Analysis of Gesture Recognition Models Using Accessible Robotic Surgical Systems

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Sara Liu

Spring 2023

Homa Alemzadeh, Department of Electrical and Computer Engineering, Computer Science

Kay Hutchinson, Department of Electrical and Computer Engineering

Introduction

Motivation

Surgical robots are complex Cyber-Physical Systems (CPS) that enhance the hand movement of surgeons and have been increasingly used to address a wide range of surgical operations (Cleveland Clinic, n.d.; Sheetz et al., 2020). As with any technology, robotic surgery comes with its benefits and drawbacks. Some major advantages include greater precision, better visualization with 3D capabilities, and small instruments that allow surgeons to perform surgery inside the body rather than outside. Additionally, robotic surgery leaves patients with less pain during recovery, smaller risk of infection, reduced blood loss, less time spent in the hospital, and smaller scars (Cleveland Clinic, n.d.). However, robotic surgery requires specially trained surgeons due to its unique functionality. It can also cause nerve damage and compression and can potentially malfunction during an operation. Additionally, if there are complications, the surgeon may need to change to an open procedure with larger incisions (Cleveland Clinic, n.d.). Thus, it is critical for safety and efficiency to be top priorities in the field of robotic surgery.

State of the Art

Surgical robots can provide both digital video and quantitative motion trajectories of instruments during surgical operations, allowing surgical actions to be analyzed in a way that is not possible with traditional methods of non-robotic surgery. Researchers can use this data to train computational models that give insights into the accuracy, quality, correctness, and efficiency of hand movements, advancing understanding and enabling the improvement of robot-assisted procedures (van Amsterdam et al., 2021). For example, researchers have analyzed kinematic data and segmented surgical operations at various levels to test models. Two of these levels are tasks and gestures. Tasks are defined as short engagements such as suturing, needle passing, and knot tying. Tasks can be divided into gestures, which are surgical movements made with a specific purpose, such as reaching for a needle with the right hand or pulling a suture with the left hand (Ahmidi et al., 2017).

Hutchinson, Li, et al. (2021) developed a rubric for identifying executional and procedural errors in tasks and gestures by analyzing public kinematic and video data. This data was collected by the da Vinci Surgical System (dVSS) and is part of the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) dataset (Hutchinson, Li, et al., 2021). Van Amsterdam et al. (2021) and Goldbraikh et al. (2022) also studied gesture recognition in different types of surgery; van Amsterdam et al. (2021) analyzed several models of automatic gesture recognition, and Goldbraikh et al. (2022) used a sensor system similar to the one I used to detect surgical gestures and tools employed during open surgery simulations.

Gestures can be further decomposed into motion primitives (MPs), which describe basic surgical actions restricted to a smaller set of modular gesture types that are applied to a tool and object (Ahmidi et al., 2017). MPs are defined to cause changes in the surgical context, which describe contact and hold relations for the left and right graspers well as the object states (Hutchinson, Reyes, et al., 2022). Hutchinson, Reyes, et al. (2022) developed an app to help label surgical context in trials from a surgical set called COMPASS, which includes tasks from three publicly available datasets – JIGSAWS, DESK, and ROSMA. They automatically translated the context data into MPs, supporting the training of action recognition models based on MP workflows instead of gestures (Hutchinson, Reyes, et al., 2022).

Problem Statement

While the current designs described above show significant progress into improving the safety of robotic surgery, they lack the data needed for research using accessible surgical systems. The studies that Hutchinson, Li, et al. (2021) and van Amsterdam et al. (2021) performed depended on the JIGSAWS dataset captured from the dVSS. However, the dVSS is proprietary to hospitals and requires additional proposals and external approval to directly gather kinematic data, so it is not easily available to researchers. Additionally, even though the JIGSAWS dataset addresses the issue of having inconsistent evaluation metrics across different studies by providing a standardized, public dataset, its small size does not work well with the training of large neural network architectures (van Amsterdam et al., 2021). While Hutchinson, Reyes, et al. (2022) aggregate data from more than just the JIGSAWS dataset, they do not explore the usage of alternative robotic surgical systems to collect data as the study relies on previously available public datasets. Goldbraikh et al. (2022) investigates the use of a newly emerging method of attaching motion sensors to the users' hands to collect data, but the study focuses on open surgery and not robotic surgery. Thus, there is an inadequate amount of exploration into the use of accessible robotic surgical systems to gather data in the current body of research.

To address the aforementioned shortcomings, I compare the accuracy of surgical activity segmentation models in identifying gestures using data collected from two different methods. I use this information to assess the feasibility of using an independent system to collect data from surgical robots and the applicability of using that system as part of future research to analyze safety incidents and provide simulation-based training and automated assessments (van Amsterdam et al., 2021).

One robotic platform that I use is the Raven, an open-architecture surgical robot which mimics the dVSS and can collect kinematic, video, and system log files (Li et al., 2019). The kinematic data from the Raven is based on motor encoder values and forward kinematic equations. The robotic surgery research community has been using the Raven since 2009, and medical students also resort to systems like the Raven to achieve adequate training due to access limitations with the dVSS (Glassman et al., 2014; Li et al., 2019). Thus, it is important to evaluate action prediction models based on the Raven due to its heavy use in the research and medical training communities.

The other apparatus I use is the Data Collection System (DCS), which is an independent system developed by UVA researchers and comprised of a trakSTAR tracking device, four sensors, and a ZED Explorer. The trakSTAR is an electromagnetic tracker which can trace the position and orientation of several sensors within a short range of motion (Ascension Technology Company, 2017). The ZED Explorer is an application that allows for live previewing and recording using a camera (Stereolabs, 2023). The DCS can also collect kinematic and video data like the Raven, but the kinematic data is collected from the set of sensors attached to the graspers of a surgical robot such as the Raven or dVSS. In this technical project, the sensors were attached to graspers controlling the Raven due to the difficulty of receiving permission to use the dVSS in the University Hospital.

The primary task of this technical project is to compare how well machine learning models perform for data collected simultaneously from the Raven and the DCS attached to the Raven to see if data from a platform-independent collection method can be used to train models with comparable performance. In order to label the video data objectively, I used a tool similar to the one that Hutchinson, Reyes, et al. (2022) developed to perform context labeling, but modified to perform gesture labeling. Afterwards, I calibrated and performed calculations on the kinematic data, trained the models using this data from the two systems, and evaluated the performance based on the gesture labels I manually generated using the gesture labeling app. Finally, I compared the outcomes between the Raven and DCS by analyzing the accuracy and edit score for each model. The results from this project can provide insight into alternative methods of data collection and aid in the improvement of safety monitoring for robotic surgery.

Methods

This section presents my methods for the collection, labeling, and calibration of data from two robotic surgical systems, the Raven and the DCS. It describes the construction and evaluation of gesture recognition models using the calibrated data.

