Triangle 2 pose (position 44-46, Orientation 47-49)
50-55	Triangle 3 pose (position 50-52, Orientation 53-55)
56-61	Pose target pole
62-67	Peg board (position 62-64, Orientation 65-67)

2.1.2 Surgeme execution using learned spline parameters:

We developed a framework to learn the trajectories that humans make when executing each surgeme. Since the space of all possible trajectories highly dimensional, we constrained the possible trajectories to the space of spline functions. Thus, the main goal of this framework is, given the initial and the target point of a surgeme, to predict of the spine parameters that fit the trajectory of that sugeme.

Two different algorithms were explored to learn the spline parameters that describe the trajectories: A linear regression and a Neural Network with Mean Squared Error (MSE) loss. Additionally, the data pre-processing had two stages: 1) Surgeme translation and 2) parameter extraction. In the surgeme translation phase, all the surgeme samples for approach, align and grasp, lift, and align and drop were subtracted with the position of peg number 1 (See the pegboard in Figure 1), making all these surgemes to be spatially clustered at the same place. This was done to reduce the complexity of the regression function that predicts the spline parameters. In the parameter extraction phase, a spline was fi to the to the original sample, then W waypoints were extracted from the spline, were $W = degree\ of\ the\ spline + 1$. These W waypoints were used as the golden label during training, given the initial and ending points of a surgeme. Figure 5. Depicts the data pre-processing.

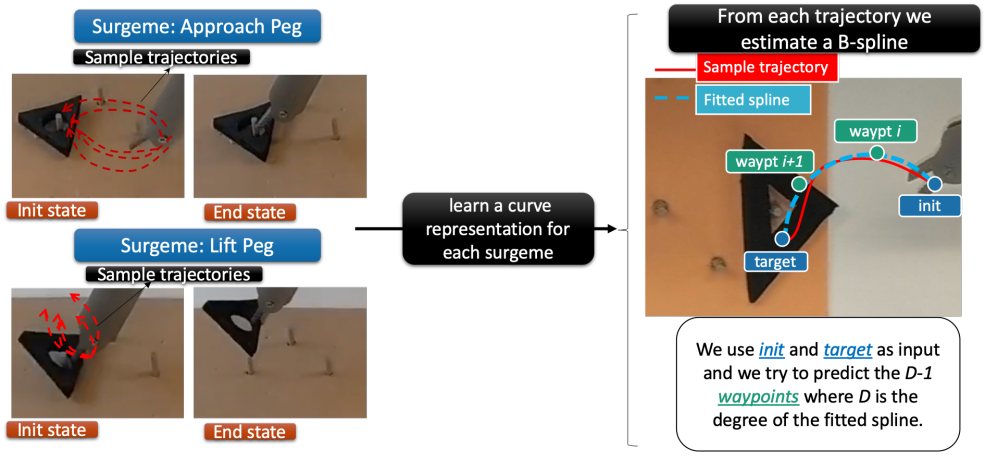

Figure 5. data pre-processing.

The Linear regression had a higher accuracy than the neural network when predicting the curve parameters. Thus, linear regression was selected as the learning method for the framework. Each surgeme was tested 10 times (independently of the other surgemes) using the Yumi, for a total of 100 execution tests. Two input parameters where given: 1) the current position of the robot and 2) the position of the target pole or triangle, which was determined using our vision framework [3]. The percentage of execution success is presented in Table 6.

for each surgeme. This framework was particularly successful at executing difficult align and grasp maneuvers, since the predicted trajectories would mimic a behavior that was very similar to the human's style. On the other hand, this framework had a very low success rate for the transfer surgemes (S4, S5). The data for these surgemes was extremely noisy, making the fitted curves in the preprocessing stage to look almost random. The low success rate of these two surgemes can be directly traced back to the regression scores of the linear regression (See Table 7). All the other surgemes have a regression score > 0.9 and that leads to a success rate of 1. For S4 and S5 the regression scores are much lower ~0.3 and ~0.6 respectively. Thus, the algorithm is not able to correctly predict the curve parameters for those trajectories.

Transferring cleaner data for S4 and S5 from other robotic datasets (Taurus or da Vinci) could lead to a better success rate. These possibilities will be explored in the next iterations.

Table 6. Sucess rate during the execution of each surgeme

Surgeme	**Percentage of successful executions**
S1	100%
S2	100%
S3	100%
S4	60%
S5	20%
S6	100%
S7	100%

Table 7. Linear Regression scores for the learned parameters of each surgeme.

Surgeme	**Left arm regression score**	**Right arm regression score**
S1	0.969	0.967
S2	0.996	0.997
S3	0.975	0.994
S4	0.389	0.331
S5	0.609	0.545
S6	0.968	0.968
S7	0.995	0.994

To reach autonomy, massive datasets of surgical data need to be collected for effective training of machine learning architectures. However, obtaining surgical data, especially for austere settings can be difficult. In previous work, we developed a transfer learning framework to

recognize surgical gestures (surgemes), with the goal of using minimal real data during training.

We expanded the experimental setup of this framework to include the Da Vinci dataset. This dataset was collected in the previous iteration and contains 144 recordings of peg transfers performed by a total of 8 subjects. Moreover, we added a Fast Fourier Transform [5,6] as a feature extractor, to further improve the surgeme recognition accuracy. Section 2.2.1 discusses these improvements in more detail and Section 2.2.2 shows the experimental results for these new pipelines.

2.2.3 Framework improvements:

The goal of this phase was to transfer knowledge for surgeme recognition using a diverse set of robots: Simulated Taurus, Real SRI Taurus, Real YuMi and the da Vinci surgical robot. We initially trained our previous framework using data from a simulated SRI Taurus and tested on a real Taurus robot, a real ABB YuMi and the da Vinci surgical robot. The surgeme classification accuracy obtained for both YuMi and da Vinci was below 30%. This low recognition can be traced to the kinematic and workspace between the simulated Taurus and the real Yumi and da Vinci. To improve the recognition accuracy, we used the FFT algorithm as a method to map the kinematic data from different robots to the same space. The FFT algorithm can map a time series to a frequency domain. FFT outputs a set of frequency component bins that can represent spatial or geometric features of the signal. Therefore, our current framework replaces the feature reduction done with PCA with a feature extraction using a histogram obtained from the FFT frequency bins.

Figure 6 shows our current framework. First, we reduced the dataset to the common kinematic features in all the robots: grippers' position, orientation, and state (open or close). The positional features were measured in the cartesian coordinate system (x,y,z), while the orientation was represented by the angles roll, pitch and yaw. This gives a total of 14 distinct features for each arm (7 each). A single 560-dimensional vector per surgeme (40 × 7 × 2) was created by concatenating 14 features for every frame. Then, we mapped these features to a common space using FFT. Further, a machine learning classifier was used to recognize the surgemes. Using a robot-agnostic set of features allows us to leverage information coming from robots of multiple domains.

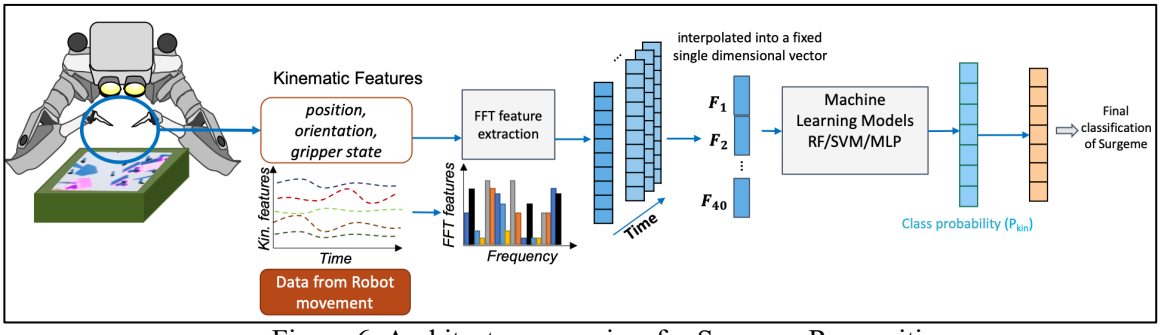

Figure 6. Architecture overview for Surgeme Recognition

2.2.4 Experimental Results:

For the transfer learning scenario, we started by training exclusively with simulation data and testing in the real robot using the architecture described in Figure 1. Then, we increasingly added data from the real scenario to the training set to simulate the effects of limited availability of the real data. We measured the presence of real data in the training model as a ratio between the real and simulated data. When the ratio value is zero, all the data comes from simulation. When the ratio value is 1, the data had a 50%-50% (50/50 = 1) distribution for real-simulated data. Figure 2 shows the classification accuracy of the models for all the real robots when they are trained using real data (orange line) against the performance when the training data is slowly added to the simulation data (blue line).

The Taurus II and da Vinci robots showed a classification accuracy of 97.5% and 93% respectively, even when there were no real examples included in the training set (data ratio=0). Adding a small number of real examples effectively improved the surgeme recognition accuracy, as shown in Figure 7. When the ratio of real to simulated data was 15%-85% (ratio=0.18), the classification accuracies went up to 99.7% for the Taurus II and 95.4% for the da Vinci. Therefore, these results show that surgeme classification on real environments can be achieved using a very small amount percentage of real data. The YuMi robot showed a slower convergence, with a ratio of 22%-78% of real to simulated data producing an accuracy of 81%. This accuracy discrepancy is likely due to the YuMi motions. The YuMi robot does not have three degrees of freedom at the gripper, making the orientation changes more abrupt. Thus, the teleoperators would choose a convenient orientation and default to translation motions. In contrast, the gripper's position and orientation changed constantly for the da Vinci, the Taurus II and the simulated Taurus. This inconsistency in the teleoperation resulted in very different FFT features for the YuMi. Thus, it needed more of its own data during training to produce an accuracy over 80%.

Both da Vinci and Taurus II show a faster convergence to a classification accuracy (≥95%), needing little to no data in the transfer setup. The FFT features allow to describe the spatial properties (shapes) of the surgemes, while retaining scale and translation invariance. This allows to describe the same trajectory or gripper aperture in robots that have different workspaces, showing significant improvement over the previous approach.

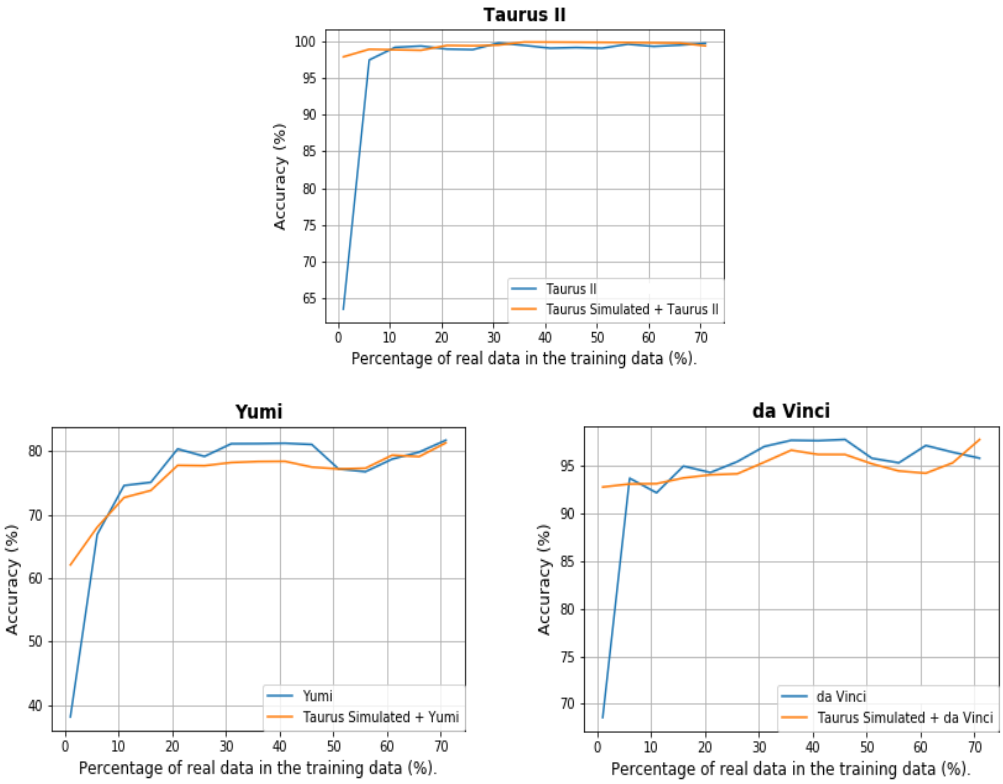

Figure 7. Performance comparison of training with the real data (no-transfer, shown in orange) vs training with only a percentage of the real data combined with simulation data (transfer learning scenario, shown in blue). The results are shown for the three real robots in the DESK dataset: the Taurus II robot (top) using SVM, the YuMi robot (bottom-left) using RF and the dVRK robot (bottom-right) using RF.

2.1.5 Peg transfer execution in a semi-submerged blood setup

To simulate a surgical scene for the peg transfer task, the setup is modified to be monochrome (red). The setup is partially submerged in artificial blood to simulate effects of blood during surgery. The artificial blood adds turbulence, scattering and reflections which the vision system has to deal with in a real surgical setup. Thus, the peg transfer setup is modified to simulate all these effects which are found in a real surgery. The modified peg-transfer setup is shown in Fig. 8 below.

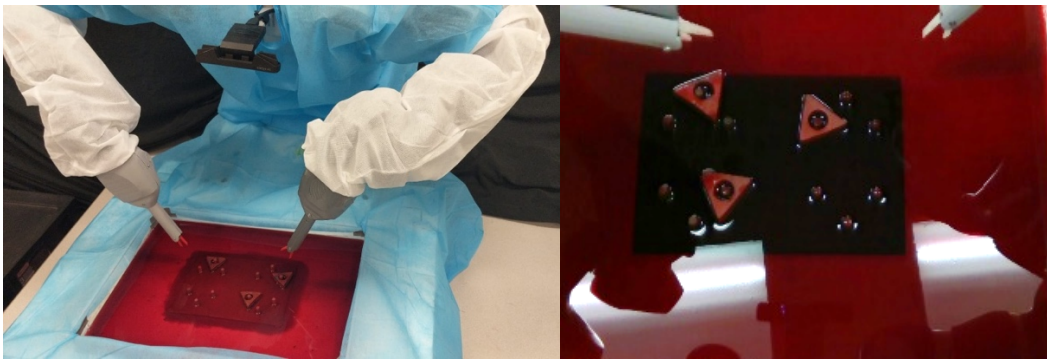

Figure 8. The system with the blood peg transfer setup and the view from the robot camera.

