

AD _____

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, 315 N. Grant Street, West Lafayette, 47907, IN

REPORT DATE: Oct 2019

TYPE OF REPORT: Annual Report

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

DISTRIBUTION STATEMENT:

Approved for public release; distribution is unlimited.

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision unless so designated by other documentation.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE Oct 2019		2. REPORT TYPE Annual		3. DATES COVERED 15-Sep-2018 to 14-Sep-2019	
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 NOTES					
14. ABSTRACT This research focuses on the "Medical Robotics Research" topic area. Within this area we will address the following objectives: 1) mitigate the deleterious effects of signal latency on complex tele-robotic surgical tasks; 2) extend to full automation when deemed necessary; 3) develop semi-autonomous robotic assistant protocols; and 4) develop models for knowledge representation of semi-autonomous medical behaviors. When tele-operation is challenged due to limited bandwidth, latency and lost-of-signal, autonomy needs to step in. While modern robots have been endowed with excellent sensing and dexterous capabilities, there is a gap in knowledge about two critical aspects: (1) how to leverage operator's expertise under limited connectivity (rather than switching on/off autonomy); (2) how to "transfer" automatically existing abundant knowledge about surgical maneuvers from the operating room (OR) to new, uncontrolled and austere settings. In this research, we will study and theorize about new approaches to predict information to account for randomly delayed and imperfect information due to bandwidth limitations. This will allow us to maximize informational content and minimize redundancy, under constrained communication settings. Based on the predicted information, we will determine at what extent a robot needs to autonomously complete surgical procedures, making use of a sequence of maneuvers associated with the required surgical procedure extracted from a previously learned library. To populate this library, we will theorize new approaches for transfer learning, where learned patterns will be projected to fundamentally different domains with variable resource constraints. This first year the effort were allocated to the creation of a database of multiple robots performing a fundamentals of laparoscopic task "peg and pole transfer" which was used to create a library of maneuvers and apply principles of transfer learning to recognize such maneuvers.					
15. SUBJECT TERMS					
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		19b. TELEPHONE NUMBER (include area code)

STANDARD FORM (SF) 298: A sample SF 298 is provided at <https://mrmc.amedd.army.mil/rrpindex.asp>. The abstract shall be provided in Block 14 and shall state the purpose, scope, and major findings and be an up-to-date report of the progress in terms of results and significance. Abstracts will be submitted to the Defense Technical Information Center (DTIC) and shall not contain proprietary information. Subject terms are keywords that may have been previously assigned to the proposal abstract or are keywords that may be significant to the research.

Pages shall be numbered. The number of pages shall include all pages that have printed data (including the front cover, SF 298, table of contents, and all appendices). Page numbers must match the numbering shown on the Table of Contents.

Purpose: We will develop a theoretical framework for supervised autonomy, capable to self-adjust its autonomous behavior and perform procedures in never seen settings using a transfer learning paradigm.

Scope: Future battlefield medical operations in the Multi-Domain Battlespace calls for robotic platforms that are near-autonomous and can provide patient support systems integrated with perception engines for accurate detection and treatment of casualties in closed-loop enroute care systems. Such an integrated system should allow safe and effective-telerobotic surgery. When teleoperation is challenged due to limited bandwidth, latency and lost-of-signal, autonomy needs to step in. While new robots have been endowed with color and depth visual, acoustic and force sensing capabilities, together with new tooltips both rigid and flexible, there is a gap in knowledge about two critical aspects: (1) how to leverage operator's expertise under limited connectivity (rather than switching on/off autonomy); (2) how to apply the existing abundant knowledge about surgical motions and maneuvers from the OR to new, Scalably Austere Environments (SAE).

Major Findings: In this report, we describe a semi-autonomous robot teleoperation framework which significantly reduces the amount of transmitted data. The framework consists of two main modules: (i) Recognition Module - which recognizes primitive actions (i.e., surges) performed at the operator end, and (ii) Execution Module – which executes the surges at the robot end (e.g., approach peg 1 on pole 5) using task contextual information. We conducted several experiments on the DESK dataset and show the effectiveness of our framework using two metrics: data compression rate and success rate of surge execution. Overall, we obtained a data compression improvement of 99:79% and an average success rate of 98% for surge execution using a model-based approach. Furthermore, a success rate of 86:6% was obtained for the peg transfer task.

TABLE OF CONTENTS

	<u>Page No.</u>
1. Introduction	3
2. Keywords	3
3. Accomplishments	4
4. Impact	77
5. Changes/Problems	79
6. Products	80
7. Participants & Other Collaborating Organizations	82
8. Special Reporting Requirements	85
9. Appendices	87

1. **INTRODUCTION:** Narrative that briefly (one paragraph) describes the subject, purpose and scope of the research.

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:** Provide a brief list of keywords (limit to 20 words).

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

3. **ACCOMPLISHMENTS:** The PI is reminded that the recipient organization is required to obtain prior written approval from the awarding agency Grants Officer whenever there are significant changes in the project or its direction.

What were the major goals of the project?

List the major goals of the project as stated in the approved SOW. If the application listed milestones/target dates for important activities or phases of the project, identify these dates and show actual completion dates or the percentage of completion.

Specific Aim 1: Data Collection and Sensory-motor Library Generation

Data collection (15-Sept-2018 – 15-Dec-2018) 90%

Library Generation (15-Sept-2018 – 15-Dec-2018) 95%

Military provides feedback (15-Sept-2018 – 15-Dec-2018) 100%

Specific Aim 2: Semantics Rules Creation

Creation of terminal symbols (surgemes) (15-Apr-2019 – 15-July-2019) 90%

Development of grammar and rules (temporal models) (15-July-2019-15-Dec-2019) 10%

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 (15-Apr-2019 – 15-Sept-2019) 90%

Specific Aim 4: Integrating predictions with semi-supervision

Find the prediction error and competitive guarantees (15-July-2019 – 15-Jan-2020) 5%

Specific Aim 5: Visual recognition and adaptive learning of primitives

Train a deep neural network (DNN) with anatomical images (15-July-2019 – 15-Oct-2019) 85%

What was accomplished under these goals?

For this reporting period describe: 1) major activities; 2) specific objectives; 3) significant results or key outcomes, including major findings, developments, or conclusions (both positive and negative); and/or 4) other achievements. Include a discussion of stated goals not met. Description shall include pertinent data and graphs in sufficient detail to explain any significant results achieved. A succinct description of the methodology used shall be provided. As the project progresses to completion, the emphasis in reporting in this section should shift from reporting activities to reporting accomplishments.

Task 1: Data Collection and Sensorimotor Library Generation

The objective of the data collection was to generate a library of surgical actions for peg transfer and suture. Any procedure can be broken down into atomic surgical actions, and these actions can be used in a sequence to complete a procedure. The data was collected from three robotic platforms (see Figure 1): a) The Taurus robot from SRI, b) Simulated Taurus Robotic environment, c) the YuMi collaborative platform, adapted for the surgical environment and d) an augmented simulation of the da Vinci robot.

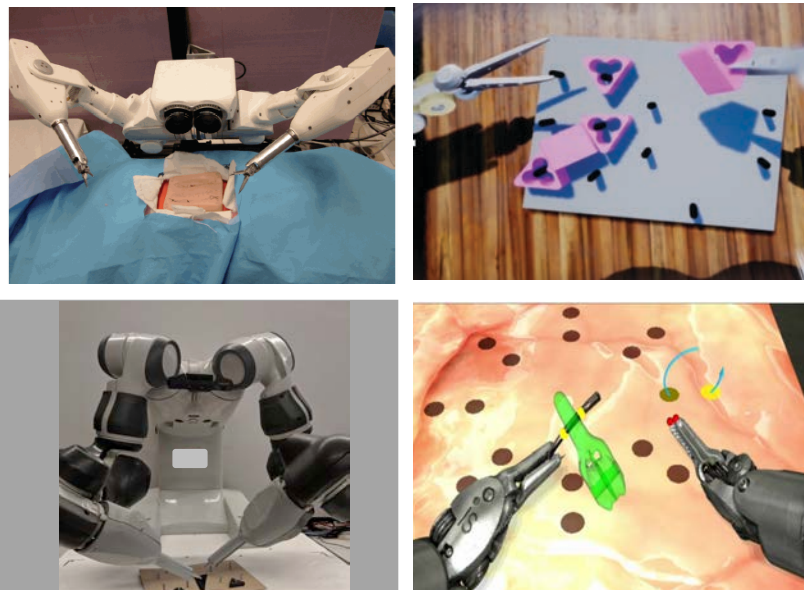


Figure 1: Top left: Taurus robot. Top right: Taurus Simulation. Bottom left: YuMi surgical adaptation. Bottom right: Da Vinci surgical robot.

The peg transfer was chosen as an initial task for two reasons. The first one is that peg transfer requires good surgical skill level from the robotics perspective. The second one is that it offers an environment that can be easily adjusted, controlled and measured. The procedure of peg transfer consists in grabbing a peg, pass it to a second gripper, and insert it on a pole on the opposite side of its initial location. To make it more challenging, we choose a setup where there are two sets of numbered poles (from 1 to 6) and the peg that is picked has to be placed in the pole labeled with the same number.

Each platform used a different subject for data collection. Each subject was trained for 20 hours of peg transfer before generating the examples. The library has a total of 408

instances of peg transfers collected from 112 trials. This experimental design allowed variability in the environment, with the purpose of getting a diverse set of examples. The following elements were randomized in the setup: 1) Peg initial positions, 2) Direction of the transfer (from left to right OR right to left) and 3) Position of the peg board (see Figure 2).

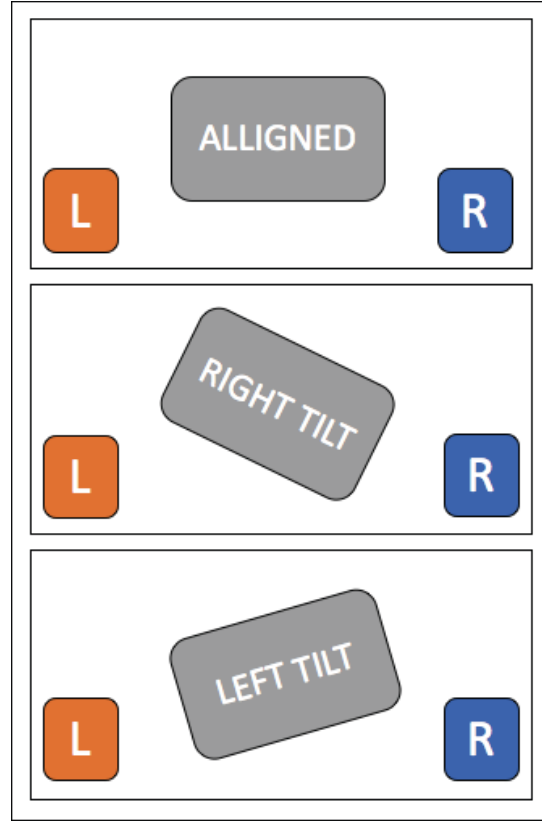


Figure 2. The data collection setup had 3 pegboard configurations: Aligned (top), Tilted to the right (middle), tilted to the left (bottom).

The information that can be collected through each robotic platform varies in format and media type. Nevertheless, the data recordings fulfill the requirements observed in other robotic datasets (Gao et al. 2014, Ahmidi et al. 2017). These requirements are summarized as following: 1) Kinematic information for all the robotic arms in use at all the captured frames, 2) RGB video information of the task, 3) Data annotations including atomic surgical actions (start frame ID, end frame ID and success or failure of the action). In the case of our dataset, the annotations are based on the video frames. All the data of this library was stored in a secure research database called PURR, hosted by Purdue University (Purr.purdue.edu, 2019). The information is stored under a project named 'Forward'. The access is restricted to the users directly listed in the database¹.

Two different simulators were developed for capturing and learning suturing skills. The first one was a mobile application, where a reinforcement learning capability was added to perform the suturing autonomously. The second one is a simulated da Vinci

environment with an integrated autonomous agent for needle driving. The subsequent sections explain the characteristics of the datasets for each robotic platform and the development of the simulation environments.

Subtask 1.1 Data collection

Data Collection for Peg Transfer Task with the Taurus Robot

The Taurus robot has two¹ 7 DOF grippers that are controlled through the Razer Hydra® controllers. The system has a foot pedal that allows to switch control between the arms and the camera. In this setup, the robot only moves when the foot pedal is pressed. The setup for recording consisted of 48 trials, each trial had 3 peg transfers for a total of 144 peg transfers. The initial position of the pegs was randomized. Half of the trials were initialized with the pegs on the left, creating left-to-right transfers and half of the trials were initialized with the pegs on the right. The place of initialization was also randomized. Figure 3 depicts the peg transfer setup.

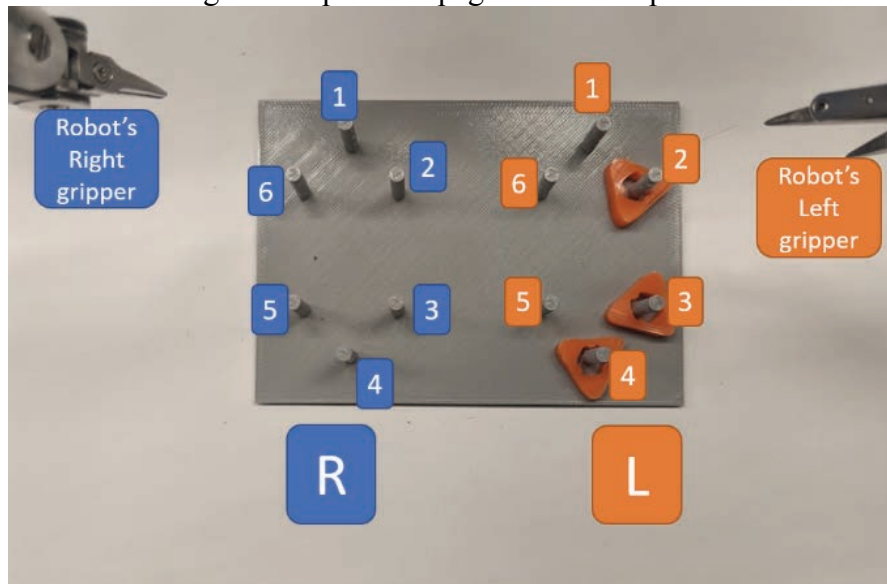


Figure 3. Peg transfer setup. The Left and right are inverted due to the camera's position (opposite to the robot's).

The data recording setup is depicted in detail in Figure 4. Each session has 7 files associated with it: An RGB video recorded from the top view recorded with the Realsense® camera (orange block in Figure 4), depth video stream also taken by the Realsense®. This device was placed at 25 cm of distance from the pegboard to maximize the accuracy of the depth sensor. The timestamps for both the video and depth were recorded. The timestamps were recorded to allow synchronization between both videos later on. The kinematic data from both arms was also recorded. This data included the rotation matrix of the tool tip, the translation in x,y,z according the robot origin and the griper state (which is a floating number between 100 and 30; 100 when the gripper is completely open and 30 when completely closed). The kinematic data was only recorded when the foot pedal was pressed, this filtered out all the unnecessary static data, since the robot was only recorded when moving.

¹ Contact gonza337@purdue.edu to get access to the database

Each database included a README file with a detailed documentation of the recorded data. Finally, the video's annotation file is included too. The Data Annotation Tool and Library Generation section describe the annotation process in more detail.

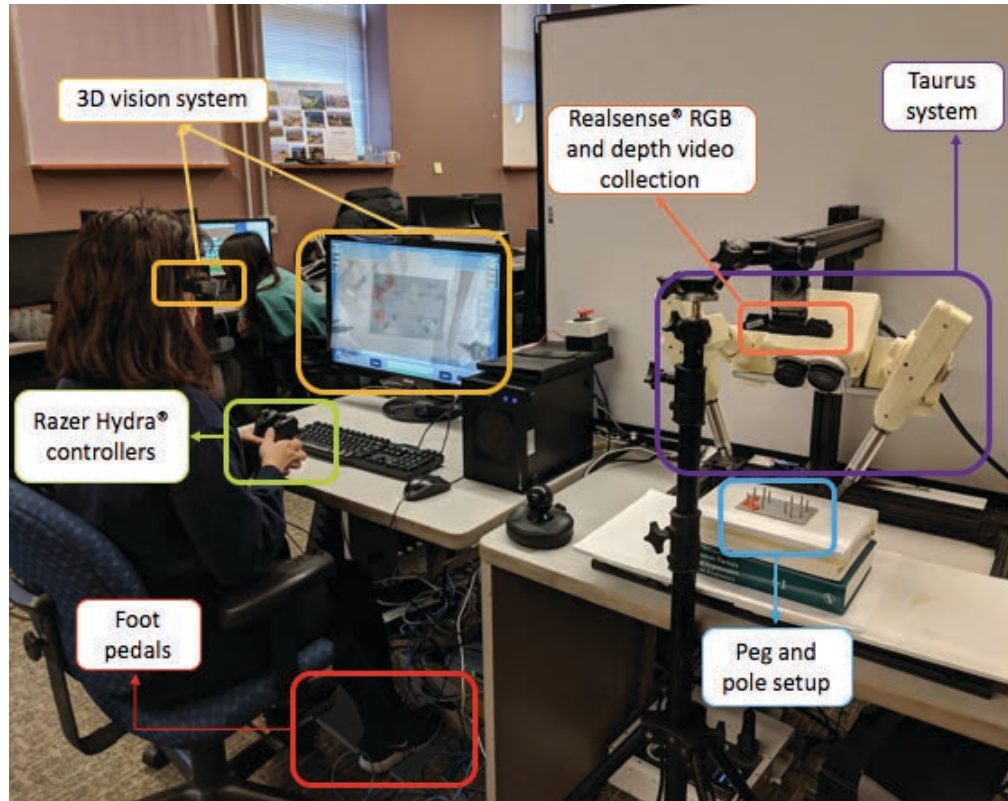


Figure 4. Taurus data recording setup

Data Collection for Peg transfer with the Yumi Robot

The adaptation of the YuMi robot (7 DOF) for surgical tasks consists of 3D printed grippers and teleoperation interface using the HTC VIVE™ motion controllers. The controller's motion is mapped to the robot's end effector while the trigger button is pressed, allowing to reset the workspace of each arm independently. Opening and closing of the gripper is controlled with the Trackpad button. 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 HTC VIVE™ reference frame and the robot gripper reference frame is depicted in Figure 5. During the data collection, the teleoperator stands in front of the YuMi robot and gets visual feedback through direct observation of the robot's workspace (See Figure 6).

A total of 40 trials consisting of 120 instances of peg transfer were recorded, following the randomization described in the previous section. The peg board setup for the YuMi robot was twice the size of the peg board used in the Taurus Data Collection.

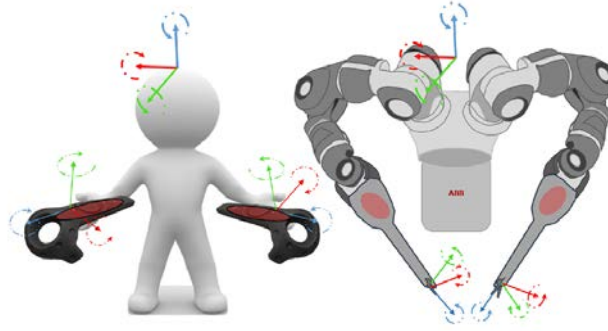


Figure 5. Coordinate frames for robot and controllers data.

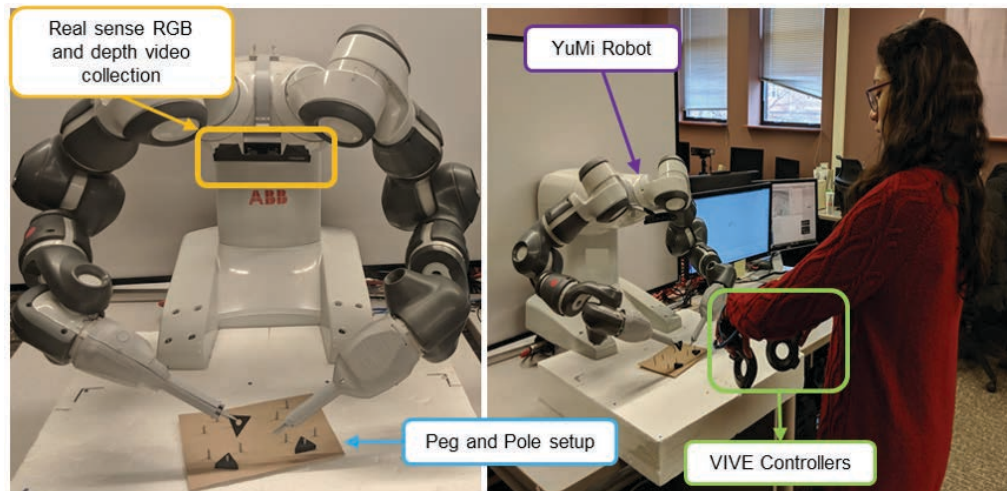


Figure 6. YuMi Robot and recording setup

The reported VIVE data includes the position, orientation and button states: trackpad button (used to open and close gripper) and trigger button (enables motion of the arms and choose the base coordinate frame for teleoperation). The data from the YuMi Robot includes position and orientation of the end effectors and the state (Opened / Closed). Color and depth data from the Real Sense were recorded at a frequency of 9 ~ 30 Hz, kinematic robot information was recorded at ~ 11 Hz, data from the VIVE controllers was recorded at ~ 10 Hz. All data was recorded with using timestamps for posterior synchronization.

Data Collection for Peg transfer with the Simulated Environment

A simulated robot environment (Figure 7) was used to collect data from the Taurus robot. The protocol of data collection for this dataset was very similar to the one used with the real Taurus robot. The Taurus robot has two 7 DOF grippers that are controlled using the Oculus Rift touch controllers. The simulated system is interfaced with a foot pedal that allows to switch control between the arms and the camera. In this setup, the robot only moves when the foot pedal is pressed. The setup for recording consisted of 24 trials, each trial had 6 peg transfers for a total of 144 peg transfers. The initial position of the pegs was randomized. Half of the trials were initialized with the pegs on the left, creating left-to-right transfers and half of the trials were initialized with the

pegs on the right. The place of initialization was also randomized. Figure 7 depicts the peg transfer setup.

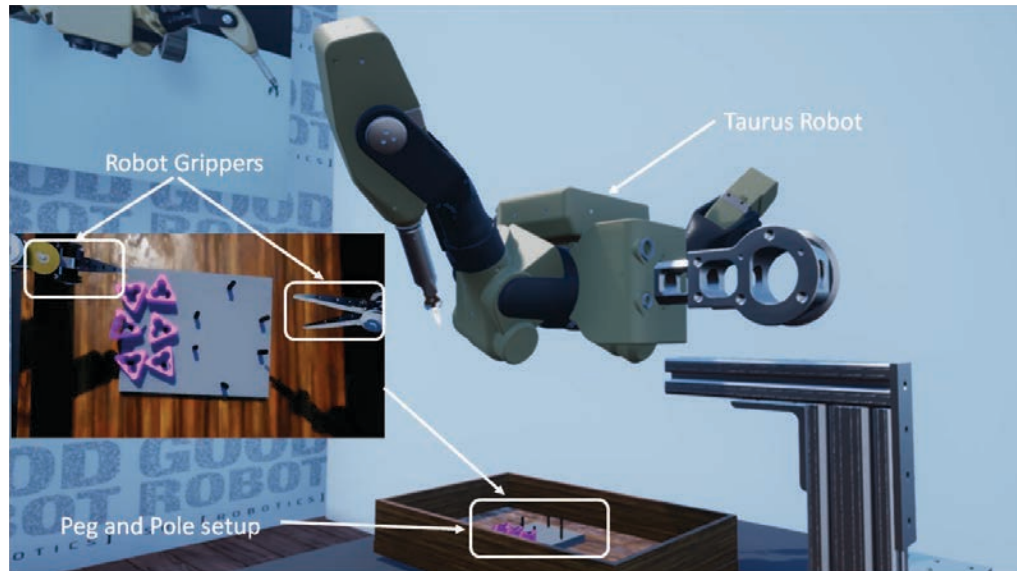


Figure 7. Taurus Simulated Environment

Each session of data recording has 9 files associated with it: An RGB video recorded from the user's virtual environment point of view, a left stereo camera view, a right stereo camera view (both the stereo views are from the Taurus robot cameras within the simulator) the two stereo videos can be used to obtain valuable depth information. The timestamps for all three videos were recorded. The timestamps were recorded to allow synchronization between all videos later on. The kinematic data from both arms was also recorded. This data included the rotation matrix of the tool tip, the translation in x,y,z according the robot origin and the griper state, which is a floating number between 100 and 30 (100 when the gripper is completely open and 30 when completely closed). Figure 8 depicts the setup for data collection.



Figure 8. Data collection setup for the simulated environment

Each database contains a README file with a detailed documentation of the recorded data. Finally, the annotation file based on the video is also enclosed. The Data Annotation Tool and Library Generation section describe the annotation process in more detail.

Visualization and Annotation Tool

The recorded data comprises color video, depth video, kinematic information of the robot and the controller trajectories for each peg transfer. The next step in the pipeline was to build a tool to visualize and annotate the recordings with respect to atomic surgical actions known as “surgesmes”. The surgesmes of the peg transfer task are explained in detail in the next section. Examples of these actions are: approaching the peg/pole, aligning and grasping the peg, exchanging the peg between the grippers, etc.

First, we listed out the desired functionalities of the tool, such as selecting the starting and ending frames of the surgesme, increasing/decreasing the speed of the video, an option to move forward/backward. Next, we tested existing video annotation tools that satisfy our requirements. It was found that the current annotation tools were customized for image semantic segmentation (i.e. drawing a bounding box on the image and providing the name of the object). Hence, we developed an annotation tool that is equipped with the required functionalities of our case study. Note that the surgesme annotation was created to operate on the color channel of the videos collected. However, the automatic surgesme classification can make use of the other modes of the data such as depth and kinematic information, in addition to the color, if needed.

The annotation tool was developed in Python programming language using the following libraries: OpenCV, and wxPython. This annotation tool will be made available to the computer vision community to encourage researchers to annotate their videos and further develop its features. A detailed description of the options and functionalities that the tool provides is given below (see Table 1).

Table 1. Functionalities of the Visualization and Annotation tool

Functionalities	Description
Video Loader	This button loads the video from the specified path in the main function of the source code as ‘video_path’. The video path itself is not GUI modifiable.

Information Bar	Displays messages about the current status of the annotation tool. This status will guide the user sequentially through the tool. Messages include: 1) Whether the video was successfully loaded and displayed 2) User's choices: start frame, end frame, surgeme and success/fail status of the surgeme 3) Next action to be taken by the user 4) End result of each action taken by the user (e.g. current frame rate and whether the limits have been exceeded) and 5) Error messages in the scenario when the user doesn't follow the recommended sequence of actions or if there are logical inconsistencies in the user's choices.
Choose Start Frame and End Frame	Toggle button that allows the user to choose the start frame of the surgeme or a single annotation record given that the video has been successfully loaded and displayed. Once the start frame has been selected, it allows the user to choose the end frame of the same surgeme or record while displaying appropriate messages on the GUI.
Choose Surgeme	A listbox that displays the list of all 7 surgemes. User can select one to annotate a video segment. This step is to be preceded by the selection of start and end frames so that the chosen frames can be associated with the corresponding surgeme in the record.
Choose Success or Fail	It's a drop-down menu which allows the user to record whether a performed surgeme was successful or not. Naturally, this follows the steps mentioned above in the same order.
Create Annotation	Once a surgeme is selected, clicking this button lets you create the annotation record. Its to be noted the record has not been entered in the file yet.

Current Annotation Display	This is a static read-only display area in the GUI that displays the current annotation record or surgeme information that the user has entered before saving it to the file. As soon as the record is saved to the file, this display clears.
Write to File	Clicking this button saves each surgeme information to a text file. The name and path of this file can be specified in the main function of the source code as 'write_path'.
Delete Last Record	This button deletes the last saved record from the text file. The deletion is not applicable to the current annotation that the user is working on.
0.5X	Clicking this button halves the speed of the video until the minimum frame rate is reached. This function works even when the video is paused or playing.
2X	Clicking this button doubles the speed of the video until the maximum frame rate is reached. This function works even when the video is paused or playing.
Last Frame	This button takes the user to the preceding frame of the video. When the video is paused, this functionality lets the user inspect the video in a precise manner. The event is initiated by either mouse click on the button or pressing left arrow key.
Next Frame	This button allows the user to move to the next frame of the video when the video is paused. It's initiated by either mouse click on the button or pressing right arrow key.
Play/Pause	It's a toggle button as in any standard video player which lets the user play or pause the loaded video. Before loading the video, the button displays 'Pause' but is not clickable (it is displayed in the information bar). Once the video is loaded, 'Pause' changes to 'Play' and while the video is playing, it once again switches back to 'Pause'.

Video scroll bar	It's a standard slider that extends from frame 0 to the end frame of the video, which is calculated as soon as the video is loaded. It displays the current frame number that is being played and can be dragged to change the frame.
All Annotations display	It's a text-display-only area where all the annotations saved to the current working text file are shown.
Video Display	It's a static display-only area where the video is being played with no scope for user modification.

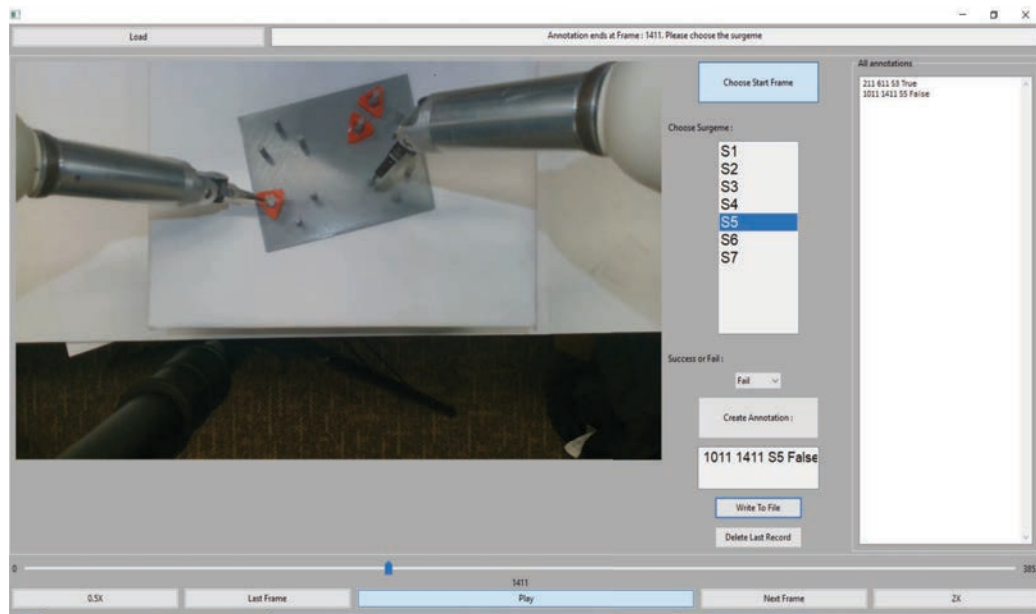


Figure 9. GUI of the visualization and annotation tool

The annotation procedure is as following: First, the annotator loads the video using the *load button* which can then be played or paused using the *toggle button* in the GUI (See Figure 9). The video can also be sped up or down by a factor of 2 using corresponding options in the GUI. The GUI annotation tool also allows the user to navigate to the next or previous frames using both mouse button clicks or side arrow keys. Using this visual feedback, the user can decide about the starting and ending frames of a particular surgeme. Then, the user can navigate to the chosen start frame. Then he/she clicks on 'Choose Start Frame' toggle button which then changes to 'Choose End Frame'. The same procedure is repeated for recording the end frame. Once this is done, the user can select the surgeme number from the 'Choose Surgeme' list given by clicking on it and

then on the success/fail option from the ‘Success or Fail’ drop-down box. Finally, the annotator can click on ‘Create Annotation’ and the current surgeme record will be displayed below that button. Once the record is verified, he/she can click ‘*Write To File*’ which will save this information to a text file in the specified path. The information entered during this workflow is fixed except for the last annotated surgeme. The ‘*Delete Last Record*’ button allows you to discard the last saved record in file. All the information in the file is displayed in the ‘*All annotations*’ text-display area at all times starting from the time the first surgeme is saved. In addition to all these functionalities, there is an information display bar at the top of the tool which shows status messages that guide the user through the workflow as well as result (success or error) of each action taken by user. This procedure is followed to annotate the data corresponding to all the three robots.

We used this tool to annotate the peg transfer dataset of YuMi, Taurus robot, and the Taurus simulator. Each task consisted of up to six peg transfers. Furthermore, there were a total of eight surgements and each surgeme was allowed to be a success or a failure. Each task is associated with an annotation file in the following format: each row in this file consists of the start and end frame IDs of the surgeme, name of the surgeme and whether the surgeme execution was a success (True) or a failure (False). All the annotated data is saved to a text file whose path has already been statically assigned in code. The format of a single surgeme or annotation record is as follows - Start frame of surgeme, End frame of surgeme, Surgeme number (S1 to S7 as shown in Figure 9) followed by True/False indicating whether the surgeme was a success or fail when it was performed. E.g. "36 52 S5 False" conveys that in the video, the surgeme with label 5 was performed from frame number 36 to 52 and was not successfully completed. All such records in each trial or video were consecutively saved as new line entries in a file in the same format and is a separate file for each of the 48 videos.

As mentioned earlier, the color, depth, and kinematic information were recorded at a different frequency. Hence, we first utilized the timestamps to synchronize these modes of the data. Next, the obtained annotations were used to segment the videos into respective surgements. In other words, all modalities of recorded data were segmented into surgements and organized in a format that is convenient for later applying machine learning algorithms for classification and prediction.

Suturing simulation data collection

To compliment the work done by the Purdue team collecting and annotating benchtop data, the JHU team has devoted their first quarter effort toward the organization of existing data for this project as well as the development of two simulation-based environments for data capture, development, and testing.

Existing benchtop data includes the publically available JIGSAWS dataset and the MISTIC data set. Each of these data sets includes multiple tasks acquired from multiple users, but they carry different types and granularities of labels. As we identify target tasks to perform, we will augment and/or adjust these labels to create a natural motion vocabulary to support the target tasks. An existing da Vinci Skills Simulator (dVSS) user study recorded simulated robot kinematics and video demonstrations of a needle

passing task from 30 clinical trainers and engineers. We will augment this dataset with demonstrations from surgeons performing a needle passing task in the dVSS. We have recorded over 300 demonstrations in a 2D suturing simulation – Needle Master -- from engineers.

To facilitate fluid algorithm development we will learn fundamental surgical skills in two simulated environments: a simplified 2D suturing environment called Needle Master (Paxton et al., 2016) and a realistic, commercial 3D robotic simulation: the dVSS.

Needle Master Environment

The Needle Master environment was developed by Dr. Paxton to investigate Imitation Learning (Paxton et al., 2016) and is available as an Android game in the Google Play Store (See Figure 10). The needle can be controlled using a touch screen or by injecting autonomous motion into the simulation. Our team has integrated the game environment with a Reinforcement Learning library and has begun training an autonomous agent to complete a needle passing task.