Data Collection

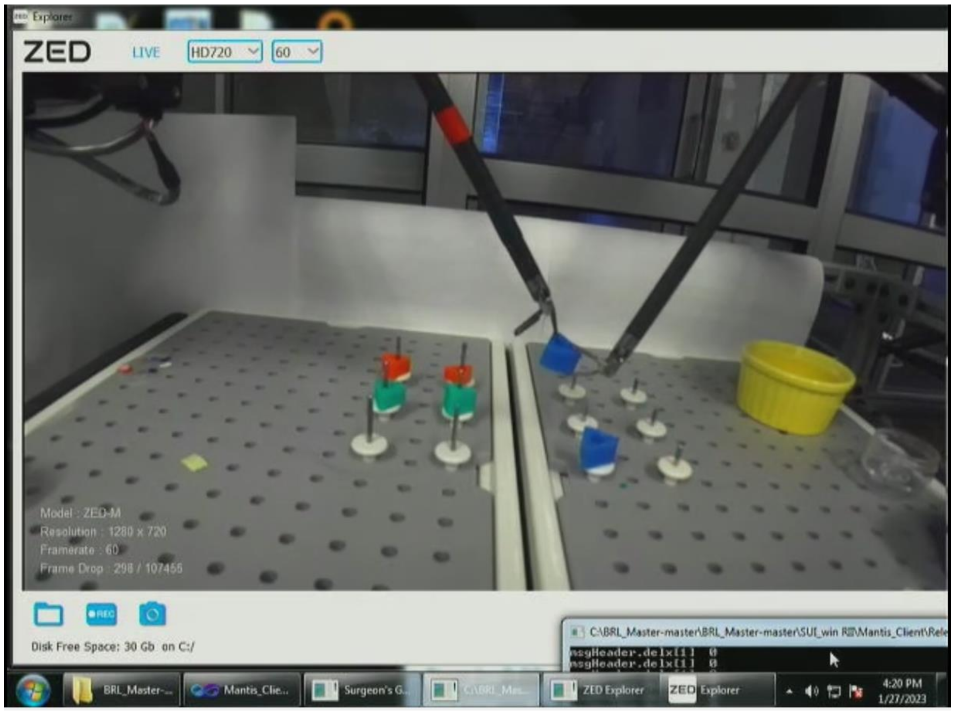
This study focuses on the task of peg transferring, which is a manual skills component that is part of the Fundamentals of Laparoscopic Surgery (FLS) exam (FLS Program, 2014). A single user performed the peg transfer task, and her dominant hand is her left hand. We placed a peg board beneath the arms of the Raven, with six pegs on each of the left and right sides. At the start of each trial, we placed six triangular objects on the pegs on the right side of the pegboard. The two pegs furthest from the camera were red, the two pegs in the middle were green, and the two pegs closest to the camera were blue.

In the first half of each trial, the user grasped each object with her non-dominant right hand and transferred the object mid-air to her dominant left hand. She then had to place the object on a peg on the left side of the pegboard. While the FLS manual states that the color and transfer order of the six objects does not matter, the user consistently picked up the objects from right to left on the right side of the pegboard, in the order of red to green to blue in each trial (FLS Program, 2014). Once the user transferred all six objects to the left side of the board, she then reversed the procedure, grasping each object with her dominant left hand, transferring the object mid-air to her non-dominant right hand, and placing the object on the right side of the pegboard. When transferring the objects back to the original side, the user did not maintain a particular order across trials, but she did still continue to pick up the objects in the order of red to green to blue. One trial consisted of transferring all six pegs from the right side to the left side of the pegboard and then transferring all of them back to the right side.

According to the FLS manual, if the user drops the object within reach, she should pick it up with the hand that dropped the object (FLS Program, 2014). However, it was often the case that the user would pick up the object with whichever grasper was closer to the object. Additionally, while the FLS manual implies that the user should not retrieve the object if it falls out of reach, we would use a human hand to place the object within reach (FLS Program, 2014). Figure 1 depicts the mid-air transfer of a blue object from the right side to the left side of the pegboard as shown on the ZED Explorer.

Figure 1

Transfer of Object During a Trial



HD720 60

Model : ZED-M
Resolution : 1280 x 720
Framerate : 60
Frame Drop : 298 / 107455

Disk Free Space: 30 Gb on C:/

```
C:\BRL_Master-master\BRL_Master-master\SUI_win R...\Mantis_Client\Rele
msgHeader.delx[1]  0
msgHeader.delx[1]  0
```

The data collection setup included the preparation of the Raven, DCS, and smartwatch systems. The front view of the Raven is shown in Figure 2, and the side view is shown in Figure 3. Figure 4 shows the DCS and teleoperation console setup. First, we turned on the Raven, the server receiving data from the Raven, the ZED Explorer, the trakSTAR, and the Alienware laptop that would store the video and data from the DCS. Then, we moved the Raven arms into place, using cables to adjust the angles, but refraining from stretching them extensively as that would impact the kinematic data. To set up the DCS, we connected the trakSTAR device to the Alienware laptop and the ZED Explorer. We oriented the camera to face the front of the Raven and ensured that the view of the pegs fit into the ZED Explorer. The user wore a smartwatch on each wrist, as this data would be used in a separate study.

The user sat in front of a teleoperation console, a user interface consisting of an image viewer showing the camera's vision of the Raven, graspers that control the Raven arms, and a foot pedal unit. To prepare the console for use, the user zeroed the left and right graspers. Before starting a trial, we honed the Raven and started a Python script on the Alienware laptop that prompted for the subject name, task, trial, and attending physician. We set the scaling factor to 0.4 in the Surgeon's GUI, which is an application located on the same computer as the ZED Explorer. This low scaling factor reduces the risk involved with the operation because the robot moves less quickly.

To start a trial, the Raven, DCS, and smartwatch systems needed to begin collecting data at approximately the same time. On the server, we ran a Python script that would collect the kinematic data from the Raven. On the Alienware, we started the video recording and kinematic data collection from the DCS. The user would start the robot by stepping on the middle pedal. During the trial, the user could reposition her hands in the teleoperation console area without moving the graspers of the Raven by stepping on the clutch pedal. After completing the peg transfer task, we stopped both scripts collecting data on the server and Alienware, and the user would stop the robot by stepping on the left pedal. We aimed to minimize the time between starting and ending the scripts, as the Raven kinematic data files were especially large.

Overall, we collected data for 10 trials of peg transferring. However, the Raven kinematic data file for trial 1 was exceptionally large (4.1 GB), compared to other trials that had a size of around 1.5 GB, as this first trial took longer due to the learning curve associated with a first attempt. Additionally, the DCS did not properly record all the necessary kinematic data for trial 8 for unknown reasons. Thus, we discarded trials 1 and 8 and had eight trials remaining for analysis. Figure 5 shows a snapshot of the raw kinematic data that the Raven collected, which includes the timestamp, XYZ positions, orientation, and gripper angle values. Figure 6 shows some of the raw kinematic data from the DCS, which also notably includes the XYZ positions and timestamp for each of the four sensors. The DCS also collected the timestamp associated with each frame number, as shown in Figure 7.