A deep neural network model is trained to detect all the components in the scene (pegs and poles). The model detects the object bounding boxes from the RGB image. The detections from the models is shown in Figure 9. The model used is the Yolo-v3 model. The Darknet (YoloV3) network which consists of 53 convolutional layers that take raw image frames as input and outputs object bounding boxes.

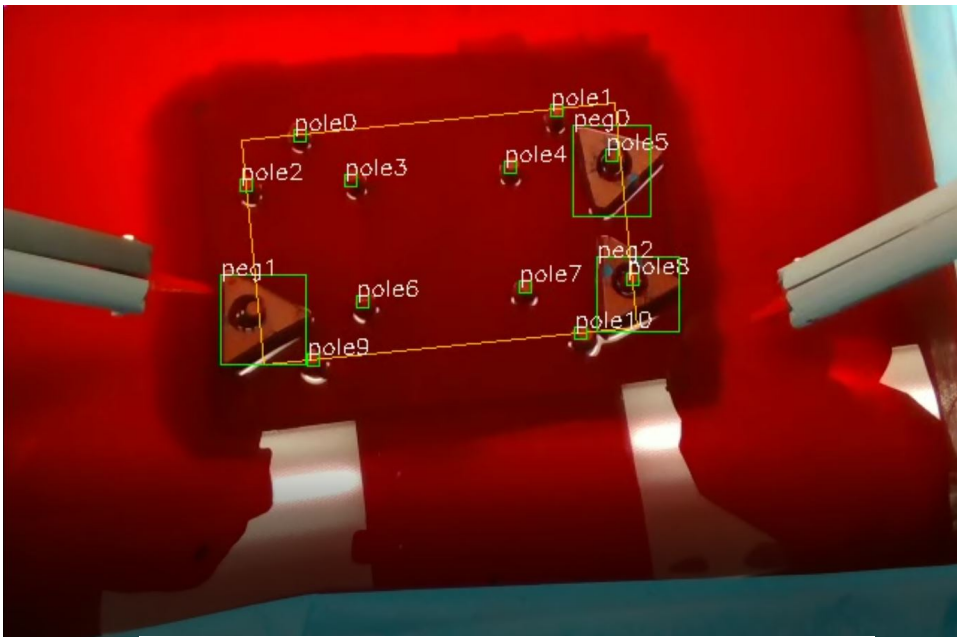

Figure 9. The detections from the yolov3 model.

The data for training is modified by flipping a few images and adding variations in illumination and saturation. Adding these variations reduced failed detections due to reflections. As multiple objects of the same class (multiple pegs, poles) exist, an object tracker is needed to uniquely identify the objects. An object tracker is created which employs centroid tracking and a Kalman filter to keep track of the objects. Filters and position history buffer are added to the tracker to account for noise, occlusions, and failed detections.

Together with the detection and tracking model, Mask-RCNN were used to perform instance segmentation of the pegs. The Mask-RCNN segments only the pixels belonging to the pegs. The segmented image is used to identify points for grasping the objects. The instance segmentation from Mask-RCNN is shown in Figure 10.

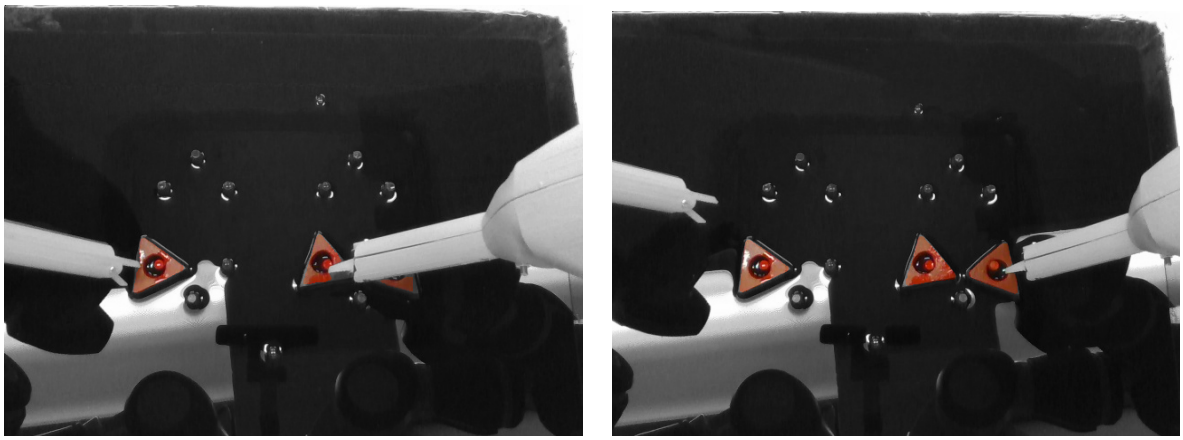

Figure 10. The instance segmentation of Mask-RCNN, showing the identified peg regions.

The liquid causes scattering and reflections which affects the depth perception of the camera. To eliminate the errors in depth the depth values outlier rejection and median filtering was performed on the bounding boxes to identify the depth value for each object (pegs and poles).

Vision Accuracy:

The previous section described the vision system for the blood peg transfer setup. The vision system has two components: a detection-tracker and an instance segmentation network. The YoloV3 network was trained with a pretrained Imagenet weights using 100 images, which were manually annotated. The model is tested using frames sampled from a video of robot interacting with the setup. The number of test images were 50,000 and the annotations for the test images are generated using a tracker and manually verified. The object detection accuracy is 97.5% with a false positive rate of 0.01%.

For the instance segmentation network, the same training images were used with a validation set of 100 more images. The training loss and validation loss are shown in Figure 6.

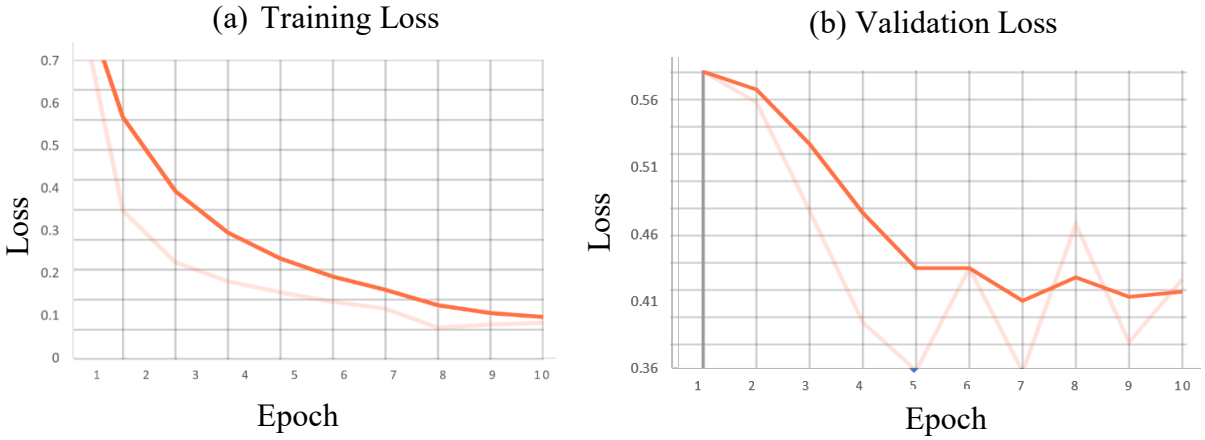

Figure 11. Training and validation losses for the Mask-RCNN model.

T2.2 Visual recognition and adaptive learning – Improved transfer Learning Using FFT features on the expanded dataset

2.2.1 Autonomous Surgeme execution using Reinforcement Learning:

We use reinforcement learning in simulation to create a framework for autonomous surgeme learning. The goal is to learn the surgeme execution in simulation from the DESK dataset and then transfer it to the real robot. As a first step, we use reinforcement learning (RL) with modelled rewards to provide a baseline method for execution learning.

The training is carried out in two phases, first in simulation, followed by training on the physical robot. This ensures faster training times as the simulation setup can be parallelized and sped up. Then using domain transfer methods, the training can be extended on the real robot to account for dynamics of the real world which are not captured in simulation.

Two simulation environments have been developed, one for the ABB Yumi Robot with the surgical grippers and one for the Taurus robot. Both the environments are developed in VREP simulator for the peg transfer task. The objects and peg board are modelled after the real ones so that the dynamics is similar to the real setup. The following section describes the algorithm used, followed by the environment setup, and finally the results from the simulator.

Proximal policy optimization (PPO) was finalized as the method for RL based on empirical comparisons between different policy gradient methods. The policy is updated based on data sampled from the current policy. Here, policy refers to the rule or function used by the agent (robot) to decide what actions to take, and it is either deterministic or stochastic. Let $\pi_\theta(a|s)$ denote a policy, it represents the probability of taking the action a given a state s, with parameters θ. Our objective is to find the policy to execute the surgeme trajectory τ that maximizes the expected reward.

$$\max_{\theta} J(\theta) = \max_{\theta} E[\sum_{t=0}^{T} R(s_t, a_t); \pi_\theta]$$

where $R(s_t, a_t)$ is the reward function for a given state s_t and action taken a_t.
This can be solved by standard approach of gradient ascent/descent by updating parameters using the rule,

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$$

Computing the objective gradient $\nabla_\theta J(\theta)$ is tricky, PPO imposes constraints on the optimization to ensure a viable step size for updating the policy.
A general form of the objective gradient is

$$\nabla_\theta J(\theta) = E\left[\sum_{t=0}^{T} \nabla_\theta \, log \, \pi_\theta(a_t|s_t)\phi_t\right]$$

Here ϕ_t is the reward for the trajectory τ i.e.,

$$\phi_t = \sum_{t=0}^{T} R(s_t, a_t)$$

It can also be shown that using the advantage function $A(s,a)$ instead of reward for ϕ_t also results in the same expected value of the policy gradient.
Let $V(s)$ be the state-value function (measures expected return of a state) and $Q(s,a)$ be the action-value function (similar to $V(s)$ measures expected return for a given state and action pair). Now,

$$V^{\pi}(s) = E(R(\tau)|s_0 = s)$$
$$Q^{\pi}(s,a) = E(R(\tau)|s_0 = s, a_0 = a)$$

The advantage function is defined as $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$. It describes how much better it is to take a specific action a in state s over a random action according to policy $\pi(.|s)$. Now the objective or loss is integration of the policy gradient i.e.

$$L = \int \nabla_{\theta} J(\theta) = E[log\ \pi_{\theta}(a_t|s_t)A_t]$$

PPO simplifies Trust region policy optimization (TRPO) by eliminating the constraint and the divergence term in the objective. It relies on specialized clipping to prevent the updated policy from moving too far from the old policy. PPO reduces the optimization problem to,

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}}\, E[L(s,a,\theta_k,\theta)]$$
$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)}A^{\pi}, g(\epsilon, A^{\pi})\right)$$

where ϵ is a small hyperparameter which controls how far away the new policy can go from the old one and,

$$g(\epsilon, A) = \begin{cases}(1+\epsilon)A & A \geq 0\\ (1-\epsilon)A & A < 0\end{cases}$$

The setup in the simulation is modeled after the real setup used for creating the DESK dataset. There is a robot, a peg board with 12 pegs (6 on left and 6 on right) and three pegs in the environment. The setup for the ABB Yumi robot and the Taurus is shown in Figure 12 left and right respectively.

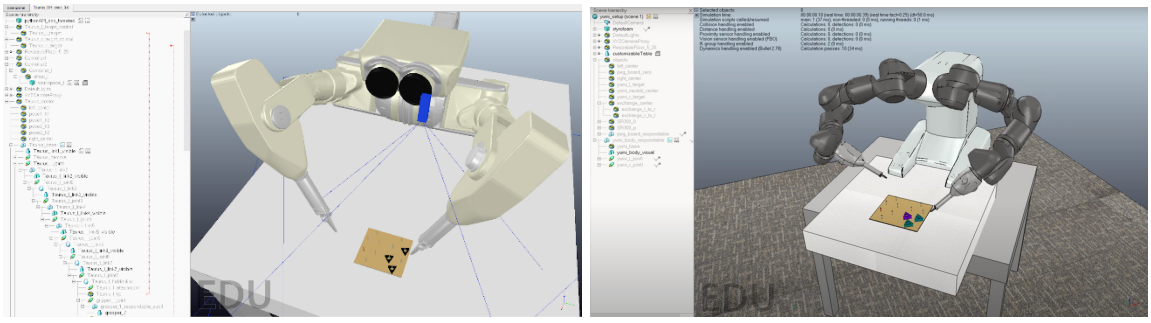

Figure 12. Simulation environments adapted for RL

The RL setup for the surgemes are as follows,

- States – The state is a 104 dimensional vector which includes 3D poses (x,y,z, roll, pitch, yaw) of the 3 objects, the 12 pegs on the board, tooltip, the 2 gripper poses and 2 gripper force values.

- Actions – The action space is defined based on joint angle changes. An action for a given arm is set of 7 joint offsets and a gripper angle. Each joint angle can be offset a maximum of 0.1 radian per timestep (tunable) based on maximum velocity settings for the robot.
- The objects are randomly initialized on 3 of the 6 pegs either on right or left. The arm on the side where the objects are initialized is considered the dominant arm.
- Reward shaping:
- The following is the reward modelled for approach object/ approach peg:

$$reward = -2|0.01 - \|A - (G_d)\|_2|$$

Where A is the position in robot space of the tool interacting with the object and G_d is the position of the desired goal object/peg in the robot frame. The offset of 0.01 is to prevent robot tooltip from colliding with the goal. Additionally, a condition on episode continuation is added to ensure the episode ends when the arm reaches the goal object/peg. The gripper state is not changed to ensure generalizability across approach object and approach peg. As seen the max reward possible is 0, and the robot is penalized harder the farther it is from the goal.