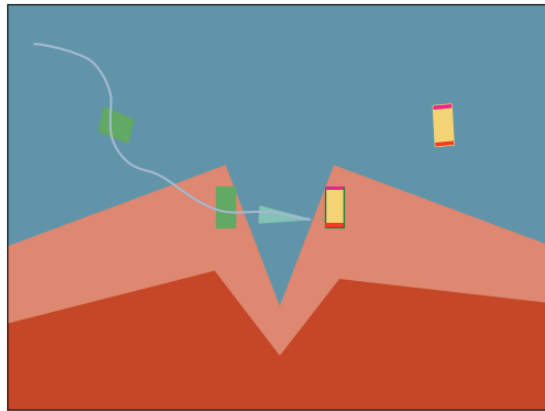


Figure 10: A screenshot from the Needle Master 2D suturing simulation. Tissue is shown in salmon. Deep (critical) tissue is shown in red. Completed gates are painted green. Incomplete gates are shown in yellow.

da Vinci Surgical Simulator Environment

The dVSS is a commercial simulation of the da Vinci Surgical Robot with a master console connected to simulated surgical tasks. Users can manipulate the simulated slave robot using the same master controller as in the da Vinci Surgical Robot. Our team's previous work augmented the dVSS with visual cues, Figure 11, in the needle driving task to indicate where to grasp the needle, the correct insertion path, etc. Our team has begun identifying how to integrate the dVSS with an autonomous agent. Once we can inject autonomous motion into the dVSS we will begin training an autonomous agent to perform needle driving using Reinforcement Learning and Imitation Learning techniques.

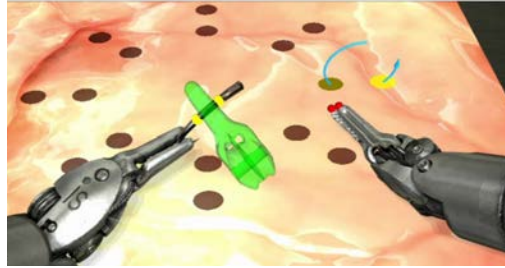


Figure 11: A needle driving task in the dVSS augmented with visual teaching cues (Malpani, Anand, et al., 2017)

Purdue has set up infrastructure to record robot kinematics, RGBD video and manually annotate task surges. In the dVSS we will also record robot kinematics, stereo video, and environment variables. In the Needle Master simulation, we can record needle trajectories, actions taken, and video. Using changes in the simulated environments we will be able to automatically segment demonstrations into surges. For example, Malpani, Anand, et al. (2017) decomposed needle driving in the dVSS into four steps: 1) align the needle with the target using the right gripper, 2) drive the needle into the tissue (bite), 3) grasp the needle with the left gripper, 4) remove the needle from the tissue. We can define each surge as seen in Fig 12 below.

Steps	Surge	Parameter 1	Parameter 2
1	Align	Right arm, grasp needle	
2	Bite	Insert needle in tissue	Release needle
3	Grasp	Left arm, grasp needle	
4	Remove	Remove needle from tissue	

Figure 12. Parametrized Surge list

We initially focus on the needle driving task in simulation which requires similar fundamental skills to the peg transfer task: grasping objects, object transport, with the added challenge of tissue interaction. In addition, there are other tasks available in the dVSS that parallel the peg transfer task like “Sea Spikes” which requires the user to grasp, remove, and place hoops on flexible “spikes”. Both simulation environments allow us to move target locations and change the experiment initializations as in the Purdue dataset (see Fig 2).

Task 1 (cont’d): Data Collection and Sensorimotor Library Generation

For this task, the library of surgical actions was expanded to contain data from multiple subjects. A total of 48 peg transfers were recorded for the real Taurus system with two new subjects. In the the simulated Taurus environment 100 peg transfers were recorded with 6 new subjects. The surgical actions were segmented and annotated. In addition, a debridement model was created for both the real and simulated scenario. The design attempted to keep the realism of the debridement challenges, while maintaining similarities between the simulator and the real scenario. Finally, a knowledge transfer model was implemented to learn from the simulated data.

Subtask 1.2 Library generation

Modeling the surgemes

Surgical tasks can be considered as a sequence of skills or motion primitives referred as surgemes or atomic surgical actions. For instance, debridement task can be decomposed into the following skills: 1. Approach the skin, 2. Identify the region of grasp, 3. Align and grasp the skin, etc. Literature shows that the surgeons learn and master these skills during their intern and resident years. These pre-learned skills allow them to perform a wide range of surgical procedures with or without the supervision of experienced surgeons. Our goal to develop autonomous surgical robots has stemmed from this idea. Robots powered with such skills can be supervised by a remote surgeon to perform surgical tasks. Hence, these skills play a crucial role in the development of autonomous or semi-autonomous behaviors in surgical robots. Our first objective is to develop a comprehensive list of surgemes that can be generalized to a range of surgical procedures. In other words, we theorize that surgical tasks constitute a sequence of these list of surgemes. We also expanded the concept of generalization within the task. We want each surgeme to have broad definition, allowing the robot to adapt the skill action to different environments. For example, let us say we have the following surgemes: approach a peg and approach a pole. This decomposition would potentially give rise to large number of surgemes and can not generalize to debridement task which requires approaching a skin. On the contrary, a surgeon who learned to execute an approach can approach peg, pole and skin. Hence, we propose a general description of surgemes by parameterizing them with respect to the location and object. In the above example, we want the robot to learn the skill known as approach irrespective of the object and location. This idea can be extended to other surgemes such as align, lift, grasp and transfer. This approach is illustrated using debridement and peg transfer tasks. Besides relying on the existing literature, we have observed several trials of these tasks in order to obtain a list of surgemes as given below (See Figures 13-15).

S. No.	Surgeme	Parameter 1	Parameter 2
1	Approach	Object	Location
2	Align	Object	Grasp/Ungrasp
3	Lift	Object	
4	Get together		
5	Exchange		
6	Incision	Orientation	

Figure 13. List of surgemes and their parameterizations

Steps	Surge	Parameter 1	Parameter 2
1	Approach	Peg	Location
2	Align	Peg	Grasp
3	Lift	Peg	
4	Get together		
5	Exchange		
6	Approach	Pole	
7	Align	Pole	Ungrasp

Figure 14. Sequence of surges for peg transfer task.

Steps	Surge	Parameter 1	Parameter 2
1	Approach	Skin	Location
2	Align	Skin	Grasp
3	Lift	Skin	Offset
4	Incision	Orientation	
5	Lift	Skin	Offset
6	Align	Region	Ungrasp

Figure 15. Sequence of surges for debridement task.

Subtask 1.3: Military provides feedback about the conceptual model

We have engaged in a series of discussions with Dr. Overholser to identify possible target procedures for a future remotely guided medical robot. Dr. Overholser proposed three common procedures: 1) needle decompression for tension pneumothorax, 2) emergency tracheotomy, and 3) arterial bleeding from GSW. We believe that emergency tracheotomy is an especially promising task for proof of concept because of easily identifiable landmarks, a dissection component, and an easily confirmable outcome.

Discussion of these procedures has led to the hypothesis that a small set of basic capabilities – dissection, pressure, retraction, and other methods to control bleeding that would comprise a basic armamentarium for a forward-field surgical robot.

With these are target capabilities, we agreed that a demonstration system that was able to perform a small set of basic manipulations of objects and tissues would have a sufficient vocabulary of basic movements to serve as a proof of concept for the scalability and generality of our proposed approach. Our initial target task will focus on the fundamental steps in needle driving, which we have existing data for in the JIGSAWS, MISTIC and dVSS datasets. A theoretical framework that can learn the fundamental steps in needle driving promises to be capable of learning basic skills to control bleeding on the battle field.

Subtask 1.1 Data collection

Enhancing the Taurus Dataset

The surgical actions for the real Taurus dataset were recorded from two new subjects. Each subject trained for 8+ hours to achieve a good level of dexterity for teleoperation. The training

sessions consisted on moving the arms through straight paths, followed by pick and place operations. Finally, the subjects trained on the peg board for 4 sessions (12 peg transfers) before recording.

A total of 48 peg transfers were recorded in total. The recording setup was the same as in the previous sessions. Each subject recorded 8 trials. Four sessions were initialized with the pegs at the left of the board and 4 sessions with the pegs at the right. At the beginning of each session the initial position of the pegs was randomized. The tilt of the pegboard was set using the same random design as in the previous data collection. Table 1 summarizes the data collected.

Table 2. New surgical actions collected.

<i>Robot System</i>	<i>N° of New Subjects</i>	<i>N° of Peg transfers</i>	<i>N° of surgical actions</i>
Real Taurus	2	48	Approach (peg): 111 Align and grasp: 115 Lift (peg): 112 Get together: 113 Exchange: 113 Approach (pole): 109 Align and place (pole): 107
Simulated Taurus (Fixed Board)	6	100	Approach (peg): 192 Align and grasp: 206 Lift (peg): 203 Get together: 180 Exchange: 175 Approach (pole): 167 Align and place (pole): 163
Simulated Taurus (Moving Board)	2	133	Approach (peg): 200 Align and grasp: 228 Lift (peg): 211 Get together: 195 Exchange: 195 Approach (pole): 199 Align and place (pole): 205

Enhancing the Taurus Simulation Dataset

The dataset from Taurus simulator was augmented with data of peg and pole task performed by 6 subjects. To add variability and gauge performance enhancement through training, 4 subjects trained for 12 peg transfers before performing to achieve good dexterity and 2 subjects trained for 6 transfers. The new dataset was performed on a modified version of the simulator where the peg board was fixed to the surface, to ease the performance of task.

A total of 100 peg transfers were recorded in total with average of approximately 17 peg transfers per subject. The recording setup was the same as in the previous sessions. Each subject recorded 3 trials with each trial consisting of 6 transfers. To add variability to data, the start and goal pegs for each transfer were randomized. Also, the direction of transfer was randomized for each transfer, i.e. pick with left arm, transfer to right arm, and place or pick with right arm, transfer to left arm, and place. The orientation of the peg board was fixed through all trials. This new data together with the data collected with free peg board contains a total of 233 peg transfers. Table 1 summarizes the data collected. It was observed that the 4 subjects who trained for 12 transfers in training had only one failed transfer across all subjects, i.e. each subject successfully completed all 6 transfers in each trial except one subject who had 1 failed transfer. In comparison, the subjects who trained for only 6 transfers had 7 failed transfers across 6 trails with an average of one failed transfer per trail. Figure 1 shows the difference between fixed and free peg board.

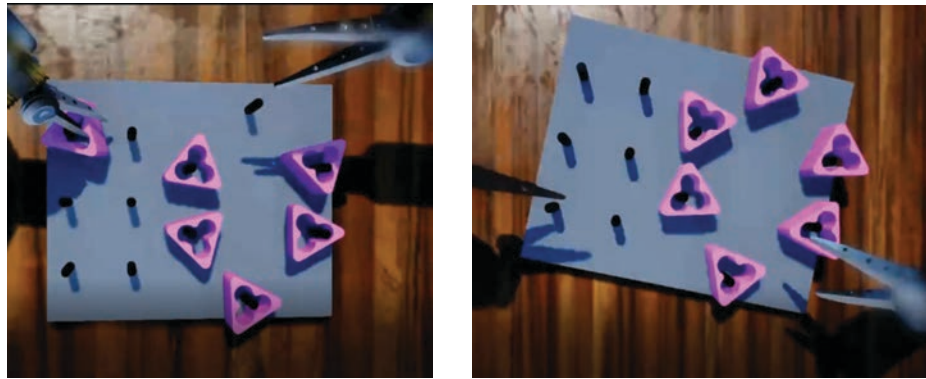


Figure 16. Left: Fixed peg board orientation. Right: Free peg board, the orientation changes during task performance

Annotation and Verification

The new data collected was annotated with respect to the surgical actions using the annotation tool developed in the previous step. Further, the entire set of annotations was verified by using a modified version of the annotation tool. This new GUI tool comprised new functionalities to delete or modify existing annotation records and add new records on top of the functionalities that were already available. A summary of the functionalities in the modified tool is given below (see Table 2). The current annotation and verification procedure is based on color but it can be based on any other feature, such as kinematic data or depth. The tool was developed in Python using OpenCV and wxPython. The new tool was made available to the research community, and is hosted through a private Github project available at https://github.com/nmadapan/Forward_Project.

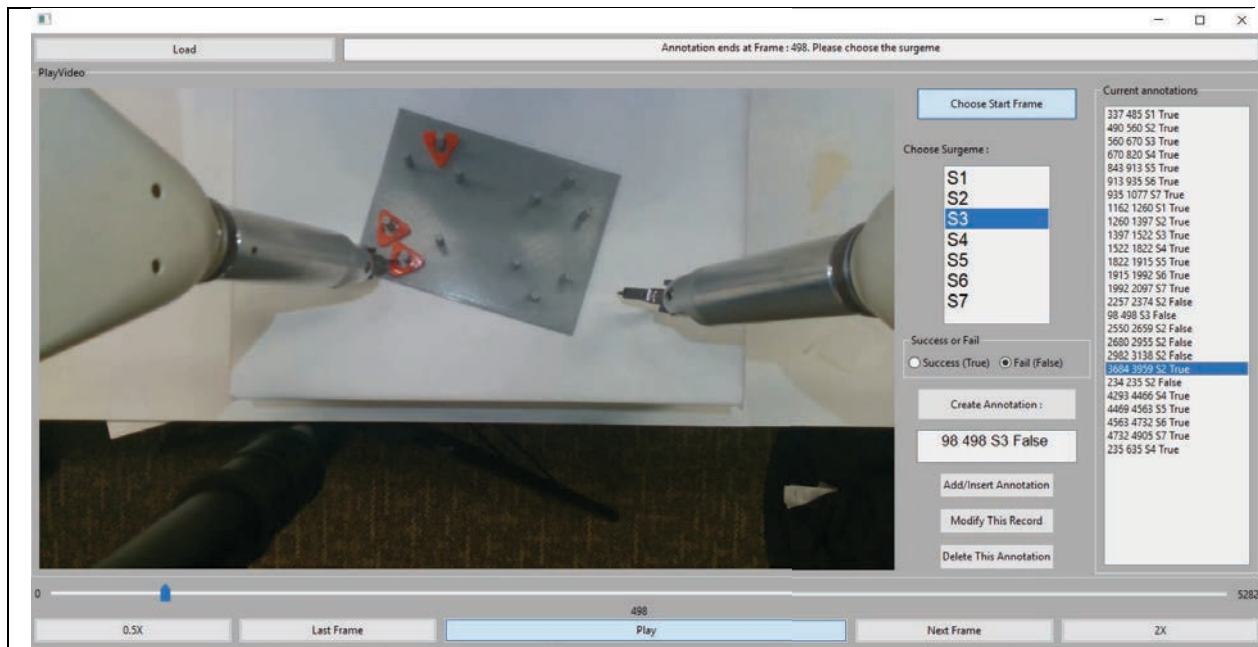


Figure 17. GUI of the annotation and verification tool

We have used this tool (see Figure 2) to annotate the new data and verify the entire enhanced dataset. While one end-user annotated the data, the annotations were verified by 3 users. Each video or task is associated with a particular file that is uniquely named with respect to the task, where each task consists of multiple peg transfers and each transfer was a composition of all the 7 surgical actions.

Debridement model

The debridement procedure is very relevant in the battlefield, since there are many elements in the environment that can severely damage the skin. This surgical technique removes the dead tissue on the superficial layers so the healthy tissue can heal underneath. Most models of debridement treat it as a pick and place operation [1-2]. These models offer little variability and have little resemblance to the actual task. Thus, we strived to create a more realistic model of debridement that would be replicable in the simulator that the teleoperator would be looking at.

Three different debridement models were tested (see Figure 18) Each model was tested four times in a teleoperation environment using the same subject. In every case, the teleoperator was able to successfully remove the dead skin. The first two models (option A and B) started with a piece of the skin already loose, only attached at the center of the patch. The first model used a foam with no elastic properties. For option B we developed a latex based skin model d, to add an elasticity that resembles human skin. Lastly, Option C did was a solid piece of simulated skin. In this model, the teleoperator must cut remove the skin from the start, taking several iterations. We decided to reproduce Option 3 in the simulator, since it was the most realistic scenario.

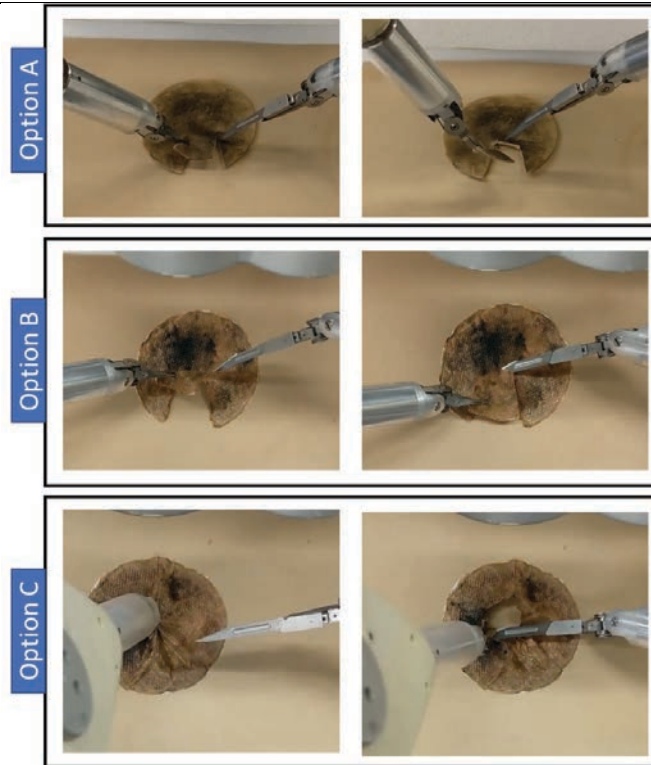


Figure 18. Three debridement models. The Pictures in the left column show the “lift (skin)” surgical action and the pictures in the right column show the “cut skin” surgical action.

Development of Debridement Simulation Model

As stated previously, debridement is very relevant to a battlefield scenario. Our architecture has the surgeon performing the task in a simulator environment, which should resemble the actual environment seen through the robot’s camera. Hence, there is a need for realistic debridement simulation model which is close to the actual model presented in previous section. To achieve a realistic simulation of debridement setup, the skin was modelled using Nvidia Flex cloth physics, which models the material as set of particles connected using springs. This model can capture stretching and deformation similar to how skin behaves.

To resemble the real debridement setup, the shape of the simulator model was modeled as a flat circular mesh. The mesh was created with Blender open source animation tool, with high density and uniform radial placement of vertices with tri-mesh connectivity, to capture small and fine deformations of the skin model. Figure 19. shows the vertex mesh and the connectivity.

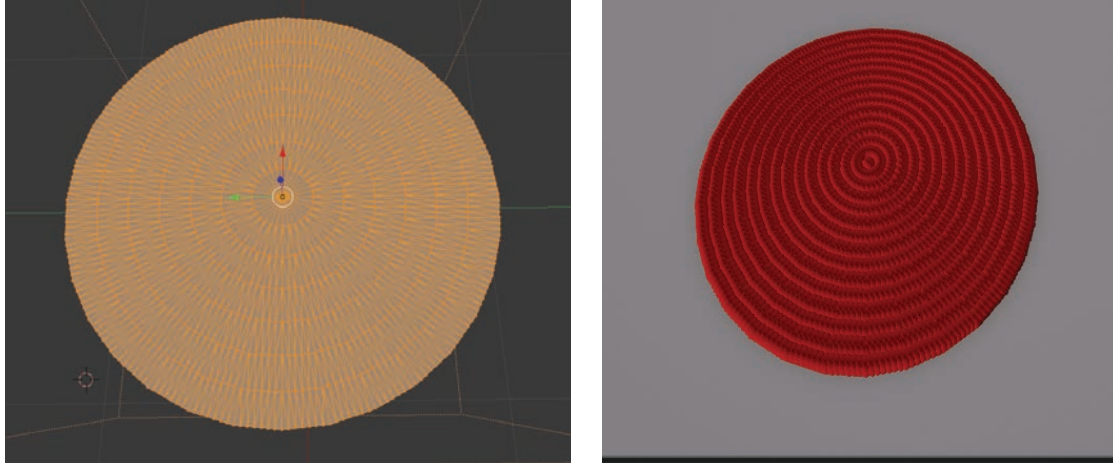


Figure 19. Left: The Mesh model with tri-mesh connectivity created using Blender. Right: The particle model created based on the mesh in Unreal Engine.

This mesh was used to simulate the debridement model. The Unreal Engine-Flex build was used to simulate the desired physics of debridement setup. A particle with mass is added at each of the vertices and a tether spring is added on each edge in the mesh. The deformation of the model is dictated by the tether stiffness, damping, and max tether deformation allowed before the tether breaks. The tearing or cutting of the skin is simulated as breaks in tether springs on interaction with the cutting tool. The tether properties (stiffness, max deformation) and particle mass were set based on deformation observed from the real debridement model. It was tuned by adding the Taurus robot tool tip and performing debridement in the simulator. Figure 4. shows the debridement as performed in the simulator. The link to the simulation vs real video can be found at [4].

Development of da Vinci Skill Simulator Environment

Controlling a 7 DOF robot in the field with resilience will require enhancement of data collection based imitation methods with reinforcement learning in a simulated environment. We make use of Intuitive's da Vinci Skill Simulator (dVSS - research edition) to allow an agent to learn surgical motions with reliability, and eventually translate them to the real world (see Figure 5). Currently, we've finished porting the simulator from Windows to Linux, and are finishing modifications of the code to allow us to connect to it as an OpenAI Gym environment for deep reinforcement learning in Python.

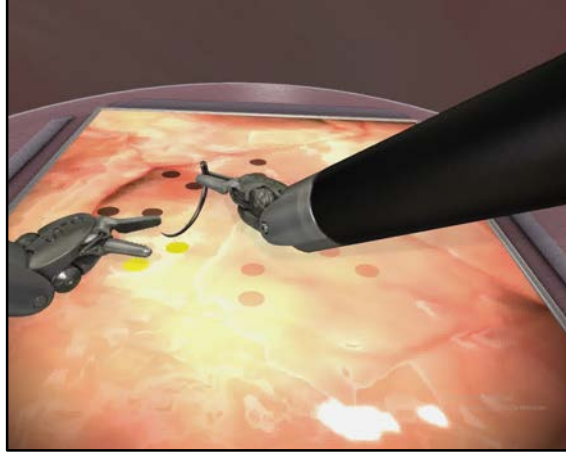


Figure 20. *Intuitive da Vinci Skill Simulator (dVSS) ported to Linux and with RL environment functions exposed.*

Recording Surgeon Demonstration of Needle Driving

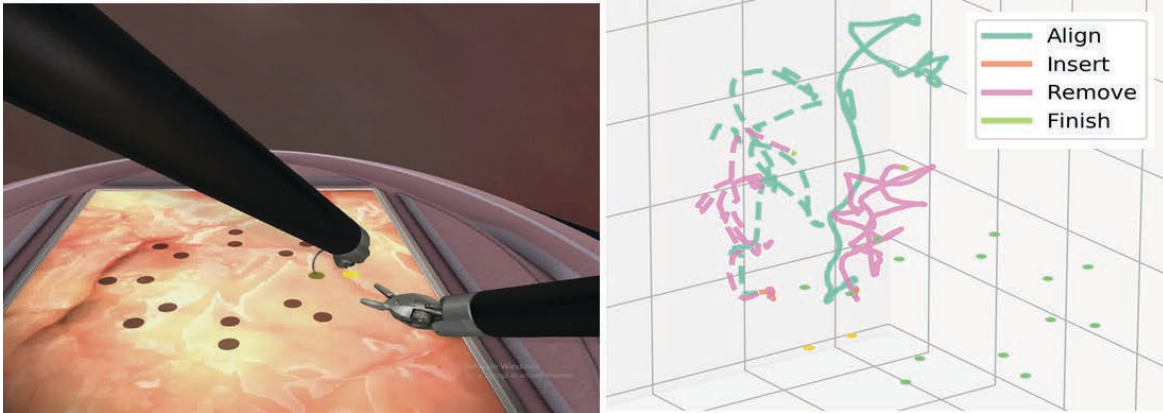


Figure 21. *Left: a screen shot during surgeon demonstration of the “Insert” surgeme. Right: 3D tool motion during the first needle drive. The target locations are shown in green and the current target is highlighted in yellow. Using the simulator we can automatically segment into the different surgemes: Align, Insert, Remove, Finish.*

We recorded demonstrations of 13 surgeons performing over 91 demonstrations of an 8-target needle driving task. This dataset includes 328,127 image frames, of which 118,870 frames have been processed and are used in Experiment 2 below. The eight targets are arranged in a circle with the insertion targets along the inner circle and the extraction targets around the outer circle. The next target is highlighted with flashing yellow markers (see Figure 22). The demonstrations capture tool pose, camera pose, needle pose and interaction with the tissue. We also recorded score metrics like path length, tissue damage, errors (needle drops). These metrics can be fed into imitation learning algorithm.

Development of a simulation environment for the YuMi robot

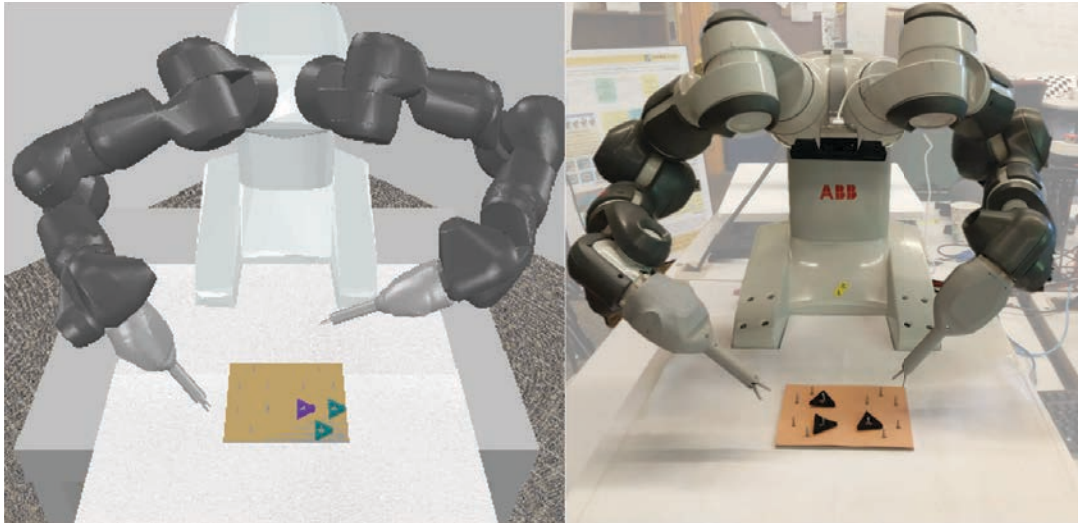


Figure 22. Simulation environment the YuMi robot versus real robot peg transfer task.

An open simulator is required for the project to test execution strategies, augment data and test reinforcement learning techniques. Thus, a simulation environment for the robot IRB 14000 ABB YuMi was created using the robotic simulation software V-REP.

The scene created resembles the setup used for the YuMi robot in the collection of the dataset, including peg board and triangle objects. The created simulator allows the user complete freedom to modify the robot, the scene objects, the control commands and the data to be collected. The scene can easily be updated when new tasks are added to the dataset. The scenes can be downloaded from this link².

Simulation scene for robot control

The simulation scene yumi-surgical.ttt (see link¹) is a simulation environment meant for testing algorithms for robot control. The robot is controlled by changing the target of the gripper tips' position and orientation, and the state of the gripper (open/close). The behavior of the robot in the environment is given by an inverse kinematics solver configured in the scene. This simulation has potential applications for testing different control approaches from action execution of small tasks to testing how the teleoperation systems handles delays between the operator and the real robot in a remote setting.

² https://www.dropbox.com/sh/hh3xpa4abadrom0/AADh1ZH8ssaQjO_4kFBTSJLua?dl=0

Data collection scene for the YuMi robot using a Virtual reality headset

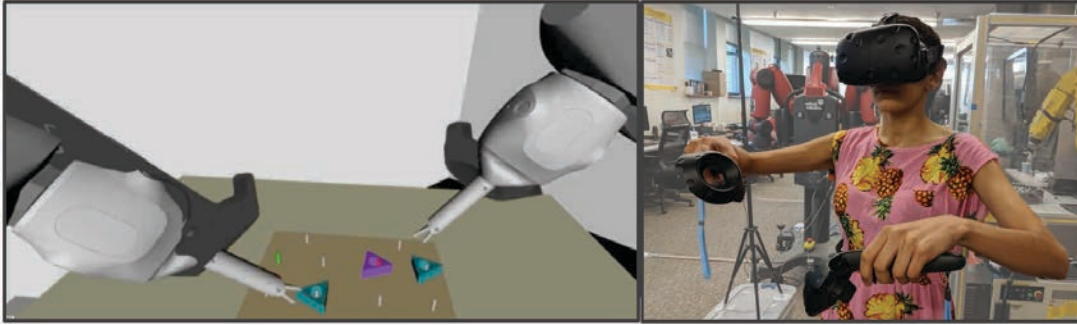


Figure 23. view of the operator during data collection. Operator using the HTC VIVE.

The scene `yumi-surgical-vive.ttt`¹ is a simulation environment meant for recording samples from robot execution. The user can perceive the simulation environment in virtual reality using the VIVE headset and can freely move around the scene to obtain a first or third-person view of the scene. The arms of the robot follow the position and orientation of the VIVE controllers, and the gripper closes when the trigger button is pressed. The data is saved at each simulation step collecting the number of the trial, the simulation time and information about the robot and the environment. Currently, the `vrep` scene captures information about the objects of interest, the target triangle to perform peg transfer, the position and orientation of the grippers, the grippers' state and the joint angles of the robot arms (as shown in Table 1).

Table 3. Information Captured. T_s is the simulation timestamp, $\sim J$ is the vector of joint angles, $\sim p$ is the position vector (x, y and z), $\sim \theta$ be the Euler angles (yaw, pitch and roll), gs is the gripper state of the end-effector

Measured objects	Variables saved
Right and left gripper	J, p, θ, gs
Triangle objects	p, θ
Peg board	p, θ
Pegs (source, target)	p, θ

The scene can be modified to add new elements and acquire different information about the scene. In addition, a top camera ($\sim$ Real sense SR300) included in the scene can be used to obtain RGB and Depth images of the robot's environment as shown in Figure 3.

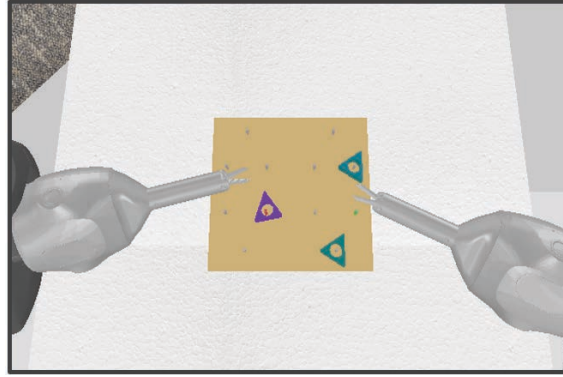


Figure 24. Top camera view of the scene.

Da Vinci Skill Simulator Reinforcement Learning Environment

In order to control a 7 DOF robot in the field, we would like to employ reinforcement learning in a simulator, so as to be able to transfer a learned policy from the simulator to the real world. We have finished transforming the dVSS (research edition) to an OpenAI-gym compatible environment (See Figure 25). While some functionality may need to be added (for example, creating a reward function tailored to specific tasks), the environment is feature complete, and we are starting to use it for RL applications.

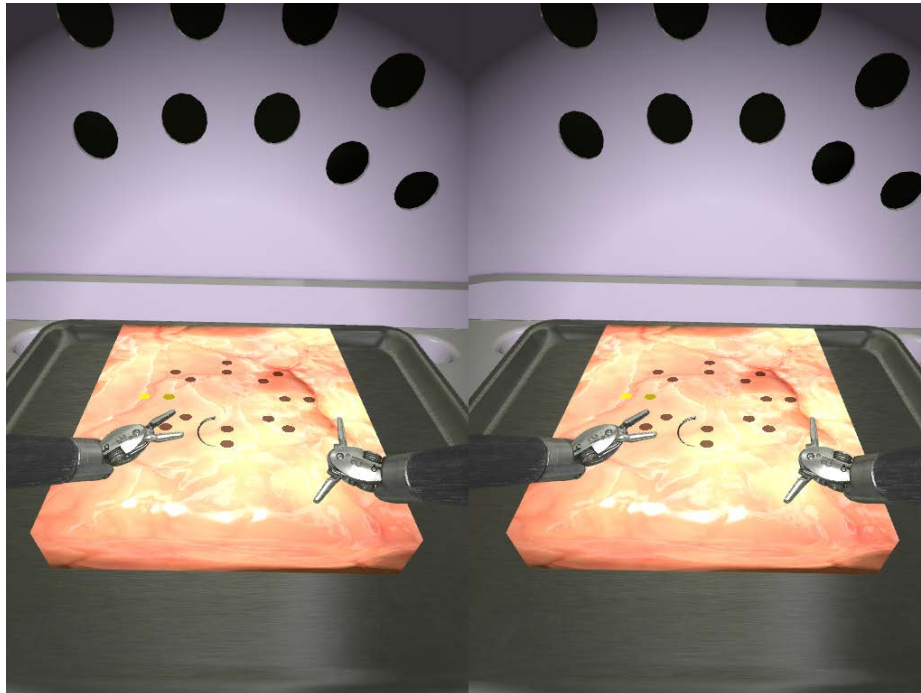


Figure 25. Intuitive da Vinci Skill Simulator (dVSS), working in stereo mode as a reinforcement learning environment.

Application of Reinforcement Learning to Surgical Environments

While we start to apply RL to tasks in the dVSS, we continue to rely on our simplified NeedleMaster environment to test out ideas on applying RL efficiently. NeedleMaster has been clarifying, in that we see the limitations of RL with regard to exploration, even on an ostensibly simple environment. While we have managed to get DQN working on NeedleMaster (see Figure 5), applying a continuous-control RL algorithm such as DDPG using image data is still a challenging task. We believe we've identified issues with the algorithm itself that we hope to address and have shifted to using the alternative stochastic version in PPO (Proximal Policy Optimization).

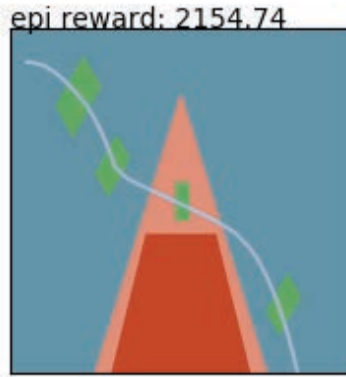


Figure 26. The NeedleMaster environment, which is used to iterate quickly over RL approaches, can now be solved using DDPG and state data.

With regard to the exploration problem, we believe that augmenting the algorithm with exploration based on our existing recorded expert trajectories, as well as our proven simplified control signal, will assist the algorithm in attaining convergence. At the same time, we are working on collecting more expert data recorded in the dVSS.

Prediction of Future States in Surgical Environments

Prediction of future state is a useful mechanism for a surgeon operating remotely, as well as for an RL algorithm as a backup mechanism for resilience. We attempted to predict future surgical state and images from current state and images both on our recorded demonstrations (from dVSS) and from the JIGSAWS datasets using convolutional networks as well as VAEs. The results are promising (See Figure 27), but require more tuning of hyperparameters.