Figure 2

Front View of the Raven

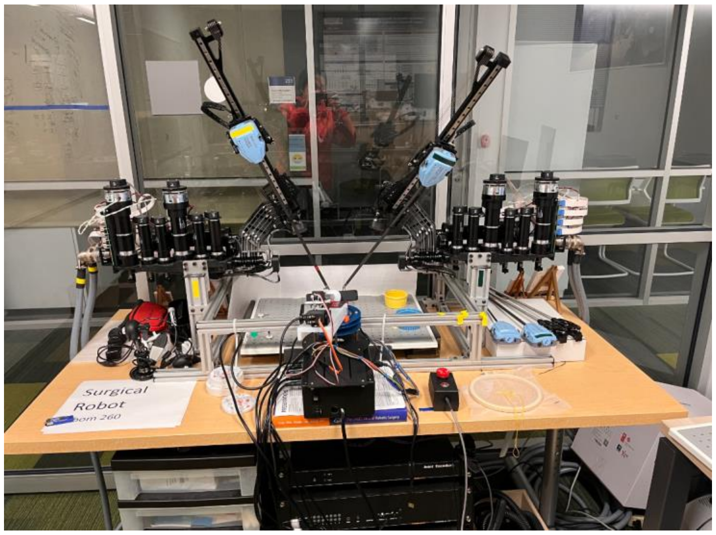


Figure 3

Side View of the Raven

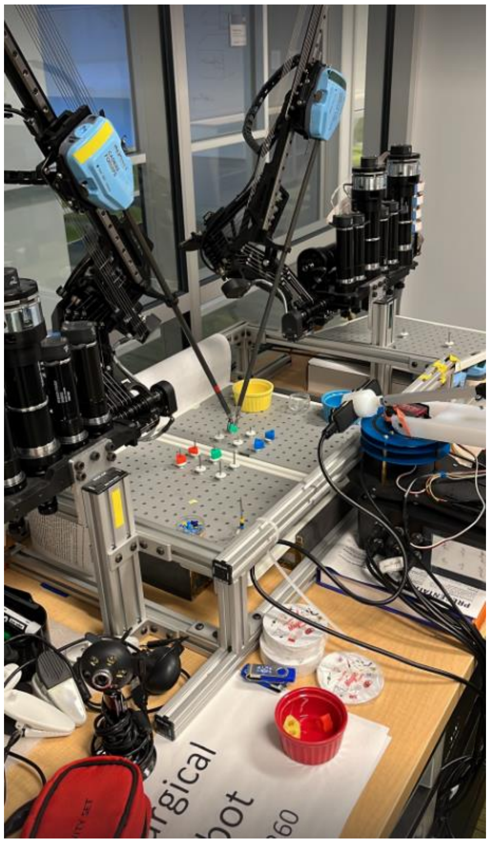


Figure 4

DCS and Teleoperation Console

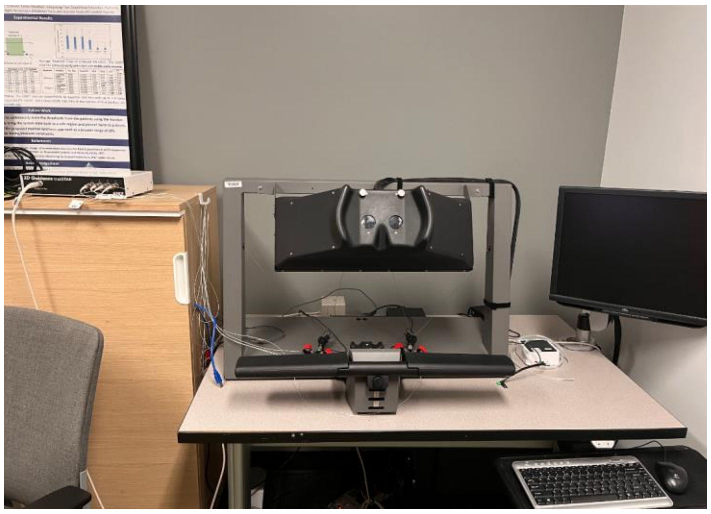
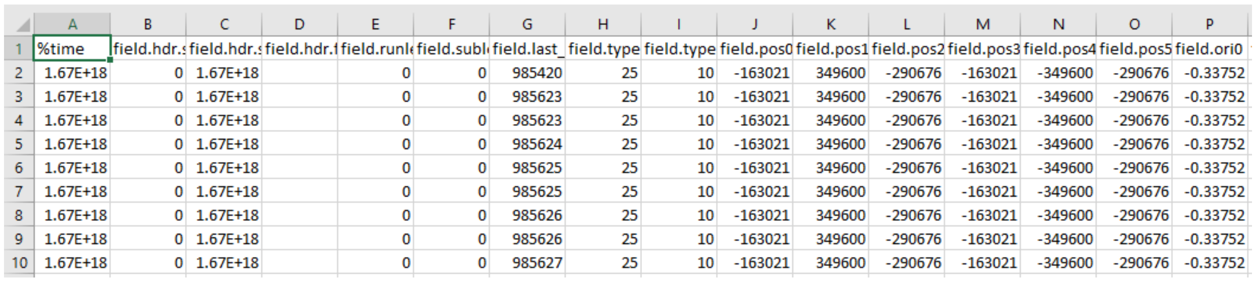


Figure 5

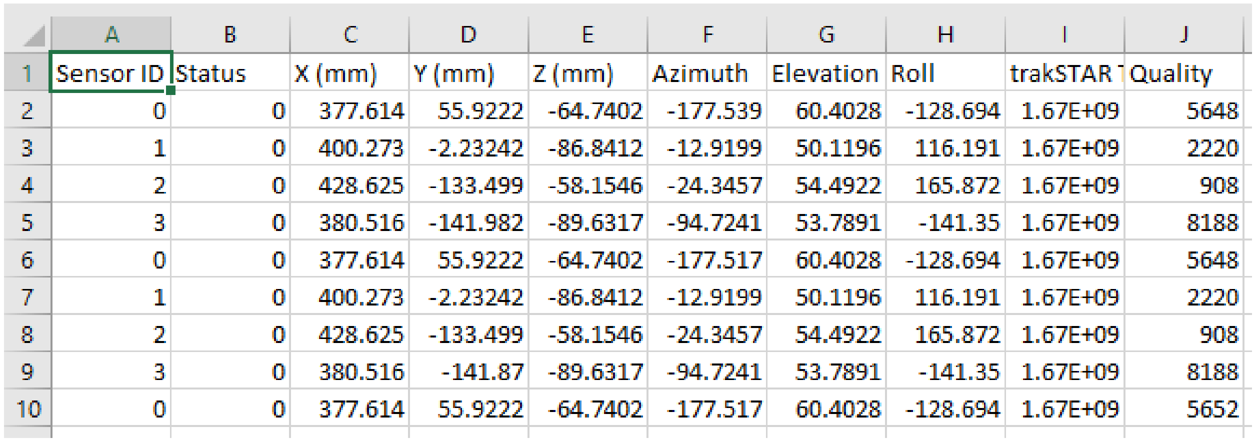
Snapshot of CSV File Containing Raven Kinematic Data



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	%time	field.hdr.s	field.hdr.s	field.hdr.f	field.runle	field.subl	field.last_	field.type	field.type	field.pos0	field.pos1	field.pos2	field.pos3	field.pos4	field.pos5	field.ori0
2	1.67E+18	0	1.67E+18		0	0	985420	25	10	-163021	349600	-290676	-163021	-349600	-290676	-0.33752
3	1.67E+18	0	1.67E+18		0	0	985623	25	10	-163021	349600	-290676	-163021	-349600	-290676	-0.33752
4	1.67E+18	0	1.67E+18		0	0	985623	25	10	-163021	349600	-290676	-163021	-349600	-290676	-0.33752
5	1.67E+18	0	1.67E+18		0	0	985624	25	10	-163021	349600	-290676	-163021	-349600	-290676	-0.33752
6	1.67E+18	0	1.67E+18		0	0	985625	25	10	-163021	349600	-290676	-163021	-349600	-290676	-0.33752
7	1.67E+18	0	1.67E+18		0	0	985625	25	10	-163021	349600	-290676	-163021	-349600	-290676	-0.33752
8	1.67E+18	0	1.67E+18		0	0	985626	25	10	-163021	349600	-290676	-163021	-349600	-290676	-0.33752
9	1.67E+18	0	1.67E+18		0	0	985626	25	10	-163021	349600	-290676	-163021	-349600	-290676	-0.33752
10	1.67E+18	0	1.67E+18		0	0	985627	25	10	-163021	349600	-290676	-163021	-349600	-290676	-0.33752

Figure 6

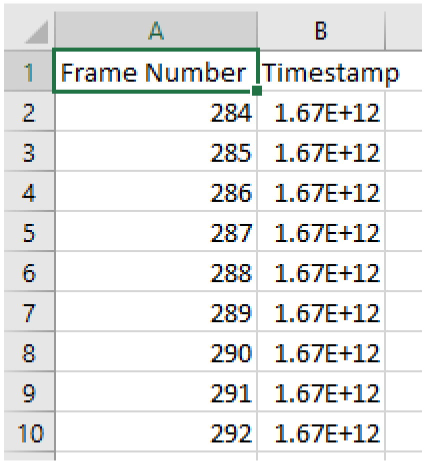
Snapshot of CSV File Containing DCS Kinematic Data