The algorithm used was PPO2 from openai/baselines [5] with a reward threshold of 0, max joint offset of 0.1 per timestep, with network as mlp, number of timesteps as 2e7 parallelized over 6 processes. The goal for each episode was randomly chosen from one of the three objects for the *approach object* surgeme or one of the 6 pegs on the same side as the hand for the *approach peg* surgeme.

Figures 13 and 14 show the reward for the approach object/peg surgeme.

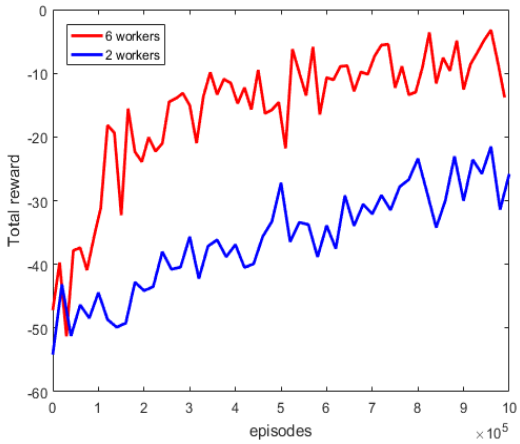

Figure 13. Reward per epoch for approaching the triangular objects in the pegboard.

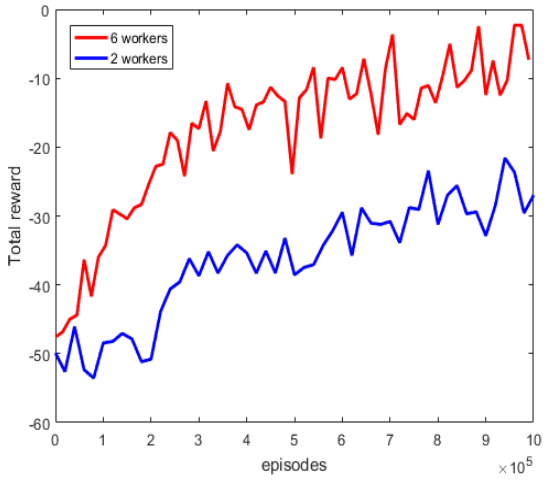

Figure 14. Reward per epoch for approaching the pegs.

2.2.2 Automatic low level surgical control Using reinforcement learning:

To develop low level control system using reinforcement learning (RL) the following phases were implemented: 1) Data collection with the da Vinci surgical simulator 2) Implementation of RL in the da Vinci simulator and 3) Testing RL on three tasks: reaching and suturing

First, we collected data using the dVSS (da Vinci Surgical Simulator) on 'Around the clock needle passing'. This was repeated 7 to 10 times. We had 15 users, 7 of which were senior residents. In total, we have 112 total trials, all providing 50hz console and needle kinematics, as well as mono video at 15fps. This data is useful for enhancing other stages of learning.

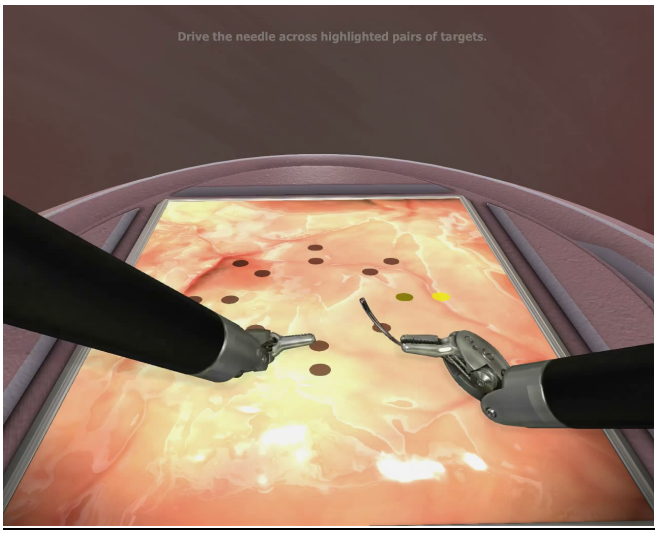

Figure 15. the da Vinci Simulator

Simulation Environment:
In order to support remote surgical control, we also focused on low-level control of robotic motion, specifically on the da Vinci Surgical Simulator (See Figure 15). The simulator is used for teaching and was adapted by us for data collection purposes. We further proceeded to convert the simulator into a fully working reinforcement learning environment.

To allow for rapid reinforcement learning, we needed to heavily modify the code of the simulator to enable off-screen OpenGL rendering. We also matched the general OpenAI Gym API, consisting of the following general instruction set:

```
state, reward, done = env.step(action)
state = env.reset()
```

The reset() function in particular presents a challenge, as there is no natural start point for the tasks we want to learn. This required adding careful state loading functionality to the simulator.

Needle Drive Task:
We focused on the task of a needle drive: moving the needle to the correct location as marked on the image, and then driving the needle through the tissue.

To prototype ideas and algorithms, we need a responsive environment that can be learned quickly yet is sufficiently close to the dVSS. For this, we use the Needle Master environment (See Figure 11).

This environment serves as a 2D approximation of the dVSS environment (See Figure 16). Like the dVSS, this environment, too, is mostly deterministic. The goal is for the needle to reach the next gate directly. Training a Needle Master environment, even a complex one, takes at most a couple of hours, or a few million steps, even from visual data, as shown in Figure 17.

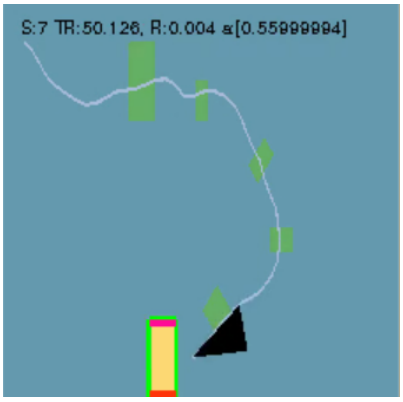

Figure 16. The Needle Master environment. This particular environment setup requires thorough exploration after reaching the 4th gate to find the next one in line. It's easier than a completely random setup though (which we also use).

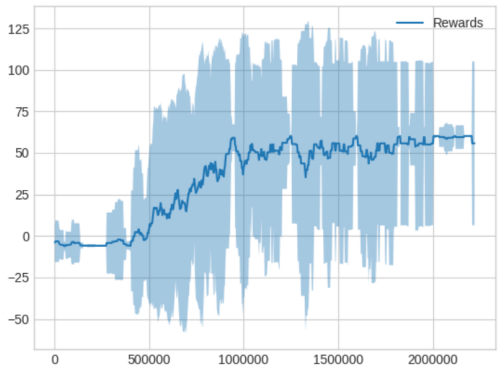

Figure 17: Reward graph for the Needle Master environment.

One important use for the Needle Master environment, was to allow us to test and tweak the batch learning replay mechanism. Only with the Needle Master environment could we iterate fast enough to try different approaches to batch learning.

Suture Task

This task involves fixed ingress and egress points. The needle starts right outside the ingress point, and the agent must learn to insert the needle and then proceed to manipulate the needle to reach the egress point (see Figure 18).

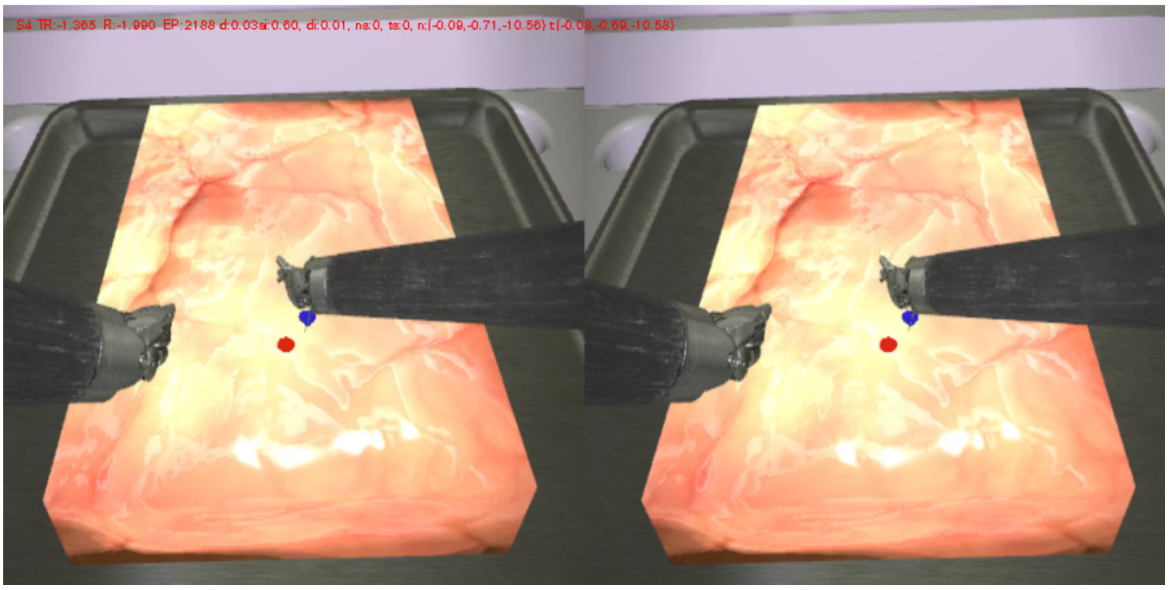

Figure 18: The Suture task as seen by the RL algorithm

The task consists of 3 distinct phases, as shown in Figure 19:

- *Phase 0*: the needle is outside the tissue.
- *Phase 1*: the needle has entered the ingress point and is inside the tissue.
- *Phase 2*: the needle has gone through both the ingress and egress points. We consider reaching phase 2 to be a 'success' for this task.

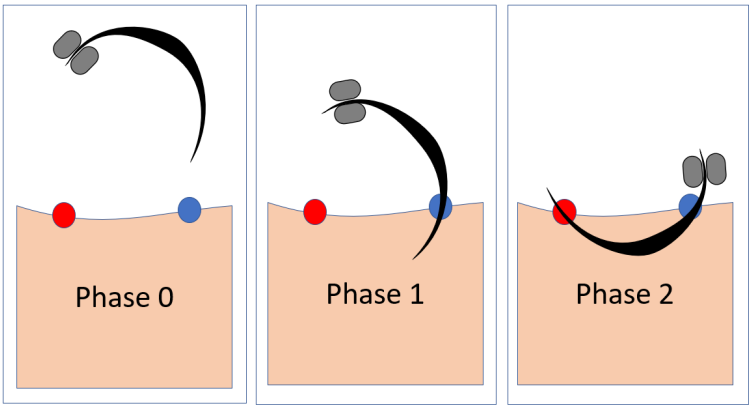

Figure 19. Suture task phases

This action space consists of full translational movement, in addition the axis of rotation needed for successful completion of the task. For Phase 0, the reward consists purely of distance to the target. A successful transition to the next phase gives the agent a reward, whereas any regression ends the episode with a penalty. The reward for Phase 1 is a weighted sum of delta distance from the needle point to the egress point and delta distance of the needle end to the ingress point.

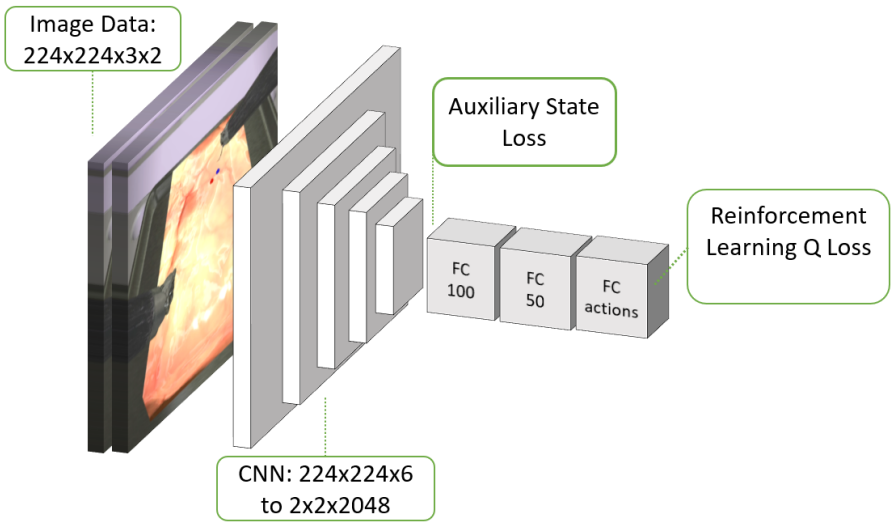

Figure 20. Deep convolutional network structure to estimate the Q-function loss.

To handle image data as input, we use a CNN followed by 3 fully connected layers, as shown in Figure 20. The CNN is supervised with an auxiliary state loss based on the state of the simulation, consisting of target and needle coordinates, whereas the fully connected part is supervised using the DDQN RL loss. We find that without the auxiliary state loss, the network refuses to learn from images.

Finally, we trained and ran on 6 parallel agents since the suture task is more demanding on the CPU. Three agents recorded, and 3 agents played back old data. The network learns by around 1.5M steps, and then tends to regress. Figure 21 shows we see the reward graph and the success graph. The success was quantified for two different states: state 1 is the needle reaching phase 1 (i.e. penetration via the ingress point) whereas state 2 is the needle reaching phase 2 (exiting through the egress point). We see that reaching phase 1 is far easier for the algorithm to learn, whereas phase 2 is achieved around 1.5M steps (the number of successes is averaged over 100,000 steps, causing the total percentage to appear lower.