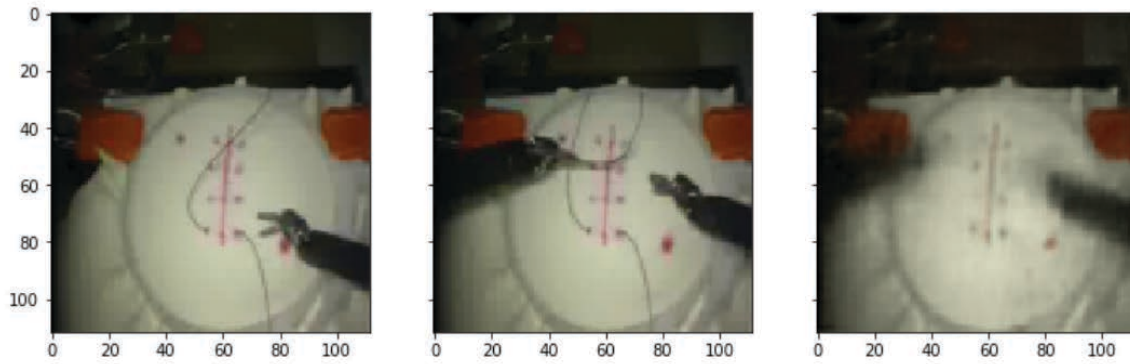


Figure 27. Prediction of future state and image from present state and movement. left: current state, middle: future state, right: predicted future state.

Surge execution

Two execution strategies were developed for the skill library. The first strategy is model based, and the second strategy is learning based. Currently, these models were tested with the surgical gestures performed during a peg transfer task. The system used was the Yumi robot with a real sense camera on top (See Figure 28). The list of surgical gestures is shown in Table 4.

Table 4. List of Surgical gestures for Peg Transfer task.

Number	Surge
1	Approach (Peg)
2	Align & Grasp (Peg)
3	Lift (Peg)
4	Get Together (Grippers)
5	Transfer (Peg)
6	Approach (Pole)
7	Insert (Pole)

Modeled approach

This approach uses knowledge about the environment, the robot and the surgical gesture to execute the action. This strategy follows the flow described in the figure below (Figure 7).

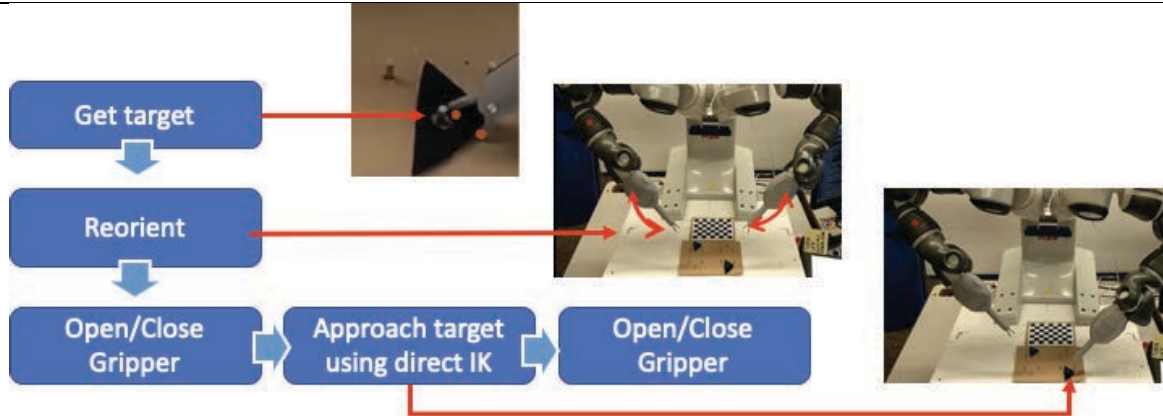


Figure 28. Steps followed during the execution of the modeled approach

For each surge a goal state is specified based on the environment state (robot pose, peg and pole positions). The locations of the poles and pegs were detected offline using machine vision. The poles are detected by thresholding of the pointcloud collected with the realsense. The peg position is classified using a re-trained YoloV3-Tiny deep architecture [1]. The training data consisted of 144 peg transfers performed with the Taurus Robot. The data was annotated semi-automatically using color filtering. The semi automatic annotation yielded an accuracy of 82%. A human annotator verified the result of the semi-automatic program to take the ground truth accuracy to 100%. The online version of the object detection system is currently in development.

Learned approach

This strategy uses the kinematic data collected for each surgical gesture to learn a model for trajectory prediction. In this model we define the prediction as:

$$\Phi_i(x_{init}, x_{end}) \rightarrow \{w_1, w_2, \dots, w_{k-1}\}$$

Where x_{init} is the current position of the robot, x_{end} is the target position detected by the vision system described in the previous section and $w_1, \dots, w_{k-1}$ are the $k - 1$ waypoints of the B-spline of degree k that best fits the trajectory to perform the surgical gesture i , as shown in figure 8. As a preprocessing step k was determined by iteratively increasing the degree of the spline. The iteration stops when the difference between the fitted curves were equal. This process yielded a $k = 3$ for the collected surgical gestures.

For this experimental design, 314 surge samples from the YuMi peg transfer were used. The data was split with an 80:20 training-testing setup. The model Φ was learned using two different approaches: a) Linear Regression(LR) b) Neural Network (NN) using Mean Squared Error (MSE) as the loss function. The neural network had a fully connected hidden layer of size 15. Finally, The model Φ was used to predict the fitted spline waypoints. Then, the x_{init} , x_{end} and the predicted waypoints (w_1 and w_2 in our case) were used to estimate the predicted B-spline.

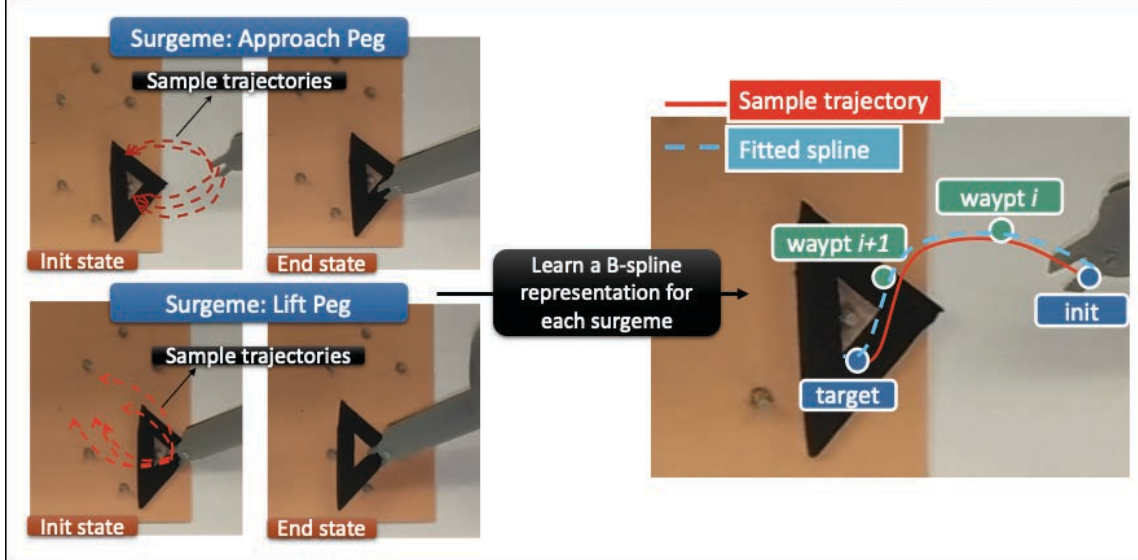


Figure 29. Left: The red dotted line shows different trajectory samples for the same surge. Right: The blue dotted line shows the fitted spline to the original trajectory (red). The models were trained to predict the waypoints (green) of the fitted spline using init and target.

The accuracy metric developed by Quiao et. al (2015) [2]. was used to measure the efficacy of the RL against the NN for estimating the model Φ . This metric compares two aligned trajectories in discrete manner. Each point f_i of the fitted spline is compared to it's pair in the predicted spline p_i . If the euclidean distance between the points is larger than a threshold θ , then we add a score of zero otherwise we add a score of 1:

$$S(f_i, p_i) = \begin{cases} 1 & \text{if } \text{eucl}(f_i, p_i) < \theta \\ 0 & \text{if } \text{eucl}(f_i, p_i) > \theta \end{cases}$$

The accuracy is defined in the equation below, where N is the number of points in the fitted or predicted trajectory (both curves use the same number of sample points).

$$Acc = \frac{\sum_{i=1}^N S(f_i, p_i)}{N}$$

Table 3. Shows the accuracy for the Linear Regression and Neural Network for two thresholds: 20 and 30 mm. The results show that the Linear Regression has a higher performance for the prediction of the surge trajectory. In the case of Grasp, lift, and insert, the Linear regression showed perfect accuracy. The RL was not successful at learning the transfer surges. In this case, the NN model performed better. Nevertheless, the transfer accuracies were much lower when compared to the other surges. The reason for this most likely lies in the nature of the data. The Get together and /transfer surges had a very high variance and the trajectories were extremely complex See Figure 9. This data might not be suitable for a trajectory-based approach.

A preliminary execution test was done using the offline vision system. Twenty peg transfers were performed with the modeled approach and another 20 with the Learned approach. For the last approach RL was used for all of the surges except for Get together and transfer. The results are presented in table 4. Both systems perform comparably well for grasp, lift,

and insert. The Modeled approach shows a visible advantage over the learned approach for the transfer action.

Table 5. Accuracy using RL and NN for different Thresholds

MODEL	Approach Peg	Grasp Peg	Lift Peg	Get Together	Transfer	Approach Pole	Insert	Threshold
RL	0.73	1	1	0	0	0.91	1	20mm
NN	0.09	0.8	0.55	0	0	0.09	0	
RL	0.91	1	1	0.17	0.5	0.91	1	30mm
NN	0.27	0.8	0.64	0.17	0.67	0.36	0.45	

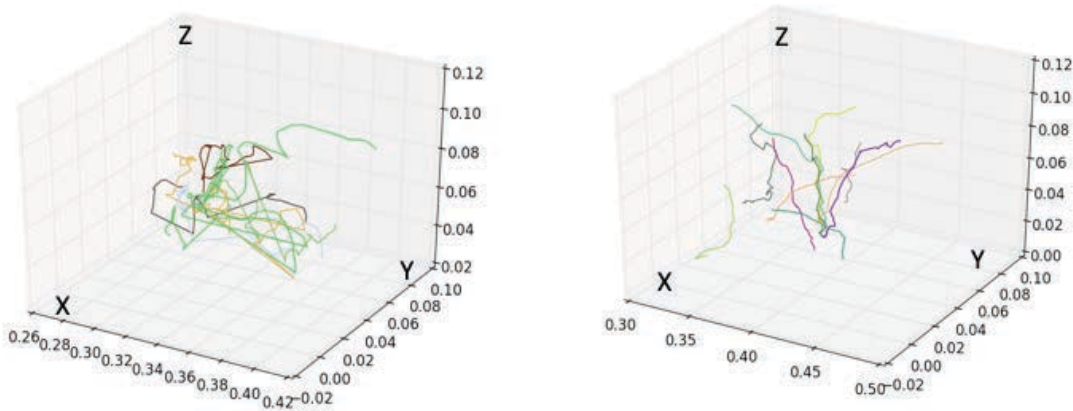


Figure 30. Left: Data samples for the Get Together surgeme. Right: Data samples for the Approach surgeme. Each data sample is plotted in a different color

Table 6. Performance for peg transfer for the Modeled and the Learned approaches.

Approach	N. Failed Grasps	N. Failed Inserts	N. Failed Transfers	N. Successful Tasks
Modeled	0	0	0	20
Learned	1	0	3	16

Autonomous Decision Module for Surgeme Execution

As mentioned above, there are two strategies for surgeme execution – model-based and learning-based. An autonomous selector has been developed to choose one of these execution modules for each surgeme command received. Each surgeme had 14 different features – the pose (position – x,y,z and orientation – roll, pitch, yaw) and gripper state (open/close) of each end-effector (left and right) – which have been recorded during data collection.

An unsupervised approach was used to cluster the surges into a strategy group. The purpose of this classification is to automatically choose which steps will be easy to learn and which steps will be more robustly performed through a modeled approach.

The samples were filtered based on feature variance and feature correlation. First, the average variance of each feature across all samples was computed along with their average correlation. Then only samples with average variance and correlation as features were selected. These samples were used as inputs to an unsupervised clustering module. Two methods were used: a) K-Means and b) Gaussian Mixture Models based. Both of them yielded comparable results. The results of the autonomous decision module are given below.

Table 7. Results of the Autonomous Decision Module for Surge Execution.

Cluster-1	Cluster-2
Approach (Peg)	Align and grasp (Peg)
Lift (Peg)	Insert (Pole)
Get together (Grippers)	Transfer (Peg)
Approach (Pole)	

Features in Cluster-1 have more variance in end-effector positions than other features and strong correlation between end-effector positions and time. They also have weak correlations between position and gripper state or position and orientation features, indicating that surges in Cluster-1 are ideal for learning/trajectory based execution. Cluster-2 surges have strong correlations between other combinations of features as well as high variance in multiple features than end-effector position alone, suggesting that they are not trajectory-dominated surges. They would be more suitable for model-based execution. The autonomous selector can be used to classify new unlabeled surges with the same 14 data features as mentioned above. This approach can be developed to include new execution strategies by employing more clusters.

Subtask 1.2 Transfer learning between robot systems

According to the specific Aim 2, the robot in the battlefield must learn to adapt to new conditions based on previously seen data. As a first approach, we implemented a system of transfer knowledge to classify surgical actions. In this system, the source domain came from the simulated Taurus, and the target domains were the real Taurus and the YuMi robot. Thus, two different transfer procedures were considered, as depicted in Figure 7: a) from the Taurus simulator to the real Taurus and b) from the Taurus simulator to the YuMi. The simulated Taurus system was selected as the source domain because it is faster, more economical and its potential for producing samples automatically.

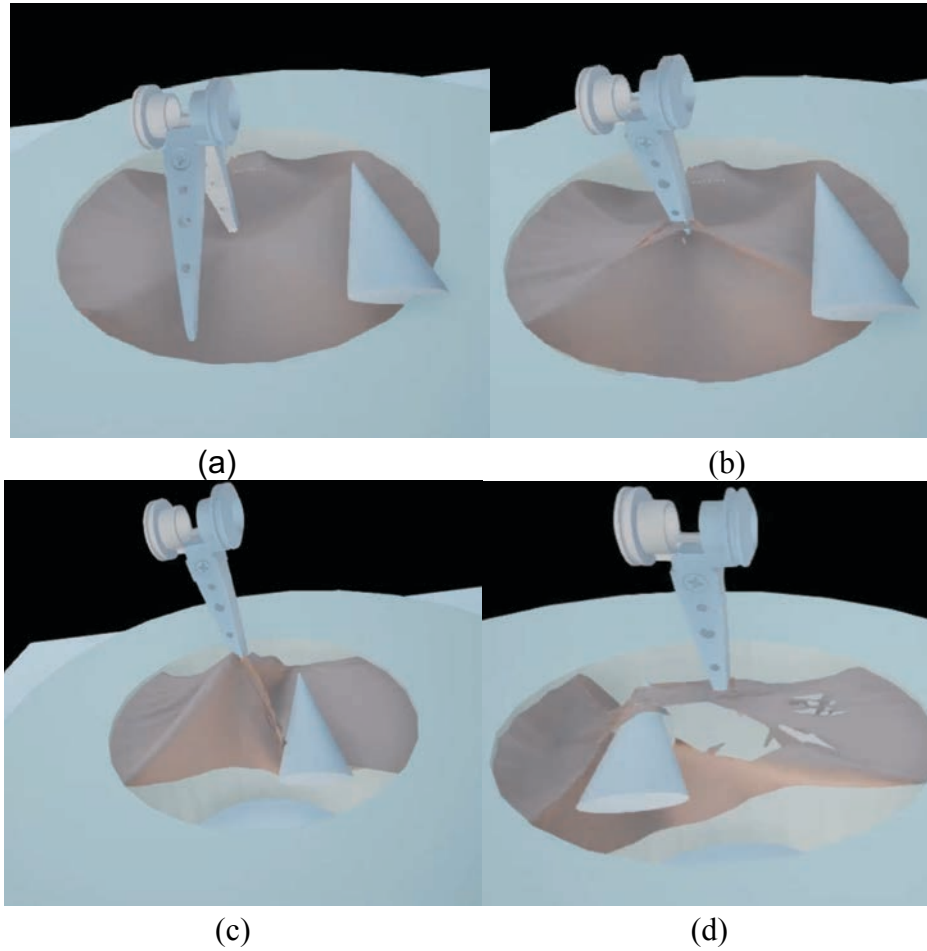


Figure 31. (a): The debridement setup showing gripper approach skin (b) “Grasp (skin)” surgical action. (c) “lift (skin)” surgical action, (d) “cut (skin)” surgical action as performed in the simulator.

The kinematic data collected from the simulator and the real robots were in different dimensions, as shown in the feature Table 3. The first step in the pipeline was to ensure that the dimensions of the data were equivalent in the three domains. Hence, we only considered the features that were commonly shared between the three domains: position of the end effector (x,y,z in robotspace), orientation of the end effector (pitch, yaw and roll) and status of the gripper (how open is it). Overall, we utilized 14 features per frame (seven features in each arm).

Table 8. Kinematic features available for each Robot platform. Kinematic variables. Note that ts is the Unix times-tamp, $\vec{J}$ is the vector of joint angles, $\vec{p}$ is the position vector (x, y and z), $\vec{\theta}$ be the Euler angles (*yaw*, *pitch*, and *roll*), gs is the gripper state of the end-effector and R be the 3×3 rotation matrix.

Taurus		Taurus Simulator		YuMi	
ID	Variable	ID	Variable	ID	YuMi
1	ts	1	ts	1	ts
2-13	R and $\vec{p}$	2-4	$\vec{p}$	2-8	$\vec{J}$
	-	5-7	$\vec{\theta}$	9-11	$\vec{p}$
14-16	$\vec{p}$	8-14	$\vec{J}$	12-20	R
17	gs	15	gs	21	gs

Two sets of experiments were conducted for this subtask: 1. Train and test on the data obtained from the same domain (no-transfer scenario) and 2. Train on one domain and test on the other (domain-transfer scenario). Furthermore, we considered two kinds of classification strategies: 1. frame-wise and 2. sequence-wise. The former strategy associates each frame to a particular class label and treats each frame as a sample point (*frame-wise instances*), while the latter method considers the entire surgical action (sequence of frames) as a single sample point (*sequence-wise instances*). Next, we used three supervised learning methods for our experiments: 1. Support Vector Machines (SVM), 2. Random Forest (RF) and 3. Multi-layer Perceptron (MLP). These three learning techniques are commonly used in the machine learning community for creating the baselines for classification. We used the scikit-learn [1] implementation of these models for our experiments.

SVMs and MLPs require the same number of features in each instance. Nevertheless, each surgical action had a variable number of frames. To deal with this issue, we re-sampled (via linear interpolation) the original instances to a fixed number of frames to generate *sequence-wise* instances. Next, we concatenated the seven features corresponding to each frame, for both the arms, to create a single feature vector. In our case, we set the number of frames to 40 and each *sequence-wise* instance is a 560-dimensional vector ($40 \times 7 \times 2$). In our experiments, we did not differentiate the surgen instances based on how well they were performed (success or failure). In addition, the following hyperparameters were used with the learning models: For the SVM, a linear classifier was used. For RF, we set $n_{estimators} = 200$ (number of trees in the forest), and maximum depth = 10. For the MLP, we used a hidden layer of size = 100, tanh as the activation function with *adam* as the optimizer.

Experiment set 1: No-Transfer (Surgical action classification)

In the first experiment, our goal was to study the classification in the *no-transfer* scenario where the learning model is trained and tested on the data coming from the same domain. In other words, the training and testing data followed the same distribution. We performed two sets of experiments: train and test on 1. simulator data ($S \rightarrow S$) and 2. real robot data ($R \rightarrow$

R). In our setup, we used a 60-40% split for training and testing respectively. Furthermore, we used a five-fold cross validation approach to tune the learning parameters (or weights) of the models.

Next, we performed the classification on both the *frame-wise* and *sequence-wise* instances, as shown in the Table 4. The *sequence-wise* performed generally better than the frame wise features, with an exception of the RF and the MLP in the Taurus robot. We conducted a two-sided paired t-test and found that SVM significantly outperforms the RF and MLP with the *sequence-wise* features ($p < 0.05$). The reason for the overall better performance of the *sequence-wise* features is that they have embedded temporal information. Therefore, we used these features for the experiments associated with the domain transfer.

Table 9. Classification accuracy on the no-transfer scenario for both the frame-wise and sequence-wise features. S is the Taurus simulator, R_1 is Taurus robot, and R_2 is the YuMi robot.

Sequence-wise				Frame-wise		
	RF	SVM	MLP	RF	SVM	MLP
$S \rightarrow S$	88 ± 2	87 ± 1	78 ± 4	86 ± 0	58 ± 1	73 ± 1
$R_1 \rightarrow R_1$	94 ± 2	92 ± 1	92 ± 2	95 ± 0	60 ± 0	92 ± 1
$R_2 \rightarrow R_2$	91 ± 1	93 ± 1	95 ± 1	88 ± 1	48 ± 1	86 ± 1

Experiment 2: Knowledge transfer for Surgical action classification

In this experiment, our goal was to build a learning model that was trained on the data obtained from one domain (source domain) and tested on the data coming from a new domain (target domain). In other words, the input data distribution is different in the train (source) and test (target) datasets. We start with the assumption that it is much easier to collect the data from the simulator when compared to the data from the physical surgical robot. Therefore, the real robot's data is assumed to be limited or not available in the extreme cases. Hence, we trained our models on the simulator data with a very little or no data from the real robot and test this model on the real robot's data.

To simulate the limited availability of the real data, we added a small fraction (α) of the real data into the training, where α was varied from zero to one. The value of α is defined as the ratio of number of examples of the real data to the simulator data present in the training. An $\alpha = 0$ indicates *complete transfer*, where there is no real data present in the training, while $\alpha = 1$ implies that the data from the simulator and real data are in equal proportions.

Moreover, *accuracy* (γ - the percentage of test examples that are correctly classified) and confusion matrices are used as metrics to evaluate how well the model is performing on the target domain. Let us define $\gamma^M_{A \rightarrow B}$ as the test accuracy for the model M trained on the source domain A and tested on the on the target domain B . Additionally, $\gamma^M_{A \rightarrow B}$ is a function of α .

Let C be a combination of the data from the simulator and the real robots, where the distribution of real data changes with the value of α . The goal this study is to determine the behavior of $\gamma_{C \rightarrow R}^M$, where the source domain is a combination of the simulated and real data and the target is only the real data) and $\gamma_{R \rightarrow R}^M$, where the source domain is a combination of the YuMi and Taurus data and the target domain is the YuMi data only.

Figure 32. shows the behavior of $\gamma_{R \rightarrow R}^{SVM}(\alpha)$ and $\gamma_{C \rightarrow R}^{SVM}(\alpha)$ for the SVM classifier. In the extreme case when there is no real data in the training process (i.e. $\alpha = 0$ – *completetransfer*), the accuracy is approximately 55%. As the value of α gradually increases from 0 to 0.2, the value of $\gamma_{C \rightarrow R}^{SVM}(\alpha)$ increases and converges in comparison to $\gamma_{R \rightarrow R}^{SVM}(\alpha)$. At $\alpha = 0.18$, the values of $\gamma_{C \rightarrow R}^{SVM}(\alpha)$ and $\gamma_{R \rightarrow R}^{SVM}(\alpha)$ are approximately equal to 80%. Moreover, this plot shows that adding the simulator data into the training procedure helps improve the accuracies considerably on the real data.

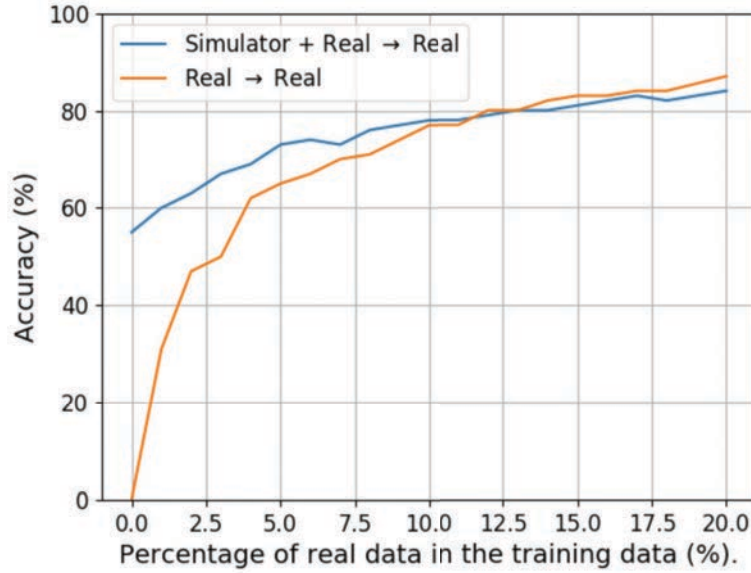


Figure. 32: Performance comparison for transfer learning from Taurus simulator to Taurus Robot using SVM. The blue curve indicates the transfer accuracy and the blue curve indicates accuracy on real data without transfer.

Table 5 shows the transfer accuracies obtained using three learning models for a range of values of α . Note that there are 1286 samples in the simulator domain. The second column (#) is the number of examples of the real data present in the training procedure. In the first row, it was assumed that there were no real data examples in the training, hence accuracy in $R \rightarrow R$ is 0. The confusion matrix presented in Figure 9 shows the transfer accuracies for all the classes at $\alpha = 0.05$. Overall, Table 5 shows that it is beneficial to augment the training data with the samples obtained from robotic simulators. Irrespective of the learning model, it helps greatly enhance the performance of those models on the real robot's data.

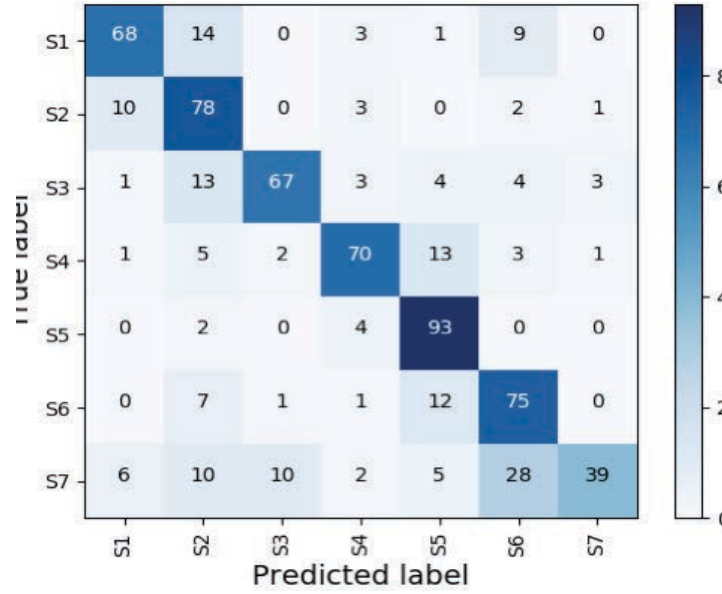


Figure 33. The confusion matrix for transfer learning with SVM using $\alpha = 0.05$

Table 10. Domain transfer accuracy when the models are trained on the domain C but tested on domain R . Note that the domain R is the real Taurus domain C is the combination of S and R .

		RF		SVM		MLP	
α %	#	$C \rightarrow R$	$R \rightarrow R$	$C \rightarrow R$	$R \rightarrow R$	$C \rightarrow R$	$R \rightarrow R$
0	0	34 ± 3	0 ± 0	55 ± 2	0 ± 0	40 ± 2	0 ± 0
3	24	53 ± 3	39 ± 4	67 ± 2	50 ± 5	57 ± 3	51 ± 4
6	49	66 ± 1	64 ± 4	74 ± 0	67 ± 3	68 ± 4	72 ± 2
9	73	77 ± 3	73 ± 5	77 ± 1	74 ± 3	74 ± 1	80 ± 3
12	98	78 ± 1	80 ± 3	79 ± 2	80 ± 1	78 ± 1	85 ± 1
15	123	84 ± 2	84 ± 2	81 ± 1	83 ± 0	82 ± 2	86 ± 1
18	147	85 ± 0	85 ± 0	82 ± 1	84 ± 1	79 ± 3	88 ± 1

In the last experiment, our goal is to verify the transfer across the robots i.e. to train the model on the simulation data of Taurus robot and test on the data of YuMi robot. The workspace, morphology and dimension of the data of these robots are significantly different. Hence, we used the principal component analysis (PCA) to rank the features based on the Eigenvalues and transform the data into the Eigenspace. Next, we chose a fixed number of features (a value of 8 is used in our experiments) in the Eigenspace in order to unify the feature dimensions. We followed a similar experimental design to test the transfer of knowledge from Taurus simulator to YuMi domain. The accuracy of complete transfer scenario is close to the random accuracy indicating that the model is overfitting to the simulator as there is no data from the real robot in the training. However, a slight increase in the number of examples of YuMi data in the training, the transfer accuracies swiftly increase. For instance, when the

number of YuMi examples = 100, the transfer accuracy is approximately 80% as shown in the Figure 10. When we add more than 200 examples of YuMi in the training, the transfer accuracies marginally surpass the accuracies in $R \rightarrow R$ scenario. This shows that transferring the knowledge between two completely different robots is a challenging task and requires relatively more examples to achieve superior accuracies.

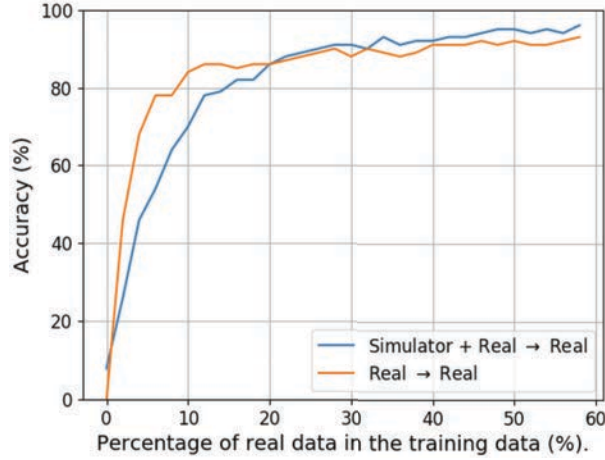


Figure.34: Performance comparison for transfer learning from the Taurus simulator to the YuMi Robot.

The transfer learning task considered in this task is an example of domain adaptation where the data distribution of the source and target domains are considerably different while the class labels remain the same. In our experiments, the source domain is nothing but the simulator where a large amount of data is available whereas the target domain is the real Taurus robot. The interaction modality of the Taurus robot and the simulated robot are completely dissimilar. Given that the configuration of the real Taurus robot and the simulator is similar, transferring the knowledge to the physical domain is relatively easier in the case of the real Taurus robot in comparison to the YuMi robot. Hence the transfer accuracy is significantly higher for the real Taurus. Furthermore, the transfer accuracy obtained on the YuMi robot increases with the increase in α and surpasses $R \rightarrow R$ scenario for $\alpha > 0.2$. In other words, the amount of real data needed to achieve the transfer accuracy of 80% is much higher for the YuMi ($\alpha > 0.3$) in comparison to real Taurus ($\alpha > 0.12$). The relatively lower transfer accuracies obtained on YuMi data show that the transfer from one robot to another robot is a challenging task. It is worth revisiting the process of creating the sequence-wise features. Given that the surgeme instances are of different length, we used linear interpolation to ensure that the instances are of uniform length. This method of interpolation, elongates the surgical actions that are shorter and compresses the ones that are longer. It is interesting to study the effect of re-sampling on the transfer accuracy. Nevertheless, the issues associated with re-sampling can be eliminated by using the deep models that are specialized for the

problems with non-uniform features lengths such as Long-Short-Term-Memory models or conditional random fields. Furthermore, we mentioned in the results that the sequence-wise features provide significantly better accuracies in comparison to the frame-wise features for the transfer learning tasks. Note that the sequence classification approach assumes that the test data is annotated beforehand with respect to the start and end of the surgical actions. However, this assumption is not valid when we want to deploy these trained classifiers in real-time as the real-time data cannot be segmented beforehand. In contrast, the frame-wise classification approach does not require the surgical actions to be segmented. Hence the frame-wise classifier can potentially act as the model that can be used to know the start and the ending point of the surgical actions.

We develop a methodology that uses compact image representations with kinematic features for surgeme recognition in the DESK dataset (Taurus II, Taurus II simulator, and YuMi robots data). This dataset offers samples for surgical procedures over different robotic platforms with high variability in the setup. We performed surgeme classification in two setups: 1) No transfer, 2) Transfer from a simulated scenario to two real deployable robots. Then, the results were compared with recognition accuracies using only kinematic data with the same experimental setup. The results show that our approach improves the recognition performance over kinematic data across different domains.

Here we discuss the details procedure and experimental results.

Methodology:

Our approach combines features extracted from the video image and kinematic data to perform transfer learning in surgeme recognition. The approach is based on robot agnostic feature extraction that can be applied to different environments in a transfer learning scenario.

An overview of our system architecture is given in Figure 10. We used a Convolutional Neural Network (CNN) to extract meaningful features from video images. The images extracted from the videos were resized from 1920×1080 to a 228×128 . We employed the CNN model ResNet18 [3] that is pre-trained on ImageNet and removed the last two layers (i.e., fully connected and softmax) to create a high dimensional feature representation for each video frame. The extracted features have dimensions of $1 \times 512 \times 4 \times 8$, which were flattened to a single vector of size $N = 16384$. Due to the large dimension in the output of the ResNet18 model a dimensional reduction module was required to reduce each image/frame features to a lower-dimensional vector ($M = 30$, where $M \ll N$). Principal Component Analysis (PCA) was used to reduce the dimension of each frame image to a lower 30-dimensional vector.

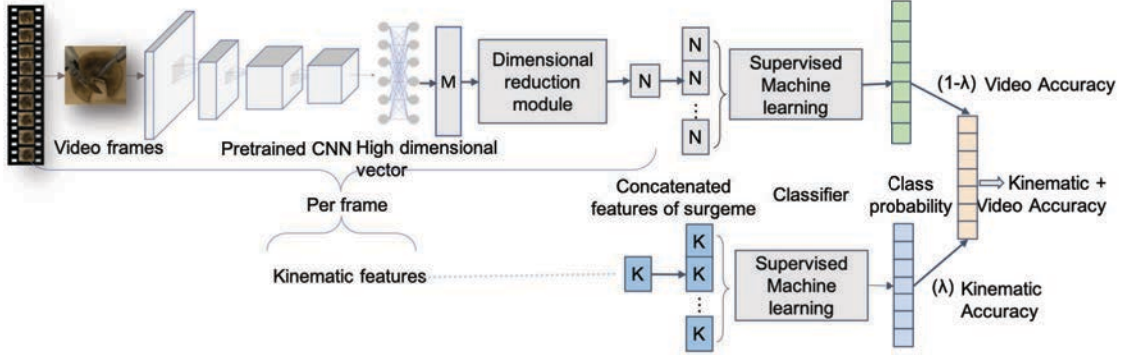


Fig. 35: Architecture overview of the approach, M: Number of dimensional features extracted using pre-trained CNN, K: number of kinematic features, N: number of image features after Dimensional reduction module.

In parallel, we extracted features from kinematic data by reducing the kinematic features from multiple domains to the commonly shared features: position, orientation and gripper status of the end-effector (14 features: seven features in each arm). Since each surgeme instance consists of a variable number of frames, we re-sampled (via linear interpolation) the original instances to a fixed number of frames (40) to generate a sequential feature instance for each surgeme. In particular, for the kinematics, we concatenated the 14 features corresponding to each frame, to create a single 560-dimensional vector per surgeme ($40 \times 7 \times 2$). On the other hand, for the video data, we concatenated the 30 features obtained after the PCA reduction into a single feature vector of 1200 dimensions (40×30) that represents each surgeme. Another PCA reduction was applied on the 1200-dimensional feature vector to obtain a 100-dimensional feature vector for the entire image sequence of 40 frames.