	A	B	C	D	E	F	G	H	I	J
1	Sensor ID	Status	X (mm)	Y (mm)	Z (mm)	Azimuth	Elevation	Roll	trakSTAR T	Quality
2	0	0	377.614	55.9222	-64.7402	-177.539	60.4028	-128.694	1.67E+09	5648
3	1	0	400.273	-2.23242	-86.8412	-12.9199	50.1196	116.191	1.67E+09	2220
4	2	0	428.625	-133.499	-58.1546	-24.3457	54.4922	165.872	1.67E+09	908
5	3	0	380.516	-141.982	-89.6317	-94.7241	53.7891	-141.35	1.67E+09	8188
6	0	0	377.614	55.9222	-64.7402	-177.517	60.4028	-128.694	1.67E+09	5648
7	1	0	400.273	-2.23242	-86.8412	-12.9199	50.1196	116.191	1.67E+09	2220
8	2	0	428.625	-133.499	-58.1546	-24.3457	54.4922	165.872	1.67E+09	908
9	3	0	380.516	-141.87	-89.6317	-94.7241	53.7891	-141.35	1.67E+09	8188
10	0	0	377.614	55.9222	-64.7402	-177.517	60.4028	-128.694	1.67E+09	5652

Figure 7

Snapshot of CSV File Containing Frame Information



	A	B
1	Frame Number	Timestamp
2	284	1.67E+12
3	285	1.67E+12
4	286	1.67E+12
5	287	1.67E+12
6	288	1.67E+12
7	289	1.67E+12
8	290	1.67E+12
9	291	1.67E+12
10	292	1.67E+12

Data Labeling

Using the gesture labeling app pictured in Figure 8, I stepped through each frame to identify which of four gesture labels was displayed:

1: Collecting block

2: Passing block R to L

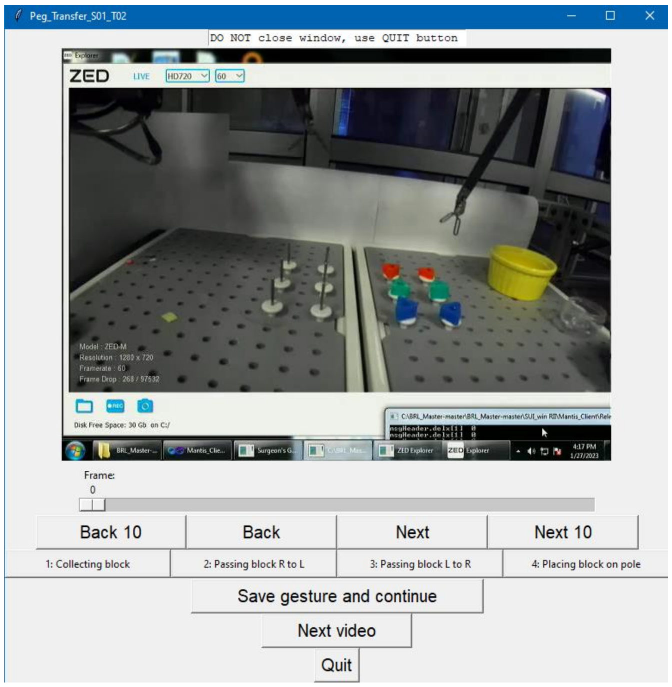
3: Passing block L to R

4: Placing block on pole

These gestures come from the Virtual Robotic Assisted Surgery Training Evaluation Dataset (V-RASTED) (Menegozzo et al., 2019). For each frame, if one of the four gestures completed, I would select that gesture and click “Save gesture and continue.” Otherwise, I would go to the next frame without clicking that button. Figure 9 depicts the gesture labels resulting from labeling trial 2. The first column is the starting frame for a gesture, the second column is the ending frame for a gesture, and the third column describes the gesture. In order to make the data match with the preprocessing script, I changed each of the third column values to a value of S1, S2, S3, or S4, with the number next to the “S” matching the number next to the verbal description.

Figure 8

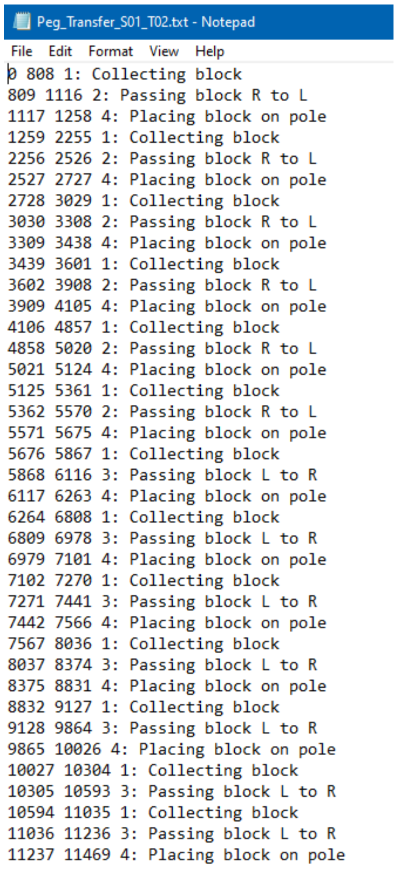
Gesture Labeling App



DO NOT close window, use QUIT button

Figure 9

Gesture Labels from Trial 2


Peg_Transfer_S01_T02.txt - Notepad
File Edit Format View Help

```
0 808 1: Collecting block
809 1116 2: Passing block R to L
1117 1258 4: Placing block on pole
1259 2255 1: Collecting block
2256 2526 2: Passing block R to L
2527 2727 4: Placing block on pole
2728 3029 1: Collecting block
3030 3308 2: Passing block R to L
3309 3438 4: Placing block on pole
3439 3601 1: Collecting block
3602 3908 2: Passing block R to L
3909 4105 4: Placing block on pole
4106 4857 1: Collecting block
4858 5020 2: Passing block R to L
5021 5124 4: Placing block on pole
5125 5361 1: Collecting block
5362 5570 2: Passing block R to L
5571 5675 4: Placing block on pole
5676 5867 1: Collecting block
5868 6116 3: Passing block L to R
6117 6263 4: Placing block on pole
6264 6808 1: Collecting block
6809 6978 3: Passing block L to R
6979 7101 4: Placing block on pole
7102 7270 1: Collecting block
7271 7441 3: Passing block L to R
7442 7566 4: Placing block on pole
7567 8036 1: Collecting block
8037 8374 3: Passing block L to R
8375 8831 4: Placing block on pole
8832 9127 1: Collecting block
9128 9864 3: Passing block L to R
9865 10026 4: Placing block on pole
10027 10304 1: Collecting block
10305 10593 3: Passing block L to R
10594 11035 1: Collecting block
11036 11236 3: Passing block L to R
11237 11469 4: Placing block on pole
```

Data Calibration

The raw kinematic data from the Raven and DCS were not synchronized with each other, because the moment at which each script started to collect data differed by up to a few seconds. Additionally, the rate in terms of frames/second at which each system collected data varied. The video component collects data at 30 Hz, and the trakStar collects data at 90 Hz. Thus, I downsampled the trakStar data to one-third of the original size by extracting the rows from the raw DCS kinematic data that had timestamps closest to each of the timestamps corresponding to a frame number from the video. Another researcher provided me with the downsampled Raven kinematic data which were obtained through a similar method. At this point, the Raven and DCS data were synchronized with each other, which could be verified due to the data having the same timestamps for each frame. The video data had more frames following the last frame in the synchronized Raven data, but the preprocessing step would ignore those additional frames.

I then calculated the position and velocity for the left and right arms in the x-, y-, and z-directions as well as the left and right gripper angles for both the synchronized Raven and DCS data. I obtained the left and right positions for the DCS data by taking the average of the position data from the two sensors on each side. To calculate the DCS gripper angle for each side, I ran a script that used the Pythagorean Theorem to find the distance between the left and right graspers and incorporated that distance into trigonometry calculations to find the angles. For the Raven, the columns field.pos0 through field.pos5 in the kinematic data correspond to the XYZ coordinates of the arms, and the columns field.grasp_d0 and field.grasp_d1 correspond to the left and right gripper angles. I extracted those columns and then performed further adjustments by taking the negative values of four columns in order to match the direction of the Raven position data to the DCS position data. The correlation of the Raven position data to the DCS position data is shown in Table 1.

Table 1

Correlation of Raven Position Data to DCS Position Data

Operation on Raven Position Data	Corresponding DCS Position Data
field.pos1	Left x-position
(Negative) field.pos2	Left y-position
(Negative) field.pos0	Left z-position
(Negative) field.pos3	Right x-position
field.pos5	Right y-position
(Negative) field.pos4	Right z-position

To derive the velocities for each frame for the Raven and DCS data, I took the mean of the position divided by time over a window of 3 frames ending on the current frame for which I was calculating the velocities. I dropped the frames that resulted in having Not a Number (NaN) values due to undefined calculations.

After the aforementioned calibration processes, I discovered that the motion occurring at a certain frame in the Raven data did not match the motion occurring at that same frame in the DCS data. I made this realization by graphing the x-position of the left grasper versus frame number for the Raven and DCS in trial 2, as shown in Figure 10. The Raven x-position curve has the same shape as that for the DCS, but it is situated to the left of the DCS x-position curve. In order to correct this, I calculated the difference in frames between the DCS and Raven data, and added that difference to each frame number in the Raven data. After regraphing the Raven x-position data shifted to the right for trial 2 as shown in Figure 11, I found that the x-position data between the two systems matched up well. I added a column in the CSV file containing the Raven kinematic data that had these shifted frame numbers. I repeated this process for all the trials. Increasing the frame numbers for the Raven data through this method led to a rough match with the DCS data for all trials except for trial 6. The x-position curve already matched for the Raven and DCS data starting from the first frame number for trial 6, so I did not make any adjustments to the data for this trial. I addressed this new correction when preprocessing the data by taking into account the "Frame" column and inserting arrays of gesture labels into the data that started and ended in the locations corresponding to the correct frames. The CSV files with the calibrated kinematic data for trial 2 that I collected from the Raven and DCS are shown in Figures 12 and 13, respectively. The magnitude of the data differs between the two files because the systems collected data in different units.

Figure 10

Pre-Shift Graph of x-Position over Frame Number

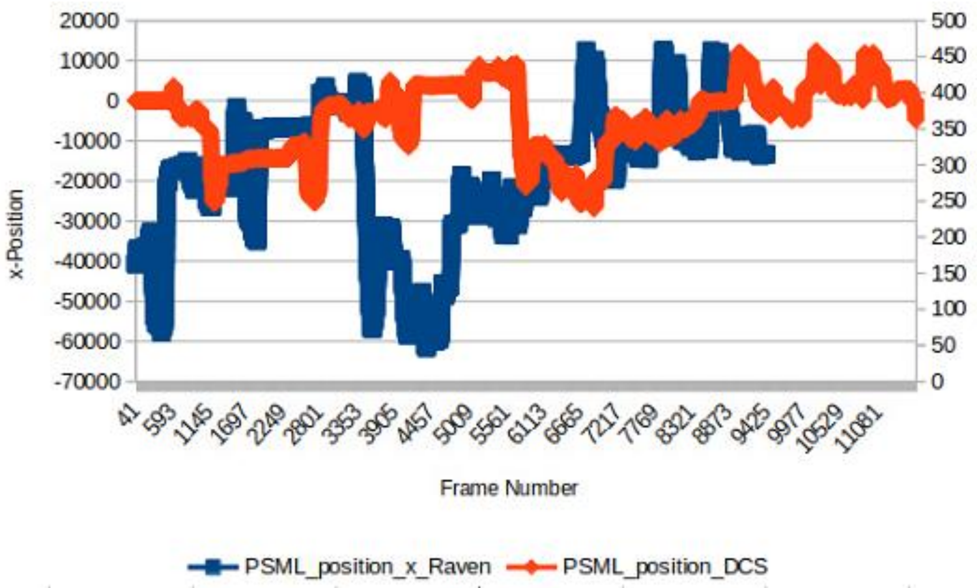


Figure 11

Post-Shift Graph of x-Position over Frame Number

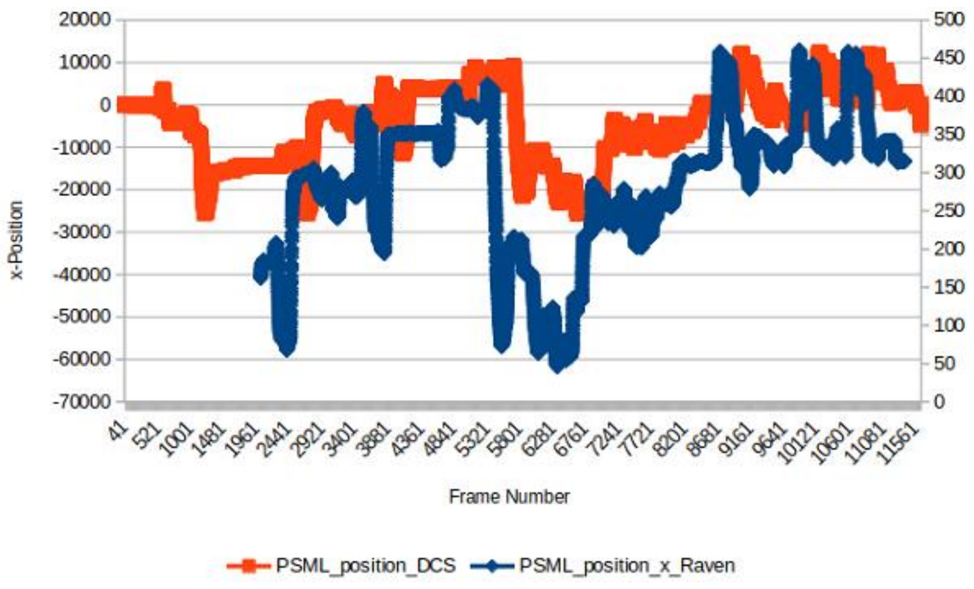
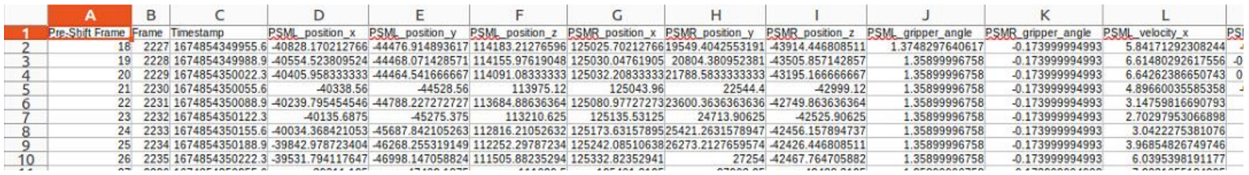


Figure 12

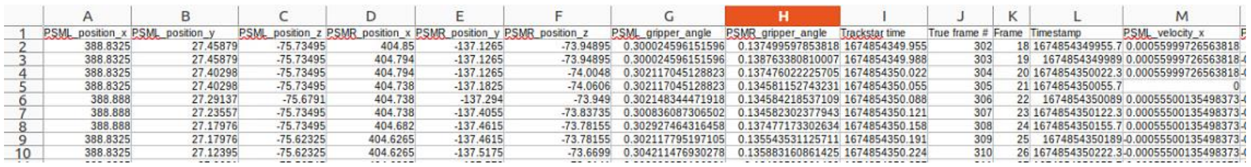
Snapshot of CSV File Containing Calibrated Raven Kinematic Data



	A	B	C	D	E	F	G	H	I	J	K	L
1	Pre-Shift Frame	Frame	Timestamp	PSML_position_x	PSML_position_y	PSML_position_z	PSMR_position_x	PSMR_position_y	PSMR_position_z	PSML_gripper_angle	PSMR_gripper_angle	PSML_velocity_x
2	18	2227	1674854349955.6	-40828.170212766	-44476.914893617	114183.21276596	125025.70212766	19549.4042553191	-43914.446808511	1.3748297640617	-0.173999994993	5.84171292308244
3	19	2228	1674854349988.9	-40554.523809524	-44468.071428571	114155.97619048	125030.04761905	20804.380952381	-43505.857142857	1.35899996758	-0.173999994993	6.61480292617556
4	20	2229	1674854350022.3	-40405.958333333	-44464.541666667	114091.08333333	125032.20833333	21788.5833333333	-43195.166666667	1.35899996758	-0.173999994993	6.64262386650743
5	21	2230	1674854350055.6	-40338.56	-44528.56	113975.12	125043.96	22544.4	-42999.12	1.35899996758	-0.173999994993	4.89660035585358
6	22	2231	1674854350088.9	-40239.795454546	-44788.227272727	113684.88636364	125080.97727273	23600.3636363636	-42749.863636364	1.35899996758	-0.173999994993	3.14759816690793
7	23	2232	1674854350122.3	-40135.6875	-45275.375	113210.625	125135.53125	24713.90625	-42525.90625	1.35899996758	-0.173999994993	2.70297953066898
8	24	2233	1674854350155.6	-40034.368421053	-45687.842105263	112816.21052632	125173.63157895	25421.2631578947	-42456.157894737	1.35899996758	-0.173999994993	3.0422275381076
9	25	2234	1674854350188.9	-39842.978723404	-46268.255319149	112252.29787234	125242.08510638	26273.2127659574	-42426.446808511	1.35899996758	-0.173999994993	3.96854826749746
10	26	2235	1674854350222.3	-39531.794117647	-46998.147058824	111505.88235294	125332.82352941	27254	-42467.764705882	1.35899996758	-0.173999994993	6.0395398191177

Figure 13

Snapshot of CSV File Containing Calibrated DCS Kinematic Data



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	PSML_position_x	PSML_position_y	PSML_position_z	PSMR_position_x	PSMR_position_y	PSMR_position_z	PSML_gripper_angle	PSMR_gripper_angle	Trackstar time	True frame #	Frame	Timestamp	PSML_velocity_x
2	388.8325	27.45879	-75.73495	404.85	-137.1265	-73.94895	0.300024596151596	0.137499597853818	1674854349.955	302	18	1674854349955.7	0.00055999726563818
3	388.8325	27.45879	-75.73495	404.794	-137.1265	-73.94895	0.300024596151596	0.138763380810007	1674854349.988	303	19	1674854349989	0.00055999726563818
4	388.8325	27.40298	-75.73495	404.794	-137.1265	-74.0048	0.302117045128823	0.13747602225705	1674854350.022	304	20	1674854350022.3	0.00055999726563818
5	388.8325	27.40298	-75.73495	404.738	-137.1825	-74.0606	0.302117045128823	0.134581152743231	1674854350.055	305	21	1674854350055.7	0
6	388.888	27.29137	-75.6791	404.738	-137.294	-73.949	0.302148344471918	0.134584218537109	1674854350.088	306	22	1674854350089	0.000555001354983731
7	388.888	27.23557	-75.73495	404.738	-137.4055	-73.83735	0.300836087306502	0.134582302377943	1674854350.121	307	23	1674854350122.3	0.000555001354983731
8	388.888	27.17976	-75.73495	404.682	-137.4615	-73.78155	0.302927464316458	0.137477173302634	1674854350.158	308	24	1674854350155.7	0.000555001354983731
9	388.8325	27.17976	-75.62325	404.6265	-137.4615	-73.78155	0.302117795197105	0.135543531125711	1674854350.191	309	25	1674854350189	-0.000555001354983731
10	388.8325	27.12395	-75.62325	404.6265	-137.5175	-73.6699	0.304211476930278	0.135883160861425	1674854350.224	310	26	1674854350222.3	-0.000555001354983731

Gesture Recognition Model

To capture the relationship between the kinematic data and gestures, I used a Temporal Convolutional Network (TCN). A TCN is a deep convolutional model that employs multiple levels of temporal convolutions and pooling layers in an encode-decoder structure to express long-range spatiotemporal relationships at various time scales in the input data. This type of model has achieved better performance than competing models in using video or sensor data to predict surgical actions. They can also compute predictions at the same time for each timestamp and thus complete training much faster than other popular methods such as Recurrent Neural Networks (RNN) (Lea et al., 2016; van Amsterdam et al., 2021).

The goal of the TCN model in this study is to predict a surgical gesture for each kinematic data segment in the test set. I used a similar TCN model as the one that Hutchinson, Reyes, et al. (2022) adopted, but with minor modifications to ensure that the model was compatible with the number of trials and the cross validation setup I used. Each of the encoder and decoder steps in the model has three convolutional layers with pooling, channel normalization, and upsampling (Hutchinson, Reyes, et al., 2022). The three layers in the encoder step have filter sizes of 64, 96, and 128, while the three layers in the decoder step have 96, 64, and 64 filters respectively. The model uses a cross-entropy loss function and was trained using the Adam optimizer. The input to the TCN model is the kinematic data over time from each robotic surgical system, x_t, and the output is a gesture label, y_t, for each time-series sample. Previous studies have found that using a combination of position, gripper angle, and linear velocity as the kinematic input to gesture recognition models generated the best results, so I also used only these kinematic variables as input to my TCN model (Hutchinson, Reyes, et al., 2022). Thus, the input size was 14, as the variables I used consisted of the position and velocity in the x-, y-, and z-directions for both the left and right graspers (2 variables x 3 directions x 2 sides = 12 inputs), as well as the left and right gripper angles (2 inputs). I created and trained the TCN model in a virtual environment on a computer with an Intel Core i7-8700K CPU @ 3.70GHz running Ubuntu 22.04.1 LTS, an NVIDIA GeForce GTX 1080 Ti GPU, and a Torch version of 1.10.2.

I evaluated the generalizability of the model using a cross validation setup of Leave-One-Supertrial-Out (LOSO). In this scheme, I created eight validation folds where each fold consisted of data from one of the eight trials. By using a different trial as the test set and the remaining trials to train the model each time, I can evaluate the generalization of the TCN models to new trials that known surgeons carry out (Hutchinson, Reyes, et al., 2022).

The learning rate and weight decay hyperparameters were chosen using a grid search of values to determine the hyperparameter numbers that would produce the best results when training on the JIGSAWS dataset with gesture labels for the LOSO cross validation setup. I kept these hyperparameter values constant across all models so that I could analyze how the type of robotic surgical system affected the performance of the same model (Hutchinson, Reyes, et al., 2022). Specifically, the learning rate was 0.00005 and the weight decay was 0.01. The model ran for 60 epochs, where each epoch had a batch size of 1.

Overall, the construction and evaluation process was comprised of four main steps: preprocessing, tuning, training, and calculation of evaluation metrics. In preprocessing, I extracted the appropriate columns, loaded applicable model parameters from a configuration file, and updated certain parameters such as input size in that configuration file. I also preprocessed the kinematic and gesture label transcript files by matching the correct label to sections of the kinematic data and saving the concatenated information into a preprocessed folder. In this preprocessing stage, I also encoded labels from the preprocessed data files and saved them to a pickle file. These steps set up many of the parameters and data needed for the model to run. After preprocessing, I tuned the batch size, number of epochs, learning rate, and weight decay through hardcoded values that have been shown to work well through previous experimentation (Hutchinson, Reyes, et al., 2022). I then trained, tested, and cross validated the TCN model with the established hyperparameters. Finally, I outputted the average metrics from the folds in the model. To run the model at each of these stages, I specified the task (peg transfer), a specific input variable (velocity) that indicated hyperparameter values, the label type (gesture), and the cross validation setup (LOSO) in the command line parameters. The complete code for my capstone can be found in the branch "sara" of the public COMPASS repository at https://github.com/UVA-DSA/COMPASS that is part of the UVA Dependable Systems and Analytics (DSA) Research Group.

Evaluation Metrics

I evaluated and compared the TCN models using two metrics, accuracy and edit score, averaged across all the folds. Accuracy is a frame-wise metric that divides the number of correct predictions by the number of total predictions (Goldbraikh et al., 2022). The formula for accuracy is $\text{Acc} = \frac{N_C}{N} * 100$, $0 < \text{Acc} < 100$, where N_C is the number of correctly labeled predictions and N is the number of total predictions. However, accuracy alone does not provide sufficient analysis of temporal predictions and robust model comparison because this metric is designed for data where the sequence does not matter. Prediction sequences with similar accuracy could have significant differences in gesture ordering and over-segmentation, which is the prediction of an excessive amount of negligible action boundaries. Thus, I also chose to evaluate the TCN model using a segmental metric, which is a value that measures the temporal sequence of action predictions (van Amsterdam et al., 2021).

The segmental metric I calculated was the edit score, which assesses the order of gestures but not the time at which they occur. Edit scores are lower when the predictions are out of order or have over-segmentation (van Amsterdam et al., 2021). The edit score makes use of the Levenshtein edit distance, *edit(G, P)*, which counts the number of insertions, deletions, and substitutions needed to convert the predicted label sequence *P* to the ground truth label sequence *G* (Hutchinson, Reyes, et al., 2022). This distance is then normalized by the maximum between the number of segments in either the predicted or ground truth label sequences. The equation for calculating the edit score is Edit Score = $\left(1 - \frac{edit(G,P)}{\max\left(len(G),len(P)\right)}\right) * 100$, 0 < Edit Score < 100 (Hutchinson, Reyes, et al., 2022; van Amsterdam et al., 2021). For both accuracy and edit score, a higher value indicates better performance of the model.

Results

In this section, I present the performance of the TCN models in recognizing gestures using kinematic data from two robotic surgical systems, the Raven and the DCS. I trained the TCN model 10 times for each of the two systems. Table 2 compares the accuracy and edit score of each run for each system, as well as the mean and sample standard deviation of these evaluation metrics across all runs for each system.

Table 2

TCN Model Results in Identifying Gestures Using Data Collected from the Raven and DCS

Run	Raven		DCS	
	Accuracy (%)	Edit Score (%)	Accuracy (%)	Edit Score (%)
1	68.397	91.015	88.574	84.068
2	65.778	88.257	87.947	88.022
3	67.032	87.550	87.249	80.083
4	68.900	89.734	86.729	82.347
5	67.386	89.638	89.497	87.185
6	67.299	88.406	88.447	85.180
7	67.644	89.694	87.238	84.661

8	68.396	88.059	88.513	85.133
9	65.949	88.448	90.407	86.709
10	66.171	90.115	88.540	84.249
Average	67.295	89.091	88.314	84.764
Std Dev	1.084	1.098	1.100	2.335

As shown in Table 2, for the Raven kinematic input data, the model predicts gestures with an average accuracy of 67.295% with standard deviation 1.084% and an average edit score of 89.091% with standard deviation 1.098%. For the DCS kinematic input data, the model predicts gestures with an average accuracy of 88.314% with standard deviation 1.100% and an average edit score of 84.764% with standard deviation 2.335%. The standard deviations for average accuracies and edit scores across both systems is less than 3%, indicating that there is low variance across runs of the TCN model for the two systems.

The average edit scores for the systems are also similar, with only a 4.328% difference between the value of 89.091% for the Raven and 84.764% for the DCS. However, the average accuracies differ significantly by 21.019%, with a much higher value for the DCS (88.314%) than the Raven (67.295%). It is interesting that the average edit scores are almost the same, but the average accuracy for the model trained with Raven data is much lower than that trained with DCS data. One potential reason for this could be additional underlying issues with the synchronization between the Raven and video data. The trakSTAR and video data were both collected as part of the DCS, so the synchronization between those values was more straightforward than that between the Raven and video data. I manually adjusted the frames of the Raven data so that the x-position curve would match that of the trakSTAR data. By the transitive property, the Raven data should have then theoretically also synced well with the video data, which included the ground truth gesture labels. However, because I verified the curve alignment visually, this method involved only an approximate frame shift. It is likely that there still could have existed some misalignment that would cause the predicted gestures from the Raven data to not equal the ground truth gestures at the right times, leading to the low average accuracy. However, the overall sequence of predicted gestures from the Raven data most likely matched that of the ground truth gestures, resulting in the high average edit score.

Discussion

In this section, I discuss the applicability of the model results to evaluating whether the Raven and DCS would be robust, accessible alternatives to collecting surgical data. Hutchinson, Reyes, et al. (2022) used a TCN model similar to the one I used but trained on the COMPASS dataset under the Leave-One-User-Out (LOUO) cross validation setup. For the peg transfer task, they observed that the model identified gestures with a 55.0% accuracy and 69.7% edit score (Hutchinson, Reyes, et al., 2022). All of the accuracies and edit scores of the models I trained on the Raven and DCS data are higher than the accuracy and edit score from the model trained on the COMPASS dataset. This indicates that the Raven and DCS are viable robotic surgical systems that can collect data for research. The DCS has particularly strong metrics to support its use in the field, providing data that can produce a TCN model with an average accuracy of 88.314% and average edit score of 84.764%. These statistics demonstrate that the DCS is a remarkably powerful, independent system that can be used to collect video and kinematic data on robotic surgery.

However, it is also important to consider flaws with the method described in this paper. We were only able to collect eight trials to use for analysis, so the model is overfitted to the training set, and the results are likely higher than they would be with a greater number of trials.

The overfit model could generate inaccurate gesture predictions when new data is inputted. Additionally, when we were collecting data for the peg transfer task, if the user dropped the object mid-air, the drop was often used as a transfer point. The user would frequently pick up the object with the hand that did not drop it, completing a transfer, even though the FLS manual states that the same hand that dropped the object should pick it up (FLS Program, 2014). This discrepancy could have interfered with the results, as there would be incorrect motions associated with passing a block from one hand to the other. Finally, it was necessary to perform manual frame shifts for the Raven kinematic data, which is a considerable drawback to this method if there are a large number of trials. If the root of the synchronization issue can be addressed, that would eliminate the need to manually correct the frame numbers.

Technical Challenges

There were a number of challenges that I overcame throughout the course of this research project. To set up a virtual environment where I could run my TCN model, I had to ensure that all of the packages required were compatible with each other. This involved several rounds of trial and error and adding the packages into the environment one at a time to satisfy compatibility requirements. I repeated this process for two servers due to resource limitations in the research lab, as the frozen package requirements from the first server were not compatible on the second server.

When we were collecting data, the initial trials took around 30 minutes to complete as we did not have much practice yet with operating the robotic surgical systems. However, with repeated execution, we were able to finish each trial in about 10 minutes towards the end of the data collection process. This was due to a combination of the user becoming more adept at performing the peg transfer task, and the rest of us handling the data collection steps faster as well. Reducing the time per trial also decreased the size of the video and kinematic data we collected.

In the fall semester, we collected 10 trials of data for the peg transfer task. However, we decided in the spring semester to adopt a new standard for this task that followed the FLS manual (FLS Program, 2014). Thus, we did not use the data from the previous semester and started from trial 1 in the spring semester. Although the time spent on collecting data in the fall did not result in data that would eventually be used in the final analysis, those trials still served as practice in becoming more familiar with using the robotic surgical systems.

Data synchronization and downsampling was a major challenge. The video, trakSTAR, and Raven components all started and stopped collecting data at different times. They also collected data at different rates. This led to discussion regarding the best approach to synchronization so that the data began at the same time and downsampling so that the data had an equal number of frames across systems. After synchronizing and downsampling the raw Raven kinematic data, we found there to be fewer frames in the Raven data than the video data. This may have been due to the fact that the video started recording data for a longer time than the Raven. We handled this issue when preprocessing the data for the model by ignoring frames that occurred outside of the frame range of other systems.

In my first attempt at training the model on the Raven kinematic data, the accuracy was only around 37%, but the accuracy of the same model trained on the DCS kinematic data was above 80%. I plotted the x-position over frame number for the two systems against each other, and after giving each system its own scale for the x-position, I found that they had a similar shape but were shifted horizontally from each other. To solve this issue, I determined a method that would shift the Raven kinematic data by enough frames so that the x-position curves approximately matched up. Another challenge was realizing that the preprocessing script assumed that the kinematic data started at frame 0, so I modified the script to take into account the shift in frames. I also altered the script to ignore any frames in the ground truth gesture label data that had been phased out in the kinematic data.

Conclusion

In this paper, I presented methods to collect, label, and calibrate data from two accessible robotic surgical systems, the Raven and DCS. I trained a TCN model with LOSO cross validation to identify surgical gestures using position, gripper angle, and velocity input data from each of these systems to determine whether they would be suitable apparatuses for data collection. The resulting accuracies and edit scores of the models in identifying gestures were significantly higher than state-of-the-art statistics. The model trained on DCS data had an especially high average accuracy and edit score. These outcomes reveal that the Raven and DCS are feasible alternatives to collect data compared to almost inaccessible systems like the dVSS. This data can be applied to safety measures by tracing

There are a number of paths that can be taken for future work. Repeating this study with a greater number of trials would strengthen the validity of this conclusion. Researchers could also use other cross validation setups such as LOUO, which requires a variety of users to generalize the model to different surgeons performing the tasks (van Amsterdam et al., 2021). The model could also be trained to identify MPs, which are more objective than gestures (Hutchinson, Reyes, et al., 2022). Future work could also tune the hyperparameters with a grid search more customized to the input data, as the hyperparameters in this study were based off the JIGSAWS dataset which does not include the peg transfer task (Gao et al., 2014). Researchers can also explore the potential of transfer learning using other datasets to improve upon the TCN model.

Acknowledgement

I would like to thank Kay Hutchinson, Hamid Roodabeh, and Keshara Weerasinghe for their help with collecting data. I also thank Kay Hutchinson for the development of a Gesture Labeling App, Keshara Weerasinghe for his work in synchronizing and downsampling the raw data we collected from the Raven, and Shrisha Yapalparvi for his work in writing a script that could downsample and extract positional calculations from the raw data we collected from the DCS.

References

Ahmidi, N., Tao, L., Sefati, S., Gao, Y., Lea, C., Haro, B. B., Zapella, L., Khudanpur, S., Vidal, R., & Hager, G. D. (2017). A dataset and benchmarks for segmentation and recognition of gestures in robotic surgery. *IEEE Transactions on Biomedical Engineering*, *64*(9), 2025-2041. https://doi.org/10.1109/TBME.2016.2647680

Ascension Technology Company. (2017). *3D guidance trakSTAR user guide* (4th ed.). Northern Digital Inc.

Cleveland Clinic. (2021, December 15). *Robotic Surgery.* https://my.clevelandclinic.org/health/treatments/22178-robotic-surgery

FLS Program. (2014, February). *FLS manual skills written instructions and performance guidelines.* Fundamentals of Laparoscopic Surgery. https://www.flsprogram.org/wp-content/uploads/2014/03/Revised-Manual-Skills-Guidelines-February-2014.pdf

Gao, Y., Vedula, S. S., Reiley, C. E., Ahmidi, N., Varadarajan, B., Lin, H. C., Tao, L., Zappella, L., Béjar, B., Yuh, D. D., Chen, C. C. G., Vidal, R., Khudanpur, S., & Hager, G. D. (2014). JHU-ISI gesture and skill assessment working set (JIGSAWS): A surgical activity dataset for human motion modeling. *M2CAI*, 1-10. https://cirl.lcsr.jhu.edu/wp-content/uploads/2015/11/JIGSAWS.pdf

Glassman, D., White, L., Lewis, A., King, H. H., Clarke, A., Glassman, T., Comstock, B., Hannaford, B., & Lendvay, T. S. (2014). Raven surgical robot training in preparation for da Vinci® use: A randomized prospective trial. *Studies in Health Technology and Informatics, 196*, 135-141. https://doi.org/10.3233/978-1-61499-375-9-135

Goldbraikh, A., Volk, T., Pugh, C. M., & Laufer, S. (2022). Using open surgery simulation kinematic data for tool and gesture recognition. *International Journal of Computer Assisted Radiology and Surgery, 17*, 965-979. https://doi.org/10.1007/s11548-022-02615-1

Hutchinson, K., Li, Z., Cantrell, L. A., Schenkman, N. S., & Alemzadeh, H. (2022). Analysis of executional and procedural errors in dry-lab robotic surgery experiments. *The International Journal of Medical Robotics and Computer Assisted Surgery*, *18*(3), 1-15. https://doi.org/10.1002/rcs.2375

Hutchinson, K., Reyes, I., Li, Z., & Alemzadeh, H. (2022). *COMPASS: A formal framework and aggregate dataset for generalized surgical procedure modeling* [Unpublished manuscript]. https://arxiv.org/pdf/2209.06424.pdf

Lea, C., Vidal, R., Reiter, A., & Hager, G. D. (2016). Temporal convolutional networks: A unified approach to action segmentation. *Computer Vision - ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III* (pp. 47-54). Springer, Cham. https://doi.org/10.1007/978-3-319-49409-8_7

Li, Y., Hannaford, B., & Rosen, J. (2019). *Raven: Open surgical robotic platforms* [Unpublished manuscript]. https://arxiv.org/pdf/1906.11747.pdf

Menegozzo, G., Dall’Alba, D., Zandonà, C., & Fiorini, P. (2019). Surgical gesture recognition with time delay neural network based on kinematic data. *2019 International Symposium on Medical Robotics (ISMR)*, 1-7. https://doi.org/10.1109/ISMR.2019.8710178

Sheetz, K. H., Claflin, J., & Dimick, J. B. (2020). Trends in the adoption of robotic surgery for common surgical procedures. *JAMA network open*, *3*(1), 1-9. https://doi.org/10.1001/jamanetworkopen.2019.18911

Stereolabs. (2023). *Get Started with ZED.* https://www.stereolabs.com/docs/get-started-with-zed/

van Amsterdam, B., Clarkson, M. J., & Stoyanov, D. (2021). Gesture recognition in robotic surgery: A review. *IEEE Transactions on Biomedical Engineering*, *68*(6). https://doi.org/10.1109/TBME.2021.3054828