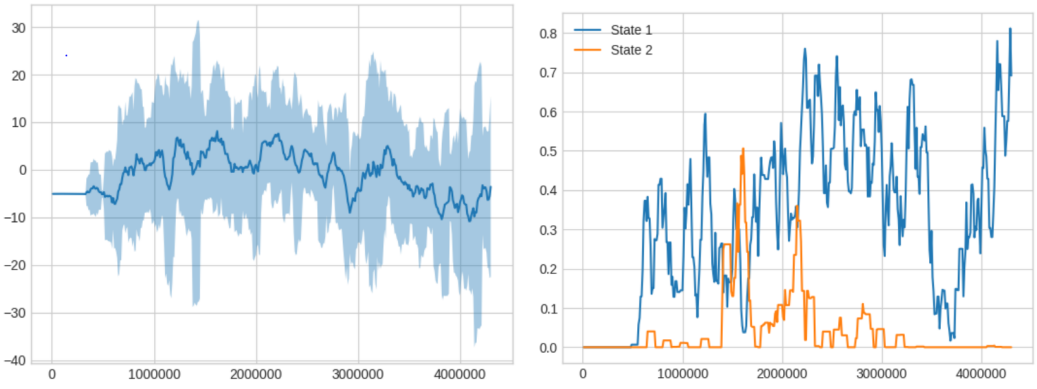

Figure 21. Left- Reward (y axis) per Epoch (y axis) for the suture task. Right – Success (x axis) per Epoch (y axis) for the suture task.

Reach Task

For the Reach task, the ingress point (blue) and egress point (red) are placed randomly on the tissue, with a fixed distance between ingress and egress (see figure 8). The right PSM, holding the needle, must approach the ingress point correctly, driven entirely by the agent

The action space of the agent consists only of translational movements. We use a discrete action space for improved sample efficiency. Each movement corresponds to roughly 1mm of motion. The reward for the agent is driven primarily by the delta distance from the target, with a heavy penalty for dropping the needle or bumping into the tissue to discourage such behavior.

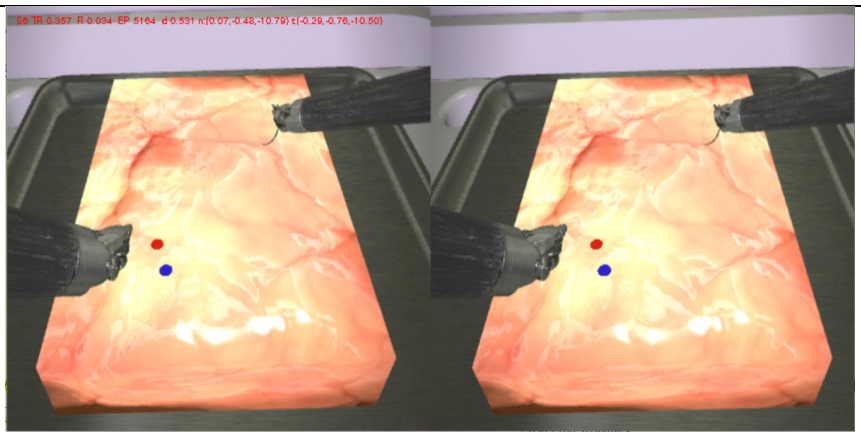

Figure 22: The Reach task as seen by the RL agent. We use stereo learning to disambiguate images that look identical in mono mode.

The network architecture developed for the suture task (see figure 22) was also used for the reach task. We trained and ran on 8 parallel agents for, with 7 agents running using the simulator, and 1 running using a playback environment for acceleration.

Figure 23 shows the reward and success data for the reach task. Both graphs show averages over 20,000 steps, with the input modalities including state, single images, stereo images, and image + depth. We define success for this task as reaching within 3mm of the target. We note that by 1.7 million steps, we reach a maximal reward and success on the best modality, which surprisingly happens to be single images. We theorize that this is due to the fact that critical features in the images, such as the needle, are only around 12 pixels, which causes a paucity of relevant data in the stereo/depth images.

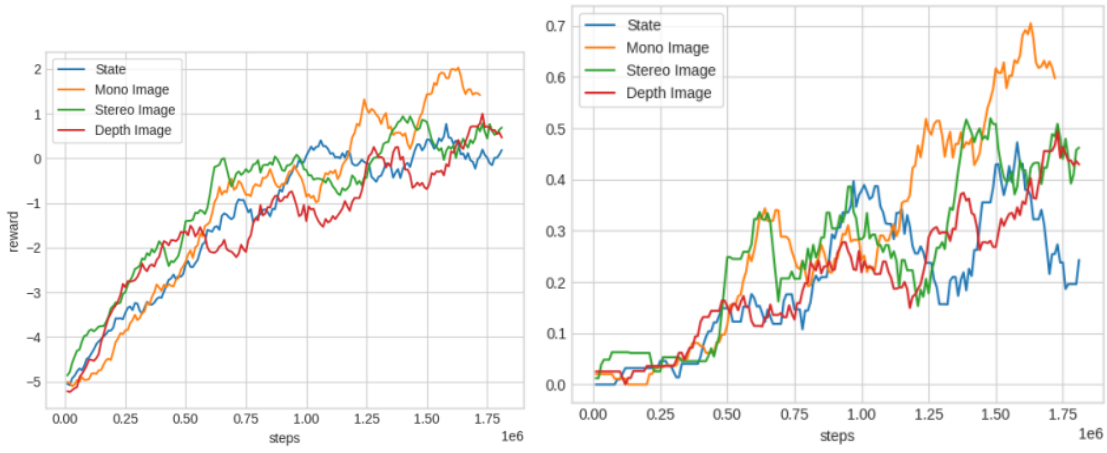

Figure 23. Left- Reward (y axis) per Epoch (y axis) for the reach task. Right - Success (y axis) per Epoch (x axis) for the reach task.

Hybrid-Batch System

Running on the simulator presented us with serious issues related to running-time. Each iteration of the experiment, whether used for results or for hyperparameter sweep, takes several days. We decided to approach the problem using the playback environments mentioned above. All transitions are recorded to long-term storage, and episodes are sampled randomly when playing back in playback environments, allowing for a mix of playback and live data. We term this a hybrid-batch system, shown in figure 24.

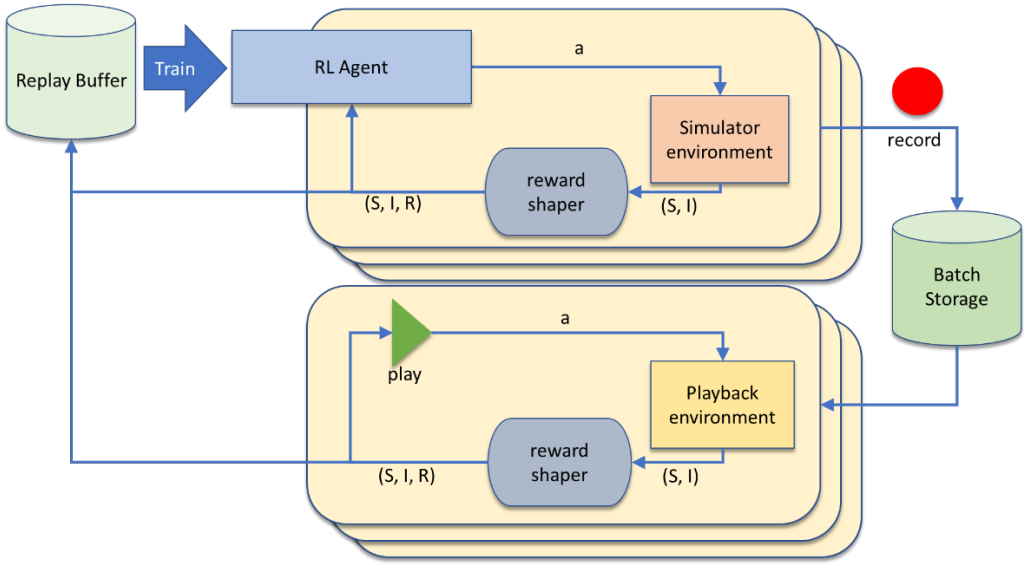

Figure 24: Hybrid-batch system used for simultaneous off-policy and on-policy learning.

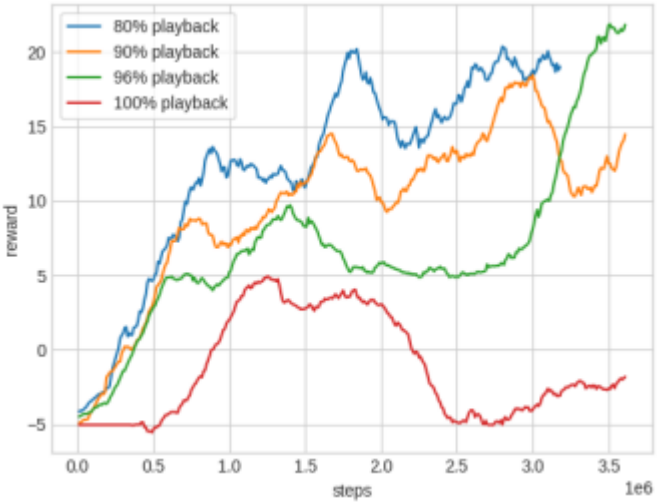

Figure 25: Comparison of performance of different playback ratios

Table 8: Comparison of time to max reward for different playback ratios

% playback	Time to max reward(hrs)
100	2
96	18.8
90	32.6
80	42

Table 8 a shows the performance of different playback ratios to live data. With 100% playback data, the algorithm fails to converge well: the data is too biased to reflect what's happening as we explore the environment. However, time-wise, we reach our (low-quality) maximum very fast, as playback environments are extremely efficient. At 80% playback, it takes 42 hours to reach maximum reward, and we do really well in terms of performance. 96% playback, however, takes only 18.8 hours, and yet eventually achieves superior performance to the 80% playback method (See Figure 25 for a detailed plot).

Needlemaster 3D

The real-time nature of the da Vinci simulator environment (dVSS-RL), together with the gap in performance between the dVSS-RL demonstrated to us the need to create an environment that is like Needlemaster i.e. it can be accelerated greatly, but one that is also three dimensional, so it is more challenging than the two-dimensional Needlemaster, and therefore closer to the dVSS-RL. Work on this environment is ongoing.

2.2.3 Live surgeme recognition:

The task is to recognize surgeme while the operator performs the surgery in a simulator. In our Vrep simulation environment, we tested our recognition model on Yumi Robotic data for peg transfer task. Our model consists of an LSTM network and do not assume on surgeme segmentation (start and end of the surgeme). We trained our model using a whole peg transfer trial and predict the label for every frame. In the context of our complete pipeline, this predicted label will be sent to the remote robot for execution at every timestep. Preliminarily, we conduct experiments on the artificially generated dataset on Yumi simulator for peg transfer task. We collected annotated surgeme by modeling the peg transfer trial automatically using simulator scripts (details in Section 2.2.1). This process serves as an alternative to human data collection and annotation. We tested our model on 100 peg transfer trial where we trained our model on 50 trials and tested on the 50 trials.

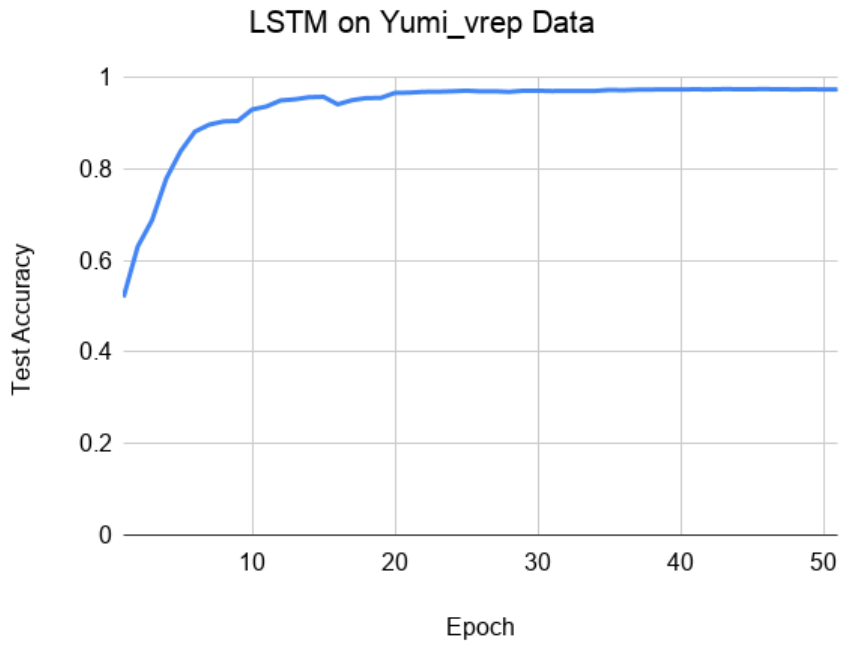

Figure 26. Results of surgeme recognition on the artificially simulated Yumi kinematics data.

We leveraged kinematics information for each frame and predicted framewise surgeme label. Figure 6 shows that the test accuracy increases as we iteratively train for more epoch. Our model converges to an accuracy of 97% after 20 epochs (see Figure 26). In the future, we will evaluate our recognition model on the human collected and manually annotated peg transfer dataset.

2.2.4 Information Reliability and information Requirement:

We designed a module that validates the reliability of the information that is received on the remote robot side. We verified the integrity of the information under two settings: 1) Information Reliability, and 2) Information requirement.

To assess the reliability of the data at the remote site, we measured the similarity between the data received in the past and the data received at present. Intuitively, the goal is to evaluate the level of surprise that the new information triggers. Thus, we would raise a reliability issue warning when the similarity difference is greater than a predefined threshold δ_i.

For this implementation, we leveraged the KL-Divergence between the old data distribution ($Q(x)$) and a new data distribution ($P(x)$), using the following equation.

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)}$$

We defined the information requirement metric, as the prediction confidence of the surgeme recognition. Our LSTM-based recognition module generates a confidence vector, where each

element is the prediction confidence of a class. This confidence vector is predicted at every frame at the operator side and sent to the remote robot. When the confidence reaches a threshold δ_r, the robot can execute the surgeme. This process would allow the remote execution to start when the operator has only performed a fraction of the step.