In our pipeline, visual features and kinematic features came from different distributions. Thus, to take advantage of both visual and kinematic modalities, we trained the supervised learning algorithms separately and combined their output class probabilities as shown in the following equation:

$$P(C) = \lambda P_{kin}(C) + (1 - \lambda) P_{video}(C),$$

where C is the class and $\{P_{kin}(C), P_{video}(C)\}$ are the probabilities given by supervised learning models using kinematic and video features respectively. Here, λ is the hyperparameter (from 0 to 1), which allows us to modify the contribution of the models based on kinematics or video frames. When λ is set to 0, the classification is based on video data alone. If $\lambda = 1$, the classification uses only kinematic data. Note that our architecture allows us to plug in separate supervised machine learning model for kinematics and video. Thus, the models for video and kinematics can be designed separately. For simplicity, in our experiments, we use the same supervised learning model for both video and kinematic features.

The image features obtained from the pre-trained ResNet18 and the extracted kinematic features are agnostic to the robot system and the surgical setup. This means we get the same feature set, even when the actions or the robot varies. Having a common set of features facilitates surgeme classification across different domains, since it helps models to leverage on the information coming from another domain.

Experimental Setup:

The proposed architecture was tested over two scenarios: no-transfer scenario (train and test data are obtained from the same domain), and domain-transfer scenario (train on one domain and test on the other). The complete domain transfer scenario used the Taurus simulator (S) data or a combination of simulation and real robot data (S+R) during training, while the algorithm was tested entirely on the real robot data (R). A value α represents the percentage of real data (i.e., target domain) that was added to the simulator data (source domain) for the model training. The rest of the real data (in the target domain) was used as the testing set. Thus, $\alpha = 0$ indicates a complete transfer. Two supervised learning methods were tested: (1) Random Forest (RF), and (2) Support Vector Machines (SVM), using the scikit-learn [4] implementation. A five-fold cross-validation approach was used with a data split of 80-20% for training and testing respectively. Furthermore, hyperparameter setting, we used a *kernel*="poly" for SVM classifier. For RF, we set n estimators = 200 (number of trees in the forest), and maximum depth = 10. We set the combined model's hyper parameter λ to 0.5 for RF and to 0.8 for SVM (empirical best).

Results:

Table 6 shows the surgeme recognition results using our approach for kinematics, video information and a combination of both modalities for the non-transfer scenario. Kinematic (Kin) features alone perform substantially better than features based on video frames (Visual). RF performs better than SVM for visual data achieving up to 63% accuracy on the Taurus real robot. Moreover, in most of the cases, accuracies improved when both kinematic data and image features are used (Kin+Visual). In general, adding visual data improved the classification accuracy of surgemes compared to the use of kinematic or visual data independently.

Two transfer learning scenarios were evaluated: (i) Taurus Simulator to Taurus real robot (Sim2Taurus) and (ii) Taurus Simulator to YuMi (Sim2YuMi). The accuracy results for the domain-transfer scenario (Sim2Taurus): simulator + real-robot (S+R) are presented in Figure 11. The results show that kinematic+visual features surpass the accuracy and consistently outperforms results that use only kinematic features.

Table 11. Classification accuracy on the no-transfer scenario using RF and SVM on kinematic only (Kin), visual only (Visual), and both (Kin+Visual) features. In most cases adding visual features with kinematic features improve the accuracy.

	RF			SVM		
Robot	Kin	Visual	Kin+Visual	Kin	Visual	Kin+Visual
Simulator (Taurus)	92.69	56.46	93.03	91.66	33.16	92.51

Taurus	94.68	63.77	96.49	91.43	35.19	92.73
YuMi	96.87	60.65	97.23	92.90	21.91	93.38

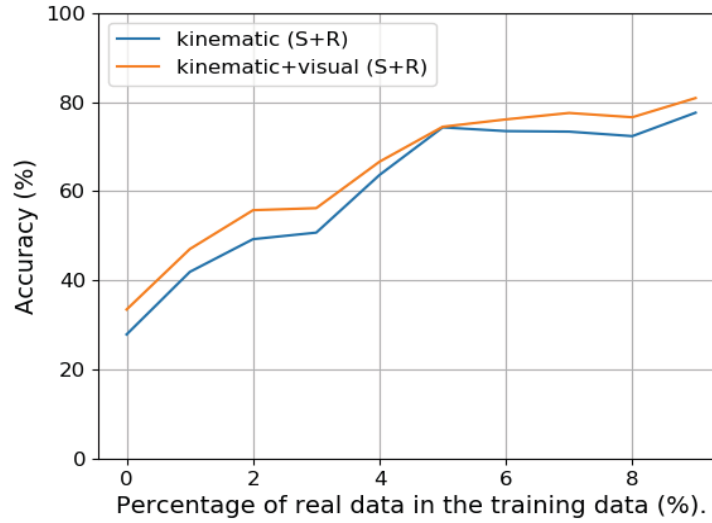


Fig. 36: Performance comparison for transfer learning from the Taurus Simulator to the Taurus Real Robot (Random Forest). Overall transfer accuracy improved quickly and it reaches over 93% with only 50% real data in the training. Details in Table 12.

Table 12. Domain transfer accuracy (Random Forest) when the models are trained on the Taurus simulator robot and tested on real robots (Taurus (Sim2Taurus) and YuMi (Sim2Yumi)). α is the percentage of real-robot data used in the training. Features: kinematic only (Kin), visual only (Visual), and both (Kin+Visual).

	Sim2Taurus			Sim2Yumi		
$\alpha(\%)$	Kin	Kin+Visual	Gain(%)	Kin	Kin+Visual	Gain(%)
0	27.92	33.51	20.00	14.20	14.80	4.24
1	41.94	47.05	12.19	25.64	33.78	31.75
2	49.27	55.22	13.17	37.42	43.07	15.08
3	50.74	56.22	10.82	39.03	46.84	20.00
4	63.65	66.62	4.67	50.38	54.14	7.46
10	78.93	82.54	4.57	74.73	73.40	-1.78

50	93.77	94.03	0.28	88.70	90.14	1.62
90	93.51	96.10	2.77	95.24	96.43	1.25

Figure 12 shows that the performance of both the (Visual) and (Kin+Visual) data increases as we add more real-robot data (YuMi) in the training (S+R). The improvement obtained for the transfer learning scenario using (Kin+Visual) data is higher than knowledge transfer based on kinematic data alone. Hence, when training data is limited on the target domain, the addition of visual data from simulation could help boost initial classification accuracy.

Table 12 shows transfer accuracy for kinematic only features (Kin) , and both kinematic and visual features (Kin+Visual). It also shows the percentage gain which is calculated as

$$Gain = \frac{A_2 - A_1}{A_1} \times 100,$$

where A1 and A2 are the accuracies using Kin features, and Kin+Visual features, respectively.

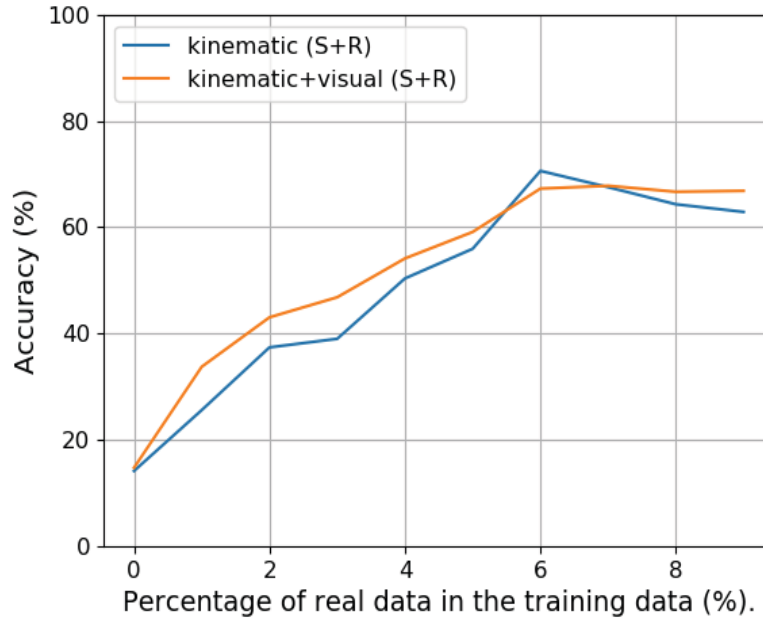


Figure. 37: Performance comparison for transfer learning from the Taurus Simulator to the YuMi Real Robot (Random Forest). Overall transfer accuracy improved quickly and it reaches over 88% with only 50% real data in the training. Details in Table IV.

It can be observed that adding visual features boosts the transfer accuracy in both Sim2Taurus and Sim2YuMi robot settings achieving up to 20% and 31% gain respectively. Another

observation is that the performance improvement is generally higher for smaller α values. Intuitively, when the target domain data is limited during training, visual information can play an important role to boost up initial transfer accuracy. As more target domain data is added to the training, both Kin and Kin+Visual accuracy increase quickly. In particular, when only 50% data is added to the simulator training, the accuracy reaches over 93% and 88% for Sim2Taurus and Sim2YuMi respectively.

Subtask 1.3 Library generation

Generating surgeime features

Once the surgeimes have been classified using kinematic feature data, we generated the same kinematic feature data using surgeime labels and other key information in an inverse mapping as well as continuous value predicting architecture. The other information that was taken are the starting and ending point of the end effector at the beginning and end of each surgeime but can be taken as current robot end-effector position and object (peg and pole) position data involved in the corresponding transfer respectively in further stages. The data currently used was a simplified version of the actual peg-and-pole task. We used a Recurrent Neural Network based one-to-many mapping procedure to generate sequential or serial data from a single label.

Recurrent Neural Networks (RNNs) are a very robust and powerful type of neural networks and are the type of networks that hold internal memory. Because of the internal memory, RNN's can remember very important feature information from the inputs given and make precise predictions with regards to the output. Especially in cases of sequential data, RNN's are widely used to predict the outcomes over timesteps. Sequential data is basically ordered data, where related things follow each other. The most popular type of sequential data is time series data which is a series of data points that are listed in discrete times in their respective order. Our dataset is also time series data. Each surgeime happens over a sequence of timesteps, the length of which can vary from one execution or task to another. For our experiments, this length was fixed to be the one that was closest to the length of maximum number of surgeime records. Every surgeime sequence was downsampled to this fixed length by taking data within the sequence at evenly distributed timesteps. In case, upsampling was required, last time step data within the sequence was repeated the required number of times.

In a regular feed forward neural network information moves from neuron to neuron. Furthermore, this information moves only in one direction i.e. information moves from the input layer to the hidden layer/layers to the output layer. During this information movement process no neuron or node is visited twice during the feed forward path. This network can be used to predict a certain label or quantity, but it cannot predict information over time because feed forward networks have no memory component. The network itself has no concept of time embedded in it. It is only the training process that ensures the optimization and storage of weights, but this cannot help us predict or generate a sequential set of data.

In RNN, the information cycles through a loop. The network takes into consideration the current input and also what it has learned from the inputs it received previously. Therefore, a Recurrent Neural Network has two inputs, the present and the recent past. This is important because the sequence of data contains crucial information about what is coming next, which is why an RNN can do things other algorithms can't. Also note that while Feed-Forward Neural

Networks map one input to one output, RNN's can map one to many, many to many and many to one. One-to-many mapping involves taking a label, which might contain one or more features repeated over a certain number of timesteps so that we can define the time dimension of the output data, and generating sequential output data which again might contain one or several features over the same number of timesteps as the input data. The input features we have used here as mentioned above are the surge number, x,y,z coordinates of the end-effector at the start and end of the surge, making a total of 7 surges. The output data contains x,y,z coordinates, yaw, pitch, roll and gripper state of the end-effector at each timestep during the surge, making a total of 7 output features as well, all of which are continuous values.

Our current RNN architecture uses 4 layers with 2 LSTM hidden layers and 1 input Time-Distributed layer and 1 output Time-Distributed layer. Long Short-Term Memory (LSTM) networks are an extension for recurrent neural networks, which basically extend their memory. Therefore, it is well suited to learn from important experiences that have very long -time lags in between. The units of an LSTM are used as building units for the layers of an RNN, which is then often called an LSTM network. LSTM's enable RNN's to remember their inputs over a long period of time. This memory can be seen as a gated cell, where cells decide whether or not to store or delete information (e.g. if it opens the gates or not), based on the importance it assigns to the information. This assignment happens through weights, which are also learned by the algorithm. This simply means that it learns over time which information is important and which one not. The time-Distributed layer applies an activation function (through Dense layer) to each temporal slice or timestep of the input. Each of the first 3 layers use 100 neurons each and the output layer uses 7, which is equal to the output feature dimension. The activation used in the output layer was linear or identity function to retain the continuous value outputs. The weights are adapted to this one-to-many mapping for the given data for sequential data generation using some backpropagation algorithm. The Adam version of Stochastic Gradient Descent was used for optimization while Mean Squared Error was used as the loss calculation function.

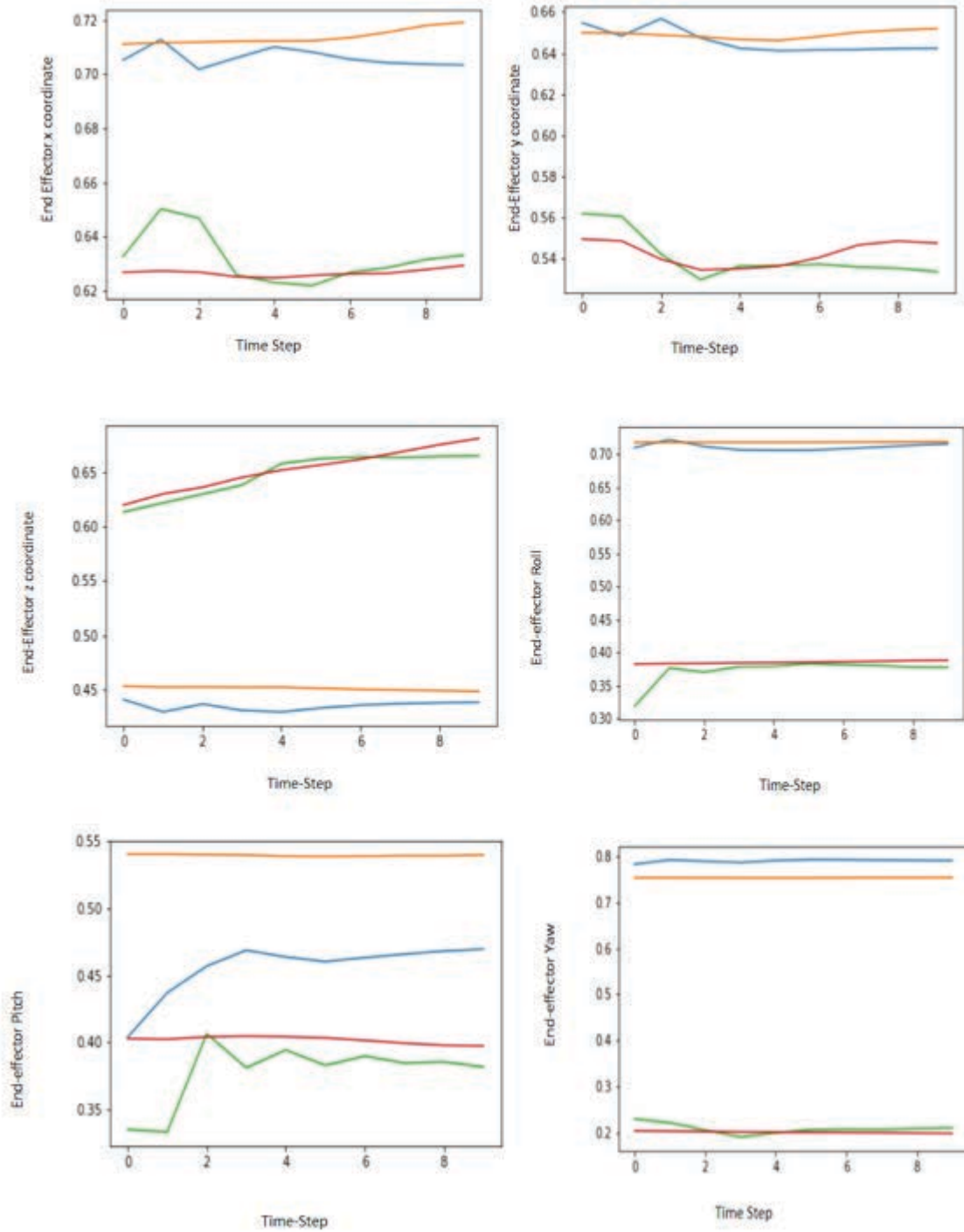


Figure 38. Actual and predicted values of features of surge 6 with the shown starting and end points. Orange and Blue line represent actual and predicted values for left arm whereas red and green represents them for right arm respectively.

The RNN was implemented using the Keras API which uses Tensorflow at its backend. Two separate networks with the same configuration were used for the left arm and the right arm of the robot. The model was trained and tested with normalized data from the Taurus simulator. The Keras implementation imposes a fixed length constraint on the sequences. Our implementation achieved an accuracy of around 98% on training dataset and an accuracy of around 95% in validation dataset. Figure 11 demonstrates the predicted and actual values of the end-effector position and orientation over the fixed number of timesteps for Surgeme 6 (align and place on pole) of a random sample from the taurus simulator dataset for both the arms.

Decoupling Visuo-motor Control in Simulation

The tasks we're planning on performing (suturing, debridement, staunching bleeding) require a safety-critical system. As such, it's not sufficient to have an end-to-end approach for handling the entire decision-making process. We want to be able to examine different stages and enable human reasoning. We propose to decouple the vision and control components into separate vision and control “blocks”. Training vision separately allow you to customize quickly for different environments. This modular architecture makes it possible to “plug-and-play” with different fine-tuned vision blocks for different environments and different control paradigms.

Vision Block

We will train a 3-layer convolutional neural network (CNN) to extract key state information from images: tool positions, needle tip positions, target positions. To start with, we predict the needle tip position from images in the Needle Master and dVSS simulated environments.

Experiment 1: Predicting needle tip pose in needle master

We use a 3-layer CNN to predict the pixel location and orientation of the needle from image frames. On an environment left out of training we predict the needle on average 12 pixels away from the true tip location. The histogram of Euclidean pixel error can be seen in Figure 13. We also show the true and predicted needle positions as depicted in Figure 12.

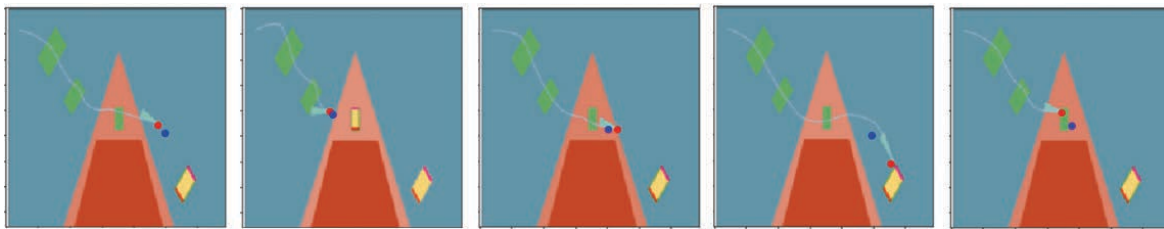


Figure 39. Frames from a Needle Master level left out of training. The red dot indicates the true needle position, the blue dot indicates the needle position predicted by the network.

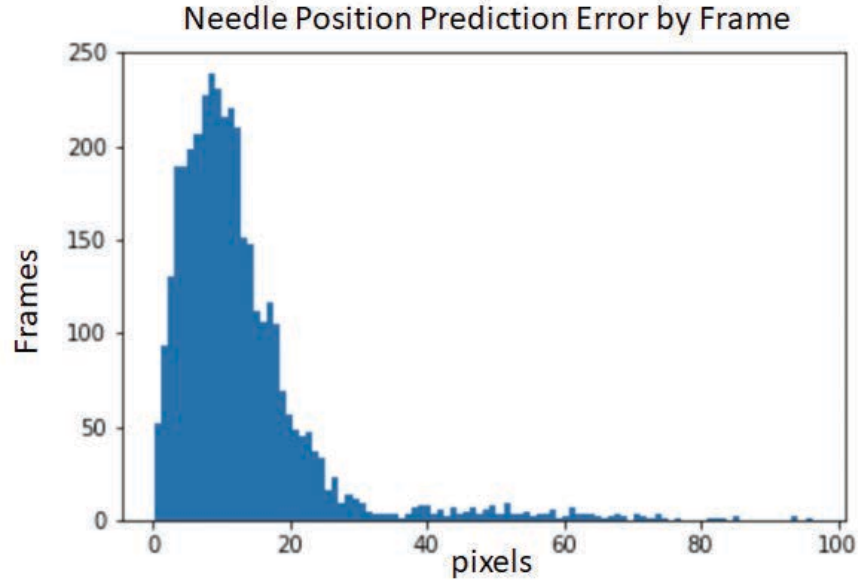


Figure 40. Needle position Euclidean error per frame.

Experiment 2: Predicting 3D needle position from monocular images

Next we train a 3-layer CNN to predict the 3D needle tip position from a monocular image. To predict a 3D position from a single image the network must leverage that the data shows many views of a static environment. Below we show the 3D target locations and the true and predicted needle positions. By projecting the needle predictions into the image view we can find the regions in the image where the network predicted the needle to be. Figure 14 (a) shows that in some cases the network successfully finds the needle in the image but it naturally has some error in predicting the actual depth of the needle. In (b) we see that sometimes the network incorrectly picks up on features on the tool. A histogram of needle position errors by axis is shown below (see Figure 15). We suspect that the poorer performance along the z-axis is due to lack of depth information.

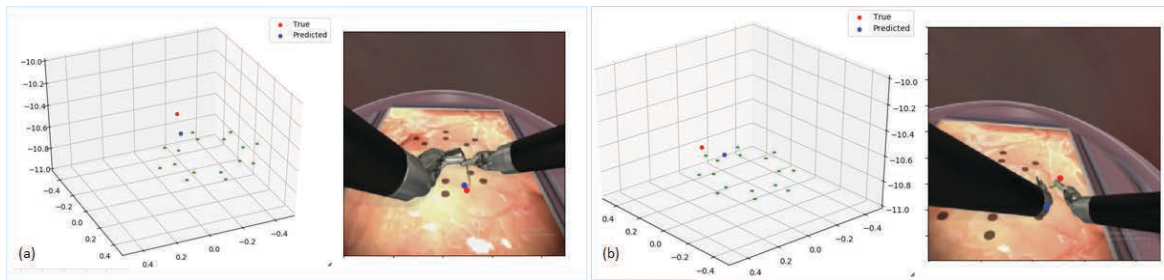


Figure 41. Two examples (a, b) of 3D needle positions predicted from image. Left: plot with target points in green, true needle position in red, predicted needle position in 3D simulated coordinates. Right: true and predicted needle positions projected onto the images. In (a) the needle tip is predicted well in the image plane but has error in the z-prediction. There is more error in the image-plane prediction in (b).

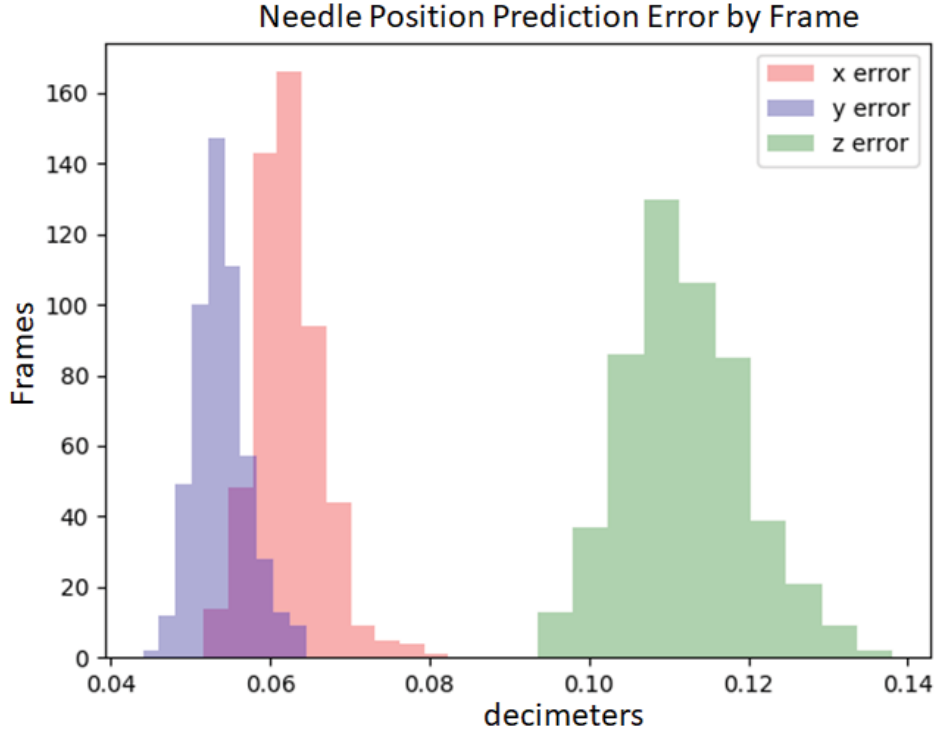


Figure 42. A histogram of distance between the true needle position and the predicted position by axis. For reference, the distance between corresponding targets on the inner and outer circles is between 0.06 and 0.09 decimeters. The diameter of each target is 0.015 decimeters.

Generating actions trajectories from PID Control

Since our plan is to generate robotic control from state, as provided by the vision block, it would be useful to guide the reinforcement learning module in its exploration rather than relying on random exploration, which is unpredictable and intractable for large spaces. We use a PID controller to augment our expert demonstrations, and to plot out a trajectory for simple actions. This controller has already been shown to work on NeedleMaster (see Figure 16), with the caveat that it does not account for tissue dynamics. The plan is therefore to plot a rough course with a mix of controller and demonstrations, and to then let the reinforcement learning algorithm optimize the trajectory further.

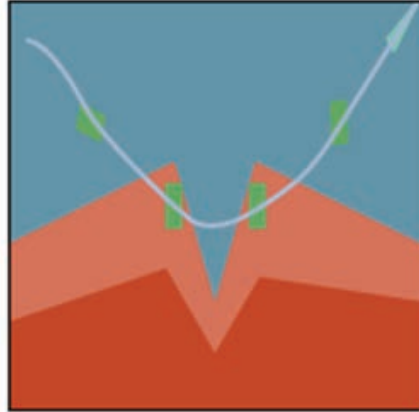


Figure 43. Needle threading in the Needle master simulator

Generating action trajectories using Reinforcement Learning

While the main da Vinci simulator is in development, we attempted to learn resilient trajectories in a simplified 2-dimensional environment called NeedleMaster (see Figure 44). Reinforcement learning tends to perform poorly even in such a simplified environment due to its large dimensionality, limited variability, and minimal rewards relative to other RL environments. Using state-of-the-art Q-learning algorithms has not proven particularly successful, even with reward shaping. DDPG [5], on the other hand, seems to do better on this task, especially when guided by a control module.



Figure 44. Needemaster simulator

Task 2: Semantic Rules Creation

Creation of Terminal Symbols and Embedding: We define the task of transferring peg between poles as a sequence of surgeme actions. Given a source and destination pole, the action starts with approaching toward the source pole (S1). The next movement for the robot arm is to align & grasp (S2) the peg. Then the robot arm lifts the peg (S3) and initiates transferring (S4, S5) process. In the next step, the robot arm with the peg approaches towards the target pole (S6) and finally finish the task by placing it to target pole (S7). In addition to that, we consider all the surgeme actions as successful and fail as in the following table.

ID	Surgeme	ID	Surgeme
S1	Approach the peg – Success	S8	Approach the peg – Failure
S2	Align & Grasp – Success	S9	Align & Grasp – Failure
S3	Lift the peg – Success	S10	Lift the peg – Failure
S4	Get together – Success	S11	Get together – Failure
S5	Exchange – Success	S12	Exchange – Failure
S6	Approach the pole - Success	S13	Approach the pole – Failure
S7	Align & Place - Success	S14	Align & Place – Failure

Figure 45. List of 14 surgemes concerned with the peg transfer task

We consider a task as complete when all the six steps are successful. In case of any failure, we keep repeating the surgeme action until we get a success or we reach a threshold (t). In case of unsuccessful task completion, we start over and start from the beginning.

Surgeme Event Embedding: To learn a surgeme event objective we leverage a Skip-gram function (Mikolov et al. 2013) as follows (Lee & Goldwasser 2018):

$$\begin{aligned}
 p(C(e)|e) &= \prod_{e' \in C(e)} p(e'|e) \\
 &= \prod_{e' \in C(e)} \frac{\exp(v_{e'} \cdot v_e)}{\sum_{e^* \in E} \exp(e^* \cdot e)}
 \end{aligned}$$

where

- e is the current surgeme event (e.g., S_3),
- $C(e)$ is the context of the surgeme event e . We define $C(e)$ as a predefined window size k . For example, if $k=3$ then the context of surgeme S_i is $S_{i-2}, S_{i-1}, S_i, S_{i+1}, S_{i+2}$.
- E is all event vocabularies which in our case S_1 through S_{12} and v_e is the vector representation of the event e

We represent each surgeme event e as a low dimensional vector (v_e). We leverage both kinematic (e.g., robot's arm position, gripper angle) and visual RGB image (e.g., pixel values) to construct vector v_e . In addition to that our surgeme data contains temporal information. Which means for each surgeme we have multiple movement primitives and sequence of images (i.e., short video) which corresponds to multiple vectors. We take an average of these vectors to get a single low-dimensional vector (v_e) for both kinematic and visual data.

In summary, we use representation-learning techniques to convert the information associated with the observed events into a semantic representation, such that similar actions can be identified. Particularly, we will embed this information in a low-dimensional vector space, by constructing a learning objective exploiting the structure of the procedure movements and their temporal relationships (Lee & Goldwasser 2018), as described in Figure 46.

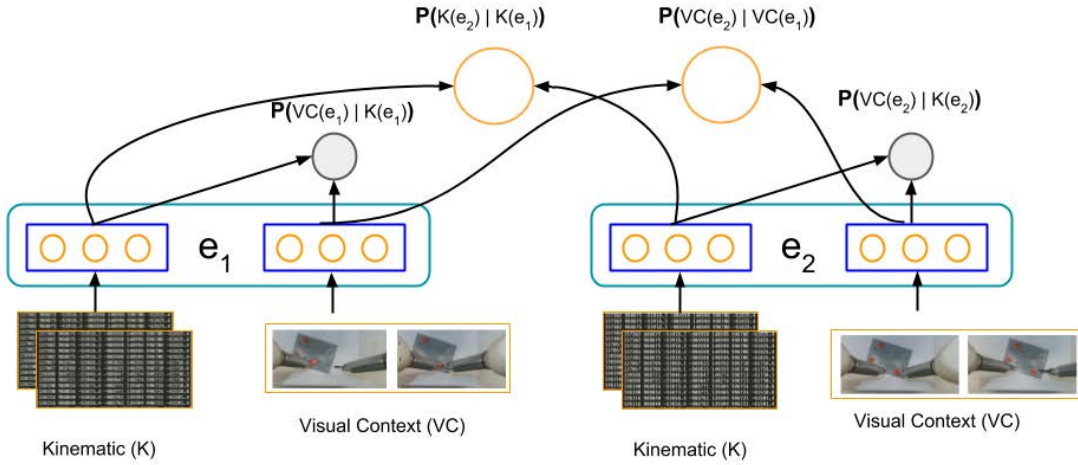


Figure 46. Representation-learning technique to convert the information associated with the observed events into a semantic representation

For both cases, we can learn this probability by minimizing the margin-based ranking loss as follows:

$$L_C(e) = \sum_{e' \in C(e)} \sum_{e^* \notin C(e)} \max(0, \delta - v_e \cdot v_{e'} + v_e \cdot v_{e^*})$$

where δ is the margin. In this case, the pair (e, e') is taken from the context thus can be regarded as a positive pair. Similarly, the pair (e, e^*) is taken outside the context thus can be regarded as a negative pair. Note that we have two objective functions: kinematic and visual image data. We can learn the joint embedding by taking a weighted summation of all the objectives as in the following equation:

$$\mathcal{L}(e) = \sum_{i \in \{\text{kinematic, visual}\}} \lambda_i L_i(e)$$

where λ is the hyperparameter which can be learned empirically.

Task 3. Learning with Imperfect and Randomly Delayed Information

In this task two different approaches to deal with delayed information were explored. The first one uses Reinforcement Learning to optimize a policy when the state is delayed. The second approach considers a scenario of surgeme classification where the information about the “ground truth” label of the previous surgemes and the current surgeme might delayed. The sections below explain both setups in detail.

Reinforcement Learning approach

Reinforcement learning (RL) is used in this work to learn the sequence of surgemes to be performed. RL is an approach through which machines can be trained like humans from actions taken and rewards associated to them. To train the robot, we need to define the current state, actions, and the rewards associated with them.

To construct the state space for learning algorithm we selected these parameters jointly to construct a state.

1. Last surgeme performed
2. Current joint angles of the robot
3. Current Coordinates and orientation of the robot's end effector
4. Current scene input from camera
5. Control instructions from teleoperator

Actions are defined as surgements to be performed. Whenever an action is performed, the state is updated via a Markov process and a reward is observed. However, in the current experiment, the observation of state update and rewards can be delayed. Rewards are a binary indicator whether the action taken was appropriate in current state or not. We further analyzed the effect of delays of observation in learning using an open-ai gym environment.

Experimental setup: Open AI gym provides fundamental environments to test functionalities of the learning algorithm. We implemented a Q-learning based algorithm for Reinforcement learning, and added random delays to update the rewards to learning.

The simulator runs the environment to control an inverted pole (See Figure 18). Each run of environment is termed as an epoch. In every run, the controller tries to take an action keeping the inverted pole stable, and observes a binary reward corresponding to event whether the pole is still stable or not. Maximum reward that can be obtained in each epoch is 200 units. The state is the current joint angle of the pole.

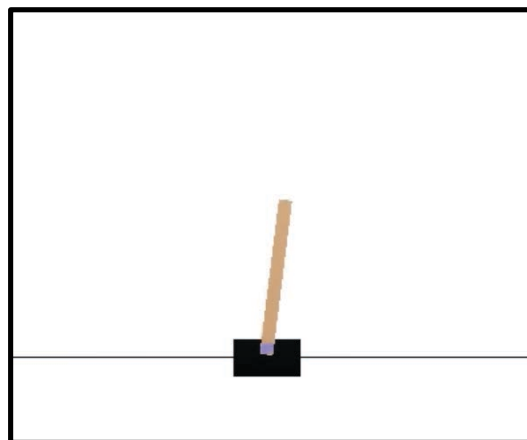


Figure 47. Simulated inverted pole control

After each action, a delay event occurs with a fixed probability. If the delay event occurs, the observation for that action will not be provided to the simulator, but stored in a buffer. If the delay event does not occur, observation for the current action and all the past actions which are still in the buffer will be fed to the simulator. The buffer is cleared after passing all the observations to simulator in case of non-occurrence of delay event. This model is similar to

jerky connections where incoming packet stream might be updated for a finite time, but all the packets arrive simultaneously after a certain delay.

We present the results (See Figure 19) for different values of probabilities (p_d) with which the delay event occurs, where $p_d = 0$ implies no delay, and $p_d = 1$ implies infinite delay where the robot is not able to learn, and performs random actions.

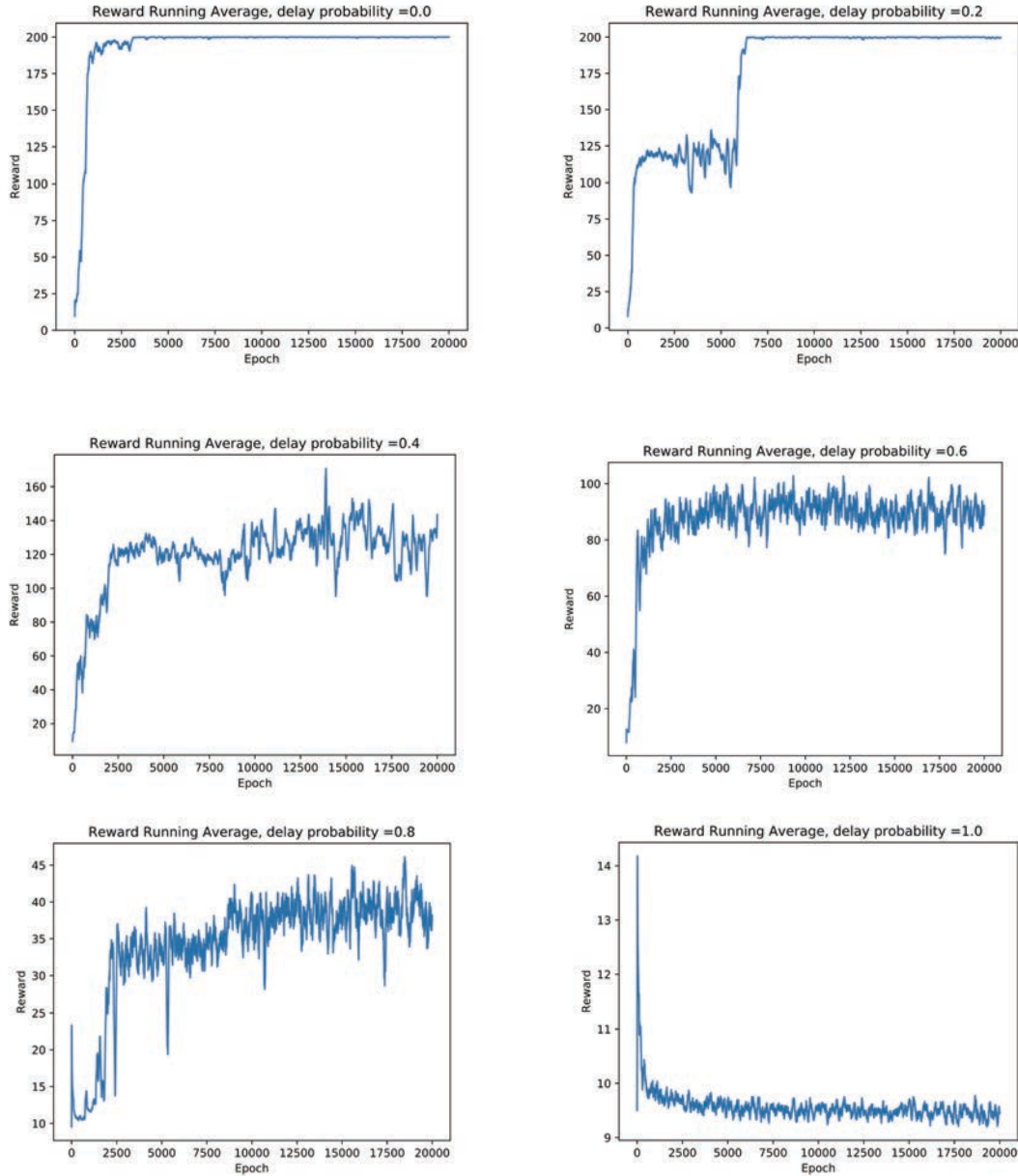


Figure 48. Cumulative reward per epoch for different probability delay values.

With increase in the delay probability, the average rewards decrease and they become less stable as the learning time remains constant. We further note that as p increases from 0 to 0.2,

the reward at the converged value is nearly the same. This indicates negligible loss in the reward obtained, while the time taken to converge to the reward is larger.

Delayed feature label approach

Experimental Setting: For each experiment, we split the data into training and testing. To emulate the real-world scenario, we maintain chronological (temporal) information. Thus from a data splitting point, we consider past data as training data and the rest as testing. We consider accuracy as the evaluation metric which is the ratio between the number of correct predictions and total testing instances. For our preliminary experiment, we use JIGSAWS suturing (Gao et al. 2014, Ahmidi et al. 2017) kinematic data. We leverage a multiclass SVM with ‘rbf’ kernel as our classifier.

Experiment 1: We split the data into three different ways as shown in the table. We evaluate each testing instance independently and calculate the accuracy leveraging the annotated JIGSAWS data. In this setting, we do not consider temporal or delay information. We observe that the classifier achieves the best result in the 90/10 (train/test) splitting with an accuracy of around 69%.

Experiment 2: Split data into train and test. Consider the previous step’s ground truth label as a feature in the training instance. The dataset splitting is similar to experiment 1. During training and testing, we consider the previous step’s ground truth label (not predicted) as an additional feature. In all the settings, we observe high accuracy in detecting surges with the best accuracy is around 98% in the 70/30 splitting set. This shows that the previous step’s ground truth label is important in classifying new surges. Thus incorporating this historical label data would give a better prediction accuracy. On the other hand, in this experiment, we assume that we would always get ground truth label. However, in case of any delay in getting the actual labels of the previous surges, we would not have the ground truth data for all the testing instances. To emulate this scenario we design the next experiment which we describe below.

Experiment 3: In this experiment, we consider both the temporal data and delay in getting the actual label of each surge from the surgeon. We design several settings corresponding to different delay length. Firstly, we consider that we will miss ground truth labels for alternating surges which is represented as “1:1 GT:MISS” in the table. We observe that the accuracy remains high despite the missing labels and the classifier achieves an accuracy of 97%. Note that, here the number of missing labels correspond to the length of the delay. Later, we increase delay and consider that we will miss three previous surges labels out of four surges. Despite this delay (indicated by “1:3 GT:MISS” in the table), we observe that the classifier’s accuracy remain almost the same. Similarly, later we introduce more delay and thus increase the missing labels. We observe that the performances decrease gradually. In the table, we see that when the missing labels go above 50, the accuracy decreases drastically. Thus varying the missing labels, we can achieve a trade-off between surges detection accuracy and delay.

Table 13. Experimental Design Emulating Delay

Exp.	Settings		Train/Test (%)	Accuracy (%)
1	No temporal information No Delay		70/30	69.06
			80/20	63.50
			90/10	69.26
2	With Temporal information No Delay		70/30	98.31
			80/20	97.84
			90/10	97.49
3	With temporal information and delay	1:1 GT:MISS	80/20	97.53
		1:3 GT:MISS	80/20	97.22
		1:10 GT:MISS	80/20	95.40
		1:50 GT:MISS	80/20	85.30
		1:100 GT:MISS	80/20	73.08

In the setting of delayed information there is very limited work ever done. One of the methods proposed in literature is to augment the underlying Markov Decision Process such that it the agent state becomes the last known environment state and the actions taken by the agent from the last known state to current time step. However, this approach is highly inefficient to implement because of the following reasons:

1. The increase in state space complexity is exponential.
2. Solution is not scalable for stochastic delays.

Further literature survey shows that algorithms take action based on the most likely current state. This scheme is scalable for stochastic delays, and doesn't cause an exponential increase in state space. However, this approach is not robust under the regimes where the most likely state itself has low probability. For example, consider a system with five state A, B, C, D, E . At a given instant the probabilities of being in state A is 40%, and being in any other state is 15%. The most likely state is A , but not being in A is much more likely than being in A .

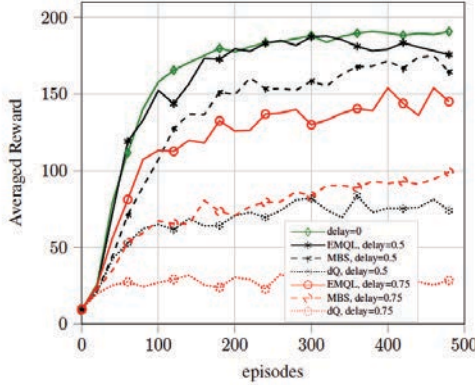
To mitigate the above mentioned issues, an algorithm was developed to take actions which maximize according to the expected equality of the action, and not the expected state. Quality of an action is defined as the expected total discounted reward, on taking action a in state s . The value of an state is defined as the total discounted reward agent can collect starting from state s .

$$Q(s, a) = E\left[\sum_{t=0}^{\infty} \gamma^t r \mid s_t = s, a_t = a\right], V(s) = \max_a Q(s, a)$$

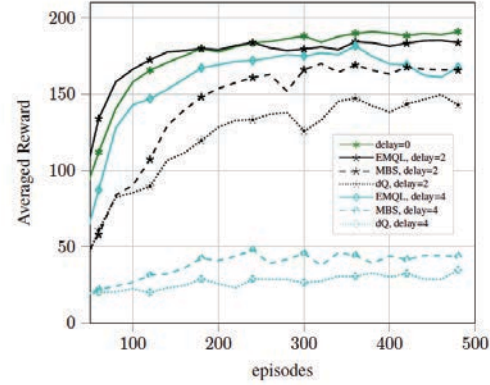
The algorithm selects an action which maximizes the expected value of the current unknown state:

$$a = \operatorname{argmax} E_s[Q(s, a) | s_{t-d}, a_{t-d}, \dots, a_{t-1}]$$

With the proposed solution (called EMQL), the agent performs better than current algorithms available. It is state efficient, scalable to stochastic delays, and robust under deviation from most likely state.



(a) Performance under stochastic delays.



(b) Performance under constant delays.

The figures above compare the following algorithms:

1. EMQL: Proposed and developed algorithm
2. MBS: Algorithm which takes action for the most likely state
3. dQ: Algorithm which is not optimizing for the delays

For stochastic delays, delay probabilities are mentioned. Averaged delay thus becomes $E[d] = 1/(\text{probability of delay})$. Two settings were compared with average delay of 2 timesteps and average delay of 4 timesteps. Corresponding constant delays were also considered. Performance for stochastic delays where average delay is same as constant delay is better. This is expected as the agent has more probability to take correct decisions.

Proposed EMQL algorithm beats the other two algorithm in performance. Also when the delays are large, the deviation is from the optimal converging point under no delays is smaller for EMQL

Task 1.2: Surge Classification

Transfer learning for surge classification using kinematics and visual data)

We develop a methodology that uses compact image representations with kinematic features for surge recognition in the DESK dataset (Taurus II, Taurus II simulator, and YuMi robots data). This dataset offers samples for surgical procedures over different robotic platforms with high variability in the setup. We performed surge classification in two setups: 1) No transfer, 2) Transfer from a simulated scenario to two

real deployable robots. Then, the results were compared with recognition accuracies using only kinematic data with the same experimental setup. The results show that our approach improves the recognition performance over kinematic data across different domains.

Task 1.3: Learning with Imperfect and Randomly Delayed Information- Theoretical results and simulations

We investigated the performance of decision-making agents for stochastic control. Based on the optimization goal for undelayed processes, we developed a policy to decide actions from imperfect and/or delayed state information. Results for them are presented in the following subsections:

1.3.1: Policy for imperfect and delayed observations

An agent at each time t is to observe the state $s_t \in S$, and take an action $a_t \in A$ to obtain a reward r_t . This action results in a transition to next state s_{t+1} with probability $p(s_{t+1}|s_t, a_t)$. The decision rule or policy $\pi: S \rightarrow A$ is the mechanism through which the agent selects the action after observing the current state. The task for the agent is to find a policy $\pi: S \rightarrow A$ which maximizes the optimization problem

$$\max_{\pi} \mathbb{E} \left[\sum_{t=1}^T \gamma^t r_t(s_t, \pi(s_t)) | s_1 = s \right]$$

Where $\gamma (< 1)$ is the weight given to future rewards to keep the summation bounded for large values of T . The optimal value of the sum is denoted as V^* . When T is large or $T \rightarrow \infty$ a function $Q: S \times A \rightarrow R$ can be defined as,

$$Q^*(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim p(s'|s, a)} [V^*(s')]$$

The optimal policy which solves the optimization problem is given as

$$\pi^*(s) = \arg \max_{a \in A} Q^*(s, a)$$

However, in our setup, the state information s_t might not be observed by the agents, instead, it receives an observation o_t at time t . For imperfect state observation o , the proposed optimal policy is given as

$$\pi^*(o) = \arg \max_{a \in A} \mathbb{E}_{s|o} [Q^*(s, a)]$$

1.3.2 Lower bounds on the policy obtained

We also provide a lower bound on the performance of this policy. We compare the performance with the instance when there exists an oracle which can provide the optimal action for the current state from the imperfect or delayed observation. With the policy $\pi(o)$, the total expected reward achieved is lower bounded as

$$V^{\pi} \geq V^* - \frac{R_{max}}{(1 - \gamma)^2} \left(1 - \frac{1}{|A|} \right)$$

where R_{max} is the maximum possible reward the agent can obtain.

1.3.3 Algorithm implementation details

To obtain the policy we utilize a model-based implementation of reinforcement learning to learn probability distributions of the next state s_{t+1} given the current state s_t and action a_t .

We implemented an algorithm (named EMQL) to simultaneously learn the policy using reinforcement learning and state distribution for discrete state spaces and using deep neural networks. The algorithm trains a deep Q network to calculate the Q function for each state and action. The algorithm, in parallel, also trains a generative neural network to generate probable current states from the imperfect/delayed observations available. Since generative neural networks do not give the probability, we generate the probable states from the last known state and action sequence. We use Monte Carlo sampling is used to generate a large number (~ 5000) probable current states. Later, from the Q function network trained in parallel, we calculate the Q-values for each action for all the probable current states generated from the generative network. Lastly, the action which maximizes the empirical mean of Q function is selected.

1.3.4 Performance evaluations using existing methods for discrete state spaces.

We evaluated the algorithm on the OpenAI Gym platform for preliminary testing. We use stochastic delays for testing purposes. Each observation is delayed by discrete time steps. The delays are generated from Geometric distribution. We denote the probability of success p as the probability of delaying the observation at any instant. Thus the average delay is $\frac{1}{1-p}$.

Further, the delays cause the state observations to become asynchronous. The algorithm always considers the latest state observation to make a decision. The stale values are used to train the generative and deep Q networks.

Figure 13 shows the performance of EMQL under expected delays of 2 units, and 4 units. The observations are averaged over 20 different runs of the algorithm. The performance of the algorithm is limited by the performance of the generative neural network. We conjecture that by improving the generative neural network, the performance of EMQL algorithm also improves, and reduces the gap from no delays as well.

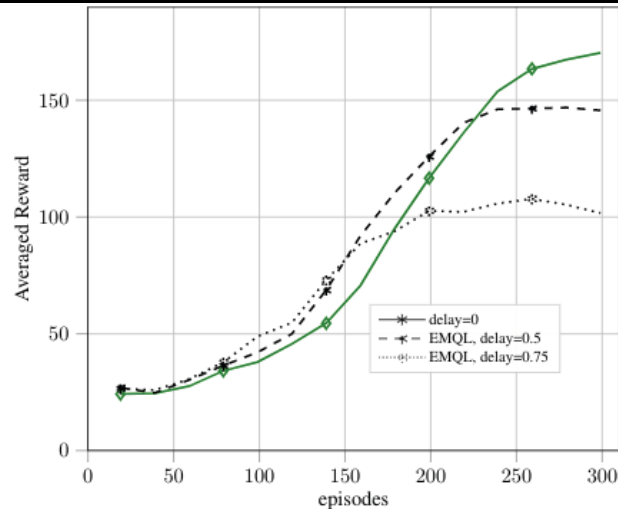


Figure 48. EMQL performance with different delay probabilities

References

1. Redmon, J., & Farhadi, A. (2018). Yolo3: 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.

TMUSMI Progress

We met with 3D systems <https://symbionix.com/> in Calgary twice to review a couple of their simulators. On the first visit, we looked at their Ultrasound simulator that would be applicable for the EFAST as well as the maternity simulator. This simulator allows for what resembles live ultrasound images with varying levels of pathology. The maternity simulator was the same. This simulator was different from other ultrasound train models in that it does not just show you a picture when the prob gets reasonably close to the region being scanned. The person using the ultrasound must be holding the prob properly or the images do not display and pathology will be missed. On the second visit, we saw the interventional simulator that allowed for the training of laproscopic procedures. All of the simulators were well received by all and were deemed to be acceptable for research purposes- TIFFFCAT. Should funds become available.

We also met with the Philips rep again and have started on the mentor side of telemedicine. We are attempting to improve how our mentors instruct and bring some consistency to this process. We have started what are calling mentor school and are examining the influence of pre-teaching on a mentors ability to instruct. This will hopefully improve our future work with the TIFFFCAT as we will be able to standardize the mentors.

In June we went to Israel and had a chance to work with Ben Gurion University. We were able to see the 3D systems simulators in use as well as the use of live models. Several things were learned during this process on IT connection, recording, scoring and screening of live models. This will greatly help us in the progression of TIFFCAT

Task 1.3 Policy for imperfect and delayed observations

We developed a neural network framework which utilizes past surgemes to predict the current surgeme.

1.3.1 Architecture:

The developed architecture is a fully connected dense neural network with one hidden layer consisting of 32 units. Input is past 'n' frames which is a controllable hyper parameter of the network. Input is fed as a n-sparse binary vector. Each surgeme is coded as a one-hot binary vector, and n surgemes are appended. The output is again a one-hot binary vector with bit for predicted surgeme set to 1. The complete architecture is presented in Figure 1.

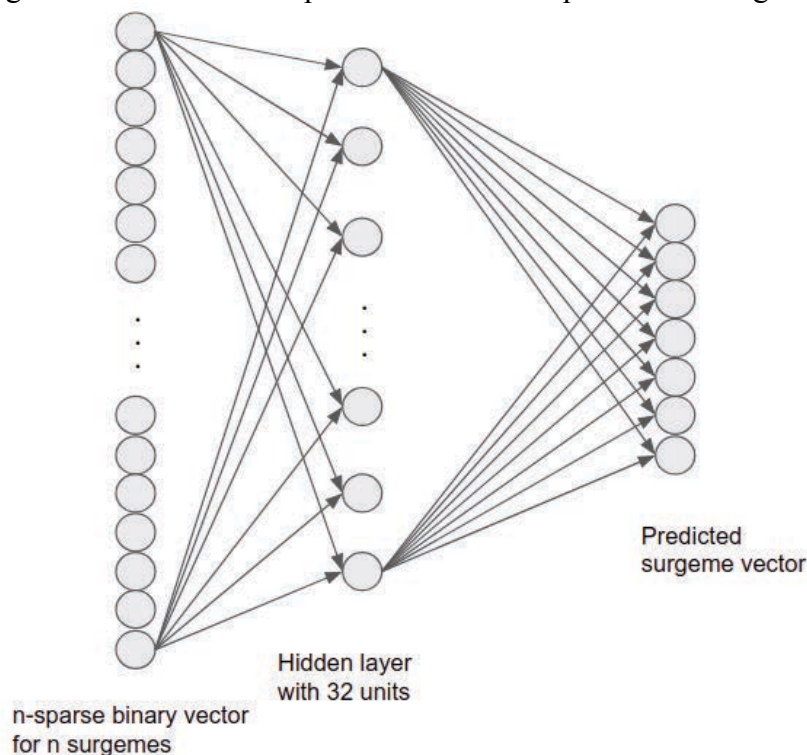


Figure 49: Architecture for predicting current surgeme using previous surgemes

1.3.2 Evaluations:

We performed experiments with the DESK dataset to test the proposed architecture. We set the hyperparameter ‘n’ to 5 and observed that the network was trained in 10 epochs. We tested the architecture for multiple values of delays starting with a best possible case of no delay or delay = 0 ms. We increased the delays upto 666.67 ms with a step size of 166.67 ms. The results are presented in Figure 2. We observe that the network correctly learns to output the received current surgeme in case of no delays, thus achieving 100% accuracy when delay = 0. Further as delays increase, we observe that the accuracy decreases as the entropy of the system increases.

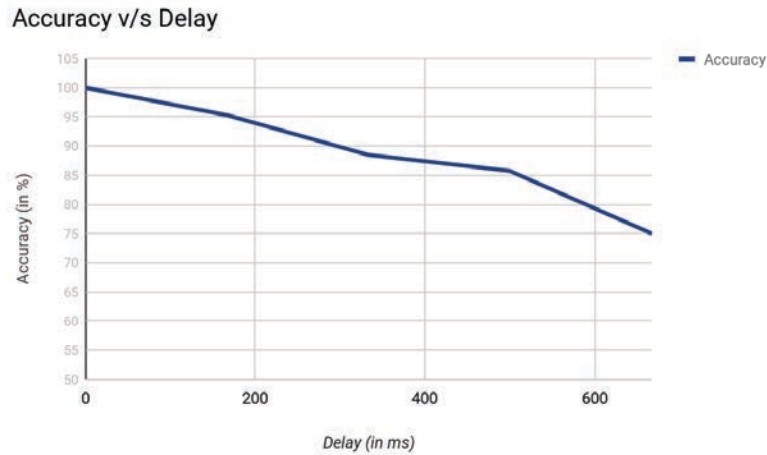


Figure 50: Accuracy v/s Delay for the proposed network.

Task 2.1 Integrating predictions with semi-supervision

2.1.1 Surgeme-Recognition Execution Framework

We developed a framework comprised of two modules: a recognition module and an execution module. The first module that recognizes the surgemes performed by the teleoperator. These surgemes are sent to the remote robot as high level commands for the robot to perform. Then, the execution module performs the received action autonomously. The following sections describe the recognition and execution frameworks in detail.

2.1.1.1 Recognition Module:

Our approach combines features extracted from video frames and kinematic data to perform surgeme classification based on the multi-modal recognition system proposed by Rahman et al. [1]. An overview of the system's architecture is given in Figure 51.

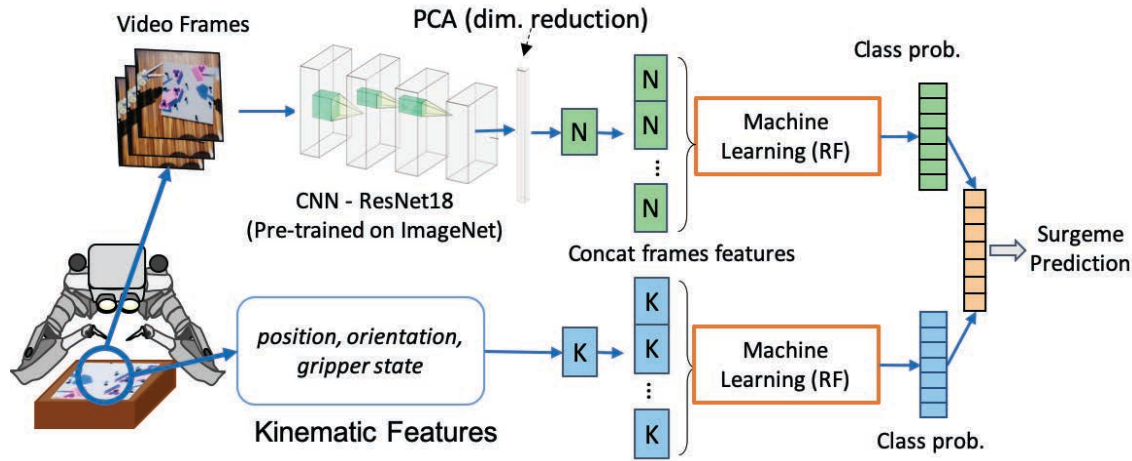


Figure 51: Recognition Module architecture. For each frame, video and kinematic features were extracted separately and then the frame sequences were combined to obtain the feature vectors per surgeme. The visual and kinematic features were used to train two separate supervised learning algorithms whose output class probabilities were then combined to obtain final class probabilities.

First, images extracted from video frames are resized from 1920 x 1080 to 228 x 128 size. This resized image is then passed to a ResNet18 [2] (pre-trained on ImageNet) to extract image features. Due to the large dimension of the output of the truncated ResNet18, a PCA model is used to reduce the dimension of the features to a lower-dimensional ($N=30$) vector.

Additionally, we extracted features from the kinematic data by encoding the features into a shared representation. This representation is composed of the position, orientation and gripper status of the end-effectors (14 features: seven features for each arm corresponding to each of the 7 DoF) [3].

The surgeme instances (frames) were re-sampled to a fixed number of frames (40) time-window to generate a sequential feature instances for each surgeme using linear interpolation.

The 14 kinematic features for each frame were then concatenated into a single 560-dimensional vector. Similarly, for image frames, 30 features were concatenated resulting in a 1200 dimensional (40×30) feature vector for each surgeme instance. The combined video features were further reduced by PCA to a 100-dimensional feature vector for the entire image sequence in that time-window.

The visual and kinematic features were used to train two separate supervised learning algorithms. One classifier is trained using the visual features and the other classifier is trained on the kinematic features.

The final class probability is calculated as a convex combination of the output class probabilities of the two classifiers with a hyper-parameter λ , as following equation

$$P(C) = \lambda P_{kin}(C) + (1 - \lambda) P_{video}(C)$$

where C is the class and $P_{kin}(C)$ and $P_{video}(C)$ are the probabilities given by the classifiers using kinematic and video features respectively. For the experiments in this paper, the value of λ is set to 0.8.

2.1.1.2 Execution Unit

The execution module consists of two main components: i) visual recognition unit, and ii) surgeme execution unit. Visual unit continuously inspects the environment and tracks the objects and tools involved in the surgical task: pole, target object, tooltip pose over time. Leveraging this information, the execution unit employs a model-based approach to execute surgues. Surgeme commands are obtained from teleoperator data which has been automatically classified by the Recognition Module. In this paper, the surgeme execution is conducted on an ABB YuMi robot with 3D printed surgical extensions. The details of this process are discussed below.

Visual Recognition Unit:

A 3D camera is installed on top of the ABB YuMi robot (Intel Realsense), which continuously streams color (RGB) and depth image frames. These RGB frames are passed into two separate networks of object detection and segmentation: Darknet (YoloV3)[4], and Mask-RCNN [5].

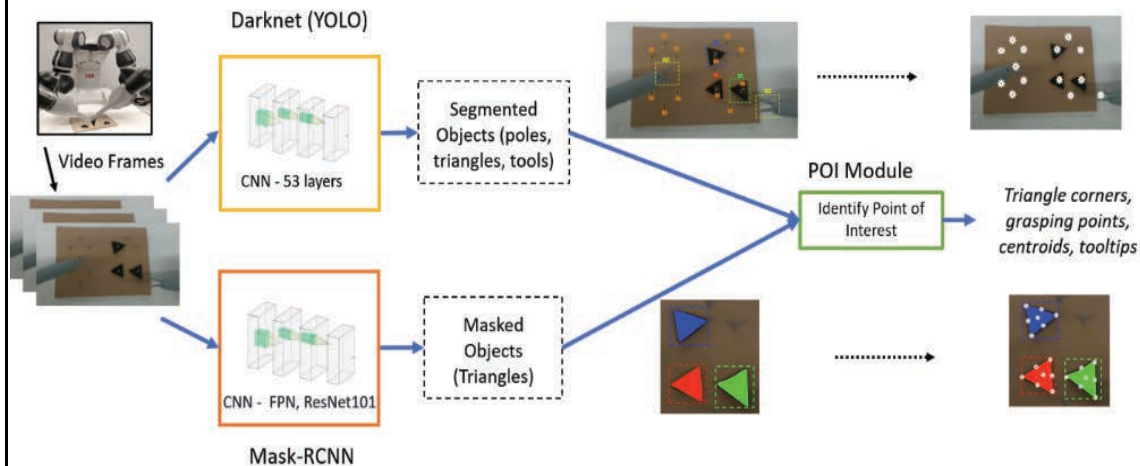


Figure 52. Overview of the Visual Recognition System

An overview of the system is given in Figure 52. The Darknet (YoloV3) network which consists of 53 convolutional layers that take raw image frames as input and outputs bounding boxes around the recognized objects. For the peg transfer task, the objects are identified as poles, triangles, and robot-tooltips.

YOLO, “You Look Only Once”, is a neural network focussed on detecting and localising various classes in an image. YOLO is very fast with detections and uses bounding boxes to localize detected classes in an image. The YOLO network architecture is inspired by the GoogLeNet model for image classification. The network has 24 convolutional layers followed by 2 fully

Starting from the pre-trained weights for COCO dataset, the Mask-RCNN was further trained on YuMi video data frames from DESK. The object detection and segmentation from the tracker and Mask-RCNN were further used as input to the Point of Interest (POI) module which determines points of interest such as grasp points on the objects, which are required for surgeme execution (marked with asterisks in Figure 4). We use OpenCV to detect the features such as corners and edges and keep the track of the corner points using the Point of Interest (POI) module. The three corner points detected (the masks obtained in shapes of triangular pegs would have three corners) are used to estimate grasp points. We find the center of the detected triangle to estimate the distance between the center and the corner points. In the case of the triangular pegs the ideal grasp points are points that are in between the two corners on any side. Once we have the three midpoints, we offset these points along the normal that passes through the center of the triangle. This offset ensures that we grasp the triangular peg along its surface rather than the edge. Figure 2 shows the corners and grasp points detected and estimated respectively. Amongst the three grasp points estimated, we choose the grasp point that is closest to the robot tooltips position.

Surgeme Execution Unit:

The execution module receives high-level surgeme commands which include the surgeme to be performed and the relevant parameters of the command. The commands received are encoded in parameters such as the target object (e.g. number of the pole, triangles) and the active hand.

The execution of the surgemes is performed by a model-based approach similar to how a priori skills were modelled in [9]. The "approach" surgeme recognizes the pole of interest related to the received command and extracts a region of interest (ROI) from the camera input that contains a triangle shaped object. From the identified ROIs, the corners of those objects are recognized, and the dominant robot tooltip is sent to one of the identified corners. In the "align and grasp" surgeme, the robot calculates feasible grasping points located on the midpoint of the lines that connect the triangle's corners, as shown in Figure 2 (Right) and the robot moves to the grasp point. For the "transfer - get together" surgeme, both arms are brought to a middle point of the work space. For the "transfer-exchange" surgeme, the non-dominant hand grasps the object while the initial hand releases it. For the "approach target" surgeme, the center of the triangle object is detected using computer vision and aligned in the horizontal plane with the target pole. The "align and place" surgeme moves down and releases the peg object on the target pole.

Surgeme-Recognition Execution Experiments

The experiments were conducted using a subset of peg transfer trials using the DESK [3] dataset. For each robotic domain, 80% of the surgeme data instances were used to train the recognition module and the remaining 20% was used for testing. The testing subset was used to evaluate the performance of the complete teleoperation framework. That is, the videos from the testing fold in the recognition module were used for the execution of experiments on the real YuMi robot. The detailed statistics of the surgemes can be found in Table I. The testing set contained 76 trials, that were evenly distributed between the Taurus robot, the Taurus simulator and the YuMi robot

Table 14. Surgemes and dataset statistics for the peg transfer

ID	Surgeme Name	# Taurus Sim	# Taurus	# YuMi
----	--------------	-----------------	----------	-----------

S1	Approach peg	170	170	117
S2	Align and grasp	171	171	123
S3	Lift Peg	165	165	123
S4	Transfer peg - Get Together	150	150	117
S5	Transfer peg - Exchange	146	146	118
S6	Approach pole	137	137	117
S7	Align & place	134	134	116

The results for the prediction module are shown in table 2, discretized according to each specific surgeme. The results are calculated over the 76 complete trials included in the testing set (trials with all S1-S7 surgemes). The surgeme prediction architecture achieves an accuracy of 100% with at least one robot for surgemes S1, S2 and S7. Overall, the surgeme recognition accuracy is 91.5% on average, for all surgemes and robotic platforms.

Table 15. Results for the recognition module per robot

Robot	S1	S2	S3	S4	S5	S6	S7
YuMi	100	96	88	92	96	96	96
Taurus	92	100	88	84	96	92	100
Taurus Sim	100	97	81	90	94	77	73
Average	97	98	85	89	95	88	90
Num Trials	76	76	76	76	76	76	76

The ground truth scenario evaluates the accuracy of the execution module when the surgemes (high level commands) are received in their totality. The table 3 shows the performance of the surgeme execution module over the ground truth scenario. Results show an average execution success of 98% for all surgemes and robotic platforms. Furthermore, the average success rate of the peg transfer task is 86.6%.

Table 16. Execution results over ground truth test data per robot

Robot	S1	S2	S3	S4	S5	S6	S7	Task
YuMi	100	96	100	91	95	100	100	83
Taurus	100	96	100	87	100	100	100	83

Taurus Sim	100	96.4	100	96.3	100	100	100	92.8
Average	100	96.1	100	91.7	98.4	100	100	86.6
Num Trials	76	76	73	73	67	66	66	76

In addition we assessed the success of the proposed approach when surges are recognized from real kinematic and visual data and evaluated as high level commands on the execution module. Table 4 shows the results and task success rate on the complete framework.

Table 17. Task completion rate of proposed framework

Robot	Prediction	Task Success	Trials
YuMi	83.3	70.8	24
Taurus	66.7	64.2	24
Taurus Sim	41.9	39.2	28
Total	63.2	53.9	76

We further studied the compression capabilities of the proposed approach. We compared the data required to send complete kinematics information to the robot from the teleoperator side vs the data required to send the surges to the robot. We also compare the compression capabilities with the state of the art Lempel Ziv compression algorithm commonly used in gzip framework by UNIX.

We define "compression achieved" as ratio of the reduced message size to the original message size. We provide the compression achieved by our approach, compression achieved by gzip algorithm, and compression achieved when comparing data required by our approach and gzip algorithm. The results are presented in Table 5. Particularly, even with gzip algorithm's efficient compression, our algorithm achieves over 98% improvement over gzip.

Table 18. Compression achieved by the proposed framework

Robot	gzip	our framework	improvement
YuMi	86.17%	99.87%	98.48%
Taurus	79.88%	99.99%	99.93%
Taurus Sim	70.61%	99.87%	99.57%

Overall, we obtained a data compression improvement of 99:79% and an average success rate of 98% for surgeme execution using a model-based approach.

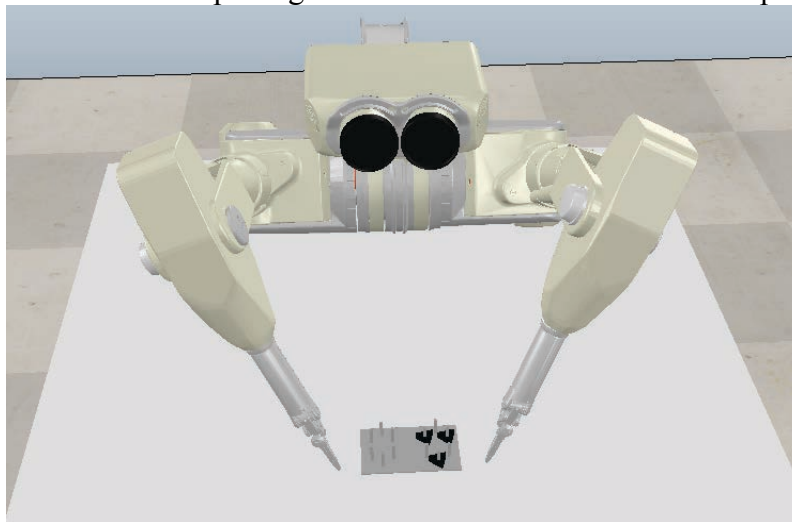
2.1.2 Surgeme recognition without segmentation

Previously, we detected surgeme type on the segmented surgeme (i.e., where the start and the end of the surgeme was given). We further conducted experiments where the surgeme is not segmented. In this case, we get a continuous data stream for a trial and detect surgeme per frame.

We developed an LSTM-based classification approach which takes data for a trial (entire peg transfer) and recognize the surgeme class per frame. We experimented with the Taurus kinematic data and observed an frame-wise accuracy of 80%. In the future, we will incorporate visual data with the kinematic and develop a multi-modal LSTM-based recognition model.

2.1.3 Creation of an open Taurus simulator

We have created a robotic simulator of the Taurus robot using the robot simulator software v-rep (See Figure). The simulator is constructed based on the CAD models provided by SRI. The robotic simulator currently had similar kinematics to the real Taurus robot and is controlled through v-rep inverse kinematics package to follow the desired end effector pose.



Currently the Taurus control is based on the direct motion of the operator's hands obtained from the virtual reality controllers and headset HTC VIVE. The VR system allows the operator to view the robot simulator environment in virtual reality and allow to collect teleoperation data of the robot in an intuitive way. The kinematics of the robot were inspired in the configuration of the real robot, and further tuning is required to obtain high correspondence of the kinematics and kinetics between the real robot and the robot simulated model.

References

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. 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.
3. 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.
4. Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767* (2018).
5. K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.
6. A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, 2016.
7. W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow." https://github.com/matterport/Mask_RCNN, 2017.
8. M. V. Balakuntala, V. L. Venkatesh, J. P. Bindu, R. M. Voyles, and J. Wachs, "Extending policy from one-shot learning through coaching," in *28th IEEE International Conference on Robot and Human Interactive Communication (Ro-Man-2019)*, 2019.
9. Zeng, A., Song, S., Welker, S., Lee, J., Rodriguez, A., & Funkhouser, T. (2018). *Learning Synergies between Pushing and Grasping with Self-Supervised Deep Reinforcement Learning*. *IEEE International Conference on Intelligent Robots and Systems*, 4238–4245.

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

If the project was not intended to provide training and professional development opportunities or there is nothing significant to report during this reporting period, state "Nothing to Report."

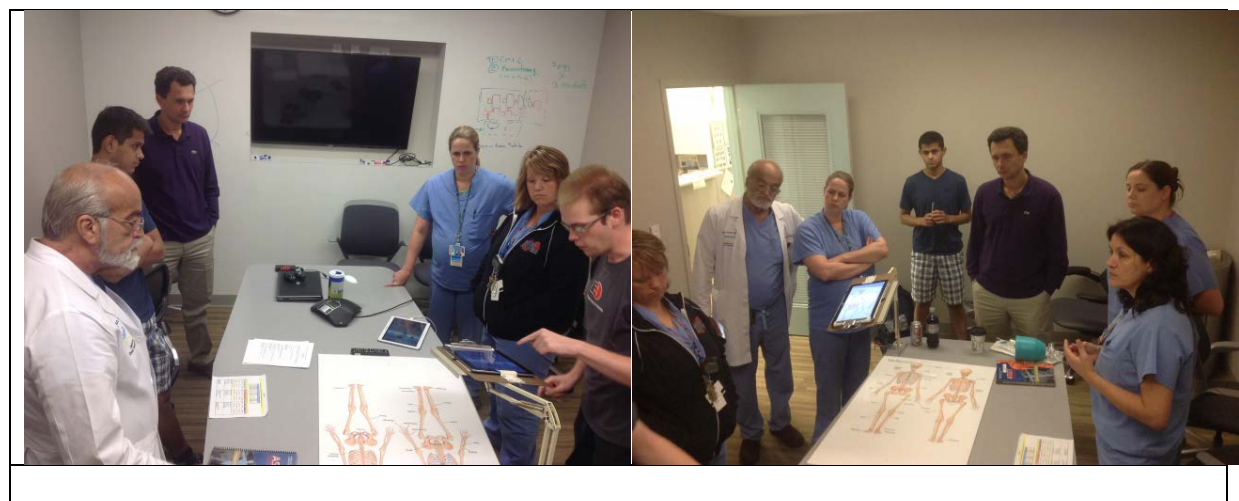
Describe opportunities for training and professional development provided to anyone who worked on the project or anyone who was involved in the activities supported by the project. "Training" activities are those in which individuals with advanced professional skills and experience assist others in attaining greater proficiency. Training activities may include, for example, courses or one-on-one work with a mentor. "Professional development" activities result in increased knowledge or skill in one's area of expertise and may include workshops, conferences, seminars, study groups, and individual study. Include participation in conferences, workshops, and seminars not listed under major activities.

Training:

The PI Wachs and co-PI Voyles, Aggarwal, Voyles and Xue are mentoring 6 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 and Natalia Sanchez are working in the problems related to imitation learning and experimental design, under the mentorship of PI Wachs;

Professional development:

The PI Wachs participated in the second phase (observed) in three opportunities of the Advanced Trauma Operative Management (ATOM) course at Ashkenazi Hospital (IUSM), with the assistance of Dr. Gerry Gomez, Sherri Marley and Dr. Brian Mullis. The ATOM course demonstrates the surgical repair of common penetrating injuries; 1) a didactic session composed of six standardized 30-minute lectures that review the basic principles of trauma laparotomy and damage control and management of abdominal and thoracic injuries including injuries to the heart and major vessels and; 2) an operative porcine laboratory experience where the human operating room is replicated and surgeons perform operative repairs on 50 kilogram swine. Operative skill is evaluated in the laboratory.





How were the results disseminated to communities of interest?

If there is nothing significant to report during this reporting period, state “Nothing to Report.”

Describe how the results were disseminated to communities of interest. Include any outreach activities that were undertaken to reach members of communities who are not usually aware of these project activities, for the purpose of enhancing public understanding and increasing interest in learning and careers in science, technology, and the humanities.

The results were disseminated in an invited talk “Cybernetic Solutions to Remote Trauma Care. Hamlyn Symposium. British Geographic Society, London, UK, 2019. June 25

A second talk presented entitled Robots without Borders: “A Brave New World in Healthcare Robotics“. Rambam Healthcare Campus department of Obstetrics and Gynecology in Haifa, Israel, Oct. 10, 2019.

Presentation at the Military Health System Research Symposium (MHSRS 2019). Poster presented.

Oral presentation at Ro-Man 2019 in New Delhi, India (August 14-16).

To be presented as oral presentation at IROS 2019 – Macao China.

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

If this is the final report, state “Nothing to Report.”

Describe briefly what you plan to do during the next reporting period to accomplish the goals and objectives.

Task 1: Data collection for Debridement in Real robot and simulator system

A debridement procedure will be added to the skill library using the previous protocol of data collection. The task will be performed by a real robot (Taurus/ABB YuMi) and its simulated version. The kinematic, RGB video and depth information will be obtained for each trial.

In addition, the procedure will also be divided and annotated into surges: 1) Approach debris 2) Align and grasp debris, 3) Lift, 4) Drop 5) Approach gauze, 6) Align and grasp gauze 7) Approach wound and 8) Clean wound.

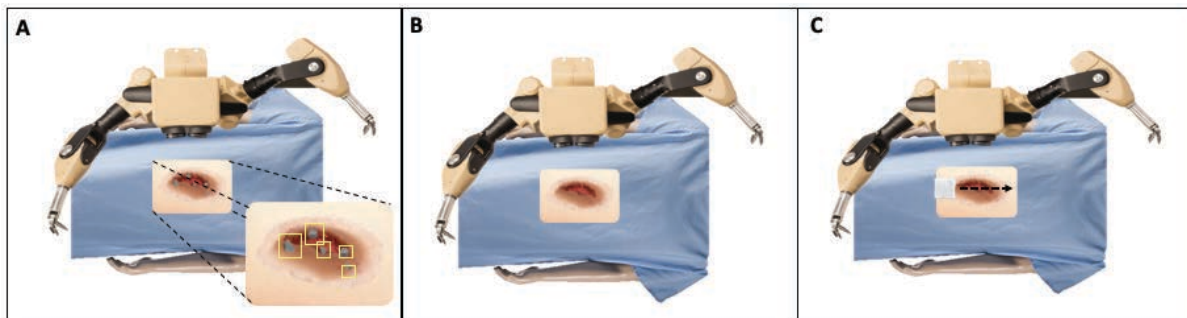


Fig 1. Debridement process. First, the debris is identified and removed and then the wound is cleaned with a gauze.

The surges 1 to 4 would be performed several times for all the detected debris (see Figure 1-A). When the wound is clear of debris (see Figure 1-B), the robot would proceed with surges 5 to 8 in order to clean the wound (Figure 1-C).

Task 2: Closing the loop with live recognition/execution in the simulator

We will implement the first stage of a closed loop execution (See Figure 2). The operator will start to execute the tasks normally in the developed simulator while a real time surge classification algorithm recognizes the high level actions that are performed (Figure 2-Left.). These high level actions or surges are sent as instructions to the remote execution side. The surge that is sent will be modeled to get delayed or lost. A Reinforcement Learning algorithm will predict the next surge in situations of delays of information loss. The execution module presented developed in Task 2.1 will be used to complete perform the surge. Then, the state of the robot and the environment will be updated in the simulator.

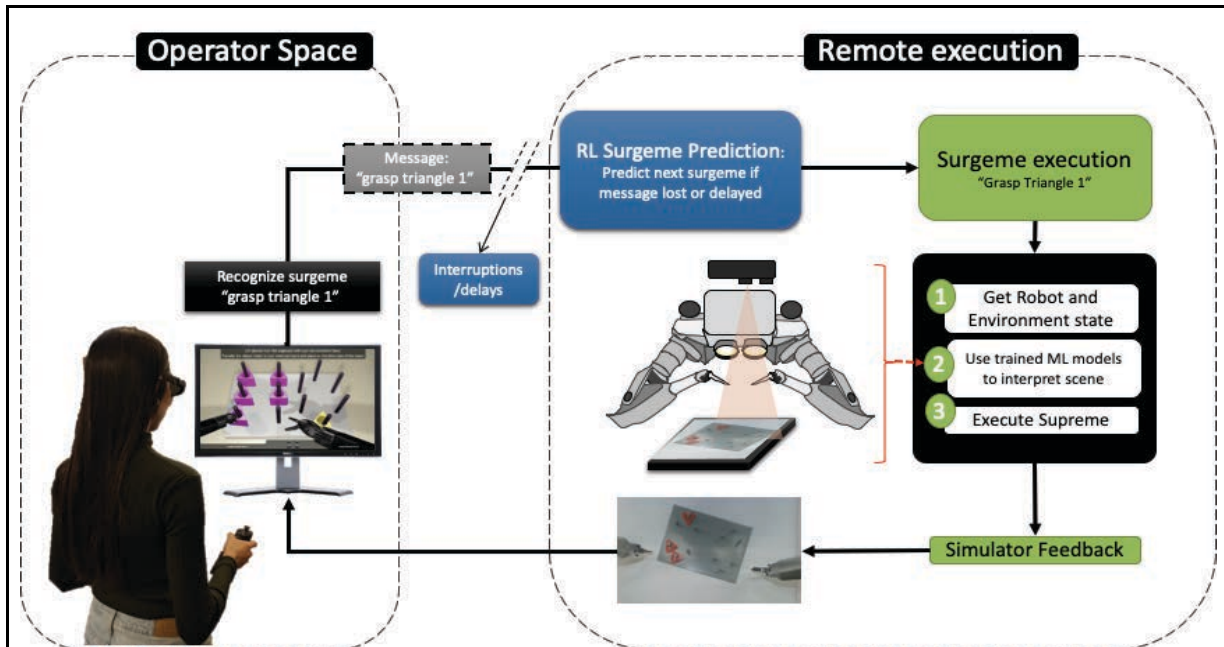


Figure 2. Initial Semi-autonomous robot teleoperation system design.

Task 3: Developing a data collection framework for the da Vinci Research Kit

We will develop a data collection protocol and a framework to include the da Vinci Research Kit (dVRK) [1] to the skill library. The use of the da Vinci has exponentially increased in minimally invasive surgery and it is becoming a common tool in surgical datasets [3,4]. The dVRK uses the first version of the da Vinci robot, with an open control system developed at Johns Hopkins University [2] (See Figure 3 left). The dVRK can perform high precision tasks that resemble the challenges of minimally invasive surgery. We will generate a system to collect peg transfer data following the same protocol for recording and randomization used for the published DESK dataset [5] (See Figure 3 right).

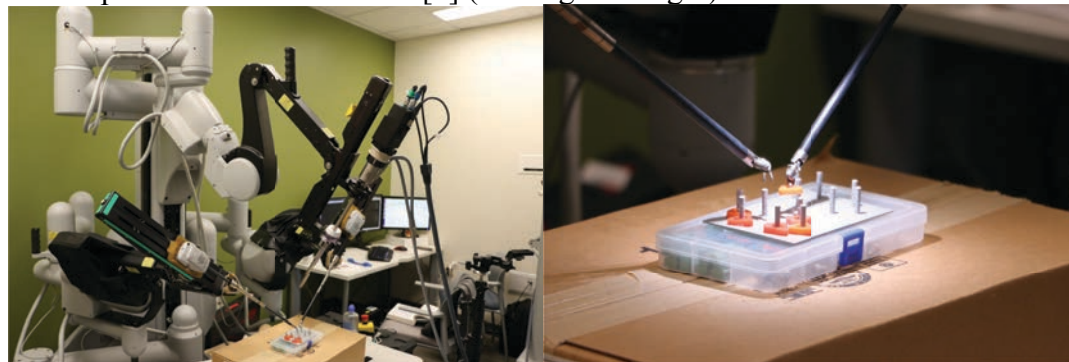


Figure 3. (Left) - Da vinci research Kit. (Right) - Peg transfer with the dVRK.

References

1. "Da Vinci Research Kit Wiki Community." *Research.intusurg.com*, https://research.intusurg.com/index.php/Main_Page.
2. "Da Vinci Research Kit." Biorobotics, <http://biorobotics.ri.cmu.edu/robots/daVinci.php>.

3. Kazanzides, Peter, et al. "An open-source research kit for the da Vinci® Surgical System." *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014.
4. Gao, Yixin, et al. "Jhu-isi gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling." *MICCAI Workshop: M2CAI*. Vol. 3. 2014.

4. **IMPACT:** Describe distinctive contributions, major accomplishments, innovations, successes, or any change in practice or behavior that has come about as a result of the project relative to:

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

If there is nothing significant to report during this reporting period, state "Nothing to Report."

Describe how findings, results, techniques that were developed or extended, or other products from the project made an impact or are likely to make an impact on the base of knowledge, theory, and research in the principal disciplinary field(s) of the project. Summarize using language that an intelligent lay audience can understand (Scientific American style).

According to our preliminary results, the performance of the recognition module is on average above 85%, the correct recognition of all surgemes in each trial is significantly lower (63.2% on average). Similarly, successful execution of the trial drops from 86.6% on average for the ground truth to 53.9%. These results are explained by the accumulation of errors in all the steps of the framework: If there are any errors on surgeme recognition during the peg transfer task, the trial cannot continue and it is therefore classified as a failure. The lack of recovery strategies causes any error in either recognition or execution to be propagated and thus, the performance of the complete framework is significantly lower than the results for recognition and execution under controlled conditions. This paper presents a framework for semi-autonomous teleoperation that significantly reduces the amount of data transmitted during robot teleoperation. The framework consists of two main modules: (i) Recognition Module- which recognize surgeme to perform at the operator side using video, and kinematics and (ii) Execution Module – which execute the surgemes sent as a high-level command through a low-bandwidth communication network at the remote robot. From the current work, we can conclude that surgical skills were recognized from videos of three different robotics platforms and executed using a model-based approach. The combined framework achieved 70.83% accuracy on execution for the YuMi robot with surgeme prediction accuracy of 83.3%, even with 99:79% compression of the transmitted data.

What was the impact on other disciplines?

If there is nothing significant to report during this reporting period, state "Nothing to Report."

Describe how the findings, results, or techniques that were developed or improved, or other products from the project made an impact or are likely to make an 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 surges 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 surge 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?

If there is nothing significant to report during this reporting period, state "Nothing to Report."

Describe ways in which the project made an impact, or is likely to make an impact, on commercial technology or public use, including:

- *transfer of results to entities in government or industry;*
- *instances where the research has led to the initiation of a start-up company; or*
- *adoption of new practices.*

Nothing to report.

What was the impact on society beyond science and technology?

If there is nothing significant to report during this reporting period, state "Nothing to Report."

Describe how results from the project made an impact, or are likely to make an impact, beyond the bounds of science, engineering, and the academic world on areas such as:

- *improving public knowledge, attitudes, skills, and abilities;*
- *changing behavior, practices, decision making, policies (including regulatory policies), or social actions; or*
- *improving social, economic, civic, or environmental conditions.*

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:** The Project Director/Principal Investigator (PD/PI) is reminded that the recipient organization is required to obtain prior written approval from the awarding agency Grants Officer whenever there are significant changes in the project or its direction. If not previously reported in writing, provide the following additional information or state, "Nothing to Report," if applicable:

Changes in approach and reasons for change

Describe any changes in approach during the reporting period and reasons for these changes. Remember that significant changes in objectives and scope require prior approval of the agency.

There are no significant changes in the objectives and scope of the project. One simple change is that in Major Task 2, instead of focusing on semantic rules creation by hand, we are relying on deep learning architectures to discover such rules.

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

Describe problems or delays encountered during the reporting period and actions or plans to resolve them.

We do not anticipate delays.

Changes that had a significant impact on expenditures

Describe changes during the reporting period that may have had a significant impact on expenditures, for example, delays in hiring staff or favorable developments that enable meeting objectives at less cost than anticipated.

No changes.

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

Describe significant deviations, unexpected outcomes, or changes in approved protocols for the use or care of human subjects, vertebrate animals, biohazards, and/or select agents during the reporting period. If required, were these changes approved by the applicable institution committee (or equivalent) and reported to the agency? Also specify the applicable Institutional Review Board/Institutional Animal Care and Use Committee approval dates.

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: List any products resulting from the project during the reporting period. If there is nothing to report under a particular item, state “Nothing to Report.”

- **Publications, conference papers, and presentations**

Report only the major publication(s) resulting from the work under this award.

Journal publications. *List peer-reviewed articles or papers appearing in scientific, technical, or professional journals. Identify for each publication: Author(s); title; journal; volume; year; page numbers; status of publication (published; accepted, awaiting publication; submitted, under review; other); acknowledgement of federal support (yes/no).*

No changes.

Books or other non-periodical, one-time publications. *Report any book, monograph, dissertation, abstract, or the like published as or in a separate publication, rather than a periodical or series. Include any significant publication in the proceedings of a one-time conference or in the report of a one-time study, commission, or the like. Identify for each one-time publication: Author(s); title; editor; title of collection, if applicable; bibliographic information; year; type of publication (e.g., book, thesis or dissertation); status of publication (published; accepted, awaiting publication; submitted, under review; other); acknowledgement of federal support (yes/no).*

No changes.

Other publications, conference papers, and presentations. *Identify any other publications, conference papers and/or presentations not reported above. Specify the status of the publication as noted above. List presentations made during the last year (international, national, local societies, military meetings, etc.). Use an asterisk (*) if presentation produced a manuscript.*

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. N. Sanchez-Tamayo, L. N. V. Venkatesh, G. Gonzalez, M. V. Balakuntala, M. M. Rahman, M. Agarwal, V. Aggarwal, Y. Xue, R. M. Voyles and J. Wachs, “SARTRES: A Semi-Autonomous Robot Teleoperation Environment for Surgery”, International Conference of Robots and Automation (ICRA-2020), Submitted in Sept 15, 2019)

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

List the URL for any Internet site(s) that disseminates the results of the research activities. A short description of each site should be provided. It is not necessary to include the publications already specified above in this section.

[Desk Dataset:](#)

Library of surgemes collected during a peg transfer procedure. The data was collected using three different robots: 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.

[Execution Pipeline Dissemination video:](#)

Dissemination video used for the ICRA conference submission. This video shows the recognition-execution framework based on surgemes that is currently developed.

[Development Repository:](#)

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

- **Technologies or techniques**

Identify technologies or techniques that resulted from the research activities. In addition to a description of the technologies or techniques, describe how they will be shared

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 three robotic platforms: 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 (e.g. from Taurus to YuMi robot) for a surgical gesture classification task with seven gestures.

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

Identify inventions, patent applications with date, and/or licenses that have resulted from the research. State whether an application is provisional or non-provisional and indicate the application number. Submission of this information as part of an interim research performance progress report is not a substitute for any other invention reporting required under the terms and conditions of an award.

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

- **Other Products**

Identify any other reportable outcomes that were developed under this project. Reportable outcomes are defined as a research result that is or relates to a product, scientific advance, or research tool that makes a meaningful contribution toward the understanding, prevention, diagnosis, prognosis, treatment, and/or rehabilitation of a disease, injury or condition, or to improve the quality of life. Examples include:

- *data or databases;*
- *biospecimen collections;*
- *audio or video products;*
- *software;*
- *models;*
- *educational aids or curricula;*
- *instruments or equipment;*
- *research material (e.g., Germplasm; cell lines, DNA probes, animal models);*
- *clinical interventions;*
- *new business creation; and*
- *other.*

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?

Provide the following information for: (1) PDs/PIs; and (2) each person who has worked at least one person month per year on the project during the reporting period, regardless of the source of compensation (a person month equals approximately 160 hours of effort). If information is unchanged from a previous submission, provide the name only and indicate “no change.”

<i>Name:</i> <i>Project Role:</i> <i>Researcher Identifier (e.g. ORCID ID):</i> <i>Nearest person month worked:</i>	<i>Juan P Wachs</i> <i>Principal Investigator</i> <i>0000-0002-6425-5745</i> <i>1.12 month</i>
<i>Contribution to Project:</i>	<i>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.</i>
<i>Name:</i> <i>Project Role:</i> <i>Researcher Identifier (e.g. ORCID ID):</i> <i>Nearest person month worked:</i>	<i>Richard Voyles</i> <i>Co-Investigator</i> <i>1.12 month</i>
<i>Contribution to Project:</i>	<i>Actively participated in and advised research assistants Vishnu and Mythra in the research and development of the robotic execution module; 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.</i>
<i>Name:</i> <i>Project Role:</i> <i>Researcher Identifier (e.g. ORCID ID):</i> <i>Nearest person month worked:</i>	<i>Vaneet Aggarwal</i> <i>Co-Investigator</i> <i>1.12 month</i>
<i>Contribution to Project:</i>	<i>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 surges.</i>
<i>Name:</i> <i>Project Role:</i> <i>Researcher Identifier (e.g. ORCID ID):</i>	<i>Yexiang Xue</i> <i>Co-Investigator</i>

<i>Nearest person month worked:</i> <i>Contribution to Project:</i>	<i>1.12 month</i> <i>Provided support and coordinated the work of Md Masudur Rahm providing supervision in all the areas related to machine learning of surgeme recognition and execution.</i>
<i>Name:</i> <i>Project Role:</i> <i>Researcher Identifier (e.g. ORCID ID):</i> <i>Nearest person month worked:</i> <i>Contribution to Project:</i>	<i>Glebys Gonzalez</i> <i>Research Assistant</i> <i>0000-0002-0181-006</i> <i>5.25 months</i> <i>Coordinating the development of the surgeme recognition and execution with all the Research Assistants. Responsible for the data collection with the Taurus robot. Developed the system for surgeme execution with Vishnu Venkatesh and Natalia Sanchez. Actively participated in the publication and dissemination of the project results in highly competitive robotic conferences.</i>
<i>Name:</i> <i>Project Role:</i> <i>Researcher Identifier (e.g. ORCID ID):</i> <i>Nearest person month worked:</i> <i>Contribution to Project:</i>	<i>Md Masudur Rahman</i> <i>Research Assistant</i> <i>0000-0002-3633-0621</i> <i>3 months</i> <i>Responsible for the design, programming, and development of the surgeme recognition module and implemented a recognition system leveraging robotic kinematic and visual information. Working on integrating the recognition module into the close loop execution pipeline.</i>
<i>Name:</i> <i>Project Role:</i> <i>Researcher Identifier (e.g. ORCID ID):</i> <i>Nearest person month worked:</i> <i>Contribution to Project:</i>	<i>Mythra V. Balakuntala</i> <i>Research Assistant</i> <i>0000-0003-2551-4780</i> <i>3 months</i> <i>Responsible for the theory and development of the synchronization of robot and remote simulation. Actively contributed in the data collection of the Taurus Simulator,</i>

	<i>creation of flexible debridement model in Taurus simulator and development of semantic scene recognition. Working on novel methods to encode robot and environment state to update simulation and synchronizing the simulation and robot. Actively participated in publications to robotic conferences.</i>
<i>Name:</i> <i>Project Role:</i> <i>Researcher Identifier (e.g. ORCID ID):</i> <i>Nearest person month worked:</i> <i>Contribution to Project:</i>	<i>Vishnunandan Venkatesh</i> <i>Research Assistant</i> <i>0000-0001-5979-5280</i> <i>3 months</i> <i>Responsible for the design, methodology and development of the Surgeme Execution Unit. Actively contributed in the data collection of the Taurus Simulator and development of scene semantic methods with Vision. Working on novel robot execution methods that leverage scene semantics to perform robot agnostic surgemes. Actively participated in publications in robotic conferences.</i>
<i>Name:</i> <i>Project Role:</i> <i>Researcher Identifier (e.g. ORCID ID):</i> <i>Nearest person month worked:</i> <i>Contribution to Project:</i>	<i>Gregory Hager</i> <i>Co-Investigator</i> <i>0000-0002-6662-9763</i> <i>1</i> <i>Coordinating work at the JHU site including development and use of the da Vinci simulator as well as the DVRK system, data collection from these systems, and work on reinforcement learning methods for surgemes.</i>
<i>Name:</i> <i>Project Role:</i> <i>Researcher Identifier (e.g. ORCID ID):</i> <i>Nearest person month worked:</i> <i>Contribution to Project:</i>	<i>Yotam Barnoy</i> <i>Research Assistant</i> <i>Development and use of the da Vinci simulator as well as the DVRK system, data</i>

	<i>collection from these systems, and work on reinforcement learning methods for surges.</i>
<i>Name:</i>	<i>Molly O'Brian</i>
<i>Project Role:</i>	<i>Research Assistant</i>
<i>Researcher Identifier (e.g. ORCID ID):</i>	
<i>Nearest person month worked:</i>	
<i>Contribution to Project:</i>	<i>Work on reinforcement learning systems and transfer learning from simulation to real-world robotic systems.</i>

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

If there is nothing significant to report during this reporting period, state "Nothing to Report."

If the active support has changed for the PD/PI(s) or senior/key personnel, then describe what the change has been. Changes may occur, for example, if a previously active grant has closed and/or if a previously pending grant is now active. Annotate this information so it is clear what has changed from the previous submission. Submission of other support information is not necessary for pending changes or for changes in the level of effort for active support reported previously. The awarding agency may require prior written approval if a change in active other support significantly impacts the effort on the project that is the subject of the project report.

<i>No changes.</i>

What other organizations were involved as partners?

If there is nothing significant to report during this reporting period, state "Nothing to Report."

Describe partner organizations – academic institutions, other nonprofits, industrial or commercial firms, state or local governments, schools or school systems, or other organizations (foreign or domestic) – that were involved with the project. Partner organizations may have provided financial or in-kind support, supplied facilities or equipment, collaborated in the research, exchanged personnel, or otherwise contributed.

Provide the following information for each partnership:

Organization Name:

Location of Organization: (if foreign location list country)

Partner's contribution to the project (identify one or more)

- *Financial support;*
- *In-kind support (e.g., partner makes software, computers, equipment, etc., available to project staff);*
- *Facilities (e.g., project staff use the partner's facilities for project activities);*
- *Collaboration (e.g., partner's staff work with project staff on the project);*

- *Personnel exchanges (e.g., project staff and/or partner's staff use each other's facilities, work at each other's site); and*
- *Other.*

Nothing to Report.

8. SPECIAL REPORTING REQUIREMENTS

COLLABORATIVE AWARDS: For collaborative awards, independent reports are required from BOTH the Initiating PI and the Collaborating/Partnering PI. A duplicative report is acceptable; however, tasks shall be clearly marked with the responsible PI and research site. A report shall be submitted to <https://ers.amedd.army.mil> for each unique award.

QUAD CHARTS: If applicable, the Quad Chart (available on <https://www.usamraa.army.mil>) should be updated and submitted with attachments.

- 9. APPENDICES:** Attach all appendices that contain information that supplements, clarifies or supports the text. Examples include original copies of journal articles, reprints of manuscripts and abstracts, a curriculum vitae, patent applications, study questionnaires, and surveys, etc.

Transferring Dexterous Surgical Skill Knowledge between Robots for Semi-autonomous Teleoperation

Md Masudur Rahman^{1*}, Natalia Sanchez-Tamayo^{2*}, Glebys Gonzalez², Mridul Agarwal³,
Vaneet Aggarwal^{2,3}, Richard M. Voyles⁴, Yexiang Xue¹, and Juan Wachs²

Abstract—In the future, deployable, teleoperated surgical robots can save the lives of critically injured patients in battlefield environments. These robotic systems will need to have autonomous capabilities to deal with communication delays and unexpected environmental conditions during critical phases of the procedure. Understanding and predicting the next surgical actions (referred as "surgemes") is essential for autonomous surgery. Most approaches for surgeme recognition cannot cope with the high variability associated with austere environments and thereby cannot "transfer" well to field robotics. We propose a methodology that uses compact image representations with kinematic features for surgeme recognition in the DESK dataset. This dataset offers samples for surgical procedures over different robotic platforms with a high variability in the setup. We performed surgeme classification in two setups: 1) No transfer, 2) Transfer from a simulated scenario to two real deployable robots. Then, the results were compared with recognition accuracies using only kinematic data with the same experimental setup. The results show that our approach improves the recognition performance over kinematic data across different domains. The proposed approach produced a transfer accuracy gain up to 20% between the simulated and the real robot, and up to 31% between the simulated robot and a different robot. A transfer accuracy gain was observed for all cases, even those already above 90%.

I. INTRODUCTION

There is an increasing interest in using teleoperated surgical robots for austere environments (such as directly on the battlefield) since they can provide timely interventions to patients with life-threatening injuries [1]. These systems are sensitive to delays intrinsic to the limited bandwidth of many austere environments [2], so there is a need for platforms with semi-autonomous capabilities that can assist the surgeon (or medic) when communication is hindered. These systems require a high level understanding of their state and environment to take over when required. In order to effectively interpret the environment, it is helpful to recognize the current and previous surgical actions, referred to as 'surgemes' [3], that are being preformed by the surgeon. In fact, the accurate recognition of surgeme-like primitives is valuable for many robot application domains beyond surgery, such as manufacturing [4] or waste handling [5].

* These authors contributed equally to this work

¹ Department of Computer Science, Purdue University, West Lafayette, IN, 47907, USA rahman64, yexiang@purdue.edu

² School of Industrial Engineering, Purdue University, West Lafayette, IN, 47907, USA sanch174, gonza337, vaneet, jpwachs@purdue.edu

³ School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, 47907, USA agarw180@purdue.edu

⁴ School of Engineering Technology, Purdue University, West Lafayette, IN, 47907, USA rvoyles@purdue.edu

The use of machine learning approaches for surgeme recognition requires collecting a substantial amount of data, which is challenging to obtain in austere settings [6]. Alternatively, it would be desirable to leverage the abundance of data available from more accessible environments to find recurrent patterns and apply such insights to new scenarios [7]–[9]. Hence, the knowledge learned from numerous accessible platforms could be transferred to deployable robotic platforms in less hospitable environments. However, field medical robots are diverse, holding different kinematic configurations, workspaces, and operate under partially unknown constraints. Such domain differences could hamper state-of-the-art approaches and prevent models learned on one platform to generalize across other platforms of disparate morphologies [10]. Addressing such challenges involves coming up with an effective transfer learning architecture than can generalize over different robots and surgical settings.

This paper presents an approach for identifying surgemes performed by a surgeon through teleoperation that can deal with variable environments and minor shifts in robot hardware. The proposed machine learning architecture leverages both images and kinematic features to transfer knowledge obtained from a surgical simulator to real deployable robots. The features were designed to be applicable to any robotic platform and surgical setup without the need of environment-specific modeling. The architecture has been tested on a surgeme classification task under two scenarios: (1) the source and target data are from the same domain (e.g. same robot) and (2) the source and target data come from different domains (e.g. different robots). The use of robot kinematic and visual data to improve knowledge transfer between different domains is a major enhancement over the current state-of-the-art. The contributions of this paper can be summarized as follows: i) a surgeme classification architecture has been developed for settings with high variability; ii) the impact of adding visual information to kinematic data for surgeme recognition during a transfer scenario is demonstrated.

The rest of this work is presented as follows: Section II discusses the prior work. Section III gives an overview of the robotic dataset used. Section IV describes the methods used for surgeme classification. Section V shows the experimental setup and results. Section VI presents discussion, and Section VII concludes the paper with a discussion on future work.

II. BACKGROUND AND RELATED WORK

Surgical skill modeling [11] consists of decomposing surgical tasks into sets of finite, well defined and quantifiable

maneuvers. Such maneuvers can be used to create datasets of surgical skills, and are referred to as surges [12]. Surge analysis has been used to model skill, to classify tasks, and to assess proficiency [13]. Furthermore, surge recognition provides the opportunity for feedback during surgical training [14] [15]. Finally, surge classification has been identified as a key step in robot automation [13] and is transferable to other domains of robot autonomy.

Commonly adopted approaches for surge recognition operate either on the user space or task space. In the task space, robotic kinematic data have been matched to templates using Hidden Markov Models (HMM), where the surges correspond to one or more states of the HMM [16], [17], [18]. Ahmidi *et al.* also used kinematics to compare benchmarks on approaches for joint segmentation and classification (sparse HMM, Markov semi-Markov conditional random field, and skip-chain conditional random field) with feature-based approaches such as bag of spatio-temporal features and linear dynamical systems [19]. Other approaches that use structured prediction include the use of a Skip-Chain Conditional Random Field (SC-CRF) [20], and a Latent Convolutional Skip-Chain Conditional Random Field (LC-SC-CRF) [21]. The SC-CRF is used to capture state transitions over large periods of time, while the latent convolutional component is added to the CRF to model multiple variations of each action.

The previously mentioned works were developed using the JIGSAW dataset (JHU-ISI Gesture and Skill Assessment Working Set) [7]. The JIGSAW dataset comprises synchronized video and kinematic information for three procedures preformed with the da Vinci Surgical System on a benchmark model: suturing, knot tying, and needle passing. DiPietro *et al.* used a recurrent neural network to classify kinematic data for activity recognition based on the JIGSAW dataset for surge classification and the MISTIC-SL dataset for longer segments called maneuvers [8].

A limiting factor of the JIGSAW dataset is that it does not contemplate variability in the environment or domain, keeping a constant configuration of objects of interest, the initial conditions of the task, and a single robot (the 'da-Vinci'). The lack of variability in task conditions hinders generalization and the capacity of transfer learning. In contrast, Madapana *et al.* [9] used a transfer learning approach based on models trained with kinematic data from a new dataset called the Dexterous Surgical Skill (DESK) dataset. Using this dataset, the authors trained a model on a simulated robot and applied it to different real robots. The approach obtained 55% accuracy in surge recognition over a teleoperated surgical robot when trained only over simulation data and achieved an improvement of 34% when a small percentage of real data was added to the training set. The approach demonstrated the power of transfer learning to augment training in austere environments. Nonetheless, this approach was based solely on kinematic information from the robot and did not consider information related to the state of the environment.

The DESK dataset is challenging because it offers high

variability by randomizing the object locations and initial conditions of the tasks over three different robotic domains (see Fig. 1). For applications with high variation in object placement and appearance, image information is of paramount importance, as it facilitates obtaining features related to changes in the scene that the kinematic information of the robot does not account for. Previous work on surge segmentation and classification has included image information, yielding equally good results as the kinematic approaches. These approaches demonstrate that both kinematics and video data capture relevant information for activity classification. The work in [22] modelled each surge video clip as the output of a linear dynamical system, extracting spatio-temporal features from each video to learn a bag-of-features model. Other methods that employ both modalities (e.g. kinematic and visual) have increased performance of surge segmentation and recognition [18], achieving better results than the structured learning approaches that use kinematic data alone. However, these methods did not evaluate the combination of kinematic and visual data to improve transfer learning across domains. The addition of image information allows to address challenging and diverse settings where visual cues might be the unique information available about the state of the environment.

We propose an architecture for surge classification across different domains, that uses a compact image representation with kinematic data. Our system is tested in a transfer learning scenario where models are trained using simulation data and tested with two real robots. Section IV describes the implementation details of the architecture.

III. DATASET DESCRIPTION

The dataset used in this paper is the DESK dataset described in [9]¹. This dataset provides synchronized RGB images, depth and kinematic information for the peg transfer task from multiple domains including two real robots (Taurus II and YuMi) and a simulation environment (Taurus II), as shown in Figure 1. The DESK dataset attempts to account for the complexity of transfer learning between dissimilar robots by introducing intentional variance in peg board configuration and object size and appearance.

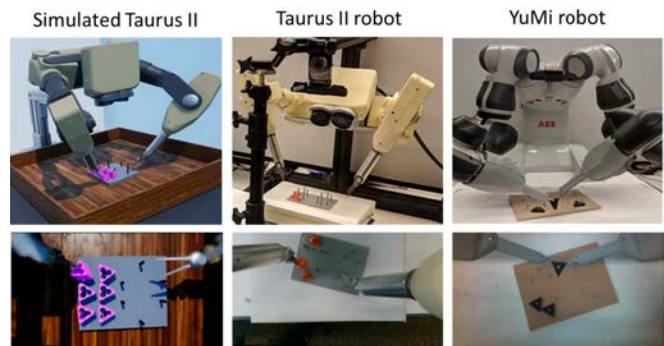


Fig. 1: Robotic system setup for peg transfer task in the DESK dataset

¹Available at https://github.com/nmadapan/Forward_Project

The RGB video, the depth video and the kinematic data are segmented according to surgemes observed in RGB video frames. Additional variability is added to the dataset by randomizing the pick and place locations for the pegs and the orientation of the board, while leaving the order of the pegs to be transferred unrestrained. In addition, successful and failed surgemes are included in the dataset along with the subsequent recovery maneuvers. From the DESK dataset, we used Taurus simulator S1-S5², Taurus S1-S8, and YuMi S1-S8. Details of the surgeme statistics can be found in the Table I. The dataset includes kinematics variables that represent robot position, orientation and gripper status as shown in the Table II.

TABLE I: Surgemes in the peg transfer task. The columns indicate surgeme ID, name of the surgeme, number of instances present for each surgeme for the simulator, real Taurus and the YuMi robot.

ID	Surgeme name	# Sim	# Taurus	# YuMi
S1	Approach peg	90	110	117
S2	Align & grasp	92	111	123
S3	Lift peg	91	111	123
S4	Transfer peg - Get together	84	111	117
S5	Transfer peg - Exchange	80	111	118
S6	Approach pole	76	109	117
S7	Align & place	75	107	116

TABLE II: Kinematic variables. Note that ts is the Unix timestamp, $\vec{J}$ is the vector of joint angles, $\vec{p}$ is the position vector (x , y and z), $\vec{\theta}$ be the Euler angles (yaw, pitch and roll), gs is the gripper state of the end-effector and R be the 3×3 rotation matrix. (adopted from DESK [9])

Taurus		Taurus simulator		YuMi	
ID	Variable	ID	Variable	ID	YuMi
1	ts	1	ts	1	ts
2-13	R and $\vec{p}$	2-4	$\vec{p}$	2-8	$\vec{J}$
	-	5-7	$\vec{\theta}$	9-11	$\vec{p}$
14-16	$\vec{p}$	8-14	$\vec{J}$	12-20	R
17	gs	15	gs	21	gs

IV. METHODOLOGY

Our approach combines features extracted from video image and kinematic data to perform transfer learning in surgeme recognition. The approach is based on robot-agnostic feature extraction that can be applied to different environments in a transfer learning scenario.

An overview of our system architecture is given in figure 2. The proposed approach uses a pretrained Convolutional Neural Network (CNN) to extract meaningful features from video images. The images extracted from the videos were resized from 1920×1080 to a 228×128 . The experiments in this paper are preformed using the CNN model ResNet18 [23] pre-trained on ImageNet. The last two layers (i.e., fully connected and softmax) of ResNet18 were removed to create a high dimensional feature representation for each video frame. The extracted features have dimensions of $1 \times 512 \times 4 \times 8$, which were flattened to a single vector of size $N = 16384$. Due to the large dimension in the output

of the ResNet18 model a *dimensional reduction module* was required to reduce each image/frame features to a lower dimensional vector ($M = 30$, where $M \ll N$). Principal Component Analysis (PCA) was used to reduce the dimension of each frame image to a lower 30-dimensional vector. PCA is a statistical method that applies an orthogonal transformation to convert a set of observations into a set of uncorrelated variables. Intuitively, this transformation extracts components (i.e., features) with the largest possible variance and thus helps classifiers to distinguish between different surgeme classes.

In parallel, we extracted features from kinematic data using a similar approach to the one proposed in the DESK [9], by reducing the kinematic features from multiple domains to the commonly shared features: position, orientation and gripper status of the end-effector (14 features: seven features in each arm). Since each surgeme instance consists of a variable number of frames, we re-sampled (via linear interpolation) the original instances to a fixed number of frames (40) to generate a sequential feature instance for each surgeme. In particular, for the kinematics, we concatenated the 14 features corresponding to each frame, to create a single 560 dimensional vector per surgeme ($40 \times 7 \times 2$). On the other hand, for the video data, we concatenated the 30 features obtained after the PCA reduction into a single feature vector of 1200 dimensions (40×30) that represents each surgeme. Another PCA reduction was applied on the 1200-dimensional feature vector to obtain a 100-dimensional feature vector for entire image sequence of 40 frames.

In our pipeline, visual features and kinematic features came from different distributions. Thus, to take advantage of both visual and kinematic modalities the supervised learning algorithms were trained separately and their output class probabilities were combined as shown in the following equation:

$$P(C) = \lambda P_{kin}(C) + (1 - \lambda) P_{video}(C), \quad (1)$$

where C is the class and $\{P_{kin}(C), P_{video}(C)\}$ are the probabilities given by supervised learning models using kinematic and video features respectively. Here, λ is the hyper-parameter (from 0 to 1), which modifies the contribution of the models based on kinematics or video frames. When λ is set to 0, the classification is based on video data alone. If $\lambda = 1$, the classification uses only kinematic data. Note that the architecture allows to plug in different supervised machine learning models for kinematics and video. Thus, the models for video and kinematics can be designed separately. For simplicity, in our experiments, the same supervised learning model for both video and kinematic features was used.

The final feature set consists of the gripper's position, orientation, status (open/closed) and image features extracted from the pre-trained ResNet18. These features are agnostic to the robot. Thus, the same set can be used even when the task or the robot varies. Having a common set of features facilitates surgeme classification across different domains, since it helps models to leverage on the information coming from another domain.

²a subset from 8 mobile pegboard subjects [9]

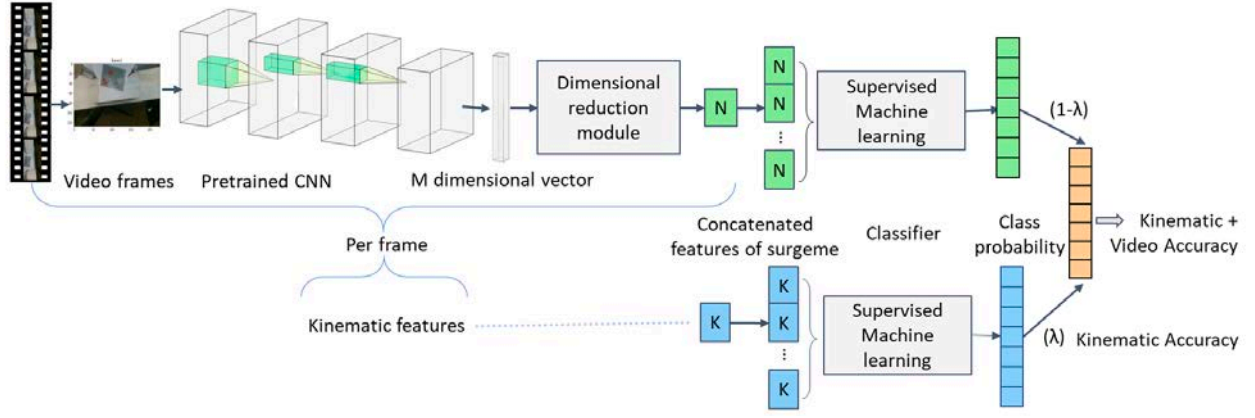


Fig. 2: Architecture overview of the approach, M: Number of dimensional features extracted using pre-trained CNN, K: number of kinematic features, N: number of image features after Dimensional reduction module.

V. EXPERIMENTS AND RESULTS

Experimental Setup: The proposed architecture was tested over two scenarios: no-transfer scenario (train and test data are obtained from the same domain), and domain-transfer scenario (train on one domain and test on the other). The complete domain transfer scenario used the Taurus simulator (S) data or a combination of simulation and real robot data (S+R) during training, while the algorithm was tested entirely on the real robot data. A value α represents the percentage of real data (i.e., target domain) that was added to the simulator data (source domain) for the model training. The rest of the real data (in the target domain) was used as the testing set. Thus, $\alpha = 0$ indicates a complete transfer.

Two supervised learning methods were tested: (1) Random Forest (RF), and (2) Support Vector Machines (SVM), using the scikit-learn [24] implementation. A five-fold cross-validation approach was used with a data split of 80-20% for training and testing respectively. Furthermore, *hyperparameter setting*, we used a *kernel = poly* for SVM classifier. For RF, we set *n_estimators* = 200 (number of trees in the forest), and maximum depth = 10. We set the combined model's hyper parameter λ to 0.5 for RF and to 0.8 for SVM (empirical best).

Results: Table III shows the surgeme recognition results using our approach for kinematics, video information and a combination of both modalities for the non-transfer scenario. Kinematic (Kin) features alone perform substantially better than features based on video frames (Visual). RF performs better than SVM for visual data achieving up to 63% accuracy on the Taurus real robot. Moreover, in most of the cases, accuracies improved when both kinematic data and image features are used (Kin+Visual). In general, adding visual data improved the classification accuracy of surgeme compared to the use of kinematic or visual data independently.

Two transfer learning scenarios were evaluated: (i) Taurus simulator to Taurus real robot (Sim2Taurus) and (ii) Taurus simulator to YuMi (Sim2YuMi). The accuracy results for the domain-transfer scenario: simulator + real-robot (S+R) are presented on Figure 3. The results show that kinematic+visual features surpass the accuracy and consistently outperforms results that use only kinematic features.

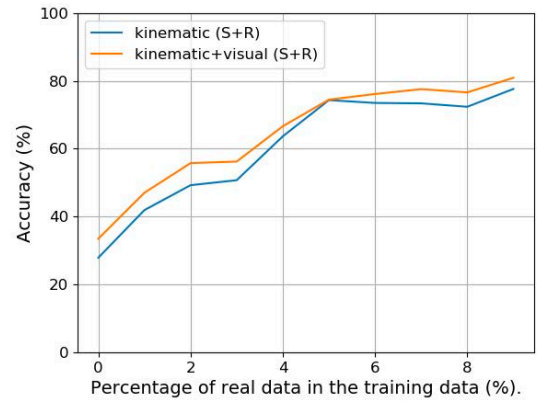


Fig. 3: Performance comparison for transfer learning Sim2Taurus (Random Forest). Overall transfer accuracy improved quickly and it reaches over 93% with only 50% real data in the training. Details in Table IV.

Figure 4 shows that the performance of both the (Visual) and (Kin+Visual) data increases as we add more real-robot data (YuMi) in the training (S+R). The improvement obtained for the transfer learning scenario using (Kin+Visual) data is higher than the knowledge transfer based on kinematic data alone. Hence, when training data is limited on the target domain, the addition of visual data from simulation could help boost initial classification accuracy.

Table IV shows transfer accuracy for kinematic only features (Kin)³, and both kinematic and visual features (Kin+Visual). It also shows a percentage gain which is calculated as

$$Gain = \frac{A_2 - A_1}{A_1} \times 100, \quad (2)$$

where A_1 and A_2 are the accuracies using Kin features, and Kin+Visual features, respectively. It can be observed that adding visual features boosts the transfer accuracy in both Sim2Taurus and Sim2YuMi robot settings achieving up to 20% and 31% gain respectively. Another observation is

³Note that this paper used a subset of simulated data, and thus the kinematic only transfer results are slightly different than that reported in [9]. For instance, RF accuracy from Sim2Taurus at $\alpha = 0$ is 28% as compared to previously reported 34%.

TABLE III: Classification accuracy on the no-transfer scenario using RF and SVM on kinematic only (Kin), visual only (Visual), and both (Kin+Visual) features. In most cases adding visual features with kinematic features improve the accuracy.

Robot	RF			SVM		
	Kin	Visual	Kin+Visual	Kin	Visual	Kin+Visual
Simulator (Taurus)	92.69	56.46	93.03	91.66	33.16	92.51
Taurus	94.68	63.77	96.49	91.43	35.19	92.73
YuMi	96.87	60.65	97.23	92.90	21.91	93.38

TABLE IV: Domain transfer accuracy (Random Forest) when the models are trained on the Taurus simulator robot and tested on real robots (Taurus (Sim2Taurus) and YuMi (Sim2Yumi)). α is the percentage of real-robot data used in the training. Features: kinematic only (Kin), visual only (Visual), and both (Kin+Visual).

$\alpha(\%)$	Sim2Taurus			Sim2Yumi		
	Kin	Kin+Visual	Gain(%)	Kin	Kin+Visual	Gain(%)
0	27.92	33.51	20.00	14.20	14.80	4.24
1	41.94	47.05	12.19	25.64	33.78	31.75
2	49.27	55.22	13.17	37.42	43.07	15.08
3	50.74	56.22	10.82	39.03	46.84	20.00
4	63.65	66.62	4.67	50.38	54.14	7.46
10	78.93	82.54	4.57	74.73	73.40	-1.78
50	93.77	94.03	0.28	88.70	90.14	1.62
90	93.51	96.10	2.77	95.24	96.43	1.25

that the performance improvement is generally higher for smaller α values. Intuitively, when the target domain data is limited during training, visual information can play an important role in increasing initial transfer accuracy. As more target domain data is added to the training, both Kin and Kin+Visual accuracy increase quickly. In particular, when only 50% data is added to the simulator training, the accuracy reaches over 93% and 88% for Sim2Taurus and Sim2YuMi respectively.

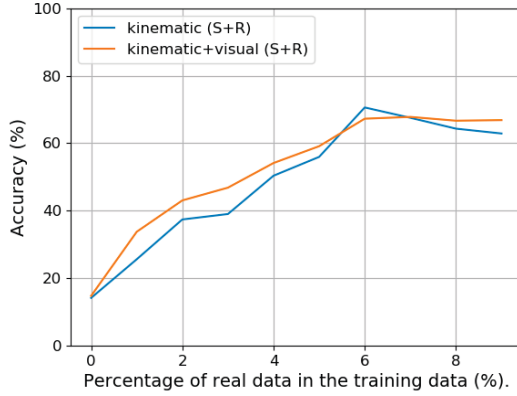


Fig. 4: Performance comparison for transfer learning from Sim2YuMi (Random Forest). Overall transfer accuracy improved quickly and it reaches over 88% with only 50% real data in the training. Details in Table IV.

VI. DISCUSSION

As shown in Section V, higher recognition was generally achieved when the predicted label is obtained from both kinematics and visual information. Even though recognition accuracy was poorer when only visual features were used for the classification model, adding visual to kinematic data incremented recognition performance. Particularly, in the experiment involving transfer learning across domains, visual information complemented and improved recognition in transfer scenarios, especially in cases where there was little to no available training information on the target domain. In

the no-transfer scenario, the recognition accuracy obtained using only kinematic features was already above 90%, so that when visual features scores were included in the prediction, results were only marginally higher.

Image features encode information about the motion of the robot along with information about the environment. However, the encoded features of the image provide a noisier representation for robot motion. Since these features are obtained from a 2D image, which is a projection of the 3D world, the input is already reducing the dimensionality of the spatial data. In contrast, the kinematic data is working at millimeter precision while the image that was fed to the network represents the robot's space at the pixel level.

Features obtained from images also capture diverse information from the scene, environment and relevant objects. These features carry more information about the environment than the kinematic features. Thus, the image features can provide knowledge about the aspects of the task that the kinematics cannot account for.

The extracted kinematic features make no assumptions on the arm's morphology. Thus, the approach can be applied to different robotic systems facilitating a transfer learning scenario. Additionally, the image features extraction module of our architecture is not constrained to any particular CNN model. Thus the ResNet18 can be replaced by other preferred CNN based models.

Although image features extracted from a pre-trained CNN network can be useful to extract meaningful features from domains with limited training data, they fall short when representing the common traits of objects of interest across domains. This is currently a limitation of this work, since the features that are obtained from the ResNet18 are not particularly designed for surgical tasks. Leveraging these commonalities across domains, beyond the differences in environment appearance, might provide information particularly helpful for the transfer learning scenario. For example, between the simulated and the real robot environment, the

appearance of the peg is different although they share the same role in the task.

VII. CONCLUSIONS AND FUTURE WORK

This paper proposed a systematic way to incorporate visual information to improve surgeme classification in surgical robotic tasks. Several experiments were conducted on datasets from three robotic domains. Results show that the visual features improve the performance of surgeme classification when used along with kinematic data. Currently the features are combined using late fusion. Future work can explore early fusion techniques to improve the classification accuracy.

In the transfer scenario, the proposed approach improved the knowledge transfer accuracy between domains by leveraging visual features effectively with the kinematic features while requiring little data from the target domain. In particular, the visual data boosted the transfer accuracy between simulator robot (Taurus) and a different robot (YuMi) by up to 31% when little real data from the target was added with the simulator data for training. For the case of transfer between Taurus simulator to Taurus real robot, an increase of up to 20% was achieved when incorporating visual data to the model.

ACKNOWLEDGMENTS

This work was supported by both the Office of the United States Department of Defense under Award No. W81XWH-18-1-0769 and by the NSF Center for Robots and Sensors for the Human Well-Being under Award No. CNS-1439717. The authors would like to acknowledge Jyothsna Padmakumar Bindu for her feedback during the preparation of this manuscript. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the funders.

REFERENCES

- [1] P. Garcia, J. Rosen, C. Kapoor, M. Noakes, G. Elbert, M. Treat, T. Ganous, M. Hanson, J. Manak, C. Hasser, *et al.*, "Trauma pod: a semi-automated telerobotic surgical system," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 5, no. 2, pp. 136–146, 2009.
- [2] J. Loyall, M. Gillen, J. Cleveland, K. Usbeck, J. Sterling, R. Newkirk, and R. Kohler, "Information ubiquity in austere locations," *Procedia Computer Science*, vol. 10, pp. 170–178, 2012.
- [3] H. C. Lin, I. Shafraan, D. Yuh, and G. D. Hager, "Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions," *Computer Aided Surgery*, vol. 11, pp. 220–230, Jan. 2006.
- [4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469 – 483, 2009.
- [5] T. Soratana, M. V. S. M. Balakuntala, P. Abbaraju, R. Voyles, J. Wachs, and M. Mahoor, "Glovebox handling of high-consequence materials with super baxter and gesture-based programming - 18598," in *Waste Management (WM 2018), 44th International Symposium on, Waste Management*, March 2018.
- [6] S. H. Y. John M. Fossaceca, "Artificial intelligence and machine learning for future army applications," 2018.
- [7] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Béjar, and D. D. Yuh, "JHU-ISI gesture and skill assessment working set (JIGSAWS): A surgical activity dataset for human motion modeling," in *MICCAI Workshop: M2CAI*, vol. 3, p. 3, 2014.
- [8] R. DiPietro, C. Lea, A. Malpani, N. Ahmidi, S. S. Vedula, G. I. Lee, M. R. Lee, and G. D. Hager, "Recognizing Surgical Activities with Recurrent Neural Networks," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016* (S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, eds.), Lecture Notes in Computer Science, pp. 551–558, Springer International Publishing, 2016.
- [9] 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," *arXiv:1903.00959 [cs]*, Mar. 2019. arXiv: 1903.00959.
- [10] N. Makondo, B. Rosman, and O. Hasegawa, "Accelerating model learning with inter-robot knowledge transfer," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2417–2424, May 2018.
- [11] G. P. Moustiris, S. C. Hiridis, K. M. Deliparaschos, and K. M. Konstantinidis, "Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 7, no. 4, pp. 375–392, 2011.
- [12] C. E. Reiley, E. Plaku, and G. D. Hager, "Motion generation of robotic surgical tasks: Learning from expert demonstrations," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 967–970, Aug. 2010.
- [13] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Unsupervised surgical task segmentation with milestone learning," in *Proc. Intl Symp. on Robotics Research (ISRR)*, 2015.
- [14] C. E. Reiley, H. C. Lin, D. D. Yuh, and G. D. Hager, "Review of methods for objective surgical skill evaluation," *Surgical endoscopy*, vol. 25, no. 2, pp. 356–366, 2011.
- [15] N. Padoy and G. D. Hager, "Human-machine collaborative surgery using learned models," in *2011 IEEE International Conference on Robotics and Automation*, pp. 5285–5292, IEEE, 2011.
- [16] C. E. Reiley and G. D. Hager, "Task versus Subtask Surgical Skill Evaluation of Robotic Minimally Invasive Surgery," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2009* (G.-Z. Yang, D. Hawkes, D. Rueckert, A. Noble, and C. Taylor, eds.), Lecture Notes in Computer Science, pp. 435–442, Springer Berlin Heidelberg, 2009.
- [17] L. Tao, E. Elhamifar, S. Khudanpur, G. D. Hager, and R. Vidal, "Sparse Hidden Markov Models for Surgical Gesture Classification and Skill Evaluation," in *Information Processing in Computer-Assisted Interventions* (P. Abolmaesumi, L. Joskowicz, N. Navab, and P. Jannin, eds.), Lecture Notes in Computer Science, pp. 167–177, Springer Berlin Heidelberg, 2012.
- [18] L. Tao, L. Zappella, G. D. Hager, and R. Vidal, "Surgical Gesture Segmentation and Recognition," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013* (K. Mori, I. Sakuma, Y. Sato, C. Barillot, and N. Navab, eds.), Lecture Notes in Computer Science, pp. 339–346, Springer Berlin Heidelberg, 2013.
- [19] N. Ahmidi, L. Tao, S. Sefati, Y. Gao, C. Lea, B. B. Haro, L. Zappella, S. Khudanpur, R. Vidal, and G. D. Hager, "A Dataset and Benchmarks for Segmentation and Recognition of Gestures in Robotic Surgery," *IEEE Transactions on Biomedical Engineering*, vol. 64, pp. 2025–2041, Sept. 2017.
- [20] C. Lea, G. D. Hager, and R. Vidal, "An Improved Model for Segmentation and Recognition of Fine-Grained Activities with Application to Surgical Training Tasks," in *2015 IEEE Winter Conference on Applications of Computer Vision*, pp. 1123–1129, Jan. 2015.
- [21] C. Lea, R. Vidal, and G. D. Hager, "Learning convolutional action primitives for fine-grained action recognition," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1642–1649, May 2016.
- [22] B. Béjar Haro, L. Zappella, and R. Vidal, "Surgical Gesture Classification from Video Data," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2012* (N. Ayache, H. Delingette, P. Golland, and K. Mori, eds.), Lecture Notes in Computer Science, pp. 34–41, Springer Berlin Heidelberg, 2012.
- [23] 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.
- [24] 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.

DESK: A Robotic Activity Dataset for Dexterous Surgical Skills Transfer to Medical Robots

Naveen Madapana^{1*}, Md Masudur Rahman^{2*}, Natalia Sanchez-Tamayo^{1*}, Mythra V. Balakuntala³, Glebys Gonzalez¹, Jyothsna Padmakumar Bindu³, L. N. Vishnunandan Venkatesh³, Xingguang Zhang¹, Juan Barragan Noguera¹, Thomas Low⁴, Richard Voyles³, Yexiang Xue², and Juan Wachs¹

Abstract—Datasets are an essential component for training effective machine learning models. In particular, surgical robotic datasets have been key to many advances in semi-autonomous surgeries, skill assessment, and training. Simulated surgical environments can enhance the data collection process by making it faster, simpler and cheaper than real systems. In addition, combining data from multiple robotic domains can provide rich and diverse training data for transfer learning algorithms. In this paper, we present the DESK (Dexterous Surgical Skill) dataset. It comprises a set of surgical robotic skills collected during a surgical training task using three robotic platforms: 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 (e.g. from Taurus to YuMi robot) for a surgical gesture classification task with seven gestures. We explored three different scenarios: 1) No transfer, 2) Transfer from simulated Taurus to real Taurus and 3) Transfer from Simulated Taurus to the YuMi robot. We conducted extensive experiments with three supervised learning models and provided baselines in each of these scenarios. Results show that using simulation data during training enhances the performance on the real robot where limited real data is available. In particular, we obtained an accuracy of 55% on the real Taurus data using a model that is trained only on the simulator data. Furthermore, we achieved an accuracy improvement of 34% when 3% of the real data is added into the training process.

I. INTRODUCTION

Minimally invasive robotic surgery has evident advantages over traditional surgery, such as quick recovery, lower risks and lower catastrophic errors for patients, and thereby have become the standard of care for a wide variety of surgical procedures [1]. However, these techniques require residents to spend a substantial amount of time practicing surgical maneuvers in simulation environments. In particular, the surgical robotic simulators play a crucial role in the training process of residents and novice surgeons, leading to significant improvement of their technical skills gradually and over time [2]. The tasks that are predominantly presented for training using the simulation and bench-top models include,

but are not limited to, peg transfer, pattern cut, suture, and needle passing [3].

There is another key-benefit in using simulation: (a) It provides unlimited amount of training time while avoiding human/animal tissue manipulation or expensive single-use mock models [4] and (b) It allows researchers to collect enormous amount of data at a lower cost and in a scalable manner [5]. While Da-Vinci datasets are publicly available including surgical maneuvers during a variety of procedures, there is a lack of datasets featuring other type of surgical robots which may be less popular. This limits the ability to generalize across other robotic platforms. This limitation is particularly critical when in-field deployable surgical robots are needed (e.g. military, disaster-relief scenarios). The scarcity of publicly available datasets prevents researchers from exploring novel strategies to transfer the knowledge gained in the surgical simulator to a variety of other robots, less common in the Operating Room (OR) but more suitable to field conditions.

The goal of this paper is to provide such dataset that can allow for "transfer learning" across robotic platforms. As a test-case study, we will rely on the peg transfer task, which is a task of primary importance in surgical skill learning. In this regard, this paper introduces a library of motions for the robotic peg transfer task obtained from multiple domains: robotic simulator and two real robots (Taurus and YuMi). We refer to this database as Dexterous Surgical Skills Transfer dataset (DESK). In addition to providing the dataset, we will present a proof-of-concept for learning to classify surges with a very limited or no training information from the real robots. The goal here is to transfer the knowledge gained from the simulator to accelerate the learning on real robots by leveraging on abundant data obtained from the simulation environment. This instance of transfer learning is known as domain adaptation, a scenario in transductive transfer learning [6], where the source and target tasks are the same, but the data distribution of target and source domain are different.

The main contributions of this work are to: 1) Provide an annotated dataset of surgical robotic skills (DESK) collected in three domains: two real robots with different morphology (Taurus II and YuMi) and one simulated robot (Taurus II), and 2) Transfer the knowledge gained from abundant simulation data to real robot's data in context of surgeme classification.

* These authors contribute equally to this work

¹School of Industrial Engineering, Purdue University, West Lafayette, IN, 47907, USA nmadapan, sanch174, gonza337, zhan3275, barragan, jpwachs@purdue.edu

²Department of Computer Science, Purdue University, West Lafayette, IN, 47907, USA rahman64, yexiang@purdue.edu

³College of Technology, Purdue University, West Lafayette, IN, 47907, USA mbalakun, jpadmaku, lvenkate, rvoyles@purdue.edu

⁴SRI International thomas.low@sri.com

II. BACKGROUND AND RELATED WORK

The advancement of surgical robotic activity recognition and semi-autonomous surgeries benefits from the the availability of robotic surgical datasets. Two such prominent datasets are the JIGSAW [7] (JHU-ISI Gesture and Skill Assessment Working Set) and MISTIC-SL (Johns Hopkins Minimally Invasive Surgical Training and Innovation Center; Science of Learning Institute). These datasets comprise procedures preformed with the da Vinci Surgical System on a bench-top model, including synchronized video and kinematic data [8]. A main advantage of these datasets is that they can allow elucidating patterns associated with skill learning. With this goal in mind, surgical tasks are decomposed into a finite set of maneuvers [7]. This process of decomposition is known as surgical skill modelling [9]. In this modelling technique, each surgical skill is represented as a sequence of atomic units referred as gestures or surgemes [7].

The MISTIC-SL dataset contains 49 right-handed trials of a suture throw followed by a surgeon's knot, employing in total four maneuvers: *suture throw*, *knot tying*, *grasp pull run suture*, and *intermaneuver segment* [10]. This dataset has been used to learn representations of surgical motions [11] and recognize surgical activities [10]. However, the MISTIC-SL dataset is not publicly available at the moment. In contrast, the JIGSAW is a public dataset that contains 39 trials of *suturing* task, 36 trials of *knot tying*, and 28 trials of *needle passing*. The data is annotated by one individual and manually segmented into 15 atomic units of intentional surgical activity called "gestures". This dataset has been used for multiple applications such as motion generation of expert demonstrations [12], recognition of surgical gestures [13], and surgical trajectory segmentation [14]. Nonetheless, the data collection of the JIGSAW dataset did not intentionally introduce variability in the environment and initial conditions of the task. In realistic surgical tasks, the videos and kinematics will vary greatly between demonstrations stressing the need of surgical datasets that include variability in the experimental setup to facilitate generalization.

The segmentation and classification of time series data on surgical datasets has been evaluated on the JIGSAW and MISTIC-SL dataset using several approaches. Previous supervised learning methods include the use of hidden Markov models [15] [16], conditional random fields [17], and bag of spatio-temporal features [13]. More recent methods include the use of Recurrent Neural Networks for recognizing surgical activities [10], [11]. Nonetheless, these approaches were tested using data from the same distribution as the training data, and do not account for the disparity encountered from randomized initial conditions.

In surgical data analysis, many state-of-the-art techniques contributions are limited to study-specific data and validation metrics [13]. Recently, several efforts have been made in autonomous classification and execution of surgical tasks using multiple surgical robots and simulation platforms [18]–[20]. However, the obtained models are specific for a given

platform and setup, and cannot be directly transferred to other robots/procedures. As surgical data sets become available, it is desirable to use principles of transfer learning to leverage previous models to accelerate learning in new domains, where it is essential to avoid collecting extensive data to retrain the models.

Research in transfer learning and domain adaptation has leveraged simulation environments to boost learning in real systems [21], [22]. Knowledge transfer between a robot and a simulated environment is challenging due to the differences in data distribution. Simulated environments and real systems are often unlike in terms of interface used for controlling, object appearance, surface dynamics or even physical behaviour of the robots. Methods of domain randomization have been proposed to increase generalization in real scenarios for models trained in simulation [5], [23], by randomizing object position and appearance over the training set.

Transfer learning between dissimilar robots has been extensively studied in the area of Reinforcement Learning (RL) [24]–[27]. The work in [28] shows a modular policy strategy for networks that allows to jointly train data from different robots with a common task, or data coming for the same robot with varying tasks. Also, the work in [29] proposed to reproduce skills of an industrial robot on a flexible robotic prototype designed for surgical tasks. Their approach uses reward profiles learned from a demonstration step (source domain) to learn over the target domain in a RL setup. In domain transfer scenarios, dimensionality reduction techniques can be used to transfer learning between different robots [30]. Bcsi et al. proposed a knowledge transfer approach that is agnostic to the robot parameters, were the source and the target dataset are reduced to a common lower-dimension manifold [30]. This method does not require any kinematic knowledge of the target domain. Instead, the users must propose a bijective mapping between each dataset and the lower dimensional manifold. In contrast with this approach, our method does not require mapping back to the original space. We take a dimensionality reduction approach that intends to preserve the common features and the transfer learning is done directly in the reduced space.

Our DESK dataset provides RGB images, depth and kinematic information for the peg transfer task from multiple domains including two real robots (Taurus II and YuMi) and a simulation environment (Taurus II). These three robotic setups possess inherent variance in peg board configuration and object size and appearance. Additional variability is added to the dataset by randomizing the pick and place locations for the pegs and orientation of the board, while leaving the order of the pegs to be transferred unrestrained. In addition, the dataset contains examples of success and failure of surgemes employed during the task and subsequent recovery maneuvers.

III. DESK DATASET

The dataset contains a library of surgical motions for the peg transfer task using the Taurus II robot from SRI, a simulated Taurus II robot, and the YuMi robot as depicted in Figure 1. The data recordings fulfill the following requirements observed in other surgical robotic datasets [7]: 1) kinematic information of the robotic arms (left and right) at all captured frames, 2) RGB video capturing the task performance, 3) data annotations concerning atomic surgical actions (start frame ID, end frame ID and success or failure). Moreover, the DESK provides both stereoscopic images for the simulated robot and the depth data for the real robots.

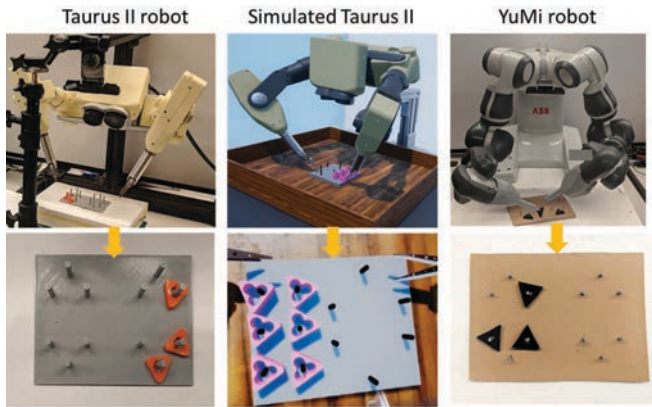


Fig. 1: Experimental setup for data collection on three different robotic domains.

A. Peg transfer surgical task

The peg transfer task is one of the five tasks present in the Fundamentals of Laparoscopic Surgery [31] and has been commonly used to train residents [32], [33]. The task consists of picking an object from a peg board with one robotic arm, transferring it to the other arm and positioning the object over a target peg on the opposite side of the board. These tasks require a high level of sensorimotor skill due to the small clearance between pegs and objects, and the limited maneuverability of the manipulator caused by multiple objects in the workspace.

The peg transfer setup for the DESK dataset has two sets of numbered poles (from 1 to 6), each object has to be picked from its peg with one gripper, transferred to the other gripper and placed in a specified peg on the other side. Variability is introduced in the dataset by randomizing the following elements in the setup: 1) Initial and final positions of objects, 2) direction of the transfer (objects on the left side are transferred to the right and vice versa) and 3) position/orientation of the peg board. Since multiple pegs were present in each trial, the order of the pegs to be transferred was selected by the user.

B. Data description

The kinematic data, the RGB video and the depth video are segmented according to surgical gestures (surges) observed in RGB video frames. A graphical tool was developed

to facilitate the surge annotation based on RGB video recordings. A total of a total of 7 surges were annotated for the peg transfer task. In addition, each surge was marked as a success or a failure. Table I and Figure 2 show the list of annotated surges for the DESK dataset. Each peg transfer video was associated with an annotation file that describes the following for each surge: name, the start and end frame, and whether the surge execution was a success (True) or a failure (False). In addition, timestamps were stored for each recording which allow to synchronize all the recordings (depth, kinematic and controller data) with respect to the RGB videos.

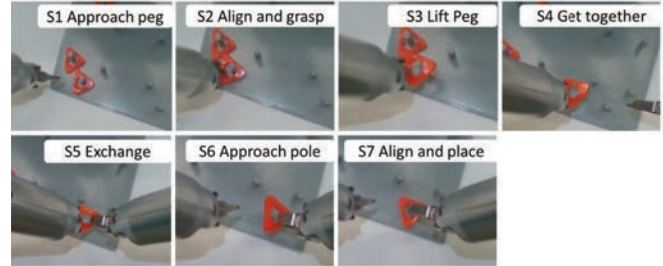


Fig. 2: Surges in the peg transfer task for the Taurus II robot.

TABLE I: Surgical gestures in the peg transfer task. The columns indicate surge ID, name of the surge, number of instances present for each surge for the simulator, real Taurus and the YuMi robot.

ID	Surge name	# Sim	# Taurus	# YuMi
S1	Approach peg	192	111	117
S2	Align & grasp	206	115	123
S3	Lift peg	203	112	123
S4	Transfer peg - Get together	180	113	117
S5	Transfer peg - Exchange	175	113	118
S6	Approach pole	167	109	117
S7	Align & place	163	107	116

C. Data collection with the Taurus robot

The Taurus II robot has two 7 degrees of freedom (DOF) arms that are controlled using the Razer Hydra[®] over a stereoscopic display. The Taurus robot is interfaced with two foot pedals that allow to switch control between the arms and the camera. A *clutch* pedal was used to toggle the robot between operation mode and standby mode to enable the user to reset to an ergonomic configuration for manipulation. In this setup, the robot only moves when the *clutch* of the foot pedal is pressed.

The DESK dataset for the Taurus robot consists of a total of 108 peg transfers, collected for 3 subjects over 36 trials, each trial included 3 randomized peg transfers. Variability is introduced in the data by randomizing three elements per trial: 1) the *tilt* of the peg board (left-tilt, no-tilt, right-tilt), 2) the *position* of the objects and 3) the *starting point* of the peg transfer task (objects on the right side or objects on the left side). The experiment followed a full factorial design for the *starting side* and *tilt* of the peg board. For the

initial positions of the objects, a set of three distinct random numbers from 1 to 6 were chosen.

The data collected from the Taurus robot includes RGB video and depth video recorded from the top view Realsense[®] camera. In addition, the kinematic data of the Taurus robot's end-effector is captured using 16 kinematic variables as shown in Table II. For each arm, we recorded: the rotation matrix of the wrist (nine values), the translation in x, y and z coordinates of the wrist with respect to the robot origin and the gripper state, which is a value between 30 and 100 (30 when completely closed and 100 when the gripper is completely open).

D. Data Collection with the Taurus Simulation Environment

The simulated Taurus robot has two 7 DOF arms that are controlled using the Oculus Rift touch controllers. Similar to the Taurus robot, the simulated Taurus system has a foot pedal that enables the motion of the robot and allows to switch control between the arms and the stereoscopic camera.

The setup for the data collection consisted of six subjects performing a total of 42 trials, where each trial had six peg transfers. Thus, a total of 252 transfers were collected. At the beginning of every trial, a set of random pickup and drop-off locations were generated for the user to follow. The peg board was rigidly attached to the ground i.e. it is at a fixed orientation in the simulator.

This dataset includes the following recordings: the kinematic data of the robot's wrist and RGB videos recorded from the point of view of the user's virtual environment, stereo view of both the left and right robot's cameras (the two stereo videos can be used to compute the depth information). The kinematic data consists of 14 kinematic variables that represent the robot's end-effector pose, as displayed in Table II. It also includes the wrist orientation (yaw, pitch and roll angles), the translation in x, y and z coordinates with respect to the robot's origin, the joint angles (7 DOF) and the gripper state (a value between 30 and 100) for both the arms.

E. Data collection with the YuMi robot

The YuMi collaborative robot has two 7 DOF arms adapted to surgical tasks using 3D printed gripper extensions [34]. The end-effectors of the robot are controlled using the HTC VIVE controllers. A peg transfer setup similar to the Taurus real robot was designed for this system and variability was introduced by randomizing the three elements described in section III-C. The data collection of the YuMi robot is comprised of 40 trials collected by one subject. Each trial included three peg transfers comprising a total of 120 peg transfers. For this setup, the recorded data includes RGB video and depth video obtained from the top view using a Realsense[®] camera. In addition, the kinematic data of each robot arm is captured using 20 kinematic variables that provide joint state information, translation of the tooltip in x,y,z coordinates, rotation matrix of the tooltip with respect to the robot's origin, and gripper state (see Table II).

The YuMi robot is significantly different from the Taurus robot. For instance, the Taurus II is designed specifically for

small dexterous tasks such as bomb disposal and surgery, while the YuMi is suitable for larger workspaces and collaborative tasks. For this reason, the setup for peg transfer using the YuMi was scaled by a factor of 2 (the size of the peg board is larger). Other differences include robot morphology and the interface used for manipulation. The pitch, yaw, and roll angles of the Taurus are computed with respect to the wrist, which causes minimal movement of the wrist when the operator changes the orientation the tool. In contrast, the YuMi robot has a kinematic control that reorients with respect to the tooltip. Given the large distance between the tooltip and the wrist, even minor changes in the orientation of the tool causes large motions at the robot's wrist and the arm configuration.

TABLE II: Kinematic variables. Note that ts is the Unix timestamp, $\vec{J}$ is the vector of joint angles, $\vec{p}$ is the position vector (x, y and z), $\vec{\theta}$ be the Euler angles (yaw, pitch and roll), gs is the gripper state of the end-effector and R be the 3 x 3 rotation matrix.

Taurus		Taurus Simulator		YuMi	
ID	Variable	ID	Variable	ID	YuMi
1	ts	1	ts	1	ts
2-13	R and $\vec{p}$	2-4	$\vec{p}$	2-8	$\vec{J}$
	-	5-7	$\vec{\theta}$	9-11	$\vec{p}$
14-16	$\vec{p}$	8-14	$\vec{J}$	12-20	R
17	gs	15	gs	21	gs

IV. EXPERIMENTS AND RESULTS

The data collected from the simulator and the real robots were in different dimensions as shown in Table II. The first step in the pipeline was to ensure that the dimensions of the data are equivalent in the three domains. Hence, we reduced the feature dimension by considering the features that are commonly shared between these two domains (position, orientation and gripper status of the end-effector). Overall, we considered 14 features per frame (seven features in each arm).

The experiments conducted in this work are two-fold: 1. Train and test on the data obtained from the same domain (no-transfer scenario) and 2. Train on one domain and test on the other (domain-transfer scenario). Furthermore, we considered two kinds of classification tasks: 1. frame-wise and 2. sequence-wise. The former method associates each frame to a particular class label and treats each frame as a sample point (*frame-wise instances*), while the latter method considers the entire surgeme (sequence of frames) as a single sample point (*sequence-wise instances*). Next, we used three supervised learning methods for our experiments: 1. Support Vector Machines (SVM), 2. Random Forest (RF) and 3. Multi-layer Perceptron (MLP). These three learning techniques are commonly used in the machine learning community for creating the baselines for classification. We used the scikit-learn [35] implementation of these models for our experiments.

Hyperparameter setting. We used a linear kernel for SVM classifier. For RF, we set $n_estimators = 200$ (number of trees

in the forest), and maximum depth = 10. For the MLP, we used a hidden layer of size = 100, $\tanh$ as the activation function with *adam* as the optimizer.

Each surge instance consists of a variable number of frames. Hence, we re-sampled (via linear interpolation) the original instances to a fixed number of frames to generate *sequence-wise* instances. Next, we concatenated the seven features corresponding to each frame, for both the arms, to create a single feature vector. In our case, we set the number of frames to 40 and each *sequence-wise* instance is a 560 dimensional vector ($40 \times 7 \times 2$). In our experiments, we did not differentiate the surge instances based on how well they were performed (success or failure).

A. Surge classification

In the first experiment, our goal is to study the classification in the *no-transfer* scenario where the learning model is trained and tested on the data coming from the same domain. In other words, the training and testing data follow the same distribution. We have two such scenarios: train and test on 1. simulator data ($S \rightarrow S$) and 2. real robot data ($R \rightarrow R$). In our experiments, we have used 60-40% split for training and testing respectively. Furthermore, we have used five-fold cross validation approach to tune the learning parameters (or weights) of the models.

Next, we performed the classification on both the frame-wise and sequence-wise instances as shown in the Table III. The *sequence-wise* features contain the temporal information embedded into them. Hence these features give superior accuracy except for the random forest approach on simulator data. Therefore, we used *sequence-wise* features for the experiments associated with the domain transfer. We conducted the two-sided paired t-test and found that SVM significantly outperforms the RF and MLP with the *sequence-wise* features ($p < 0.05$).

TABLE III: Classification accuracy on the no-transfer scenario for both the frame-wise and sequence-wise features. S is the Taurus simulator, R_1 is Taurus robot, and R_2 is the YuMi robot.

	Sequence-wise			Frame-wise		
	RF	SVM	MLP	RF	SVM	MLP
$S \rightarrow S$	88 ± 2	87 ± 1	78 ± 4	86 ± 0	58 ± 1	73 ± 1
$R_1 \rightarrow R_1$	94 ± 2	92 ± 1	92 ± 2	95 ± 0	60 ± 0	92 ± 1
$R_2 \rightarrow R_2$	91 ± 1	93 ± 1	95 ± 1	88 ± 1	48 ± 1	86 ± 1

B. Domain transfer: Simulator to real robot

In this experiment, our goal is to build a learning model that was trained on the data obtained from one domain but tested on the data coming from a new domain. In other words, the input data distribution is different in the train and test datasets. We start with the assumption that it is much easier to collect the data from the simulator when compared to the data from the physical surgical robot. Therefore, the real robot's data is assumed to be limited or not available in the extreme cases. Hence, we trained our models on the

simulator data with a very little or no data from the real robot and test this model on the real robot's data.

To simulate the limited availability of the real data, we added a small fraction (α) of the real data into the training, where α was varied from zero to one. The value of α is defined as the ratio of number of examples of the real data to the simulator data present in the training. An $\alpha = 0$ indicates *complete transfer*, where there is no real data present in the training, while $\alpha = 1$ implies that the data from the simulator and real data are in equal proportions.

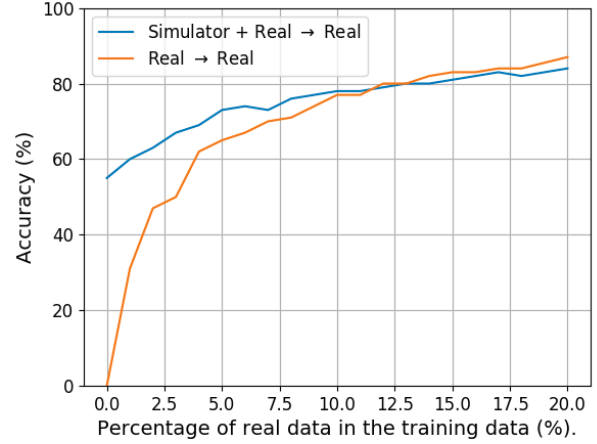


Fig. 3: Performance comparison for transfer learning from Taurus simulator to Taurus Robot using SVM. The blue curve indicates the transfer accuracy and the blue curve indicates accuracy on real data without transfer.

TABLE IV: Domain transfer accuracy when the models are trained on the domain C but tested on domain R . Note that the domain R is the real Taurus domain C is the combination of S and R .

α %	#	RF		SVM		MLP	
		$C \rightarrow R$	$R \rightarrow R$	$C \rightarrow R$	$R \rightarrow R$	$C \rightarrow R$	$R \rightarrow R$
0	0	34 ± 3	0 ± 0	55 ± 2	0 ± 0	40 ± 2	0 ± 0
3	24	53 ± 3	39 ± 4	67 ± 2	50 ± 5	57 ± 3	51 ± 4
6	49	66 ± 1	64 ± 4	74 ± 0	67 ± 3	68 ± 4	72 ± 2
9	73	77 ± 3	73 ± 5	77 ± 1	74 ± 3	74 ± 1	80 ± 3
12	98	78 ± 1	80 ± 3	79 ± 2	80 ± 1	78 ± 1	85 ± 1
15	123	84 ± 2	84 ± 2	81 ± 1	83 ± 0	82 ± 2	86 ± 1
18	147	85 ± 0	85 ± 0	82 ± 1	84 ± 1	79 ± 3	88 ± 1

Moreover, *accuracy* (γ - the percentage of test examples that are correctly classified) and confusion matrices are used as performance metrics to evaluate how well the model is performing on the unseen data. Let us define $\gamma_{A \rightarrow B}^M$ as the test accuracy of the model M trained on the data from domain A and tested on the data from domain B . Note that $\gamma_{A \rightarrow B}^M$ depends on the value of α . Also, let us define C as the domain that is a combination of both real and simulator domains. The goal of this study is to determine the behavior of $\gamma_{C \rightarrow R}^M$ (trained on both simulator and real data and tested on the real data) and $\gamma_{R \rightarrow R}^M$ (trained and test on real data) as the value of α increases.

Figure 3 shows the behavior of $\gamma_{R \rightarrow R}^{SVM}(\alpha)$ and $\gamma_{C \rightarrow R}^{SVM}(\alpha)$ for SVM classifier. In the extreme case when there is no real data in the training process (i.e. $\alpha = 0$ - *complete transfer*), the accuracy is approximately 55%. As the value of α gradually increases from 0 to 0.2, the value of $\gamma_{C \rightarrow R}^{SVM}$ increases and converges in comparison to $\gamma_{R \rightarrow R}^{SVM}$. At $\alpha = 0.18$, the values of $\gamma_{C \rightarrow R}^{RF}$ and $\gamma_{R \rightarrow R}^{RF}$ are approximately equal to 80%. Moreover, this plot shows that adding the simulator data into the training procedure helps improve the accuracies considerably on the real data.

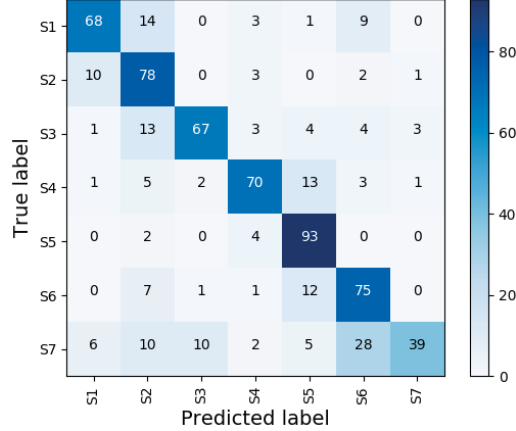


Fig. 4: Confusion matrix for transfer learning with SVM using 5% real Taurus robot data.

Table IV shows the transfer accuracies obtained using three learning models for a range of values of α . Note that there are 1286 samples in the simulator domain. The second column (#) is the number of examples of the real data present in the training procedure. In the first row, it was assumed that there were no real data examples in the training, hence accuracy in $R \rightarrow R$ is 0. The confusion matrix presented in Figure 4 shows the transfer accuracies for all the classes at $\alpha = 0.05$. Overall, the Table IV shows that it is beneficial to augment the training data with the samples obtained from robotic simulators. Irrespective of the learning model, it helps greatly enhance the performance of those models on the real robot's data.

In the last experiment, our goal is to verify the transfer across the robots i.e. to train the model on the simulation data of Taurus robot and test on the data of YuMi robot. The workspace, morphology and dimension of the data of these robots are significantly different. Hence we used the principal component analysis (PCA) to rank the features based on the Eigen values and transform the data into the Eigen space. Next, we chose a fixed number of features (a value of 8 is used in our experiments) in the Eigen space in order to unify the feature dimensions. We followed a similar experimental design to test the transfer of knowledge from Taurus simulator to YuMi domain. The accuracy of *complete transfer* scenario is close to the random accuracy indicating that the model is overfitting to the simulator as

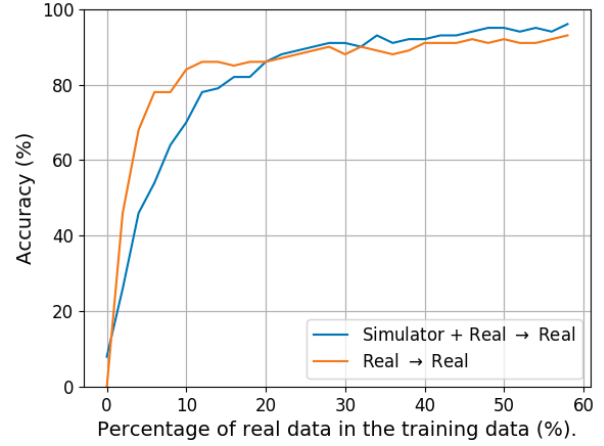


Fig. 5: Performance comparison for transfer learning from the Taurus simulator to the YuMi Robot.

there is no data from the real robot in the training. However, a slight increase in the number of examples of YuMi data in the training, the transfer accuracies swiftly increase. For instance, when the number of YuMi examples = 100, the transfer accuracy is approximately 80% as shown in the Figure 5. When we add more than 200 examples of YuMi in the training, the transfer accuracies marginally surpasses the accuracies in $R \rightarrow R$ scenario. This shows that the transferring the knowledge between two completely different robots is a challenging task and requires relatively more examples to achieve superior transfer accuracies.

V. DISCUSSION AND FUTURE WORK

The transfer learning task considered in this paper is an example of *domain adaptation* where the data distribution of the source and target domains are considerably different while the class labels remain the same. In our experiments, the source domain is nothing but the simulator where a large amount of data is available whereas the target domain is the real Taurus robot. The interaction modality of the Taurus robot and the simulated robot are completely dissimilar. Given that the configuration of the real Taurus robot and the simulator is similar, transferring the knowledge to the physical domain is relatively easier in the case of the real Taurus robot in comparison to the YuMi robot. Hence the *transfer accuracy* is significantly higher for the real Taurus. Furthermore, the *transfer accuracy* obtained on the YuMi robot increases with the increase in α and surpasses $R \rightarrow R$ scenario for $\alpha > 0.2$. In other words, the amount of real data needed to achieve the *transfer accuracy* of 80% is much higher for the YuMi ($\alpha = 0.3$) in comparison to real Taurus ($\alpha = 0.12$). The relatively lower transfer accuracies obtained on YuMi data show that the transfer from one robot to another robot is a challenging task.

It is worth revisiting the process of creating the *sequence-wise* features. Given that the surgeme instances are of different length, we used linear interpolation to ensure that the instances are of uniform length. This method of interpolation,

elongates the surges that are shorter and compresses the surges that are longer. It is interesting to study the effect of re-sampling on the *transfer accuracy*. Nevertheless, the issues associated with re-sampling can be eliminated by using the deep models that are specialized for the problems with non-uniform features lengths such as Long-Short-Term-Memory models or conditional random fields.

Furthermore, we mentioned in the results that the *sequence-wise* features provide significantly better accuracies in comparison to the *frame-wise* features for the transfer learning tasks. Note that the sequence classification approach assumes that the test data is annotated beforehand with respect to the start and end of the surges. However, this assumption is not valid when we want to deploy these trained classifiers in real-time as the real-time data cannot be segmented beforehand. In contrast, the *frame-wise* classification approach does not require the surges to be segmented. Hence the *frame-wise* classifier can potentially act as the model that can be used to know the start and the ending point of the surges.

In this regard, we have made our database publicly available to encourage researchers to further investigate this problem of transfer learning between the robots or from simulator to a real robot. In addition to the RGB-D videos and the kinematic data, we also provide bounding boxes of the objects and pegs for each frame. These annotations are created in a semi-autonomous manner i.e. first, the color-based image processing techniques were used to create the bounding boxes automatically and next, a human annotator was asked to verify those annotations and manually annotate the frames with erroneous annotations. The instructions to obtain the data are available at https://github.com/nmadapan/Forward_Project.git.

The potential future works and applications of our dataset include: 1. Incorporating the object bounding boxes and visual data (RGB-D images) into the feature vectors to improve the transfer accuracies, 2. Developing task specific deep models with the goal of transferring the knowledge from one robot to the other, 3. Learn to predict the surges from one physical robot instead of the simulator and test on an other physical robot (in our case, train on YuMi and test on Taurus), 4. Learning to predict the surges only with the partial information, and 5. Developing the learning models that do not require the surges to be segmented beforehand.

VI. CONCLUSIONS

The main goal of this paper is to learn to transfer the knowledge (surgical skills in our case) from one domain (either simulator or a physical robot) to another domain (physical robot). Previous datasets concerned with surgical tasks were mainly focused on a unique robotic platform (e.g. da-Vinci). This limits the researchers to explore novel ways to transfer the surgical skills learned from one platform to the other. Therefore, it is essential to have the data collected from various robotic platforms. Hence, we created a database DESK of surgical robotic skills (peg transfer task) collected from three domains: simulated Taurus robot, real Taurus

robot and YuMi industrial robot. In addition to developing the dataset, this paper proposes a, simple yet effective, transfer learning methodology to improve the learning for a surgical gesture classification task over real robot's data using the data obtained from the simulation. We conducted an experiment on the dataset and presented three supervised models as baselines for surge classification. Results show that augmenting the training data with simulator data considerably improves the accuracies of prediction on the real data. More specifically, in the extreme case when there is no real Taurus robot's data present in the training, the *transfer accuracy* on the real Taurus data is 55%. Furthermore, the *transfer accuracy* increases significantly as the real robot's data is added to the training process.

ACKNOWLEDGMENTS

This work was supported by both the Office of the Assistant Secretary of Defense for Health Affairs under Award No. W81XWH-18-1-0769. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the funders.

REFERENCES

- [1] A. Chellali, L. Zhang, G. Sankaranarayanan, V. S. Arikatla, W. Ahn, A. Derevianko, S. D. Schwartzberg, D. B. Jones, M. DeMoya, and C. G. L. Cao, "Validation of the VBLAST peg transfer task: a first step toward an alternate training standard," *Surgical Endoscopy*, vol. 28, pp. 2856–2862, Oct. 2014.
- [2] R. Korets, A. C. Mues, J. A. Graversen, M. Gupta, M. C. Benson, K. L. Cooper, J. Landman, and K. K. Badani, "Validating the Use of the Mimic dV-trainer for Robotic Surgery Skill Acquisition Among Urology Residents," *Urology*, vol. 78, pp. 1326–1330, Dec. 2011.
- [3] A. J. Hung, I. S. Jayaratna, K. Teruya, M. M. Desai, I. S. Gill, and A. C. Goh, "Comparative assessment of three standardized robotic surgery training methods," *BJU International*, vol. 112, no. 6, pp. 864–871, 2013.
- [4] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, May 2018.
- [5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, Sept. 2017.
- [6] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, Oct. 2010.
- [7] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Bjar, and D. D. Yuh, "JHU-ISI gesture and skill assessment working set (JIGSAWS): A surgical activity dataset for human motion modeling," in *MICCAI Workshop: M2CAI*, vol. 3, p. 3, 2014.
- [8] Y. Gao, S. S. Vedula, G. I. Lee, M. R. Lee, S. Khudanpur, and G. D. Hager, "Unsupervised surgical data alignment with application to automatic activity annotation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4158–4163, May 2016.
- [9] G. P. Moustiris, S. C. Hiridis, K. M. Deliparaschos, and K. M. Konstantinidis, "Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 7, no. 4, pp. 375–392, 2011.
- [10] R. DiPietro, C. Lea, A. Malpani, N. Ahmidi, S. S. Vedula, G. I. Lee, M. R. Lee, and G. D. Hager, "Recognizing Surgical Activities with Recurrent Neural Networks," in *Medical Image Computing and Computer-Assisted Intervention MICCAI 2016* (S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, eds.), Lecture Notes in Computer Science, pp. 551–558, Springer International Publishing, 2016.

- [11] R. DiPietro and G. D. Hager, "Unsupervised Learning for Surgical Motion by Learning to Predict the Future," in *Medical Image Computing and Computer Assisted Intervention MICCAI 2018* (A. F. Frangi, J. A. Schnabel, C. Davatzikos, C. Alberola-Lpez, and G. Fichtinger, eds.), Lecture Notes in Computer Science, pp. 281–288, Springer International Publishing, 2018.
- [12] C. E. Reiley, E. Plaku, and G. D. Hager, "Motion generation of robotic surgical tasks: Learning from expert demonstrations," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 967–970, Aug. 2010.
- [13] N. Ahmidi, L. Tao, S. Sefati, Y. Gao, C. Lea, B. B. Haro, L. Zappella, S. Khudanpur, R. Vidal, and G. D. Hager, "A Dataset and Benchmarks for Segmentation and Recognition of Gestures in Robotic Surgery," *IEEE Transactions on Biomedical Engineering*, vol. 64, pp. 2025–2041, Sept. 2017.
- [14] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Transition State Clustering: Unsupervised Surgical Trajectory Segmentation for Robot Learning," in *Robotics Research: Volume 2* (A. Bicchi and W. Burgard, eds.), Springer Proceedings in Advanced Robotics, pp. 91–110, Cham: Springer International Publishing, 2018.
- [15] C. E. Reiley and G. D. Hager, "Task versus Subtask Surgical Skill Evaluation of Robotic Minimally Invasive Surgery," in *Medical Image Computing and Computer-Assisted Intervention MICCAI 2009* (G.-Z. Yang, D. Hawkes, D. Rueckert, A. Noble, and C. Taylor, eds.), Lecture Notes in Computer Science, pp. 435–442, Springer Berlin Heidelberg, 2009.
- [16] L. Tao, E. Elhamifar, S. Khudanpur, G. D. Hager, and R. Vidal, "Sparse Hidden Markov Models for Surgical Gesture Classification and Skill Evaluation," in *Information Processing in Computer-Assisted Interventions* (P. Abolmaesumi, L. Joskowicz, N. Navab, and P. Jannin, eds.), Lecture Notes in Computer Science, pp. 167–177, Springer Berlin Heidelberg, 2012.
- [17] L. Tao, L. Zappella, G. D. Hager, and R. Vidal, "Surgical Gesture Segmentation and Recognition," in *Medical Image Computing and Computer-Assisted Intervention MICCAI 2013* (K. Mori, I. Sakuma, Y. Sato, C. Barillot, and N. Navab, eds.), Lecture Notes in Computer Science, pp. 339–346, Springer Berlin Heidelberg, 2013.
- [18] B. Kehoe, G. Kahn, J. Mahler, J. Kim, A. Lee, A. Lee, K. Nakagawa, S. Patil, W. D. Boyd, P. Abbeel, and K. Goldberg, "Autonomous multilateral debridement with the Raven surgical robot," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1432–1439, May 2014.
- [19] J. Schulman, A. Gupta, S. Venkatesan, M. Tayson-Frederick, and P. Abbeel, "A case study of trajectory transfer through non-rigid registration for a simplified suturing scenario," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4111–4117, Nov. 2013.
- [20] D. Seita, S. Krishnan, R. Fox, S. McKinley, J. Canny, and K. Goldberg, "Fast and Reliable Autonomous Surgical Debridement with Cable-Driven Robots Using a Two-Phase Calibration Procedure," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6651–6658, May 2018.
- [21] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4243–4250, May 2018.
- [22] K. Fang, Y. Bai, S. Hinterstoisser, S. Savarese, and M. Kalakrishnan, "Multi-Task Domain Adaptation for Deep Learning of Instance Grasping from Simulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3516–3523, May 2018.
- [23] S. James, A. J. Davison, and E. Johns, "Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task," *arXiv:1707.02267 [cs]*, July 2017. *arXiv: 1707.02267*.
- [24] S. Barrett, M. E. Taylor, and P. Stone, "Transfer learning for reinforcement learning on a physical robot," in *Ninth International Conference on Autonomous Agents and Multiagent Systems-Adaptive Learning Agents Workshop (AAMAS-ALA)*, vol. 1, 2010.
- [25] S. Mahadevan, "Enhancing transfer in reinforcement learning by building stochastic models of robot actions," in *Machine Learning Proceedings 1992*, pp. 290–299, Elsevier, 1992.
- [26] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," *arXiv preprint arXiv:1610.04286*, 2016.
- [27] M. E. Taylor, H. B. Suay, and S. Chernova, "Integrating reinforcement learning with human demonstrations of varying ability," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 617–624, International Foundation for Autonomous Agents and Multiagent Systems, 2011.
- [28] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2169–2176, IEEE, 2017.
- [29] M. S. Malekzadeh, D. Bruno, S. Calinon, T. Nanayakkara, and D. G. Caldwell, "Skills transfer across dissimilar robots by learning context-dependent rewards," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1746–1751, Nov. 2013.
- [30] B. Bocsi, L. Csató, and J. Peters, "Alignment-based transfer learning for robot models," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, IEEE, 2013.
- [31] E. M. Ritter and D. J. Scott, "Design of a Proficiency-Based Skills Training Curriculum for the Fundamentals of Laparoscopic Surgery," *Surgical Innovation*, vol. 14, pp. 107–112, June 2007.
- [32] N. A. Arain, G. Dulan, D. C. Hogg, R. V. Rege, C. E. Powers, S. T. Tesfay, L. S. Hynan, and D. J. Scott, "Comprehensive proficiency-based inanimate training for robotic surgery: reliability, feasibility, and educational benefit," *Surgical Endoscopy*, vol. 26, pp. 2740–2745, Oct. 2012.
- [33] R. A. Joseph, A. C. Goh, S. P. Cuevas, M. A. Donovan, M. G. Kauffman, N. A. Salas, B. Miles, B. L. Bass, and B. J. Dunkin, "Chopstick surgery: a novel technique improves surgeon performance and eliminates arm collision in robotic single-incision laparoscopic surgery," *Surgical Endoscopy*, vol. 24, pp. 1331–1335, June 2010.
- [34] N. Sanchez-Tamayo and J. P. Wachs, "Collaborative Robots in Surgical Research: A Low-Cost Adaptation," in *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, HRI '18*, (New York, NY, USA), pp. 231–232, ACM, 2018. event-place: Chicago, IL, USA.
- [35] 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.

Background

Surgical robotics has improved the rate of success and recovery of patients in minimally invasive surgery. These accomplishments usually involve controlled environments, where the teleoperator is in the same room as the robot and the patient. However, the conditions in the field can severely hinder the teleoperation process. Thus, relying on autonomy in situations of limited bandwidth, latency, and loss-of-signal is critical for telerobotic systems. Obtaining surgical data in battlefield austere scenarios can be very challenging. Conversely, data collection on large amounts of procedures can be done seamlessly in conventional Operating Rooms (OR). Da-Vinci datasets are publicly available, including surgical maneuvers for multiple procedures. These datasets, though rich in amount of procedures, severely lack the information to generalize across robotic platforms. This limitation is particularly critical for in-field deployable surgical robots, since they can potentially have different morphologies (e.g. Taurus, Phantom, and Raven robots). The scarcity of publicly available datasets prevents researchers from exploring knowledge transfer strategies against a benchmark. This includes knowledge transfer between simulated vs real data using the same platform and scenarios with dissimilar robots.

Moving from robotic surgery in the OR to the battlefield can benefit from transfer learning. The knowledge obtained from controlled scenarios can be leveraged to learn surgical procedures in different robotic platforms that could be deployed in the battlefield. This work addresses the problem of transfer learning across different domains, from robot simulation (Taurus II simulated robot) to two real robots with different morphologies: (Taurus II robot and the YuMi robot). Thus, we collected kinematic and video data from multiple robots in a wide range of environments.

Methods

Our work presents a test-case study for transfer learning between different robotics domains, for the problem of surgical gesture classification in the peg transfer task. Peg transfer is one of the five fundamental tasks present in laparoscopic training. This task is composed of 7 surgical gestures (surges) that encompass its main segments. We explored three different transfer learning scenarios: 1) No transfer (for the Taurus robot and the YuMi robot), 2) Transfer from the simulated Taurus to the real Taurus and 3) Transfer from simulated Taurus to the YuMi robot. In the no-transfer scenario, the training and testing data were obtained from the same domain; whereas in the domain-transfer scenario, the training data comes from one domain (source) and the testing from another (target). The collected dataset DESK (Dexterous Surgical Skill) comprises a set of surgical robotic skills using three robotic platforms: the Taurus II robot, Taurus II simulated robot, and the YuMi robot. Additionally, it facilitates developing novel approaches for surgical classification and learning tasks for diverse robot modalities.

The classification task associates the entire surgical gesture to a particular class, treating the sequence of frames as a single sample point. The transfer learning approach reduces the dimensionality of the data by considering the features that are commonly shared

between the source and target domains. In the case of knowledge transfer between the simulated and the real Taurus, only the position, orientation and gripper status of the end-effector were considered. For the transfer between the simulated Taurus and the YuMi the workspace, there was an additional challenge involving their different morphologies. Therefore, principal component analysis (PCA) was used to rank and unify the features' dimensions from the source and target domains. Then, three supervised learning methods were used to perform surgical classification: 1. Support Vector Machines (SVM), 2. Random Forest (RF) and 3. Multi-layer Perceptron (MLP). Different percentages of source domain data were used to test the effectiveness of the transfer strategy.

Results

Results show that using simulation data during training enhances the performance on the real robot where limited real data is available. We obtained an accuracy of 55% with the real Taurus data as the target domain and the simulated Taurus as the source domain. Furthermore, we achieved an accuracy improvement of 34% when 3% of the target domain data was combined with the source domain data for training. In the case of when Yumi was the target domain, the simulated source domain did not provide enough information to achieve an accuracy higher than random ($1/7 = 14\%$ for 7 classes); indicating that the model was overfitting to the simulator when no target data was included in the training set. However, a slight increase in the number of examples of the target YuMi data (15%) in the training improved the classification accuracies up to 80%. For instance, when the number of YuMi examples = 100, the transfer accuracy is approximately 80%.

Conclusions

Our experimental results focus on the knowledge transfer of surgical skills from one domain (source) to another (target). In our particular case, the source was a simulated Taurus platform, and the target was in the real world. Two different target domains were tested, a real Taurus system and a Yumi robot which as a different morphology. We created a dataset of surgical robotic skills called DESK with data from all three domains. We trained three supervised models and presented them as baselines for surgical gesture classification. The potential for transfer learning of the dataset was tested by training only with a small percentage of the target domain. In the extreme case when there was no real Taurus robot's data present in the training, the transfer accuracy on the target Taurus data was 55%. This accuracy increased to 89% when 3% of the target Taurus domain was added.

The results show that the source domain can be used to augment the training data of the target domain. This solidifies the idea that the surgical data from the OR can be used for deployable surgical robots in the battlefield. Future work focuses on expanding the dataset to include more complex surgical procedures, that are more relevant in the battlefield scenario, such as debridement. This procedure has been found essential for initial wound management in trauma and military wounds. Moreover, the debridement

task presents many difficulties due to the dynamics of cutting and manipulating flexible tissue which are a fundamental challenge in more realistic surgical tasks. Further developments in the system include an early surgical gesture detection scheme trained using only partial gesture information. Such system could potentially assist a remote teleoperator in real time and reduce the latency effects in the communication with the semi-automated robot in the battlefield.