Experiments

We conducted a simulation experiment to validate the proposed methods. For this experiment, we considered the surgeme "approach". The environment consists of three target locations (towers). Given a starting position and goal location (from three targets), the task is to reach the goal. For simplicity we modeled the actions in the simulated robotic agent as follows: at each time step the agent will go in a straight line towards the goal with added Gaussian noise (mean=0, standard deviation=0.5).

The LSTM model used the gripper trajectory as an input and predicted the target tower (there are three target towers). The added noise makes the trajectory challenging for the recognition model. Figure 27 shows a sample trajectory of the simulated agent with Gaussian noise.

In this experiment, we consider the predicted probability from the LSTM-based recognition module as the information being sent to the remote side. We measured the amount of surprise of in the new information given the history of past information. Figure 28 shows that in general, the KL-Divergence varies little which is the indication of the consistent information.

At the 97th frame, we intentionally swapped two prediction values in the confidence vector and sent them to the remote robot. This causes a spike near to 100th frame (see Figure 28). The results show that KL-Divergence can be potentially used as a metric for reliability. We will explore this possibility in detail with various adversary scenarios.

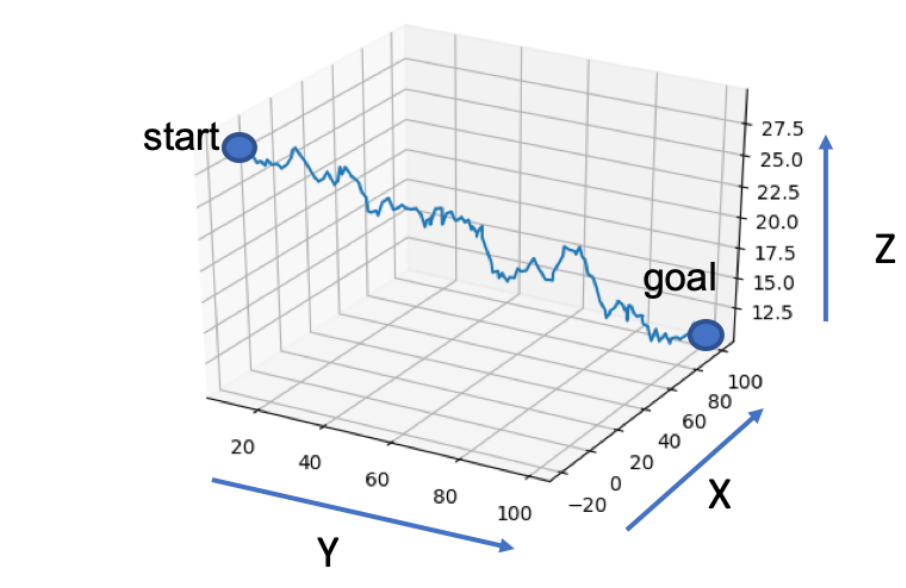

Figure 27: Sample trajectory for simulated agent with gaussian noise

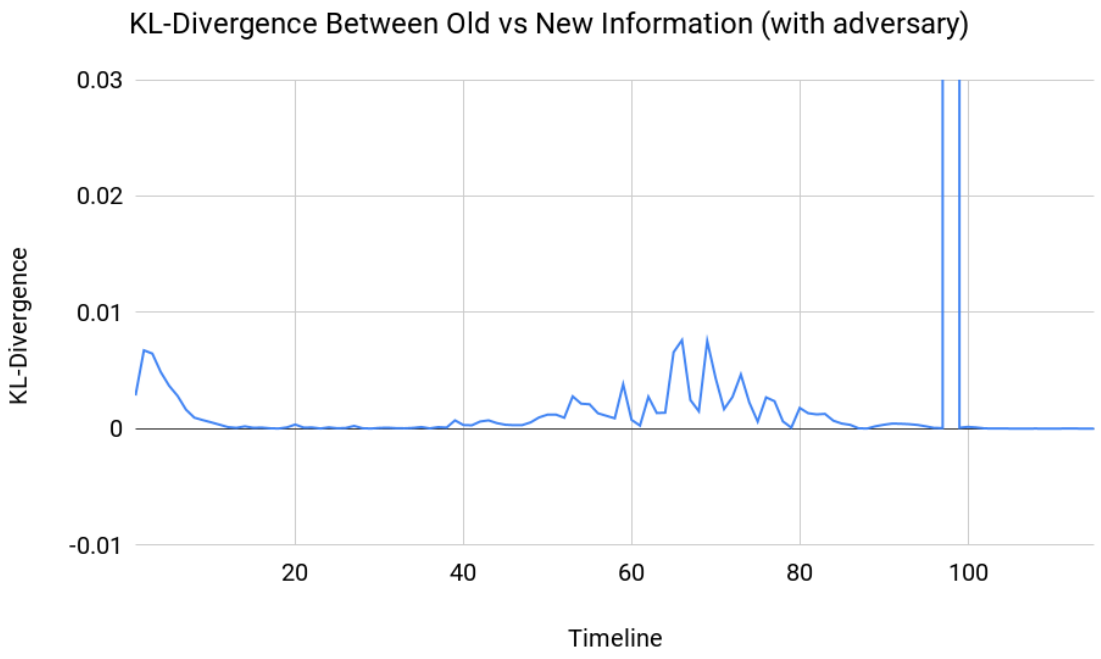

Figure 28: Variation of KL-Divergence over time

We performed a second experiment, were we assessed the amount of information that is needed to start executing a surgeme. In this setup, we used the LSTM prediction probability as the confidence score. Figure 29 shows that the surgeme target can be accurately recognized after only seeing 80 frames. This result shows promise for using the confidence vector as a metric of required information for surgeme execution. When the confidence is high (greater than a predefined threshold δ_r), the remote robot can start the execution of the corresponding

surgeme.

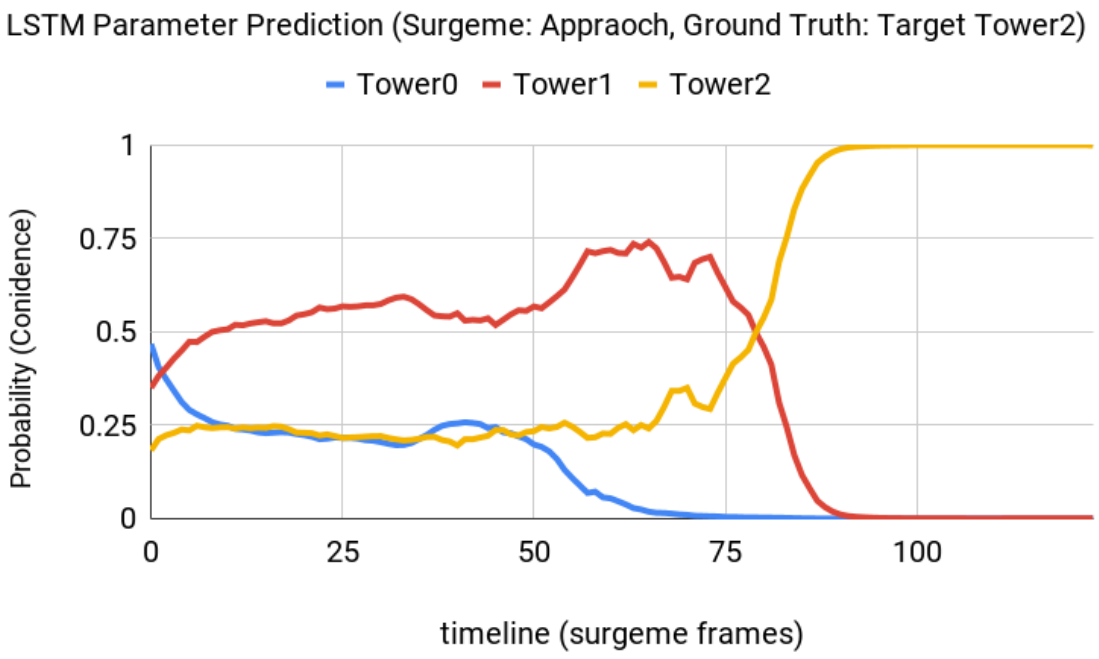

Figure 29: Variation of confidence over time

2.2.5 Surgeme recognition with the feedback system:

We conducted two experiments on Yumi simulation data for live surgeme recognition. In the first setup, we collected the data with the simulator component running exclusively. In the second setup, we collected the data while the simulator was incorporated with the execution through a communication network (full loop). This setup, shown in Figure 30, impacts the simulator movement by the natural delay generated from the communication network, execution module, and the feedback. In the first setup, three subjects collected a total of 36 trials (12 each), and in the second setup, two subjects collected a total of 24 trials (12 each). We annotated all the data for recognition module training. In both setups, we used an LSTM based recognition module where 80% of data was used to train the model, and 20% of data was used for testing. The first setup achieved a testing accuracy of 74% while the second setup achieved a testing accuracy of 85%.

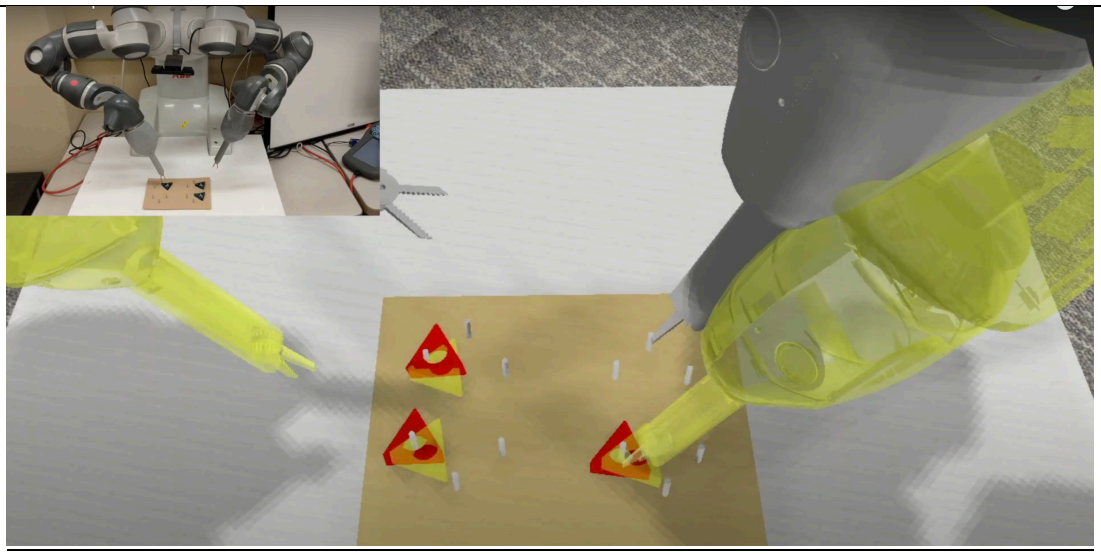

Figure 30: Feedback system

2.2.6 Surgeme target prediction:

We note that the prediction from the surgeme prediction using a LSTM based architecture creates artificial delays in execution. The delays arise from the fact that the parameters associated with surgeme are only identified when the surgeme is completed in the simulator, defeating the purpose of the early prediction module. To overcome this issue, we attempted two different approaches.

1. Target label prediction using LSTM: Inspired by the success of the surgeme prediction module, we attempted to train another LSTM based classifier to learn the target labels for the parameters associated with the surgemes. However, we noticed that the performance of the neural network is not good. The reason we associated as the large class imbalance. From the 7 surgemes defined for our framework, only 2 of them require a parameter namely, surgeme 1 and surgeme 6 to identify which target object to approach. For the peg and pole task in DESK dataset, surgeme 1 has 3 possible pegs to approach, and surgeme 6 has 12 possible pegs to approach. For the remaining surgemes, the module was trained to predict a NULL class. This results in the total number of classes to be 16. Because of the large class imbalance, the neural network overfits to predict NULL class every time as visible in Figure 31.

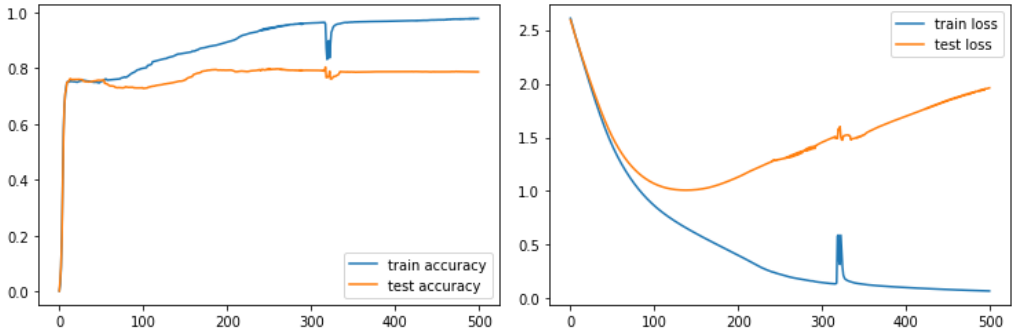

Figure 31: Accuracy and loss v/s training epochs

2. Target position prediction using LSTM: We note that the large class imbalance issue could be treated by converting to an analog domain where we predict where the robot will end up after certain time steps. This approached proved to be useful where we trained the LSTM neural network to learn the target positions instead of the target labels. We first remove the noise from the input by putting an exponential filtering block on the input data. This helps in increasing the learnability of data. From the experiments conducted, this approach proves to better than the previous approach. The results are provided in Figure 32.

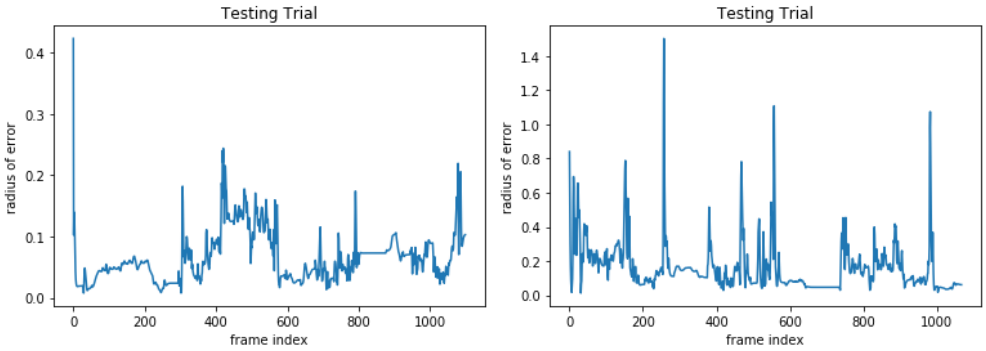

Figure 32: Left subfigure shows radius of error (in meters) for LSTM with an exponential filtering block which as about 4 times lesser error compared to the right subfigure which shows radius of error(in m) for LSTM without exponential filtering.

2.2.7 Cricothyroidotomy task:

Obstruction of the airway is a preventable cause of death, especially on the battlefield. Restoring the airway often necessitates surgical intervention. Due to high risk of grave collateral damage, specially trained personnel only can administer such lifesaving care. Cricothyroidotomy is one such surgical procedure which is used to re-establish the airway, when all other methods such as Laryngeal mask airway and laryngeal intubation fail. Cricothyroidotomy involves making incision on the cricothyroid membrane to construct an alternative air passage and then inserting a tube to restore airway. Due to risks associated with tracheal damage, Esophageal injury, and injury to neurovascular structures, this procedure remains a high-risk undertaking. Hence, the need for expert surgical skills is imminent for this procedure. Robots, with precise force control, can assist and fulfill the need for a surgeon, especially in the austere and hostile conditions of a battlefield for such procedures.

The surgeme-based encoding of surgical actions provides a mode of high-level communication to teleoperate the robot. This provides an important extension of the current work into contact-intensive procedures that are germane to remote medical and specifically surgical procedures. Increasing levels of contact complexity in remote surgical procedures have traditionally required doctors to adapt their skills to substitute vision for touch. In effect, these are environments austere in sensing. This increasingly allows us to investigate environments that are not only austere in communication, but austere in sensing, as well.

The difference between peg-transfer task, which is simulated laparoscopy, and cricothyroidotomy is mode of sensing. In cricothyroidotomy the surgeon uses palpation to understand the profile of the trachea and identify the location for incision. Thus, the robot is required to use force information in addition to vision to perform the task. Extending the proposed FORWARD framework of agile and robust surgical trauma management to such time-critical emergency procedures, we aim to develop a fully automated system capable of working under such hostile conditions. New surgemes parametrized by force feedback will be developed for this undertaking. The surgemes involved in the cricothyroidotomy tasks are as follows,

(a) Trachea registration (palpation):

We are prototyping the development of the palpation surgeme with tactile inspection using an external, high-fidelity SynTouch tactile probe. Palpation is a hybrid surface exploration which requires both kinematic and force control modes. Palpation is performed through tactile inspection using a Syntouch force sensor. The goal of palpation is feature discovery, i.e. to identify the anatomy. Specifically, to identify and register thyroid cartilage, cricoid cartilage, and 2-3 tracheal rings.

(b) Skin Incision:

The second surgeme is to perform an incision on the skin at the registered location (between 1st and 3rd tracheal ring). The surgeme has bounds on force and depth. The goal of surgeme is to perform a 2.5 cm vertical incision on skin at the registered location.

(c) Expose cricothyroid (using tool):

The goal of this surgeme is to employ a surgical retractor to expose the cricothyroid membrane. The robot will use self-retaining retractor to hold the edges of the incised skin and spread it until the cricothyroid membrane is visible.

(d) Cricothyroid membrane Incision:

This surgeme is similar to the skin incision. The surgeme is to perform a horizontal incision of roughly 5mm on the membrane.

The methods used for the palpation, skin and membrane incision is deep reinforcement learning (DRL) models.

Setup and Dataset creation:

To develop a DRL model for execution of the surgemes the robot can be trained using a shaped reward. But, without an initial model, a random initialization will result in large number of timesteps for learning the surgeme. To reduce the timesteps required for learning the surgeme, data from demonstrations or teleoperated robot data can be used. In this regard, we developed a hybrid force controller to perform the palpation surgeme with known position information.

A torque controller is used in the contact direction, the model for this is as follows,

$$\tau = J(\theta_t)^T F_d + K_1 \left(f(x^d, s*) + K_2 \dot{f}(x^d, s*) \right)$$

Here τ is the desired joint torque, $J(\theta)$ is the jacobian, f is the feedback function, K_1, K_2 are gains for the feedback errors.

A similar position controller is used to move along the trachea,

$$x_{\{t+1\}} = x_t + k_1(f(x^d, s*) + k_2\dot{f}(x^d, s*))$$

Here x_t is the position at time t, x^d is the desired position, f is the feedback errors and k_1, k_2 are the feedback gains.

The setup for the cricothyroidotomy is shown in Figure 33.

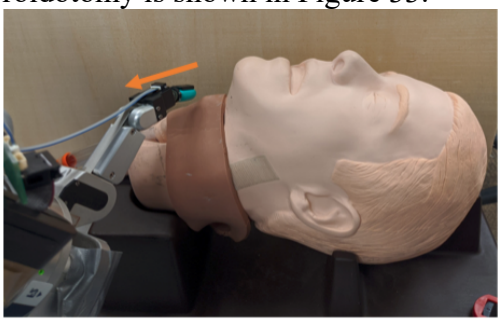

Figure 33: The robot performing palpation surgeme with high-fidelity Syntouch probe.

Different thickness skins for the neck was developed for the creation of the dataset. The thickness of the skin simulates the variation in fat content in humans. The various skin models are shown in Figure 34.

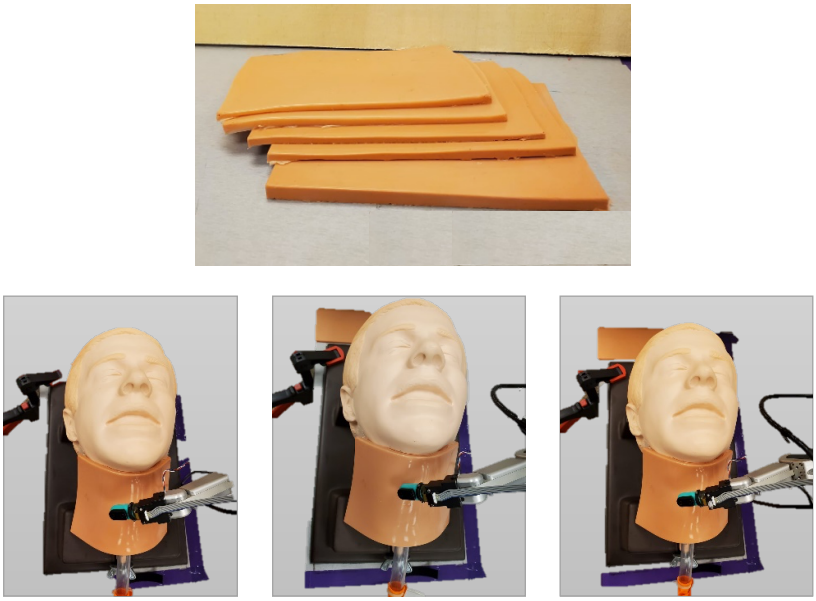

Figure 34: Skins of different thickness for robot to learn a generalized feature registration.

The SynTouch force sensor is housed on a Barrett Hand which is mounted on Rethink Sawyer arm. The variations in force profile is used to identify the different tracheal rings. The variations in force is shown in Figure 35.

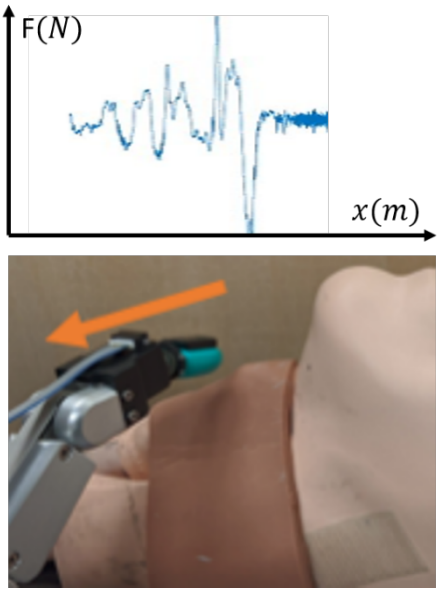

Figure 35: The force feedback during the palpation surgeme.

The data collection is conducted for 8 different skin thickness values. For each skin model the data is collected for 25 trails. A total of 200 trails are recorded for palpation surgeme. The data collected will be used to learn sparse rewards for the DRL model for the surgeme. The baseline hybrid controller developed will be used as the initialization policy for RL. Future work will include training DRL models for palpation and the incision using the data collected. The surgeme execution will be integrated with the full semi-autonomous teleoperation framework.

T3.1 Motor command generation in constrained and new settings

Our framework relies on surgemes or surgical maneuvers as the fundamental action units that a robot can execute autonomously to complete a surgical procedure. Surgemes are used to describe surgical tasks at a higher level of abstraction while requiring a reduced amount of information exchange between the operator and the robot. We build on the concept of surgeme recognition and generation to achieve a semi-autonomous behavior that can perform basic surgical tasks with minimal manual intervention under low bandwidth scenarios. The framework includes the following modules: 1) Surgeme recognition, 2) Action prediction and 3) Surgeme execution and 4) Live feedback. The subsections below describe the development of these semiautonomous framework.

3.1.1 Live Surgeme recognition:

In this task, we trained an LSTM-based recognition model on the Yumi simulated data collected from the human users. This trained model is used to perform recognition of surgeme while the operator performs the surgery in a simulator. In this experiment, do not assume on surgeme segmentation (start and end of the surgeme). We trained our model using a whole

peg transfer trial and predict the label for every frame. In the context of our complete pipeline, this predicted label is sent to the remote robot for execution at every timestep. We experimented on a 110 peg transfer trial collected from 3 human users. We used 80% trial for training and 20% for testing. We leveraged kinematics information for each frame and predicted framewise surgeme label. The recognition model achieves a test accuracy of 74%. In the future, we will incorporate visual data (video image) from simulator with the kinematic information and build a multi-modal live recognition system.

3.1.2 Live Surgeme feedback:

Initially, we calibrate and arrange the simulator environment (Triangles, Poles, Pegboard) based on the remote side environment when the connection is established for the first time. Through a feedback system, the remote robot environment is shown in the alpha-blended layout along with the simulator robot. Alpha blended objects are updated at regular timestep when the feedback information is received at the operator side. The minimal information is sent from the remote side. These are the remote robot gripper tip, and objects (i.e., triangle, peg) positions. From this information, we reconstruct the movement of the remote robot in simulation using inverse kinematics. We update the position of the objects by mapping the real robot and simulator coordinate system (done during calibration).

3.1.3 Live system control:

To improve the robustness and speed of the live simulator control, we implemented an interface using the Razer Hydra sensing™ controller. This interface increased the controller's refresh rate from 15Hz(using the HTC VIVE™) to 60Hz (using the Razer Hydra™). The previous system required a pre-processing step in unity, where the 3D pose of the HTC VIVE™ was computed and sent to the simulator. This step produced delays in the overall pipeline. In contrast, the new implementation obtains the position and orientation of the controller directly through Razer Hydra's driver and the pose of the robot is directly computed in the simulator. In addition, the previous system required external motion sensors, causing interruptions when the control was occluded (ie. The user put the controllers under the table to rest their arms). The new interface does not possess these vulnerabilities.

During teleoperation, the translation of the robot's end effector corresponds to the translation of the controllers in the real world, while the changes in orientation for each gripper are based on the controller's coordinate frame. The relationship between the Razer Hydra™ reference frame and the robot gripper reference frame is depicted in Figure 36.

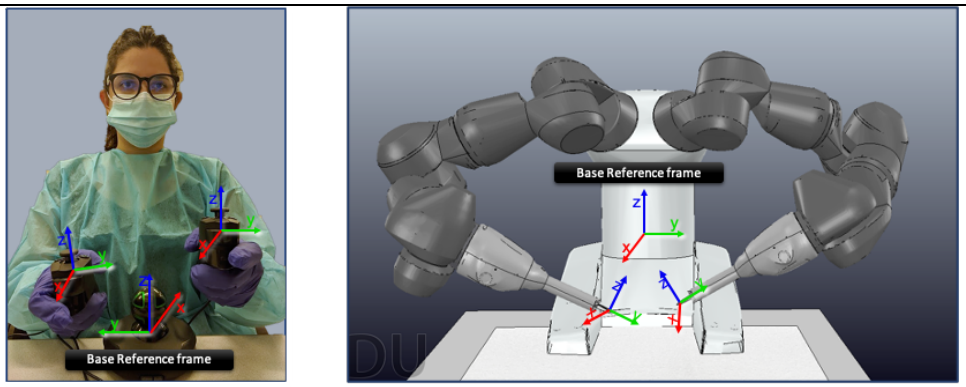

Figure 36. Operator control reference frame (left) vs robot control reference frame (right)

Finally, a pedal system was added to control the robot motion. The robot follows the motion of the controllers only when the pedal is pressed, allowing the user to reconfigure the hand to a comfortable place when the pedal is released (See Figure 37).

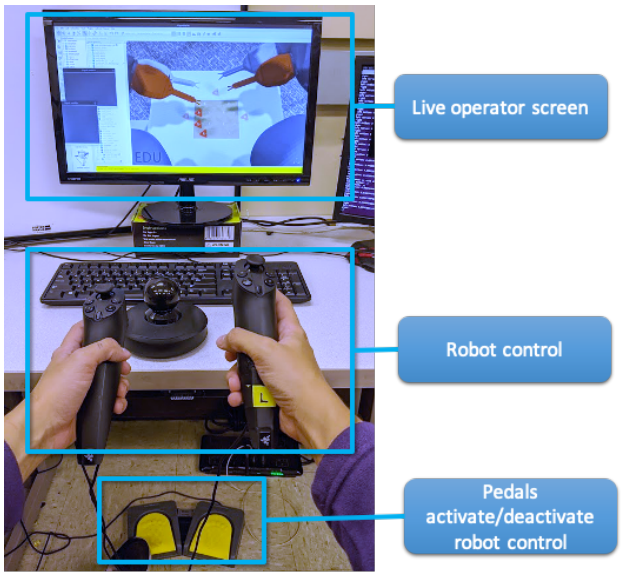

Figure 37. Interface for the live simulator control

Task 3.2 Communication Framework and experimental design

We developed a communication tool for sending messages between the robot and the simulator and created an experimental design based on the network statistics obtained from the framework.

3.2.1 Design Requirements

1. *Robust under change of IP addresses*: Since we consider the robot will operate in an austere setting, it might have its IP changed due to a reestablishment of connection.

2. *Ensuring the identified surgeme reaches remote the robot*: If the connection between the robot and the simulator is broken, the robot should get the entire sequence of surgemes sent by the simulator after the connection is reestablished.
3. *Ensuring that latest feedback from the robot reaches the simulator*: If the connection between the robot and the simulator is broken, after connection reestablishment, the robot should send the latest available feedback to the simulator to ensure continuous operation.

3.2.2 Design details

To mitigate the Impact of dynamic IP addresses in the client, we choose a server-client model, where both the robot and the simulator act as clients and clients communicate with each other via the server. We made a server run on a machine with a fixed IP address.

To ensure that the surgeme is always received by the robot from the simulator, we used TCP connections between the clients and the server. Moreover, to ensure that the messages are not lost if the connection is not available, the clients and the server maintain a pending message queue guaranteeing that the messages are sent in correct order of generation.

Finally, to ensure that the simulator always receives live information from the remote robot, we used a UDP connection for the feedback model. If a message is lost during the communication, the clients will not waste time and resources to ensure that the lost message is sent again. Instead it will move on to the next message.

3.2.3 Implementation details:

We implemented two servers: one for the surgeme messages sent from the simulator to the robot, and one for the feedback messages sent from the robot to the simulator. Each server allows only one simulator and one robot connection to ensure security. Once the robot and the simulator are connected, no other connection is allowed. Also, the servers maintain a queue of pending messages. First, every received message received is appended to the queue. Then, if the message is sent successfully, the message popped off the surgeme server queue. For the feedback server, the message is always popped, as the UDP connection does not send an acknowledgement when the message is received by the client.

For our client implementation, we used an object-oriented design. After an object of the communicator class is created, the methods can be used for sending and receiving the messages from the server.

The object creates a separate thread, which constantly communicates with the server to fetch any newly available messages. If a message is received, it is stored in the object's message queue. Whenever the get_message() method of the object is called, it returns the message from the queue.

To send packets, a send_mesasge() method is provided. The method first pushes the message to a pending message queue. The thread created then peeks the queue and sends any available message. If the message is successfully sent, the message is popped from the pending message queue.

Each message is appended with a 16 byte string storing the UNIX timestamp of the message creation. Whenever a message is received by the server, this header can be decoded to fetch the timestamp and compute the delays.

3.2.4 Improving Teleoperability of Taurus Simulator

The control of the Taurus robot has been changed from direct teleoperation to an intuitive teleoperation system. The new system controls the position of the robot based on the displacement of the controllers in the user's environment (see Figure 38). In contrast, the orientation of the robot grippers is changed based on each controller change in orientation (see Figure 39).

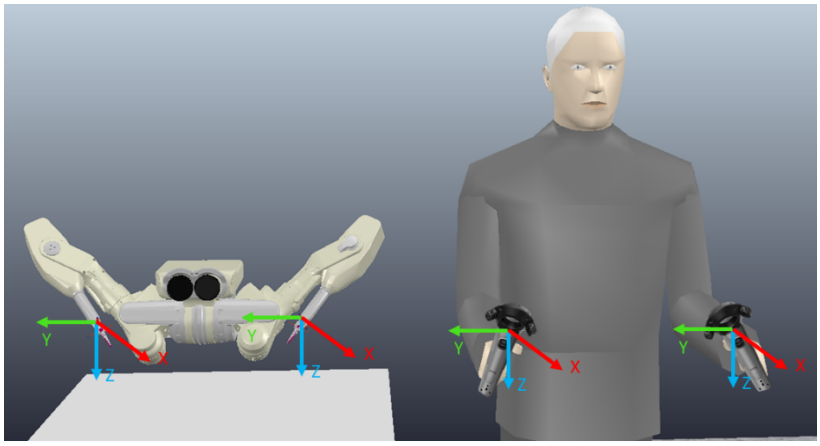

Figure 38. Coordinate spaced for position control (aligned in the environement).

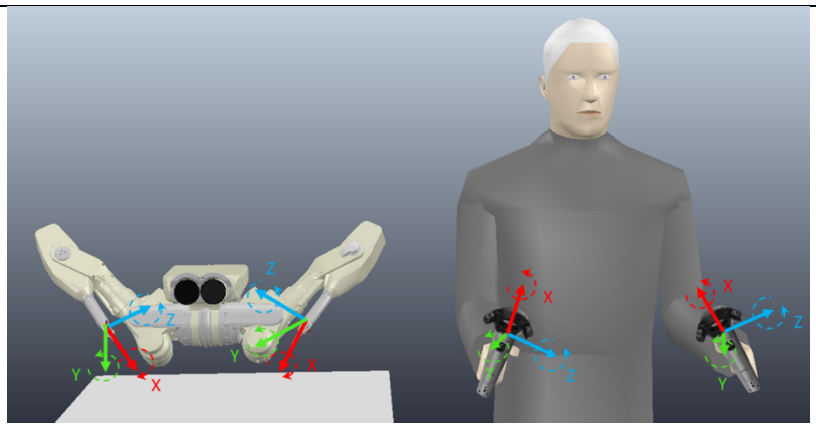

Figure 39. Coordinate frame for orientation control (with respect to each object gripper/controller).

3.2.5 Live data transmission experimental design:

We evaluated the impact of local processing times in teleoperations. For the full loop with live video feedback, we note that sending raw images from cameras is highly inefficient. A color image from a camera is a tensor of size 640x480x3 bytes for 8 bit encoding. At 30 fps, the camera generates a stream of about ~25 MBps. Hence, we chose to compress the images before sending them over the network. We specifically choose JPEG compression on each image of the video instead of compressing blocks of images. This allows us to mitigate the impact of random packet losses where only a few frames could be lost instead of an entire block.

We studied the impact of the frame rate of the video streaming on the latency. We kept the client terminal and the server 4 hops apart over a network. The client compresses the image tensor into a vector of variable length using JPEG compression of Q value 25. To push the test framework to the limit, we send a full HD color video stream of size 1920x1080 pixels. Additionally, we also sent a string of the maximum length observed in the experiments to identify the latency of the network.

Table 9: Statistics for latency when sending strings and compressed images 20 and 30 fps.

	String (in seconds)	20 fps(in seconds)	30 fps(in seconds)
mean	0.040	0.105	0.135
std	0.031	0.028	0.057
min	0.0012	0.056	0.050
max	0.224	0.263	0.430

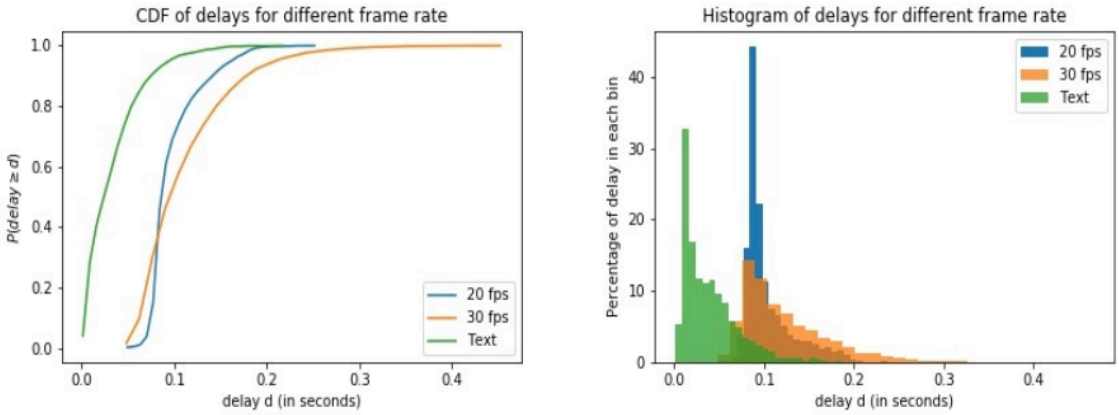

Figure 39: CDF and Histogram for various transmission schemes for video feedback. Local processing to compress and decompress the images using JPEG compression consumes a non-trivial amount of time. Additionally the 30 fps frame rate has fat tail.

We provide the results of the experiments in Table 9 and Figure 39. From Figure 1, we note that the delay distribution for sending image vectors is shifted to the right. This is because we include the processing times in the latency.

Based on the findings, we concluded that the impact of computation time for compressing and decompressing the video feed is important. Further, to make a latency sensitive system such as ours, we need to work with faster compression algorithms.

3.2.6 Live execution:

We implement the proposed strategy using 3 different systems. One system hosts the simulator, second systems hosts the robot interface and the 3 system works as the server. Note that adding the third system as server allows us to work with portable systems with changing IP address. We now describe the working of the robot and the simulator system in details.

The robot system starts three services, 1) vision service that identifies and tracks objects, 2) communication service that constantly sends the state of the robot and object id and locations to the simulator, and 3) execution platform which on connects with the surgeme server and executes the command received from the simulator.

The simulator system has two sub systems, 1) a display system which reproduces the state of the robot and the environment after receiving the feedback from the robot and 2) a recognition module which identifies the surgeme and the parameter associated with the surgeme and sends it to the surgeme network to pass it to the robot for execution .

Based on our experiments, we observed the following latency and jitter values (See Table 10 and Figure 40) for the feedback communication system with UDP protocol and the surgeme communication system with TCP protocol:

Table 10. Latency and jitter for different communication schemes

	Feedback communication system(UDP)	Surgeme communication system(TCP)

Latency(in ms)	1.67	31.62
Jitter (in ms)	1.54	25.28

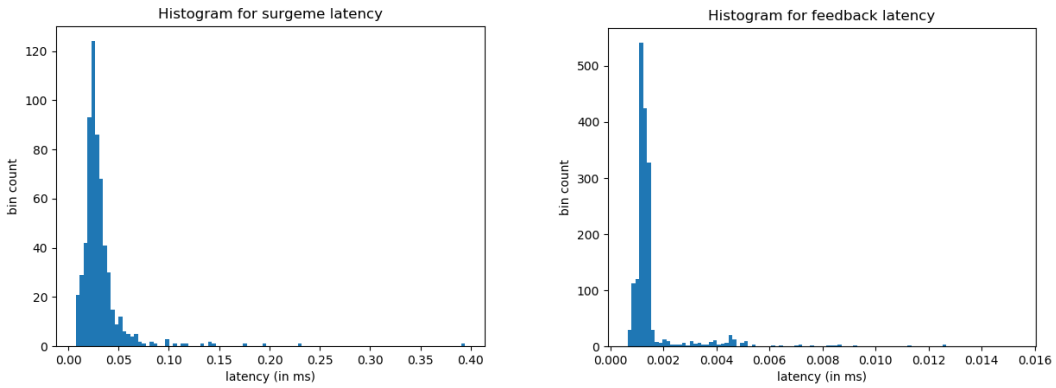

Figure 40: Histogram figures for the latency values measured over the live communication network. Surgeme communication uses a TCP protocol and has an additional overhead of the handshakes. Feedback communication uses UDP framework and does not uses any handshake mechanism and thus enjoys a lower latency by allowing packet drop.

References

1. *Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.*
2. *Qiao, S., Shen, D., Wang, X., Han, N., & Zhu, W. (2014). A self-adaptive parameter selection trajectory prediction approach via hidden Markov models. IEEE Transactions on Intelligent Transportation Systems, 16(1), 284-296.*
3. *K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.*

4. *F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.*
5. *Gruber, Marvin HJ. "Statistical digital signal processing and modeling." (1997): 335-336. Hayes MH. Statistical digital signal processing and modeling. New York: John Wiley & Sons; 1996.*
6. *Herff, Christian, and Dean J. Krusienski. "Extracting Features from Time Series."*
7. *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.*
8. *Brockman, Greg, et al. "Openai gym." arXiv preprint arXiv:1606.01540 (2016).*

What opportunities for training and professional development has the project provided?

Training:

The PI Wachs and co-PI Voyles, Aggarwal, Voyles and Xue are mentoring 4 graduate students: (1) through a research assistantship, computer science, polytechnic, and engineering students are working on all the problems related to computer vision, robotics and machine learning under the mentorship of co-PI Yexiang Xie; (2) through a research assistantship, industrial engineering PhD student Glebys Gonzales is working in the problems related to robot teaching and experimental design, under the mentorship of PI Wachs.

How were the results disseminated to communities of interest?

1. SARTRES: A Semi-Autonomous Robot TeleopeRation Environment for Surgery. Masudur Rahman, Mythra V. Balakuntala, Glebys Gonzalez, Mridul Agarwal, Upinder Kaur, Vishnunandan L. N. Venkates, Natalia Sanchez-Tamayo, Yexiang Xue, Richard M. Voyles, Vaneet Aggarwal and Juan Wachs. International Conference on Medical Image Computing & Computer Assisted Intervention (MICCAI) Joint AE-CAI workshop. 2020 *(accepted for publication).*
2. From the Dexterous Surgical Skill to the Battlefield-A Robotics Exploratory Study. Glebys Gonzalez, Upinder Kaur, Masudur Rahman, Vishnunandan Venkatesh, Natalia Sanchez, Gregory Hager, Yexiang Xue, Richard Voyles and Juan Wachs. Journal of Military Medicine (MILMED). 2020 *(accepted for publication).*
3. Blind Decision Making: Reinforcement Learning with Delayed Observations. Mridul Agarwal and Vaneet Aggarwal Journal of Pattern Recognition Letters (elsevier) *(submitted by: July 19 2020).*

What do you plan to do during the next reporting period to accomplish the goals?

Development of a Debridement model in Real robot and simulator system

A debridement procedure will be added to the skill library using the previously developed protocol of data collection. The task will be performed in the current system using a Taurus model. Thus, we will develop simulated and real debridement setups based on the design of Murali et. al[1]. This procedure will be divided and annotated using the following surgemes: 1) Approach debris 2) Align and grasp debris, 3) Lift attached tissue, 4) Cut 5) Lift removed tissue (see Figure 1).

The kinematics, RGB video and depth information will be obtained for each trial. The surgemes will be performed several times for all the damaged tissue, until the remaining is clear.

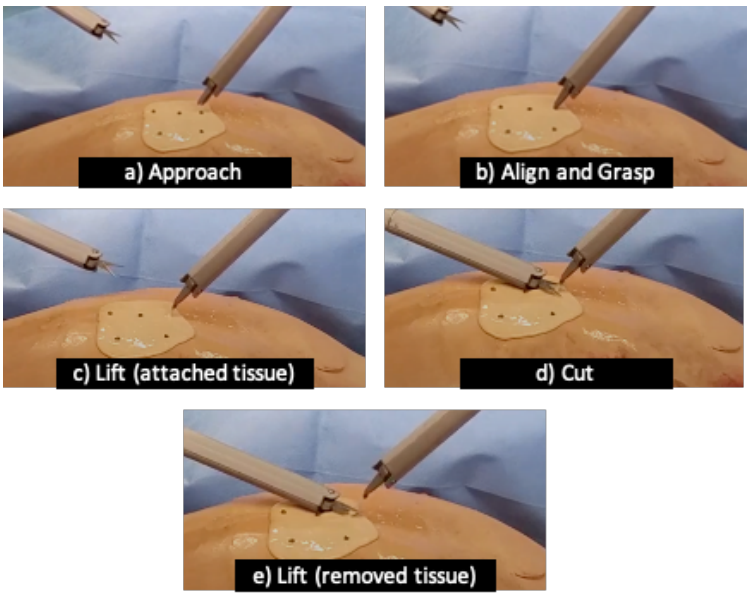

Fig 1. Debridement Model.

References

1. Kehoe, B., Kahn, G., Mahler, J., Kim, J., Lee, A., Lee, A., ... & Goldberg, K. (2014, May). Autonomous multilateral debridement with the raven surgical robot. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1432-1439). IEEE.

4. IMPACT:

What was the impact on the development of the principal discipline(s) of the project?

This work presents a surgeme library and a machine learning framework to transfer knowledge from different domains in surgical tasks. The dataset includes variability through random changes in the task setup, the orientation of the peg board, and by using different robotic setups with different operators. The dataset has observations from four different robots performing a peg transfer task (624 transfers in total): a simulated Taurus II, a real Taurus II, a YuMi robot and a dVRK surgical robot. The previously available datasets for surgical procedures used a single robotic system for data collection, when skill transfer is required for military robots which have not been used during surgical controlled data collection in the operating room. In the case of remote teleoperation, this constrain is exacerbated as the `deployable robot could significantly

differ in its kinematic chain configuration and operational space from the surgical robots in an operating room (i.e., da Vinci). Therefore, a surgical task dataset with a higher number of robots and intentionally embedded variability represents a valuable contribution to the research community. In addition, we presented a baseline framework for surgeme classification in a transfer learning scenario. The transfer model produced an accuracy of 93% and 97.5% for the da Vinci and Taurus II respectively in the extreme case of training with no real data. In addition, the results indicate that using a mix of simulated and real data in the training set can yield higher accuracies. Specifically, training with a ratio of real to simulated data of 22%-78%, YuMi accuracy was boosted from 63% to 81%. With a lower real to simulated ratio (15%- 85%), the da Vinci and Taurus II obtained accuracies of 95.4% and 99.7% respectively, as their kinematic data has less discrepancies with the simulated data than the YuMi platform. The results of this work show the potential of using simulated environments to generate data for real, distinct autonomous robots that will be deployed in a remote area.

What was the impact on other disciplines?

An initial formative experiment conducted during the telerobotic surgical skill training for the peg & pole task, provided some initial understanding about the role of high level and low level instructions affect performance. Improved understanding of the factors affecting design and use of explicit commands as well as the physical and cognitive requirements for this interaction will be crucial to introduce physical interaction with devices in teleoperation. We expect a significant breakthrough in this knowledge after our second experimental design, which consists of collecting the surgemes involved in debridement's tasks performed while interacting with the robot and a simulated tissue model. It is expected that the use of high-level interfaces and the surgeme learned through deep learning frameworks will increase the understanding about the different uses of autonomous control in the operating setting, with extensions to other high-risk/high-stakes scenarios.

What was the impact on technology transfer?

Nothing to report.

What was the impact on society beyond science and technology?

The new fundamental scientific theories and algorithms developed through the two years of funding through this FORwARD Award will have applications to the broad field of communications and machine learning, but specially will be suitable to the robotic surgical domain in the austere setting. At the fundamental level, our work will address challenges such as “what is the tradeoff between specificity and generality of data for maximum knowledge transfer between domains”, or “how to measure diversity of actions/objects among a dataset” (e.g. entropy would be a good starting point), “how to compress the information so the semantic content is not affected”, and “how to deal with imperfect and randomly delayed information to the robot”. Our research strategy will also suggest a potential plan to implement these principles into a semi-autonomous robot capable of performing common surgical tasks associated with the battlefield. The direct outcome of this will be an improvement in public knowledge about semi-autonomous robotics.

5. CHANGES/PROBLEMS:

Changes in approach and reasons for change

No changes

Actual or anticipated problems or delays and actions or plans to resolve them

We do not anticipate delays. (Covid-19 expected delays?)

Changes that had a significant impact on expenditures

No changes.

Significant changes in use or care of human subjects, vertebrate animals, biohazards, and/or select agents

Significant changes in use or care of human subjects

No changes.

Significant changes in use or care of vertebrate animals.

No changes

Significant changes in use of biohazards and/or select agents

No changes.

6. PRODUCTS:

- **Publications, conference papers, and presentations**
  Journal publications.

No changes.

Books or other non-periodical, one-time publications.

No changes.

Other publications, conference papers, and presentations.

1. 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.

2. 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.)

3. Natalia Sanchez-Tamayo, Glebys Gonzalez, Naveen Madapana, Molly O'Brien, Yexiang Xue, Richard Voyles, Vaneet Aggarwal, Gregory Hager, Andrew W Kirkpatrick, Steve Overholser, Juan Wachs. From the Desk (Dexterous Surgical Skill) to the Battlefield -A Robotics Exploratory Study. Presented in Poster Format at the Military Health System Research Symposium, Kissimmee, Florida, Aug 22, 2019. MHSRS-19-02537.

4. M. M. Rahman, N. Sanchez-Tamayo, G. Gonzalez, M. Agarwal, V. Aggarwal, R. M. Voyles, Y. Xue, and J. Wachs. DESK (Dexterous Surgical Skill) to the Battlefield - A Robotics Exploratory Study. Journal of Military Medicine (2020 MHSRS Supplement).

5. M. M. Rahman, N. Sanchez-Tamayo, G. Gonzalez, M. Agarwal, V. Aggarwal, R. M. Voyles, Y. Xue, and J. Wachs. (2020) SARTRES: A Semi-Autonomous Robot TeleopeRation Environment for Surgery. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization. (accepted).

- **Website(s) or other Internet site(s)**

Forward Website
Project dissemination website, summarizing the goals and accomplishments of the grant.

Desk Dataset:
Library of surgemes collected during a peg transfer procedure. The data was collected using four different robots: A da Vinci Research Kit, An SRI Taurus, a simulated SRI Taurus and an ABB YuMi robot.
Data collection Protocol:
Reference document that explains in detail the collection protocol for the Desk Dataset. The information includes the pegboard setup for every subject, direction of the transfer (left-to-right or right-to-left) and the camera configuration with respect to the robot.

Development Repository:
This repository contains all the surgeme execution development. The repository is private. (Contact gonza337@purdue.edu to get access).

- **Technologies or techniques**

As a result of our research, we have developed and implemented an annotation the DESK (Dexterous Surgical Skill) dataset. It comprises a set of surgical robotic skills collected during a surgical training task using four robotic platforms: the da Vinci Robot, the Taurus II robot, Taurus II simulated robot, and the YuMi robot. This dataset was used to test the idea of transferring knowledge across different domains (for a surgical gesture classification task with seven gestures.

- **Inventions, patent applications, and/or licenses**

We intend to fill a patent with the prototype of the FORWARD system that we developed. This process has not started yet.

- **Other Products**

Databases, videos, raw images and recording of the FORWARD sessions (3) are located at the PURR repository.

7. PARTICIPANTS & OTHER COLLABORATING ORGANIZATIONS

What individuals have worked on the project?

Name:	*Juan P Wachs*
Project Role:	*Principal Investigator*
Researcher Identifier (e.g. ORCID ID): 0000-0002-6425-5745	
Nearest person month worked:	*1.12 month*
Contribution to Project:	*Supervising the overall performance of the project. Coordinated visits to IUSM. Working with Maria Eugenia in all the aspects of gesture recognition and one shot learning. Working with Aditya Shanghavi for the design of the large interaction table. Helping with the journal publication.*
Name:	*Richard Voyles*
Project Role:	*Co-Investigator*
Researcher Identifier (e.g. ORCID ID):	
Nearest person month worked:	*1.12 month*
Contribution to Project:	*Actively participated in and advised research assistant Mythra Balakuntala in the research and development of the first prototype of the execution by Reinforcement Learning; in designing, conducting, and analyzing the results of user studies aimed at assessing the*

	robotic system; in disseminating the project results in a journal paper.
Name:	*Vaneet Aggarwal*
Project Role:	*Co-Investigator*
Researcher Identifier (e.g. ORCID ID):	
Nearest person month worked:	*1.12 month*
Contribution to Project:	*Supervised and actively participated in research for theoretical limits for performance of reinforcement learning algorithms under delays. Guided development of experiments for performance of learning algorithms under delays. Supervised research for algorithms for detecting and predicting surgemes.*
Name:	*Yexiang Xue*
Project Role:	*Co-Investigator*
Researcher Identifier (e.g. ORCID ID):	
Nearest person month worked:	
Contribution to Project:	*Actively participated in and advised research assistant Md Masudur Rahman and others in the research and development of the recognition module and the first prototype of the overall pipeline; in designing, conducting, and analyzing the results of the recognition module and the overall pipeline; in disseminating the project results in conference papers.*
Name:	*Gregory Hager*
Project Role:	*Co-Investigator*
Researcher Identifier (e.g. ORCID ID):	
Nearest person month worked:	
Contribution to Project:	*Actively participated in and advised research assistant Yotam Barnoy in the development of automatic execution of surgical tasks in the da Vinci simulatpr*
Name:	*Glebys Gonzalez*
Project Role:	*Research Assistant*
Researcher Identifier (e.g. ORCID ID):	*0000-0002-0181-006*
Nearest person month worked:	*6 months*
Contribution to Project:	*Coordinating the development of the surgeme recognition and execution with all the*

	Research Assistants. Actively participated in the development of the automatic execution system of the remote robot and the feedback simulated system. Actively contributed in the data collection of the Taurus Simulator and development of scene semantic methods with Vision.
Name:	*Natalia Sanchez*
Project Role:	*Research Assistant*
Researcher Identifier (e.g. ORCID ID):	
Nearest person month worked:	*3 months*
Contribution to Project:	*Developed the initial architecture of the feedback simulated system.*
Name:	*Md Masudur Rahman*
Project Role:	*Research Assistant*
Researcher Identifier (e.g. ORCID ID):	*0000-0002-3633-0621*
Nearest person month worked:	*6 months*
Contribution to Project:	*Developed the learning algorithms for surgeme classification on the live. the simulated system. In addition Masud Rahman actively participated in the development of the feedback system*
Name:	*Mythra V. Balakuntala*
Project Role:	*Research Assistant*
Researcher Identifier (e.g. ORCID ID):	0000-0003-2551-4780
Nearest person month worked:	*6 months*
Contribution to Project:	*Formulated the methods for the perception module of the robot. Designed and developed the object detection, tracking and instance segmentation models for scene understanding. Developed the blood peg transfer setup and the modifications to the perception system for the same.Formulated the theoretical aspects of the Reinforcement learning models for learning the surgemes. Developed the simulated environment to deploy RL algorithms to learn the surgemes and trained surgemes using different DRL models. Designed and developed the cricothyroidotomy setup. Responsible for creation of the dataset and surgeme modelling for the cricothyroidotomy task. Setup the feedback from the robot to the simulator to*

update the simulator. Contributed to the data collection for peg-transfer data collected in the Yumi simulator.
Name: *Vishnunandan Venkatesh* *Project Role:* *Research Assistant* *Researcher Identifier (e.g. ORCID ID):* 0000-0003-2551-4780 *Nearest person month worked:* *6 months* *Contribution to Project:* *Developed the model based surgeme execution unit which integrates information from the teleoperated robot and the visual scene information to execute the required surgeme with the physical robot. Developed the hydra controller interface which uses the hydra controllers to operate the robot in the simulator and perform the surgemes. Also worked in setting up the data collection for the same using the yiumi simulator and hydra controllers. Contributed to the data collection and annotation of the collected data.*
Name: *Mridul Agarwal* *Project Role:* *Research Assistant* *Researcher Identifier (e.g. ORCID ID)* *Nearest person month worked:* *6 months* *Contribution to Project: Responsible for the development of the communication sytem and the surgeme prediction with delays. Actively contributed in the data collection of the Taurus Simulator and development of scene semantic methods with Vision. Actively participated in publications in robotic conferences.*

Has there been a change in the active other support of the PD/PI(s) or senior/key personnel since the last reporting period?

No changes.

What other organizations were involved as partners?

Nothing to Report.

8. SPECIAL REPORTING REQUIREMENTS

COLLABORATIVE AWARDS:

QUAD CHARTS:

9. APPENDICES: