

AI-Enhanced Control for Surgical Robots - Calibration, Tracking, and Haptics

Haonan Peng

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2026

Reading Committee:

Blake Hannaford, Chair

Sam Burden

Eli Shlizerman

John Raiti

Program Authorized to Offer Degree:
Electrical and Computer Engineering

©Copyright 2026

Haonan Peng

University of Washington

Abstract

AI-Enhanced Control for Surgical Robots -
Calibration, Tracking, and Haptics

Haonan Peng

Chair of the Supervisory Committee:
Professor Blake Hannaford
Electrical and Computer Engineering

With worldwide implementation, millions of surgeries are assisted by surgical robots. The cable-drive mechanism on many surgical robots allows flexible, light, and compact arms and tools. However, the slack and stretch of the cables and the backlash of the gears introduce inevitable errors from motor poses to joint poses, which affect the accuracy of the pose and orientation of the end-effector. Since the reported end effector position of surgical robots like Raven-II is directly calculated using the motor encoder measurements and forward kinematics, it may contain relatively large errors up to 10 mm. On the other hand, wiring complexity and sterilization also prevent the implementation of sensors on the end-effector, which have caused a lack of haptic feedback in many surgical robots ever since the beginning. When manually teleoperated, the system is closed-loop by well-trained operators, and the cable-driven inaccuracy and lack of haptic feedback are less of a concern. However, as semi-autonomous functions are introduced into abdominal surgeries, such as debridement and suturing, better closed-loop performance with better positional accuracy and force estimation is desired. However, obstacles of no direct measurements on joints and end-effectors still exist. Indirect measurements on motors rely on accurate modeling of the robot dynamics and kinematics, which can also be difficult due to cable-driven mechanisms. In contrast, machine learning methods show great potential to utilize indirect features, finding nonlinear relations among multiple factors. Thus, it can be desirable to develop a learning-based calibration that can take the original inaccurate robot states as input, and output corrected robot positional states and estimation of

the contact force on the end-effector. This dissertation first develops an efficient data-driven calibration framework for the joint positions of the Raven-II surgical robot. External joint encoders are used to collect ground-truth data during calibration, while the trained models only require the original robot states during operation. The effects of calibration trajectory design, model structure, input-output selection, and long-term operation are evaluated to improve both accuracy and practical efficiency. Second, this dissertation studies how different robot-state features contribute to learning-based calibration. Through feature ablation studies, joint positions and motor torques are identified as important inputs. By selecting useful features and excluding unnecessary ones, the learning models have improved accuracy and reduced training and inference costs. Third, this dissertation investigates surgical instrument tracking through endoscopic instrument segmentation, using active learning and copy-and-paste synthetic image generation to reduce manual annotation effort. Finally, a motor-cable force actuation system and preliminary distal force sensing framework are developed for Raven-II, enabling controlled external force application and data collection for distal force estimation and haptic feedback.

TABLE OF CONTENTS

	Page
List of Figures	iii
List of Tables	xi
Chapter 1: Introduction	1
1.1 Background and Motivation	1
1.2 Main contributions	3
1.3 Organization of the dissertation	7
Chapter 2: Background and Literature Review	8
2.1 Pose Calibration of Cable-Driven Surgical Robots	8
2.2 Instrument Segmentation, Active Learning, and Synthetic Images	9
2.3 Haptics, Distal Force Sensing, and Cable Force Actuation for Surgical Robots	11
Chapter 3: Efficient Data-Driven Calibration of Joint Positions	13
3.1 Method	13
3.2 Experiments	22
3.3 Results	25
3.4 Discussion	38
3.5 Conclusion	40
Chapter 4: Feature Ablation Study of Data-Driven Calibration	46
4.1 Method	46
4.2 Experiments	55
4.3 Results	59
4.4 Discussion	68
4.5 Conclusion	71
Chapter 5: Instrument Segmentation of Surgical Robot End-Effector	76
5.1 Method	76

5.2	Experiments	84
5.3	Results	89
5.4	Discussion	97
5.5	Conclusion	99
Chapter 6:	Distal Force Sensing and Force Actuation System	104
6.1	Method	104
6.2	Experiments and Preliminary Results	114
6.3	Conclusion	118
Chapter 7:	Conclusion and Future Work	122
7.1	Conclusion	122
7.2	Future Research Directions	123
Appendix A:	Real-time Data Driven Precision Estimator for Raven-II Surgical Robot and Visual Tracking of End Effector Position	142
A.1	Method	142
A.2	Experiments and Results	151
A.3	Discussion	154
A.4	Conclusion	155
Appendix B:	Code and Data Availability	157
Appendix C:	How AI Helped My Research and Teaching in 2023-2026	158

LIST OF FIGURES

Figure Number		Page
3.1	The general-purpose controller and relative tools developed for the Raven-II surgical robot, based on ROS, Python, and CRTK API. Compared to the original C++ controller that only supports servo incremental control, this new controller has more control, visualizations, data recording, and safety functions.	14
3.2	Workflow of the calibration. The time cost mainly depends on the collection of training data. The time cost of data processing and model training may vary with different CPUs. GPU is not necessary for the training of neural network models in this work, unless stated otherwise. Designed for surgical purposes, the first 3 joint axes of Raven-II always intersect at the same location which is called the remote motion center and the surgical instrument will always pass this location as the incision site (top left). In order to possess compact and light arms, the motors and encoders are mounted on the robot base, and joints are driven by cables, which results in considerable errors in joint positions obtained by remote motor encoders (top middle). Without calibration, the average positional error of Raven-II's end-effector is around 20 mm (top right).	16
3.3	The joints and cable connections of the Raven-II surgical robot. Compared to joints 2-7, joint 1 has a considerably short cable length (alpha wrap). Joint 3 is prismatic and has a large range for insertion, while all the rest of the joints are rotational. Joints 1-3 are positioning joints and have major control of the end-effector's position. Joints 4-7 are concentrated near the end-effector and have only 1 non-zero link length of 13 mm in the kinematic model. Thus, the last 4 joints mainly control the orientation of the end-effector and have much less influence on the positional accuracy of the end-effector. Joints 6 and 7 control the left and right fingers of the end-effector and can be considered as one joint in kinematics. Thus, Raven-II has 6 joints in kinematic models instead of 7. The left arm and the right arm of Raven-II are symmetric.	18
3.4	Examples of calibration trajectories with different directions (top) and sparsity (bottom). Please note that the trajectories are defined in joint space instead of Cartesian space. Different directions cause different coverage in different joints. Smaller sparsity results in better coverage of the joint space, but also takes a longer time to execute. 19	

3.5	Model of regression-based calibration (left) and learning-based calibration (right). The outputs of the DNN model and regression model are the errors of the joint positions, these errors are added back to Raven-II's original inaccurate joint positions to obtain the calibrated joint positions. The output of the regression models can be either the same or end-to-end output of joint positions (details in 3.3.6).	23
3.6	The external joint encoders that are used to collect ground truth. Raven-II has threaded holes for easy installations of joint encoders. After training, they can be removed before operation.	24
3.7	Calibration performance with different sparsities of the calibration trajectory. For sparsity 1/2, 1/3, 1/4, 5 independent calibration trajectories were recorded, and for sparsity 1/5, 1/6, 3 independent were recorded to prevent long-time operating. Combinations of trajectories with different sparsities, no longer than 1013 seconds to record, were also evaluated. Average RMSE is shown in solid lines and the standard deviations are shown in shades.	27
3.8	Calibration performance during 6-hour continuous unloaded operating, 500g loaded operating, and idleness. The same sparsities were used as in 3.3.2 and shown in the box plot. The baseline results were obtained by fixed offset compensation. Accuracy of idleness was only evaluated by test trajectories in hours 0 and 5, and hours 1-4 were interpolated. The arrows in the box plot indicate training trajectories longer than 800 sec had >20% better calibration accuracy than trajectories shorter than 500 sec, and the difference between the 1st and 3rd quartile is larger than 0.2 (deg/mm).	29
3.9	Example of calibrated joint position trajectories. The models were trained by the calibration trajectory with sparsity $\frac{1}{6}$. 3 sections are shown horizontally. The first section is the immediate performance right after calibration. The second and third sections show performance in hour 2 and 5, respectively.	30
3.10	The example of the calibrated end-effector position trajectories corresponding to Fig. 3.9, represented in Frame 0 of Raven-II. Frame 0 is located at the remote motion center (described in Fig. 3.2, upper left) of the robot arm. It is worth noting that in the front view, the X-axis is pointing vertically up and the Z-axis is pointing right, determined the joint 1 axis and its positive direction of rotation. Only around 10-minute testing trajectories are shown for clear visualization, the entire testing trajectories were recorded throughout 6-hour continuous operating.	31
3.11	Calibration performance of models trained by selected important features (joint positions and motor torques), and by all features (details in 4.1.1) in the robot state. Besides the original DNN model, a larger DNN model was also used to accommodate more input dimensions. The box plot shows the average RMSE among 6-hour unloaded operating with different calibration trajectory sparsities, as in Fig. 3.8. The arrows in the box plot indicate training trajectories longer than 800 sec had >20% better calibration accuracy than trajectories shorter than 500 sec, and the difference between the 1st and 2nd quartile is larger than 0.2 (deg/mm).	35

3.12	Calibration performance after 0-5 robot homing procedures. Different combinations (<20 min) of training trajectories with various sparsities were tested and the average is shown with standard deviation as shade. The 2nd order polynomial regression model is not shown because of unstable performance. Two datasets were used for training, one had the training set with no homing, and another one had the training set with 1 homing. To study the effectiveness of homing, baseline results with the same operating time but no homing in both training and test sets are also included.	35
4.1	Relationship of features in the Raven-II robot states. The control command is included in this figure for an explanation of robot states but is not included in the robot state. The control command is the end-effector pose increments in Cartesian space. And it can also be given as increments in joint space. In this case, the desired joint positions will be directly determined by current joint positions and the control command, without forward and inverse kinematics. If joint-level control is applied, the current end-effector position and orientation will still be computed but the desired position and orientation will be set equal to the current values.	50
4.2	Model of the DNN and GRU in learning calibration. The outputs of the models are the errors of the joint positions and are added back to the original joint position estimation for correction.	52
4.3	Examples of zig-zag training trajectories and random test trajectories. Note that the trajectories are shown in joint space instead of Cartesian space.	52
4.4	External joint encoders are mounted on the first 3 joints of Raven-II. Actual joint positions are measured to provide ground truth during training and testing of the learning calibration.	56
4.5	Root mean square error (RMSE) evaluation of the DNN (left) and GRU (right) calibration on the test set, under removal feature ablation (top) and inaccurate feature ablation (bottom). Baseline results were learning calibration obtained with no modification on any features and are marked with '(B)'. All ablation results were based on the learning calibration with removed or inaccurate features. The color of each entry is determined by the percentage increase of the error compared to the baseline result, instead of by the value itself. On the vertical sections separated by black lines, the 1st section is the performances of baseline results and no ablation was performed. The 2nd section is the performances of the ablations on feature types. The selected features were removed or inaccurate for all 3 joints. The 3rd section is the performance of ablation on joints. The selected joints had all features removed or inaccurate.	58
4.6	Two datasets used to study the effect of homing on the learning calibration, without homing in training (left), and with homing in training (right).	62

4.7	Robustness of calibration error against robot re-homings when using different features as input of the DNN and GRU model. DNN1 and GRU1 were trained by Dataset 1. DNN2 and GRU2 were trained by Dataset 2, as shown in Fig. 4.6. Six individual test sets were collected with robot re-homing in between. In the selected features, encoder values and offsets as well as the 4th motor position (spare and not used) were excluded. For a clearer comparison of learning calibration improvement, the error following an offset compensation was used as the baseline (black line), rather than the error from the original joint positions recorded in the robot's state.	63
4.8	Motor encoder values of the first 3 joints affected by robot homing. The mean encoder value and offset compensated mean encoder value of the training set and 6 test sets after 0-5 robot homing are shown. The power of robot hardware stayed on during the collection of training and test sets.	64
4.9	DNN calibration: example joint trajectories from 6-hour continuous operations utilizing calibrations with different model sizes, training sets, and input features. The 3-letter legend represents "model size - training set - input features". Model size can be large (L) or small (S), corresponding to the large models (Fig. 4.2 and 5.2.1) utilized in most of the experiments or the lightweight models developed in 4.3.5, respectively. The training set can also be large (L) - a 2508-second trajectory with multiple sparsities, or small (S) - a 476-second trajectory with a sparsity of 1/4. The input features can be all (A) features in the robot state, selected (S) important features by ablation study, or missing (M) important pose features. Compared to fixed offset compensation, most of the calibration configurations had a considerable improvement in accuracy, except for "S-S-A" - small model, small training set but all robot states as input, and "L-L-M" - large model, large training set, but the important pose features were missing.	69
4.10	GRU calibration: example of joint trajectories in 6-hour continuous operation utilizing calibrations with different model sizes, training data, and input features. The legends have the same meaning as Fig. 4.9. The performance of the GRU calibration configurations was similar to the DNN calibration.	70
5.1	Workflow of the system. The segmentation model is first trained by labeled images. Uncertainty estimation is performed on the unlabeled real images to select the most informative ones, which are then manually labeled. Next, synthetic images are generated from the labeled images. There are 2 types of synthetic images. Type-1 syntheses have the same instrument as the original real image while Type-2 syntheses have the same background.	77

5.2	Generation of a synthetic image. Note that y^v is only for visualization, where the solid green line indicates the outline of the instrument, the yellow area is from the background image x^b and the blue area is from the instrument image x_c^i . The transition area can be found around the boundary of the instrument on the synthetic image.	78
5.3	Multi-blending: images on the left and right are similar because they have the same size and position of the instrument and background, however, the instruments are blended on the background by different blending methods and parameters (average fusion on the left and Gaussian fusion on the right). According to [1], when training deep learning models, using multi-blended synthetic images can prevent the negative effect of blending artifacts.	80
5.4	Original real images (left), Type-1 synthetic images (center) and Type-2 synthetic images (right). Compared with the real images, Type-1 has different backgrounds and Type-2 has different instruments.	81
5.5	Generation of inpainting backgrounds (right). The instruments on the original images (middle) are removed by self-rotation / flipping or existing backgrounds (left).	83
5.6	The distribution of BALD acquisition values on unlabeled images (left) and examples of queried images with the softmax probability masks (right) in early, middle, and late AL iterations. 395 real annotated images were used in the training and generation of synthetic images. The segmentation model was retrained for 70 epochs after each active learning iteration.	85
5.7	Example images from Sinus-Live (top) and Sinus-Cadaver (bottom) datasets. . . .	86
5.8	Evaluation of the model trained by different budgets of real images, with and without active learning (BALD) and synthetic images (SYN). The solid lines indicate the average over 5 repetitive tests and the shades indicate the standard deviation. The horizontal axes are nonlinear for better visualization. The training on the EndoVis 2017 dataset did not converge on the 1% or 2% budget because of too few images. Thus, the evaluation of this dataset started at 5%. The baseline result is obtained by training the model on 100% real images (no synthetic images), regardless of budget.	90
5.9	Segmentation result (mDSC) with different backgrounds. The baseline result is obtained by training the model on 100% real images (no synthetic images).	92

5.10	Comparison of segmentation performance with state-of-the-art approaches. Sinus-Live dataset: CycleGAN [2], LC-GAN [3], and fully supervised MFFA [4]. EndoVis 2017 dataset: AGSD [5], FUN-SIS [6], PAT-Ext [7], Multi-blending and Mix-blending [1, 8]. Evaluation metrics (mDSC or mIoU) are chosen based on the commonly used ones in the compared methods. Note that LC-GAN used annotated images from the cadaver dataset and FUN-SIS used annotated images from other existing datasets. Multi-blend and Mix-blend methods used 14180 auto-labeled instrument foregrounds to generate synthetic images. For the Sinus-Cadaver dataset, to the best of the authors' knowledge, no unsupervised, semi-supervised, or GAN-based learning methods were reported at present.	93
5.11	Confusion matrix of the image review result from 3 expert surgeons. 200 synthetic and 100 real images were reviewed. For each image, experts were asked to choose whether the image was more likely to be 'real' or 'synthetic'.	95
5.12	Examples of the most realistic synthetic images that received 3 out of 3 realism ratings (top), and less realistic synthetic images received 0 out of 3 realism ratings (bottom).	95
5.13	Realism ratings from the image review. Synthetic images with 1 or more realism ratings are considered 'more realistic', and otherwise 'less realistic', as the yellow line indicates.	96
5.14	Principal component analysis on extracted features of real images in the training and test set, more realistic synthetic images, and less realistic synthetic images, corresponding to Table 5.3. Despite that less realistic synthetic images expand the training set slightly better than more realistic synthetic images, the distributions of the two image sets are close. This is notable considering their significant differences in human-perceived realism, as depicted in Fig. 5.12.	98
6.1	The hardware and setup of the force actuation system. There are 6 motor-cable units (marked by red circles) installed around the robot workspace, and all cables are connected to the robot end-effector. By adjusting the tensions on cables, different forces can be applied to the end-effector.	105
6.2	The control workflow of the force actuation system. The higher-level feedback control is on the force command, while the lower-level feedback control is on each single cable tension.	106
6.3	Cable tension output under the same motor torque command but different directions of motor rotation. If the motor rotation and torque have the same direction, the actual cable tension is smaller than the command. If the motor rotation and torque have opposite directions, the actual cable tension is larger than the command. . . .	107
6.4	320 Hz load cell drivers for feedback control on cable tensions.	108
6.5	Simulation of the force actuation system with the bounding boxes of Raven-II surgical robot and its workspace.	110

6.6	Monte-Carlo simulation of the workspace of Raven-II. The blue box is approximate size of the robot base as a size of reference. For better visualization, the blue, green, and red edges are generated by having joint 1, 2, and 3 as variable, respectively, while setting the remaining 2 positional joints at maximum or minimum.	111
6.7	Measurement of motor locations with intermediate frame reference plates. The MicroScribe MX device and the frame reference plates can be located at any convenient locations, as long as the locations are fixed while obtaining measurements of reference points. By measuring known reference points on the robot base, the frame of MicroScribe can be obtained using the Kabsch algorithm. If the motor is far away from the robot base so that the MicroScribe cannot reach the robot base and motor unit at the same time, the frame reference plates can be used to extend the measurement range. Multiple frame reference plates can be used to further extend the range.	112
6.8	Chain of spatial relationship from the robot base frame to the motor locations, using intermediate frame reference plates. 0P_1 means the reference points of frame 1, represented in frame 0. 0_1T means the transformation matrix from frame 0 to 1. Frame 0 is the robot base frame. Frames 1, 2, ... are the intermediate reference frames. Frames d01 mean the device frame when measuring intermediate reference frames 0 and 1.	113
6.9	Joint torque in robot state without local motion when external forces apply. Although various external forces are still applied in the static status, the joint torques have no considerable changes, because the external forces are passively resisted by the joint frictions.	115
6.10	Joint torque in robot state with local motion when external forces apply. Compared with Fig. 6.9, changes in joint torques can be observed in both moving and static status. However, it should be noticed that the joint torques in the static status are caused by both local motion and external force.	116
6.11	Recorded robot trajectory with external force. The "stars" are for studying the static condition of the robot, in which the robot is not moving while the external force keeps changing. Both position and force are normalized by their ranges for better visualization.	117
6.12	Example of force estimation performance when the robot has no local motion in static status. Without local motion, the neural network model cannot estimate the external force in a static status because external forces are not reflected in joint torques, as shown in Fig. 6.9.	119
6.13	Example of force estimation performance when the robot has local motion in static status. With local motion, the neural network model can estimate the external force in a static status, because the joint frictions are overcome by local motion, so external forces can be reflected in joint torques.	120
6.14	Example of force estimation performance when the robot is moving.	121

6.15	Example of force estimation performance when the robot is static.	121
A.1	Workflow for proposed online position precision estimation system. Pipeline entails vision based ground truth measurement (left) and data driven robot position estimator (right). Training labels passing from left to right are true Raven-II position information.	143
A.2	16 marker points used for camera pose calibration are placed at 8 ground locations in robot workspace.	143
A.3	Geometric relations among image plane, camera and marker.	144
A.4	Circle detection method and the preprocessing steps. Color image (1) is first converted to gray and Gaussian blur is applied (2). Then Canny edge (3) is taken from the gray image. In the other pathway, the color image (1) is divided into red, green, and yellow color channels (4). Then Gaussian blur with σ of 10% radius followed by binarization is applied (5). Image (6) is elementwise multiplication of (3) and (5). Hough circle detection is applied in each channel in image(6), which combines to result in (7). Finally, image (8) is a color check near the circle border to eliminate false positives.	144
A.5	Comparison of Canny edge on original image (left), normal blurred image (middle) and processed image by our system (right).	146
A.6	Geometric relations between ball centers and end effector position.	148
A.7	The performance of different hyperparameter sets. The legend ‘L654_sigmoid’ means that there are 3 hidden layers of 600, 500 and 400 units with sigmoid activation. ‘_elu/_relu’ refers to ELU[9] and RELU[10] activation functions	150
A.8	Experimental setup.	152
A.9	Comparison of actual position, motor encoder based forward kinematics position, and neural network estimation. The neural network estimator decreases the positional RMS error by 83.6% and the standard deviation of error by 59.4%.	154
A.10	RMS error and standard deviation of error in end effector position with and without neural network online estimator.	154

LIST OF TABLES

Table Number	Page
3.1 Learning and Regression Calibration Error with Different Directions of Trajectories	42
3.2 Calibration Performance of Models Trained on Joint Position Error or End-to-end .	43
3.3 Direct Comparison to the State-of-the-art in Joint Space	43
3.4 Indirect Comparison to the State-of-the-art in Cartesian Space	44
3.5 Deep Neural Network VS. Linear Regression	45
4.1 Features in Raven-II State	46
4.2 Calibration Error under Modified Motor Torque Features	73
4.3 Calibration Error under Modified Joint Position Features	74
4.4 Calibration Error of Lightweight Models with All or Selected Features	75
5.1 Segmentation Performance with Different Budgets	101
5.2 Segmentation Performance with Different Types of Synthetic Images and Multi-blending	102
5.3 Segmentation Performance with Different Syntheses Realism	103
A.1 The Error of Measurement System	153

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Professor Blake Hannaford, for his guidance, support, and mentorship throughout my Ph.D. and master's journey. His insight, patience, and encouragement have shaped not only this dissertation but also my growth as a researcher, engineer, and educator. I am especially grateful for the freedom to explore challenging research questions and for the thoughtful advice that helped me navigate both technical and professional decisions. Prof. Hannaford gave me the opportunity to join the BioRobotics Lab in 2018 when I could not even use Arduino or code in Python. His insight, kindness, and support gave me a smooth learning curve for this dissertation. As many Ph.D. students might feel, I also felt stress and anxiety sometimes. But with Prof. Hannaford's support and suggestions, motivation, curiosity, sense of accomplishment, and happiness have always been the dominant themes of my study, research, and life. He served not only as a role model for me in research and teaching, but also taught me the importance of treating others with kindness, as well as how to encourage and guide them. I will strive to approach my future students in this very spirit.

I am sincerely grateful to the members of my dissertation committee, Professor Blake Hannaford, Professor Sam Burden, Professor Eli Shlizerman, Professor John Raiti, Professor Per Reinhall and Professor Byron Boots, for their time, feedback, and valuable perspectives. Their suggestions have helped strengthen this work and broaden my understanding of the field.

I would also like to thank the faculty, staff, and colleagues in the Department of Electrical and Computer Engineering at the University of Washington for creating a supportive academic environment. I am especially thankful to the members of the BioRobotics Lab, past and present, including Dun-Tin Chiang, Andrew Lewis, Niveditha Kalavakonda, Melody Su, Shan Lin, Yana Sosnovskaya, Boyang Li, Justin Su, Armina Moghadasi, Joseph Garret Sulliva, Zijie Jin, Amber Chou, Annika Pfister, Siena Villancio-Wolter, for their collaboration, technical discussions, encouragement, and friendship. Working alongside such talented and dedicated people has been one of the most reward-

ing parts of my graduate experience. I am very grateful for Dun-Tin's excellent work and we have been working back-to-back in the lab for 2 years. The force actuation system cannot be built without his knowledge, hardworking and innovative ideas. I would also like to express my gratitude to Andrew who taught me how to use the Raven-II surgical robot and fixed the robot many times when I broke cables. I would also like to thank Niveditha for all her suggestions and support throughout my entire Ph.D. journey, especially for completion of Ph.D. and job hunting. Finally, I would like to express my gratitude to Melody and Shan for their close collaboration on research projects and suggestions, which helped me to be more and more mature in research and collaboration. Especially, I would also like to thank Xingjian Yang for collaboration and friendship for a decade at UW.

I would like to extend my appreciation to my friends, both inside and outside the university, who offered companionship, encouragement, and much-needed moments of balance during the long years of graduate school. Their support helped me stay grounded through both the exciting and difficult stages of this journey. Especially, I would like to express my gratitude to Guotao Xu, Sihan Qu, Qian Qian, Bari Demo, Haoming Yan, Xiaodong Dai, Guanyu Chen, and Yanru Wang for more than a decade of friendship since my childhood.

Most importantly, I am deeply grateful to my family for their unconditional love, patience, and belief in me. Their sacrifices and constant encouragement have given me the strength to continue moving forward. This dissertation would not have been possible without their support.

Finally, I would like to thank everyone who has helped me along the way, whether through mentorship, collaboration, friendship, or a simple word of encouragement. This dissertation represents not only my own effort, but also the support and kindness of many people who have accompanied me throughout this journey.

DEDICATION

to my parents, Yan Peng and Lifeng Ren,
your support has given me the courage to pursue my dreams, the strength to overcome difficulties,
and the confidence to keep moving forward.

Chapter 1

INTRODUCTION

1.1 Background and Motivation

Robot-assisted Minimally Invasive Surgery (RAMIS) offers the possibility of accurate and reliable operations between surgeons and surgical robots [11, 12, 13], with improved outcomes for patients [14], [15]. In RAMIS, millimeter-level accuracy is preferred [16]. In surgical robotics mechanisms, the highly elongated form factors of endoscopic surgical instruments, and the necessary re-location of sensing and actuation to more proximal locations, impose fundamental limitations on mechanical accuracy. Cable drive mechanisms, traditionally used in robotic laparoscopic instruments, such as Raven-II [17] and da Vinci Research Kit (dVRK) [18], are problematic due to long runs of highly loaded drive cables, and their routing around numerous pulleys whose diameter is substantially below standard cable design guidelines [19]. Accurate manipulation can still be achieved in conventional bilateral teleoperation (human-in-the-loop control) under this inaccuracy, as the human operator can compensate for the error with visual feedback [20], typically obtained through endoscopes [21]. However, as the automation of surgical robots has been increasingly drawing attention in recent years [22, 23], such as suturing [24, 25, 26], debridement [27, 28], and peg transfer [29, 30, 31, 32], accurate robot pose becomes more demanding [33].

In addition to using forward and inverse kinematics [34, 35], accurate state estimation for cable-driven surgical robots like Raven-II usually relies on the modeling of cable-driven effects [36, 37]. Numerous parameters and measurements are necessary to build the model-based estimation of robot pose. Some parameters could be obtained from the computer-aided design (CAD) model, such as initial inertia matrices, mass, and center of mass. On the other hand, it can be more difficult to obtain parameters such as spring and damping constant, as well as Coulomb and Viscous friction [38]. Furthermore, cable-pulley friction is relevant to cable tension that is dynamic during robot operation [39]. As a consequence, inaccurate or experimentally measured parameters can undermine the performance of model-based state estimation.

Recent efforts use data-driven and machine learning calibration to reduce the error caused by the cable-driven mechanism and achieve competitive performance [40, 41]. Unlike model-based methods, learning-based methods can establish end-to-end models to provide more accurate robot poses. Various input features can be utilized by the learning models, such as control command [42] and robot state [43]. However, what types of information features of the robots contribute to the learning calibration, as well as how the features contribute, remains undiscovered.

Different surgical robot arms usually have different arm designs, motors, and controls. Even for different individual robots of the same type, the difference in cable tension still results in different kinematics and dynamics. This makes it difficult to generalize calibration models of one robot to other robots. Furthermore, cable characteristics and calibration accuracy can change over time and operating [44, 45]. Therefore, re-calibration may be necessary for individual robots before each operation. Due to the data hunger of learning-based calibration and the need of re-calibration, efficiency becomes important for feasible implementation in real practice.

On the other hand, visual tracking is also used as feedback of end-effector pose. Endoscopes are commonly used to provide visualization of the surgical site in real time. Segmentation of instruments is critical to the interpretation of endoscopic surgical images, and deep learning has been applied to this task [46, 47, 48, 49, 50]. In medical practice, labeled data is costly, and typically only trained experts can accurately annotate the images [51, 52]. Consequently, small training sets are a frequent challenge to surgical instrument segmentation [53].

Recent efforts use synthetic data to alleviate the workload of annotating [54, 55, 56]. Synthetic images generated from simulation have accurate labels without manual work, but [2] suggest that the domain gap between synthetic and real datasets reduces the performance of a model trained by synthetic images. Domain adaptation techniques, such as generative adversarial networks (GANs) [57, 58], can alleviate the domain gap and artifact of synthetic images. However, for endoscopic sinus surgery, because of reflections on metallic instruments, as well as blur and liquids on the tissue-instrument boundary, GAN-based methods may generate inaccurate synthetic images while the ground truth segmentation masks remain the same, which may confuse the training [3].

Besides GAN-based methods, the method of generating synthetic images by copying and pasting real object images onto real background images generalizes to many segmentation tasks [59, 60] and efficiently generates synthetic images with reduced concern for domain gap and inaccurate ground

truth [1], thereby enhancing the reliability and effectiveness of segmentation models. Furthermore, when background images are difficult to acquire, image inpainting methods such as patch-based image synthesis [61] can generate vivid backgrounds from cropped images.

When the resources for annotating real images are limited, the effectiveness of blending synthetic images through the copying and pasting method depends on the strategic selection of real labeled images. Active learning (AL) enhances this process by selecting the most uncertain or informative samples from an unlabeled dataset for manual annotation [62], thus efficiently reducing the reliance on extensive real labeled data [63, 64]. When AL is combined with deep learning techniques, it facilitates faster convergence and achieves superior performance with less labeled data.

Haptic perception is a fundamental but still underdeveloped capability in robot-assisted minimally invasive surgery. In conventional surgery, surgeons rely on direct tactile and kinesthetic feedback to regulate tissue interaction forces, identify tissue properties, and avoid excessive loading on delicate anatomical structures. In teleoperated surgical robots, however, the surgeon is physically separated from the instrument tips, and many widely used robotic systems provide limited or no direct force feedback, requiring surgeons to infer contact forces primarily from visual cues [65, 66, 67]. This limitation is especially important for cable-driven surgical robots, where distal force sensing is difficult because sensors near the end-effector must satisfy strict constraints on size, sterilization, wiring, biocompatibility, and integration with compact instruments. As a result, force information is often unavailable or indirectly estimated from proximal measurements such as motor currents, joint torques, robot states, or visual deformation, which can be challenging due to cable friction, hysteresis, backlash, and interaction with the surgical environment [68, 69]. Reliable and accurate distal force sensing or sensorless force estimation is therefore not only valuable for restoring haptic feedback to the surgeon, but also essential for closed-loop force control, safe tissue manipulation, palpation, suturing, and future semi-autonomous surgical tasks.

1.2 Main contributions

To improve the position accuracy, an efficient data-driven joint calibration of Raven-II is proposed. Zig-zag calibration trajectories are used for desirable coverage of the workspace. By studying the effectiveness of the direction and sparsities of the calibration trajectories, the calibration can be

finished in 8 to 21 minutes and has an accuracy of 0.104° , 0.120° , and 0.118 mm in joints 1, 2, and 3, respectively, reducing more than 76% of the error. Although continuous operating and load cause decreased accuracy, with sufficient training data the accuracy remains within 0.855° , 0.432° , and 0.181 mm for the positioning joints in 6-hour heavily loaded operating. To achieve robust calibration performance, the choice of input-output of the calibration models, and the robot homing procedure are also studied. This work makes the following contributions: 1) an efficient joint calibration pipeline that costs less than 21 minutes and improves the accuracy significantly, which make re-calibration possible before each operation; 2) evaluation of calibration performance on 6-hour idleness, unloaded, and loaded operating, in the time scale of real surgeries; 3) besides the DNN calibration with best accuracy, a linear regression calibration established with slightly compromised accuracy but fast enough inference speed to meet the robot's 1000 Hz servo control; 4) a general software controller for Raven-II robot which to the best of the author's knowledge, is the only one providing a Python API. To improve the accuracy of cable-driven surgical robots, compared to model-based calibration methods, the proposed data-driven calibration method achieves better accuracy and does not require studying and testing sophisticated cable characteristics that can vary with individual robots. Compared to end-to-end learning-based calibrations on end-effector pose, the proposed method focuses on the more direct source of error - joint positions, and thus allows smaller models, faster training and inference, while maintaining desirable accuracy through long operating time.

Although current model-based and learning-based approaches suggest desired advantages in the calibration of cable-driven surgical robots, some limitations still exist. Model-based methods suffer from complicated cable characteristics and coupling, which results in less accurate calibration. Time-consuming experiments are also needed for individual robots to determine some parameters of the cables. In contrast, learning-based methods usually achieve better accuracy without the necessity to analyze complicated models. However, the learning models' nature of data hunger and computational cost also pose obstacles in real practice. In the proposed work, ablation studies are performed to find important features in the robot states and significantly reduce the input dimensions of the learning calibration models. When the training trajectory is shortened to 8 minutes, the calibration model with all robot states as input shows a significant drop in accuracy, while the model with only the important states as input retains desirable performance, and the training time on CPU

is reduced to 1 minute. With the short training time, the calibration accuracy remains desirable in 6-hour continuous operating, which suggests the feasibility of learning-based calibration in the time scale of real practice. In terms of identifying important robot states, we find that removal ablation - removing features in both training and testing, is not sufficient to determine whether a feature is important. Thus, we also develop inaccurate ablation - adding noise to features only in testing, and compare the results. Based on the different results of the two ablation methods, we further explore how the features contribute to the task by human knowledge of the robot mechanism and system.

To reduce the workload of manual annotation in training of instrument segmentation model, a feasible method is developed that combines active learning (AL) with copy-and-paste image syntheses, overcome the difficulty in the generation of synthetic images for endoscopic sinus surgeries [2, 3], and further improve the performance of instrument segmentation. We utilize active learning to choose informative unlabeled images to annotate, and a copy-and-paste method to generate synthetic images that have instruments and tissue backgrounds inherited from original real images, which can improve the utilization of the selected real images. We aim to test the hypothesis that the segmentation model trained with the synthetic images and a smaller number of selected real images has competitive performance compared to models trained on fully labeled real datasets. We also aim to measure the relative effectiveness of different types of synthetic images, external backgrounds, and realism on instrument segmentation accuracy. We also investigate the effects of fusion near the boundary of the instrument and multi-blending on the visible boundary artifact on the performance of segmentation near the boundary. Three open-source datasets are used in the experiments - the UW-Sinus-Surgery Cadaver Dataset, Live Dataset [48], and the EndoVis 2017 Dataset [70]. Among approaches for the generation of accurate, labeled synthetic images in endoscopic surgery, we leverage active learning to minimize the workload of manual segmentation, then copy surgical instruments and paste them into a surgical background image at random positions and orientations. The background image can be a video frame where a human confirms that no instrument is present, or it can be created from a segmented image by infill from adjacent backgrounds at the instrument location. This image synthesis method is relatively easy to implement and adjust. In comparison, while unsupervised learning eliminates the need for annotating training data, it may incur effort and cost to acquire pseudo labels or labeled data from other datasets and simulations. In contrast, the proposed method, requiring a small amount of labeled data, demands no additional human effort.

Motivated by the need for haptic feedback, this dissertation also investigates a force actuation and distal force sensing framework for Raven-II, where controlled external forces are applied to the end-effector to collect representative data for learning-based force estimation and to study how surgical robot states can be used to infer contact forces without permanently adding sensors to the surgical instrument.

To summarize, this dissertation makes the following contribution:

- An efficient data-driven calibration for joint position estimation for Raven-II surgical robot is developed. The calibration procedure needs less than 20 minutes, and the calibration accuracy remains desirable in 6-hour continuous operating.
- A feature ablation study is performed for the learning calibration model, in which joint position and motor torque are found to be most important. By excluding other unnecessary features, the accuracy and efficiency of the calibration model are improved simultaneously.
- A pipeline for generating synthetic endoscopic images in sinus surgeries. The synthetic images are generated from real images guided by active learning, in which the most informative real images are chosen by active learning, and then synthetic images are generated based on the informative real images to "make the best use of them". This can reduce the workload of manual annotation of training data.
- A Python-based controller for Raven-II is developed, using CRTK control API. Compared with the original C++ controller, this new Python controller has more high-level control functions, such as trajectory following and data recording.
- A motor-cable force actuation system is developed. It can apply known and controllable forces on the end-effector of Raven-II, and meanwhile follows the movement of Raven. This system allows data collection on Raven with various external forces for training force estimation models.
- A low-cost visual tracking system for Raven-II end-effector is developed, which only uses 4 webcams that are easy to calibrate. This method was cited, used, and improved by research

labs in other universities.

1.3 Organization of the dissertation

This dissertation is organized as follows. Chapter 2 reviews the technical background and related literature on pose calibration and haptics of cable-driven surgical robots, as well as surgical instrument segmentation, active learning, and synthetic image generation. Chapter 3 presents an efficient data-driven calibration method for the joint positions of the Raven-II surgical robot, including the proposed methodology, experimental setup, results, and discussion. Chapter 4 further investigates the calibration problem through feature ablation studies, with emphasis on identifying important robot-state features and improving the efficiency and robustness of learning-based calibration. Chapter 5 focuses on surgical instrument segmentation, proposing a framework that combines active learning with copy-and-paste synthetic image generation to reduce annotation effort while maintaining strong segmentation performance. Chapter 6 introduces a distal force sensing and force actuation system developed to support future force estimation and haptic interaction studies in surgical robotics. Finally, Chapter 7 concludes the dissertation and discusses directions for future research. In addition, Appendix A presents earlier work on real-time data-driven precision estimation and visual tracking of the Raven-II end-effector, Appendix B provides all code and data used in this dissertation (except for the force actuation and estimation work, which is under patent application), and Appendix C reflects on how AI tools supported the research process.

Chapter 2

BACKGROUND AND LITERATURE REVIEW

2.1 Pose Calibration of Cable-Driven Surgical Robots

To alleviate the inaccuracy of cable-driven surgical robots, both model-based and data-driven methods show desirable performance. Model-based methods benefit from expert analysis of the robot design and have better explainability in general. However, accurate modeling of cable-driven mechanisms can be challenging with consideration of practical cable effects [36, 37, 71]. To model cable effects, Miyasaka et al. [72] utilize a Bouc-Wen hysteresis model and a linear damper to model longitudinally loaded cables, in which 9 hysteresis model parameters are optimized by a genetic algorithm. Haghighipanah et al. [38] apply an unscented Kalman Filter (UKF) to obtain a more accurate estimation of the positioning joints of Raven-II. A significant improvement in accuracy can be found in joints 2 (1.669°) and joint 3 (0.928 mm), however, joint 1 (1.295°) sees less reduced error due to relatively higher joint stiffness. Baek et al. present ViO-Com, a vision-based optimized feed-forward compensation method for cable-driven surgical robots that enhances control accuracy by reducing hysteresis using a Bouc-Wen model, optimized with CycleGAN for segmentation of the surgical manipulator and a Siamese Convolutional Neural Network to estimate joint angles, significantly improving precision than single vision feedback compensation [73]. Lee et al. present a method to model and compensate for non-linear hysteresis for tendon-sheath-driven manipulators, including backlash and dead zone effects, using motor current for parameter identification, improving the accuracy of robotic control without external sensors [74].

On the other hand, by utilizing features from robot states, data-driven (preferably learning-based) methods can achieve state-of-the-art accuracy with sufficient training data [41, 42]. With the effectiveness of data-driven methods proved by many, the efficiency of calibration time, training data, and real-time inference remains less investigated. Huwang et al. present a calibration based on a recurrent neural network that reduced the estimation error of the dVRK end-effector position from 2.96 mm to 0.65 mm [41]. This calibration takes 31 minutes and further helps automate surgical peg

transfer and achieve superhuman performance [32]. Seita et al. present a two-phase calibration, a DNN model for coarse calibration and then a random forest model for fine training [28]. Mahler et al. develop a Gaussian process regression calibration for Raven-II, in which velocity is found to be an important input for the model [42]. Cursi et al. propose a method to improve model learning and control of tendon-driven surgical robots by incorporating a backlash compensation technique alongside a feedforward artificial neural network to address nonlinearities caused by tendon backlash [75]. To ensure the accuracy of supervised data-driven calibration, reliable ground truth is of importance. Visual-based tracking shows considerable feasibility and accuracy. In [43] and [41], ground truth is obtained by tracking colored spheres mounted on the robot end-effector using 4 monocular cameras or 1 RGB-D camera. In [42], ground truth is obtained by tracking LEDs mounted on the end-effector.

By analogy with ablative brain surgery [76], ablation studies in Artificial Intelligence remove certain components of the AI model and compare the performance, to evaluate and understand the contribution of the components [77], [78]. A unit ablation study on computer vision classification accomplished by Meyes et al. [79] suggested that the more weight distribution changed during training, the more contribution that unit had. Kauchak et al. [80] proposed an ablation study on features in a learning-based medical text difficulty classification. When removed, some features caused more changes in performance than other features.

2.2 Instrument Segmentation, Active Learning, and Synthetic Images

Using synthetic images is an intuitive approach to reducing the annotation workload. [1] proposed a similar ‘cut’ and ‘paste’ method to generate synthetic data for kitchen object detection. Their result showed that simply copying and pasting could result in artifacts such as aliasing of boundaries, which decreased learning performance. By improving the blending between sprite and background, their approach reached competitive performance combined with 10% real data. However, [59] showed that pasting without any blending had a similar performance to blending.

In further related studies, [60] described object instance segmentation with weakly-supervised cut-and-paste adversarial learning: a discriminator was used to distinguish between real and synthetic images. GANs [81] have been also implemented to generate synthetic medical images [82, 83]

based on a similar discriminator. Recent implementations include image-to-image translation from simulated images to real images for cataract surgery [84] and laparoscopic surgery [85], as well as from cadaver images to live images [3] for sinus surgery.

Active learning is dedicated to selecting and labeling the most informative training images that can reach near-optimal performance with the fewest annotations (human effort) [86, 87]. Typically, in active learning, unlabeled images are selected by criteria such as maximum entropy and least confidence [88, 89, 90]. In some cases, however, these criteria do not outperform random selection [52, 91]. Thus, more advanced criteria such as Bayesian active learning by disagreement (BALD) are proposed [92]. The BALD criterion combines a high overall uncertainty with a term that increases the weight of disagreement among the population. [93] presented a study on semantic segmentation of prostate medical images with active learning, and the BALD criterion outperformed maximum entropy, especially when the budget for annotation was small. [94] proposed a Bayesian generative active deep learning, which combined active learning and GAN data augmentation. The evaluation of image classification tasks suggested that the combined method outperformed each single method. [95] developed an active learning approach based on Deep Bayesian Networks for instrument presence detection and surgical phase segmentation. The experiments suggested that with the same amount of training data, active learning outperformed random selection in training the surgical workflow analysis model.

With plenty of unlabeled data but expensive annotation, unsupervised and semi-supervised learning are also preferred in medical imaging [96, 97]. [5] proposed an unsupervised learning method for surgical instrument segmentation that used generated anchors as pseudo labels with ambiguity resolved by temporal coherence. Instead of using pixel-level annotations, [98] developed an approach utilizing weak annotations provided as stripes over the different objects in the image. Recent efforts also include utilizing the local center of mass [99], 2D points of interest [100], and existing annotations from other datasets [6]. With minimized usage of annotation, unsupervised and semi-supervised learning can achieve comparable performance to fully supervised learning. Momentum Contrast (MoCo) [101] for unsupervised visual representation learning also suggested close performance to supervised representation learning on medical imaging datasets [102, 103]. Besides the current frame, external information such as robot pose [104, 105] and coherent frames [4, 106] can also be utilized to improve the performance of segmentation in supervised and self-supervised

learning models.

2.3 Haptics, Distal Force Sensing, and Cable Force Actuation for Surgical Robots

Haptic feedback is an important but still limited capability in robot-assisted minimally invasive surgery. In conventional surgery, surgeons can directly perceive tool-tissue interaction forces, while in teleoperated surgical robots this information is largely removed and the surgeon must often infer contact force from visual cues [65, 66]. This lack of force feedback can affect delicate tasks such as palpation, suturing, tissue manipulation, and dissection. Recent reviews and meta-analyses suggest that haptic feedback can reduce applied forces, reduce peak forces, and improve task accuracy and success rate in robot-assisted surgical tasks [67, 107]. Therefore, reliable force sensing and haptic feedback remain important directions for improving surgeon-robot interaction and surgical safety.

A key challenge is that the most useful force information is the distal interaction force at the instrument tip, but distal sensing is difficult to integrate into surgical tools. Sensors near the end-effector must satisfy strict constraints on size, sterilization, wiring, durability, biocompatibility, and cost. Direct sensing approaches can provide high-quality measurements in research systems; for example, Chua and Okamura developed a modular 3-DOF force sensor for robot-assisted minimally invasive surgery research [68]. However, permanent distal sensing remains difficult for compact cable-driven surgical instruments. As an alternative, sensorless force estimation methods infer contact force from available robot states such as joint motion, motor current, joint torque, or cable tension. Model-based approaches require accurate modeling of friction, gravity, cable transmission, and trocar interaction, while learning-based approaches can compensate for some nonlinear and unmodeled effects when representative training data are available. For example, Yang et al. proposed a hybrid model and learning-based framework for surgical robot force estimation that combines robot modeling with learned compensation for environmental effects [69].

Cable-driven force actuation systems provide a useful way to generate controlled external forces for force sensing and haptic applications. In cable-driven parallel robots, multiple motor-driven cables generate multi-directional forces while maintaining positive cable tensions, offering advantages such as low moving inertia, large workspace, and flexible reconfiguration [108]. Recent studies have continued to address key challenges in these systems, including tension distribution, cable

relaxation, unknown geometric or dynamic parameters, and sensorless tension control [109, 110]. Although these systems differ from the external actuation system used in this dissertation, they provide relevant design principles for applying, sensing, and regulating forces through motor-cable units.

Chapter 3

EFFICIENT DATA-DRIVEN CALIBRATION OF JOINT POSITIONS

Accurate joint position estimation is crucial for the control of cable-driven laparoscopic surgical robots like the Raven-II. However, any slack and stretch in the cable can lead to errors in kinematic estimation, complicating precise control. This chapter proposes an efficient data-driven calibration method, requiring no additional sensors post-training. The calibration takes 8–21 minutes and maintains high accuracy during a 6-hour heavily loaded operating. The Deep Neural Network (DNN) model reduces errors by 76%, achieving accuracy of 0.104° , 0.120° , and 0.118 mm for joints 1, 2, and 3, respectively. Compared to end-to-end models, the DNN achieves better accuracy and faster convergence by correcting original inaccurate joint positions. Additionally, a linear regression model offers 160 times faster inference speed than the DNN, suitable for Raven’s 1000 Hz control loop, with slight compromises in accuracy. This approach should significantly enhance the accuracy of similar cable-driven robots.

3.1 Method

3.1.1 General-purpose Controller for Raven-II Surgical Robot

To achieve efficient control and data collection on the Raven-II robot, a general-purpose software controller is developed, based on ROS, Python, and Collaborative Robotics Toolkit (CRTK) [111]. CRTK is a common control API for two surgical robot research platforms, the Raven-II and the da Vinci Research Kit. The controller has the following parts (shown in Fig. 3.1).

CRTK Python core establishes communication with the CRTK API and ‘ravenstate’ – a ROS topic containing the states and measurements of the robot, including joint positions, end-effector poses, run-level, etc. It also generates and sends control commands to the robot via CRTK API. The CRTK Python core achieves semi-servo control of Raven-II, around 500 Hz. The control command is scaled by a factor computed in each control loop, compared to Raven-II’s 1000Hz servo interface.

Keyboard controller allows the user to control Raven-II using local keyboard input, which

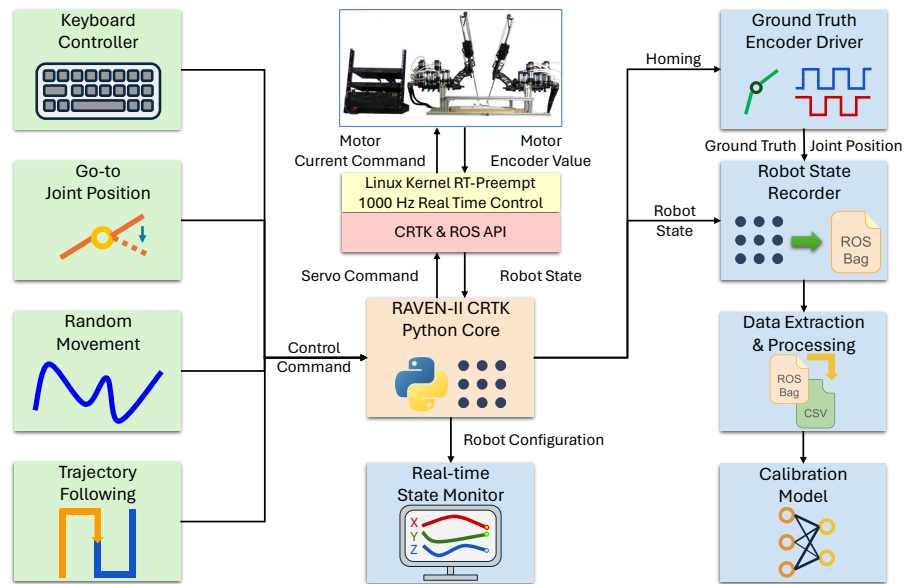


Figure 3.1: The general-purpose controller and relative tools developed for the Raven-II surgical robot, based on ROS, Python, and CRTK API. Compared to the original C++ controller that only supports servo incremental control, this new controller has more control, visualizations, data recording, and safety functions.

enables easy testing of the functions of the robot. The robot will follow a predefined constant joint or Cartesian velocity while the corresponding key is holding pressed. The keyboard controller can also change the run level of the robot, replacing the physical pause control pedal.

Go-to joint position controls the robot to reach a target joint configuration, regardless of the current position. While far away from the target configuration, joints have constant velocities. The velocities get reduced while approaching the desired position, and then the joints stop within a predefined tolerance.

Random sinusoidal joint movement controls the robot joints to run random sinusoidal trajectories. All joints are controlled independently. In each iteration, a target joint position is randomly chosen with a random velocity, both in predefined ranges.

Trajectory generation & follow. The purpose of this part is to control the robot to follow a target trajectory with a desired speed to collect training data for calibration. A trajectory is given by a sequence of configuration vectors of joint positions. The controller first moves the robot to the initial configuration using the 'go-to' control. Next, in each control loop, the current position of the robot is obtained via 'ravenstate' and CRTK API. Then, the target position is found in the trajectory, and the control command is generated based on the difference between the current position and the target position, with a desired smooth velocity.

Runtime monitor provides real-time visualization of the joint positions, end-effector poses, and control commands. It obtains robot states and plots them in an animated manner.

External joint encoder. The controller also includes an external joint encoder driver PCB and software for ground truth data collection on Raven-II. In Raven-II's initialization and homing, all joints go to the limit to register motor encoders. The external joint encoders are registered in the same way and readings are then published via ROS message.

Robot state recorder records robot states as ROS bags via CRTK API and 'ravenstate' ROS topic, as well as external joint encoder positions. Selected CRTK joint positions and external joint positions are recorded at 100 Hz, and 'ravenstate' is recorded at 30 Hz due to high dimensions. **Data extraction** processes the ROS bags and synchronizes all robot states.

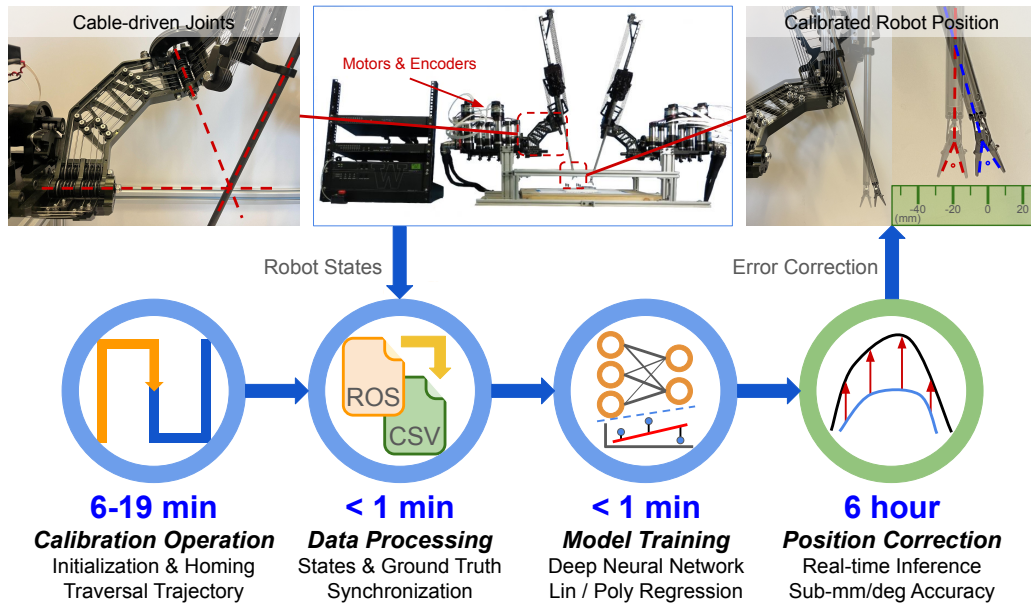


Figure 3.2: Workflow of the calibration. The time cost mainly depends on the collection of training data. The time cost of data processing and model training may vary with different CPUs. GPU is not necessary for the training of neural network models in this work, unless stated otherwise. Designed for surgical purposes, the first 3 joint axes of Raven-II always intersect at the same location which is called the remote motion center and the surgical instrument will always pass this location as the incision site (top left). In order to possess compact and light arms, the motors and encoders are mounted on the robot base, and joints are driven by cables, which results in considerable errors in joint positions obtained by remote motor encoders (top middle). Without calibration, the average positional error of Raven-II's end-effector is around 20 mm (top right).

3.1.2 Calibration Workflow

The objective of the proposed data-driven calibration is to achieve better accuracy of the positioning joints of the Raven-II surgical robot in an efficient manner. As shown in Fig. 3.2, the calibration workflow contains the following sequential procedures:

Calibration operation: Firstly, it takes less than 2 minutes to initialize the robot and perform the homing procedure, during which the robot explores joint limits and registers the motor encoders. If calibration is to be performed, temporary encoders for ground truth collection are mounted on the joints and are registered to joint positions at the same time. Then, to collect training data, the robot follows predefined zig-zag trajectories which have even distribution in the workspace of the robot (details in 3.1.3 and 3.1.1). Meanwhile, the robot states and ground truth joint positions are recorded as ROS bags [112]. Depending on the sparsity and the size of the training dataset, this procedure takes 4-17 minutes. Longer trajectories with smaller sparsity have better coverage of the robot workspace, but also require a longer time to record.

Data processing: This step extracts data from the recorded ROS bags from the previous step and saves them into .csv files. Synchronization between robot states and ground truth joint positions is also used to construct training pairs of input robot state and output joint positions. This step takes less than 1 minute with the largest dataset in this work.

Model training: With the dataset from the previous step, calibration models can be trained/fitted. Deep neural network (DNN), linear regression, and 2nd-order polynomial regression models are studied in this work (details in 3.1.4). The models take robot states as input and output calibrated joint positions. With optimized input dimension and DNN structure, all models can be trained within 1 minute (without GPU for the DNN model).

Position correction: After training, the calibration models are able to work in parallel with the robot control system in real time. The models take the original robot states as input so that no additional sensor is needed. The output is the corrected joint positions with sub-mm/deg accuracy and the accuracy remains desirable in 6-hour continuous loaded and unloaded operating.

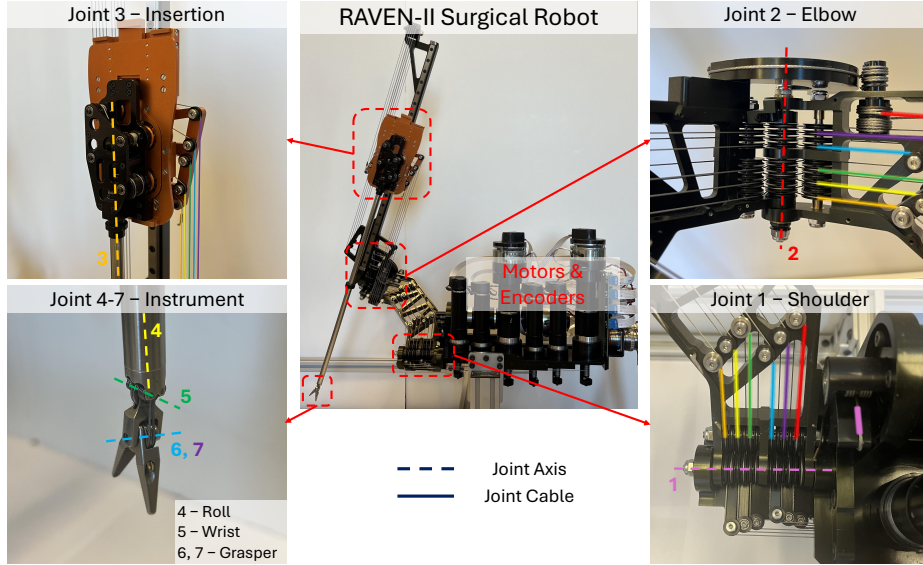


Figure 3.3: The joints and cable connections of the Raven-II surgical robot. Compared to joints 2-7, joint 1 has a considerably short cable length (alpha wrap). Joint 3 is prismatic and has a large range for insertion, while all the rest of the joints are rotational. Joints 1-3 are positioning joints and have major control of the end-effector's position. Joints 4-7 are concentrated near the end-effector and have only 1 non-zero link length of 13 mm in the kinematic model. Thus, the last 4 joints mainly control the orientation of the end-effector and have much less influence on the positional accuracy of the end-effector. Joints 6 and 7 control the left and right fingers of the end-effector and can be considered as one joint in kinematics. Thus, Raven-II has 6 joints in kinematic models instead of 7. The left arm and the right arm of Raven-II are symmetric.

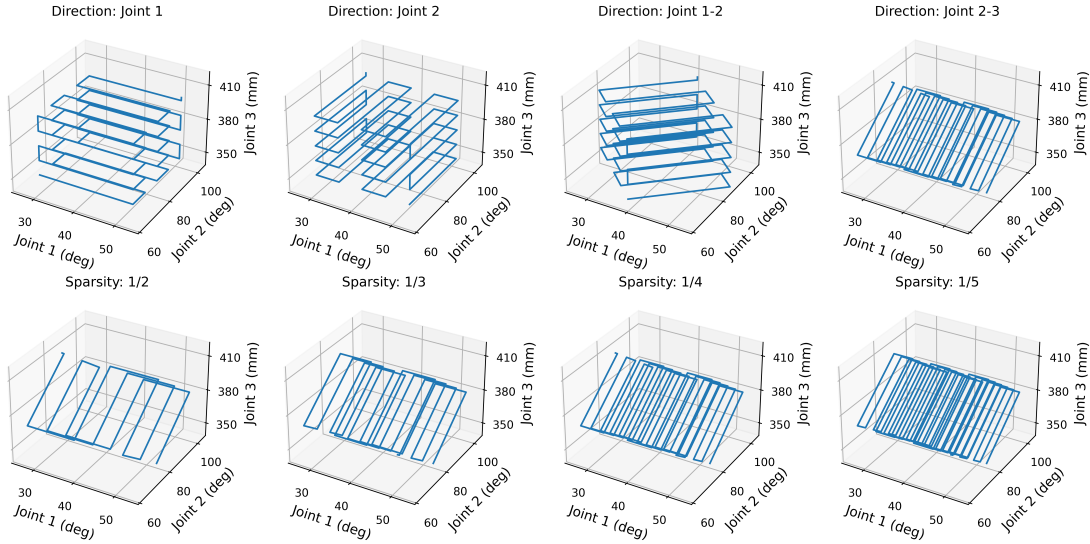


Figure 3.4: Examples of calibration trajectories with different directions (top) and sparsity (bottom). Please note that the trajectories are defined in joint space instead of Cartesian space. Different directions cause different coverage in different joints. Smaller sparsity results in better coverage of the joint space, but also takes a longer time to execute.

3.1.3 Robot Workspace and Calibration Trajectories

This work studies the positioning joints (joints 1-3) of the Raven-II robot. Fig. 3.3 shows the joints and cable connections of Raven-II. Designed for RAMIS, the last 4 joints of Raven-II are concentrated near the end-effector with very small (13 mm) or zero link lengths, and thus have much less effect on the location of the end-effector, compared to the first 3 joints.

Zig-zag trajectories with different directions and sparsity are chosen as calibration trajectories for data collection, as Fig. 3.4 shows. With limited calibration time, zig-zag trajectories ensure even coverage of the robot workspace and prevent undiscovered areas. However, because trajectories are continuous, different directions of the trajectories have different coverage in different joints. For example, as the top left sub-plot in Fig. 3.4 shows, the zig-zag trajectory has a sparsity in the direction of joints 2 and 3, but has no sparsity in the direction of joint 1. The bottom row of Fig. 3.4 shows different sparsity, smaller sparsity gives better coverage of the workspace but requires a longer time for the calibration.

The calibration zig-zag trajectories are defined as a set of sequential 3D vectors representing

points in the workspace of the positional joints of the robot. All trajectories are obtained by rotating and translating $\mathbb{T}_1$ – a zig-zag trajectory with the direction of joint 1, and are first defined on a normalized range $(-0.5, 0.5)$ and will be scaled according to each joint's limit. For example, the trajectory with the direction of joint 2 is obtained by

$$\mathbb{T}_2 = T\left(\begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}\right)R(J_3, 90^\circ)\mathbb{T}_1 \quad (3.1)$$

where $T()$ and $R()$ are operators of translation and rotation, respectively, J_3 means the axis of joint 3 in the joint space, the trajectory with the direction of joint 1 – $\mathbb{T}_1$ is a $4 \times N$ matrix with the direction of joint 1, in which the first 3 rows are positions of joint 1, 2, and 3, respectively. In our implementation, both translation and rotation are applied by homogeneous transformation matrices.

Similarly, trajectories with the direction of joint 1-2 and joint 1-2-3 are generated by the following equations, respectively.

$$\mathbb{T}_{1-2} = T\left(\begin{bmatrix} 0.5\sqrt{2} \\ 0.5\sqrt{2} \\ 0.5 \end{bmatrix}\right)R(J_3, 45^\circ)\mathbb{T}_1 \odot \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 1 \end{bmatrix} \quad (3.2)$$

$$\mathbb{T}_{1-2-3} = T\left(\begin{bmatrix} 0.5\sqrt{3} \\ 0.5\sqrt{3} \\ 0.5\sqrt{3} \end{bmatrix}\right)R(J_2, 45^\circ)R(J_1, 45^\circ)\mathbb{T}_1 \odot \begin{bmatrix} 1/\sqrt{3} \\ 1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix} \quad (3.3)$$

where $\odot$ indicates elemental-wise multiplication. After Equation (3.1)-(3.3), joint limit of all trajectories are within $(0, 1)$, though with different direction.

In this work, the following directions of zig-zag trajectories are used: **1) single direction:** Joint 1, Joint 2, and Joint 3; **2) double direction:** Joint 1-2, Joint 2-3, Joint 1-3; **3) triple direction:** Joint 1-2-3. Given:

$$c = 0.5(J_{max} + J_{min}) \quad (3.4)$$

$$r = J_{max} - J_{min} \quad (3.5)$$

where c and r are the center position and range of a joint, J_{max} and J_{min} are the maximum and minimum position of a joint. Then trajectories are translated and scaled so that a single direction has a range $(c - \frac{1}{2\sqrt{3}}r, c + \frac{1}{2\sqrt{3}}r)$, a double direction has a range $(c - \frac{\sqrt{2}}{2\sqrt{3}}r, c + \frac{\sqrt{2}}{2\sqrt{3}}r)$, and a triple direction has a range $(c - \frac{1}{2}r, c + \frac{1}{2}r)$. These different ranges result in all trajectories having the same center and covering the same volume in the normalized joint workspace.

3.1.4 Calibration Models

The calibration models based on DNN and regressions are shown in Fig. 3.5. The details about the hyperparameters of the DNN model can be found in 3.2.1. The input of the models is obtained from the robot state 'ravenstate' which is a ROS topic published at 1000 Hz in real time. Based on the ablation study in Chapter 4, the following features are used as input: **Current joint positions** are the built-in joint positions derived from motor positions, and the accuracy is degraded by the cable-driven mechanism. A typical arm of Raven-II has 7 joints. The positioning joints (joints 1-3) mainly determine the position of the end-effector, while the orientational joints (joints 4-7) mainly determine the orientation of the end-effector. **Motor torques** are derived from motor current control commands. Raven-II does not have any direct measurement of joint torques. The control commands are reflected in the desired joint positions, but this robot state is not used as input. This is because Raven-II has a 1000 Hz servo control frequency, which results in current and desired joint positions being very close. For example, a joint velocity of 10 deg/s only causes a 0.01 deg difference between the current and desired joint positions. More details about the robot states are in 4.1.1, and the coupling effects between joints in the data-driven calibration are studied in Chapter 4.

It is possible to include all features in the robot state as input for the calibration models. However, for linear and 2nd-order polynomial regression models, an increase in the input dimension can result in significantly more parameters. With limited training data, there can be a higher risk of overfitting, especially for 2nd-order polynomial regression. Choices of input features are studied in 3.3.4.

Instead of end-to-end calibration models that output calibrated joint positions, the output is the error of the original joint positions in the robot state. Then, the calibrated joint positions are obtained

by using the output error to correct the original inaccurate joint positions. Training on errors allows better performance for the DNN models, while it is optional for regression models (details in 3.3.6).

Besides learning and regression calibration, fixed offset compensation is also applied as a baseline method. This calibration uses the same training data and computes the average bias between the robot’s original joint positions and the ground truth joint positions. Then each joint position is compensated with the average bias when inferred.

3.2 Experiments

3.2.1 Experiment Setup and Model Parameters

The proposed methods were implemented and evaluated on a Raven-II surgical robot, which is fully cable-driven. Only the left arm of the robot was used. To collect reliable ground truth of joint positions, Avago Technologies AEDA-3300 encoders were installed on the rotational joints 1 and 2, with a resolution of 80000 PPR. Mercury II 1600 was installed on the translational joint 3, with a resolution of 5 μm , as shown in Fig. 3.6. As described in 3.1.1, the external encoders were registered during the initialization of Raven-II, in which the limit of each joint was explored and reached. In this work, temporary joint encoders were used for the best accuracy and reliability of the ground truth. The Raven-II robot has threaded holes on the arm links to support easy installation. Plug-in and use hardware and software drivers for encoders are also open-sourced (Appendix B). For other robot arms that do not support easy installation of joint encoders, for better feasibility, visual tracking might be preferred [41, 42, 43].

The hyperparameters of the DNN model were chosen empirically and are inspired by our previous work [43, 113] and the state-of-the-art work [28]. Unless stated otherwise, the DNN model for calibration in this work has 2 fully connected hidden layers with 100 units each. The sigmoid function was used as activation for all hidden units. Mean squared error was used as the loss function. L2 kernel regularization was applied with a factor of 0.0005. The DNN model was trained by 200 epochs with a learning rate of 0.001 and batch size of 1024. Adam [114] was used as the optimizer, and the exponential decay rates of the 1st and 2nd order moment estimates were 0.9 and 0.999, respectively.

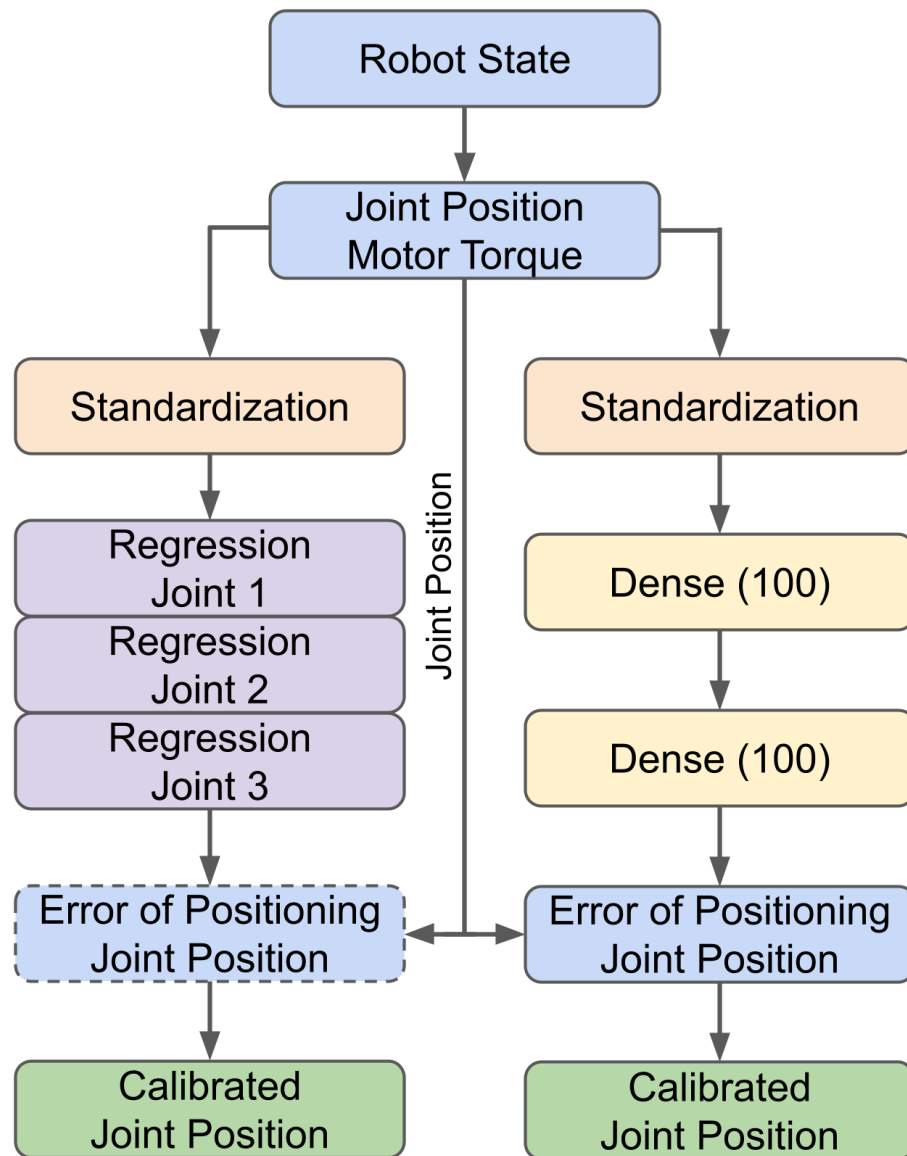


Figure 3.5: Model of regression-based calibration (left) and learning-based calibration (right). The outputs of the DNN model and regression model are the errors of the joint positions, these errors are added back to Raven-II’s original inaccurate joint positions to obtain the calibrated joint positions. The output of the regression models can be either the same or end-to-end output of joint positions (details in 3.3.6).

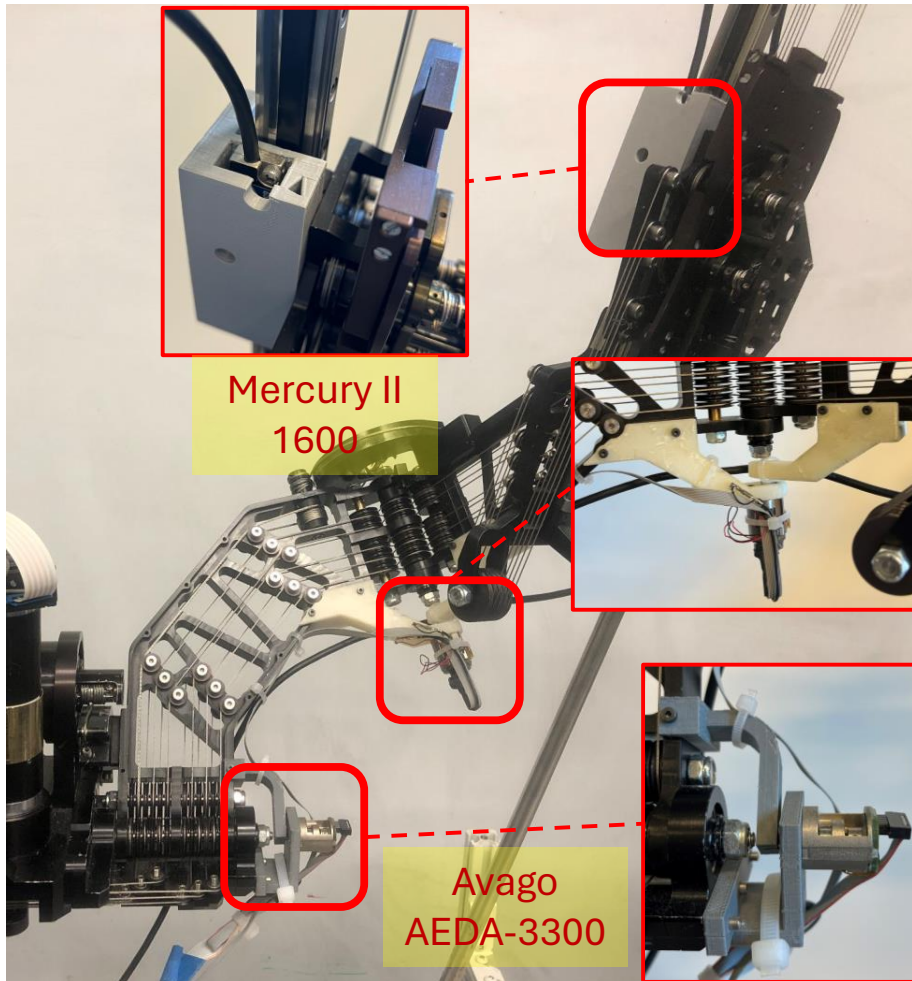


Figure 3.6: The external joint encoders that are used to collect ground truth. Raven-II has threaded holes for easy installations of joint encoders. After training, they can be removed before operation.

3.2.2 Datasets

Data was collected as ROS bags (details in 3.1.1) and then processed to form training and test datasets. Details about the experiment setup can be found in 3.2.1, The external joint positions were recorded under 100Hz, while the robot state 'ravenstate' was recorded under 30Hz due to high dimensions. Synchronization was performed when processing ROS bags. In the experiments, different training trajectories with different directions and sparsity were recorded and the details are included in the following sections, together with experiment results.

All testing trajectories used in this work are random sinusoidal trajectories in which each cycle has a different velocity and target position (details in 3.1.1). Compared with predefined zig-zag training trajectories, random testing trajectories have the smallest risk of similar patterns with the training trajectories so that the result evaluation can suggest better generalization ability with less risk of overfitting. All testing random trajectories were at least 20 minutes long in all experiments reported in this work to be representative to the workspace of the Raven-II robot. And the comparisons between different calibration methods and baseline results were from the same random trajectories that could ensure fairness.

3.3 Results

3.3.1 Directions of Calibration Trajectories

As described in 3.1.3 and Fig. 3.4, different directions of calibration trajectories cover the joints with different sparsity. Thus, the performance of the calibration was evaluated using trajectories with 7 different directions. For each direction, 3 calibration zig-zag trajectories with sparsity of $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$ were collected, with an independent 20-minute random movement trajectory as the test trajectory. All calibrations were performed using the combination of the 3 zig-zag trajectories. Evaluation of the DNN model was the average 5 repetitive training and evaluations with different global random seeds, which applied to all evaluations of DNN models in this work.

The calibration accuracy (RMSE) of different models trained by zig-zag trajectories with different directions is shown in Table 3.1. For joint 1, fixed offset compensation reduced the raw error from $\sim 2^\circ$ to $\sim 0.9^\circ$. The best performance of DNN calibration was achieved by the direction J2-J3 of the training trajectory, with an accuracy of 0.104° . Compared to fixed offset compensa-

tion, DNN calibration further reduced 87.9% of the error. The linear regression model also had the best accuracy of $0.174^\circ (-79.8\%)$ trained with the same trajectory direction. 2nd-order polynomial regression model achieved its second-best accuracy of $0.118^\circ (-86.3\%)$, while the best accuracy $0.114^\circ (-87.7\%)$ was trained by direction J1-J3 for this model.

For joint 2, the fixed offset compensation reduced the raw error from $\sim 8.0^\circ$ to $\sim 1.1^\circ$. The best accuracy $0.120^\circ (-89.1\%)$ was achieved by the DNN model trained with direction J2-J3. When trained with direction J1-J2-J3, the 2-nd polynomial regression and linear regression models saw their best accuracy $0.148^\circ (-85.8\%)$ and $0.169^\circ (-83.8\%)$, respectively.

Similarly, for joint 3, the fixed offset compensation reduced the raw error from ~ 11.7 mm to ~ 0.5 mm. The best performance of all DNN, 2nd-order polynomial regression, and linear regression was obtained by direction J3, with the accuracy of $0.111(-77.6\%)$ mm, $0.141(-71.7\%)$ mm, and $0.134(-73.1\%)$ mm, respectively. The second-best performance of all models was found with direction J2-J3, with accuracy DNN $-0.118(-76.0\%)$ mm, 2nd-order polynomial regression $-0.173(-64.9\%)$ mm, and linear regression $-0.166(-66.3\%)$ mm.

Overall, the direction of J2-J3 showed the best overall performance in all 3 joints among all 3 calibration models, reducing more than 64.9% of error, and thus was chosen for all the rest of the experiments.

3.3.2 Sparsity and Calibration Time

As shown in Fig. 3.4, trajectories with smaller sparsities have better coverage of the robot workspace, but also require more time to collect data in the learning calibration. This experiment studied the calibration performance of the DNN and regression models trained by different trajectories, with sparsities $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, $\frac{1}{5}$, and $\frac{1}{6}$. Combinations of different sparsities shorter than 1013 seconds were also evaluated. To rule out the time effect (details in 3.3.3), independent 20-minute random trajectories were collected for each sparsity as test sets. The result is shown in Fig. 3.7. For sparsities $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, 5 independent calibration trajectories were recorded, and for sparsity $\frac{1}{5}$ and $\frac{1}{6}$, 3 independent calibration trajectories were recorded to prevent long-time running.

For joint 1, DNN and linear regression models showed considerable improvement in all sparsities. When sparsity was $\frac{1}{2}$, it only took 209 seconds to record the calibration trajectory. DNN cali-

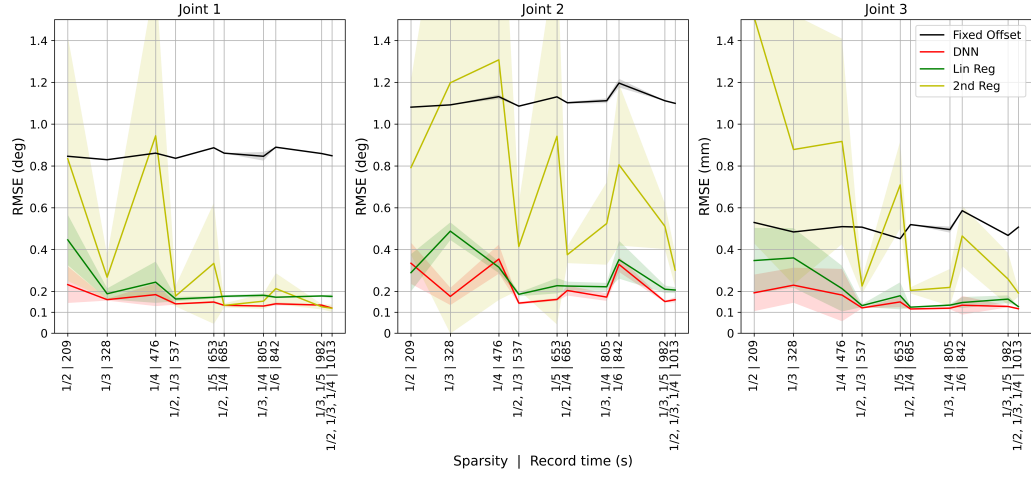


Figure 3.7: Calibration performance with different sparsities of the calibration trajectory. For sparsity 1/2, 1/3, 1/4, 5 independent calibration trajectories were recorded, and for sparsity 1/5, 1/6, 3 independent were recorded to prevent long-time operating. Combinations of trajectories with different sparsities, no longer than 1013 seconds to record, were also evaluated. Average RMSE is shown in solid lines and the standard deviations are shown in shades.

bration achieved RMSE of 0.232° . Compared to the RMSE of fixed offset compensation – 0.846° , the linear regression model had RMSE of 0.446° , while 2nd-order polynomial regression showed no significant improvement – 0.834° . As sparsity got smaller, the performance of all learning and regression methods was improved in both average RMSE and standard deviation. At the longest tested trajectory (1013 seconds to record), the combination of sparsities $\frac{1}{2}$, $\frac{1}{3}$ and $\frac{1}{4}$, all 3 models showed significant improvement in accuracy, DNN – 0.120° , 2nd-order polynomial regression – 0.119° , and linear regression – 0.176° . Compared to the shortest trajectory with sparsity $\frac{1}{2}$, the longest combined trajectory had improvement in accuracy of 0.112° , 0.727° , and 0.270° in DNN, 2nd-order polynomial, and linear regression, respectively. The performance of fixed offset compensation did not see improvement with smaller sparsity. Standard deviation also got reduced with sparsity in general, while a significantly larger standard deviation could be observed in 2nd-order polynomial regression when shorter calibration trajectories were used.

For joint 2, when trained by trajectories shorter than 500 seconds, the DNN model and linear regression model had an average accuracy of 0.288° and 0.364° , respectively. On the other hand,

when trained by trajectories longer than 800 seconds, the average calibration RMSE reduced to 0.203° (DNN) and 0.248° (linear regression). The performance of 2nd-order polynomial regression was very unstable with large standard deviations among most of the sparsities and only achieved comparable accuracy 0.302° when trained by the longest trajectory, though still did not outperform the other 2 models.

For joint 3, similarly, when trained by trajectories shorter than 500 seconds, the DNN model and linear regression model had an average accuracy of 0.202 mm and 0.307 mm, respectively. As the calibration trajectory increased to 537 seconds, the combination of sparsities $\frac{1}{2}$ and $\frac{1}{3}$, the accuracy was improved to DNN – 0.121 mm and linear regression – 0.132 mm. No considerable further improvement could be found with longer calibration trajectories. Although the 2nd-order regression model had reduced RMSE with longer calibration trajectories, the accuracy was overall unstable and did not outperform the other 2 models.

3.3.3 Long-time Idleness, Unloaded and Loaded Operation

The previous experiments 3.3.1 and 3.3.2 showed that data-driven calibrations significantly reduced the error of the joint positions. However, the feasibility of the calibrations also depends on how the calibration accuracy drops with operating time. Due to the relatively small sheave and cable diameter ratio, high tension, large wrap angle, etc., cables of Raven-II have a short fatigue life and cable characteristics can change over time [44]. Generally, the average operating time of laparoscopic RMAIS is less than 4 hours [115, 116, 117]. To study the influence of operating time on the scale of real surgeries, calibration trajectories in the direction of J2-J3 with different sparsities were recorded first. The robot was then continuously operated by random trajectories for 6 hours to collect the test set. Unloaded and loaded (500g weight scale hanging on the end-effector) configurations were tested, as well as unloaded idleness for comparison. To the best of the authors' knowledge, this is the current longest test trajectory set on data-driven calibration of laparoscopic surgical robots. The result is shown in Fig. 3.8. The same sparsities were used as in 3.3.2 and shown in the box plot. The arrows in the box plot indicate training trajectories longer than 800 sec had $>20\%$ better calibration accuracy than trajectories shorter than 500 sec, which suggests considerably better performance can be obtained by longer calibration trajectories. As shown in

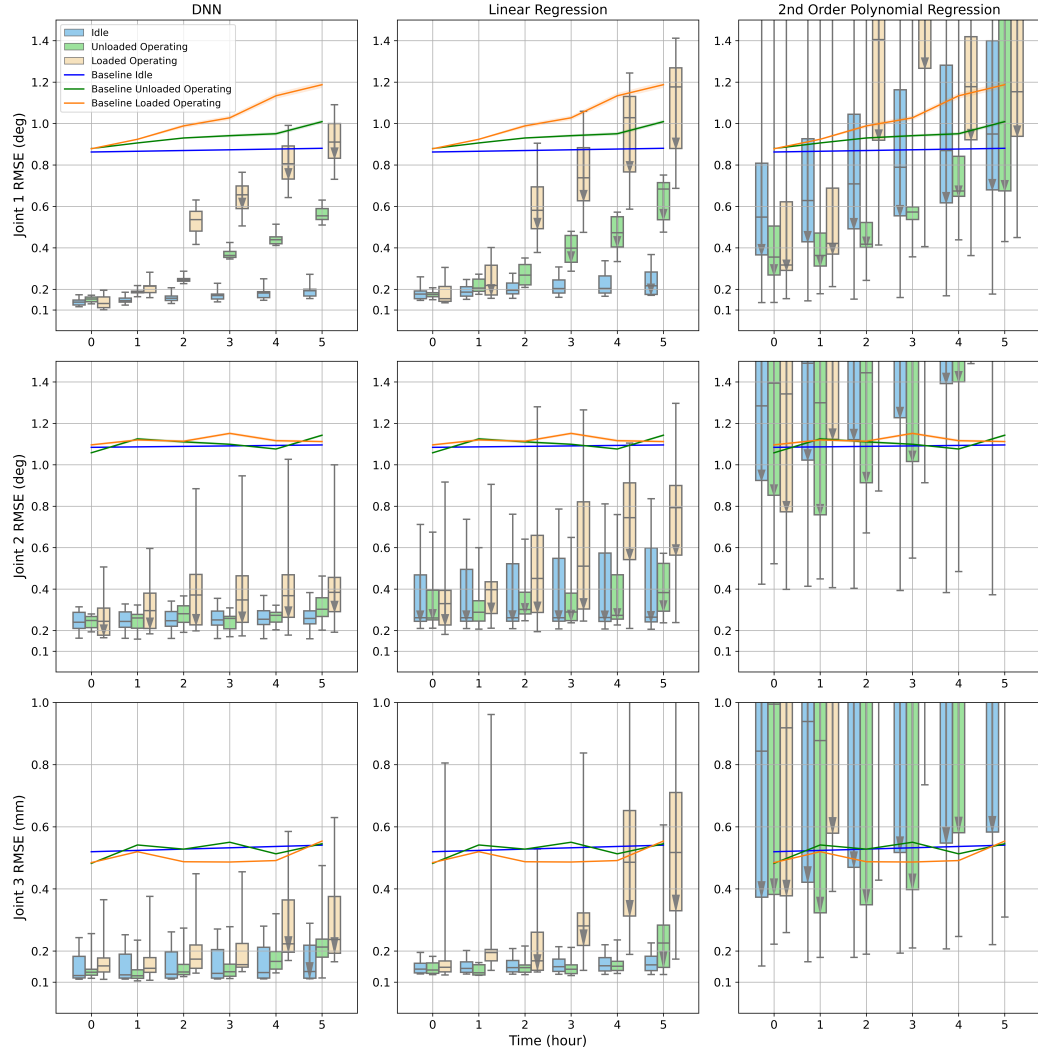


Figure 3.8: Calibration performance during 6-hour continuous unloaded operating, 500g loaded operating, and idleness. The same sparsities were used as in 3.3.2 and shown in the box plot. The baseline results were obtained by fixed offset compensation. Accuracy of idleness was only evaluated by test trajectories in hours 0 and 5, and hours 1-4 were interpolated. The arrows in the box plot indicate training trajectories longer than 800 sec had $>20\%$ better calibration accuracy than trajectories shorter than 500 sec, and the difference between the 1st and 3rd quartile is larger than 0.2 (deg/mm).

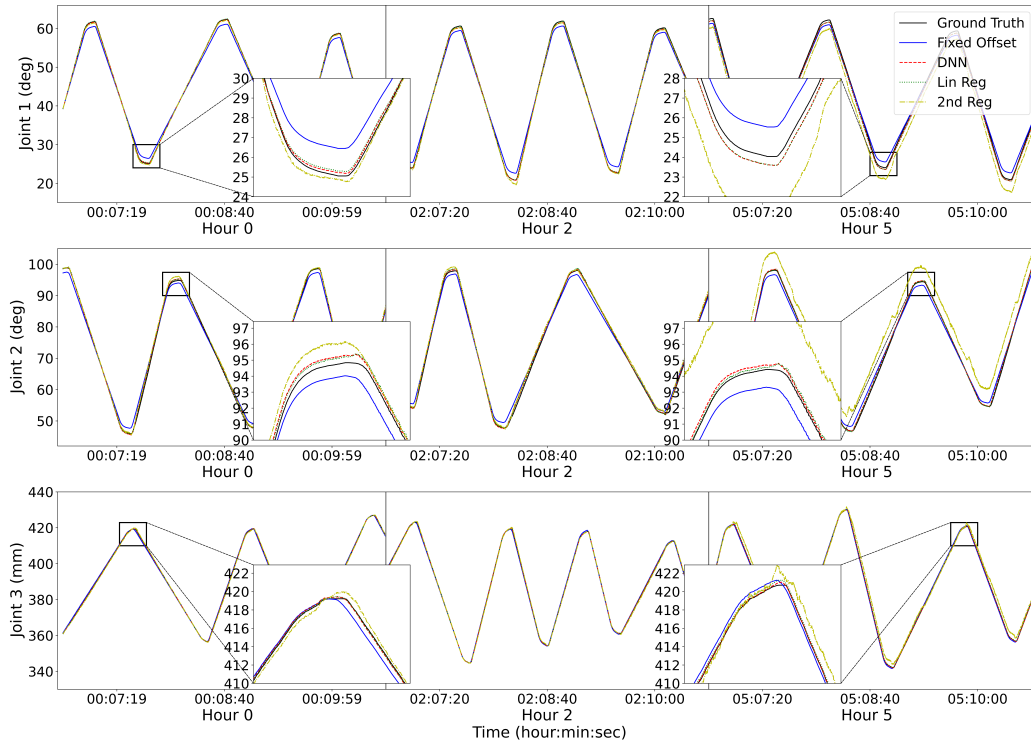


Figure 3.9: Example of calibrated joint position trajectories. The models were trained by the calibration trajectory with sparsity $\frac{1}{6}$. 3 sections are shown horizontally. The first section is the immediate performance right after calibration. The second and third sections show performance in hour 2 and 5, respectively.

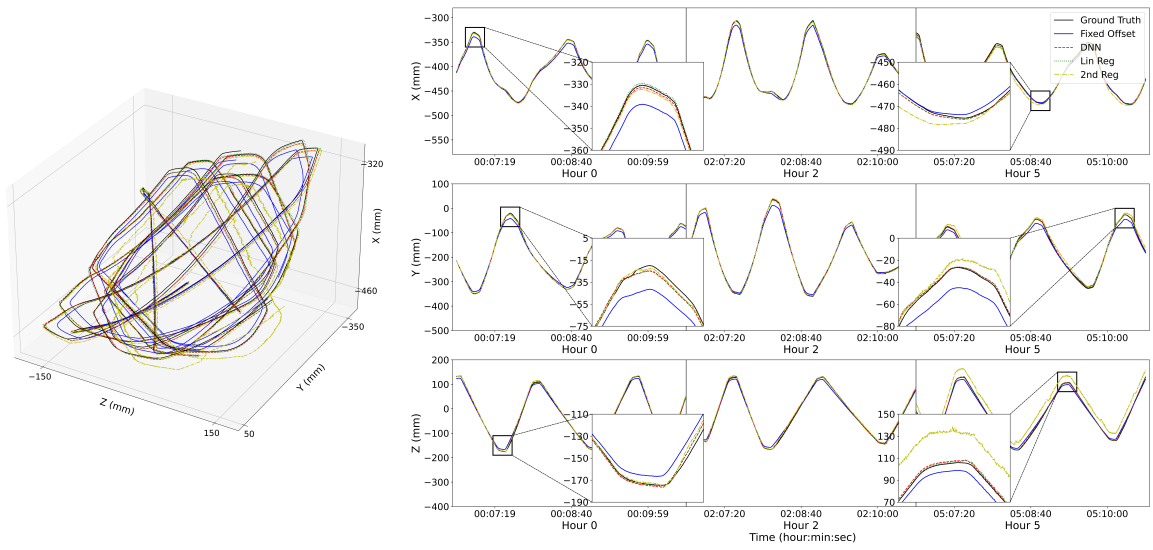


Figure 3.10: The example of the calibrated end-effector position trajectories corresponding to Fig. 3.9, represented in Frame 0 of Raven-II. Frame 0 is located at the remote motion center (described in Fig. 3.2, upper left) of the robot arm. It is worth noting that in the front view, the X-axis is pointing vertically up and the Z-axis is pointing right, determined the joint 1 axis and its positive direction of rotation. Only around 10-minute testing trajectories are shown for clear visualization, the entire testing trajectories were recorded throughout 6-hour continuous operating.

Fig. 3.8, longer calibration trajectories achieved better performance in many entries, especially for regression models. The RMSE described in this section is the average RMSE of models trained by the 3 longest trajectories, unless stated otherwise.

For joint 1, for idleness of 6 hours, the baseline result – fixed offset compensation had no obvious decrease in accuracy from hour 0 – 0.863° to hour 5 – 0.879° . The DNN model saw a small drop of 0.048° in accuracy from hour 0 – 0.124° to hour 5 – 0.172° . A similar trend was observed in the linear regression model, from 0.162° to 0.193° . However, the 2nd-order polynomial regression model saw a significant decrease in accuracy, from 0.241° to 0.493° even when the robot stayed idle for 6 hours. When the robot was operated for 6 hours without load, a considerable drop in accuracy could be found in all 3 calibration models. Compared to fixed offset compensation that dropped 0.121° in 6 hours (from 0.879° to 1.000°), the accuracy of the DNN model dropped 0.411° (from 0.142° to 0.553°), the linear regression model dropped 0.358° (from 0.169° to 0.527°), and the 2nd-order polynomial regression model dropped 1.294° (from 0.235° to 1.529°). It is also worth noticing that after hour 3, longer trajectories could result in better performance in the linear regression model, while the DNN model is less affected by the amount of training data. When the robot was operated for 6 hours with a 500g load on the end-effector, a larger decay in accuracy could be found in all models. The fixed offset compensation had accuracy dropped 0.296° (from 0.878° to 1.174°), the DNN model dropped 0.741° (from 0.114° to 0.855°), the linear regression model dropped 0.772° (from 0.144° to 0.916°), and the 2nd-order polynomial regression model dropped 0.824° (from 0.221° to 1.045°).

For joint 2, with 6-hour idleness, no decay in accuracy could be found in fixed offset compensation – from 1.084° to 1.094° , DNN model – from 0.211° to 0.217° , and linear regression model – from 0.237° to 0.232° . The performance 2nd-order polynomial regression model was much worse and unstable than the other calibration models in all configurations and thus will not be compared. When operated for 6 hours without load, the drops in accuracy were: fixed offset compensation – 0.083° (from 1.056° to 1.139°), DNN – 0.099° (from 0.232° to 0.331°), linear regression – 0.106° (from 0.247° to 0.353°). When operated for 6 hours with 500g load, the drop in accuracy was: fixed offset compensation – 0.017° (from 1.092° to 1.109°), DNN – 0.181° (from 0.251° to 0.432°), linear regression – 0.322° (from 0.266° to 0.588°). It could be found that although the fixed offset compensation did not see a big difference, a larger drop in accuracy was seen by the DNN and lin-

ear regression models. Overall, in the configurations of idleness and unloaded operation, the DNN model achieved desirable performance in all calibration trajectories, while longer trajectories were preferred by the linear regression model to achieve comparable accuracy. In the loaded configuration, longer calibration trajectories were preferred by both models.

For joint 3, with 6-hour idleness, no decay in accuracy could be found in fixed offset compensation – from 0.520 mm to 0.541 mm, DNN model – from 0.113 mm to 0.113 mm, and linear regression model – from 0.131 mm to 0.132 mm. The performance 2nd-order polynomial regression model was again excluded for comparison because of undesirable performance. When operated for 6 hours without load, the drop in accuracy was: fixed offset compensation – 0.063 mm (from 0.482 mm to 0.545 mm), DNN – 0.039 mm (from 0.126 mm to 0.165 mm), linear regression – 0.068 mm (from 0.132 mm to 0.200 mm). When operated for 6 hours with a 500g load, the drop in accuracy was: fixed offset compensation – 0.069 mm (from 0.485 mm to 0.554 mm), DNN – 0.059 mm (from 0.122 mm to 0.181 mm), linear regression – 0.335 mm (from 0.138 mm to 0.473 mm). Both DNN and linear regression models showed desirable accuracy with all calibration trajectories in the configuration of idleness and unloaded operation. However, for loaded operation, longer calibration trajectories were necessary to obtain better accuracy.

Overall, with sufficient training data, the calibration models improved the accuracy significantly among 6-hour idleness, unloaded, and loaded operating. Examples of calibrated joint position trajectories in the test set can be found in Fig. 3.9, and the corresponding end-effector trajectories in Cartesian space can be found in Fig. 3.10.

3.3.4 Robustness of Unnecessary Input Features

As described in 3.1.4, there are numerous types of information in the robot state (details in 4.1.1), including the robot run level, joint position, velocity, motor torque, end-effector pose, and so on. Among these types of information, based on the ablation study in Chapter 4, only joint positions and motor torques were found important to the calibration. However, the ablation study also took a considerable effort. Without the ablation study, in order to not miss critical input features, all information in the robot state should be given as input to the calibration models, which will increase the parameters of the models. Under the same amount of training data, more parameters may result

in a higher risk of overfitting. Thus, this experiment utilized the same 6-hour unloaded operation data in 3.3.3. The DNN, linear regression, and 2nd-order regression models were trained using only selected useful features (joint positions and motor torques, dimension of 16), and using all features in the robot state (dimension of 138). Besides the original DNN model (2 layers, 100 hidden units each), a larger DNN model was also used to accommodate more input dimensions, which has 3 layers with 600, 500, and 400 units, respectively. Regularization rates were also applied, kernel L1 10^{-5} L2 10^{-4} , bias L2 10^{-4} , and activity L2 10^{-5} . The other hyperparameters stayed unchanged. GPU acceleration was utilized for the training of the larger DNN model.

The evaluation result is shown in Fig. 3.11. When the model input was only joint positions and motor torques, the DNN model had desirable average accuracy in 6-hour unloaded operating among all calibration trajectories from short to long, with the RMSE of joint 1 – 0.328° , joint 2 – 0.264° and joint 3 – 0.167 mm. When all features in the robot state were used as input, the input dimension was 8.6 times larger from 16 to 138, but the number of model weight parameters was only 2 times larger from 12103 to 24303. A considerable drop in accuracy could be observed in all 3 joints. The average RMSE when trained by the 3 longest calibration trajectories was joint 1 – 0.700° , joint 2 – 0.754° , and joint 3 – 0.388 mm. When the 3-layer larger model was used with all features as input, the weight parameters increased dramatically to 585503. The average RMSE when trained by 3 longest calibration trajectories improved to joint 1 – 0.397° , joint 2 – 0.565° , and joint 3 – 0.229 mm, though the performance was still not as good as the small model with selected features.

The linear regression model had 17 parameters for each joint when trained by selected features, and had desirable average accuracy in 6-hour unloaded operating among all calibration trajectories, with the RMSE of joint 1 – 0.361° , joint 2 – 0.355° and joint 3 – 0.167 mm. When all 138 features in the robot state were used as input, the number of parameters also increased linearly to 139. A significant drop in accuracy could be observed, and the average RMSE trained by the 3 longest calibration trajectories was joint 1 – 1.985° , joint 2 – 1.834° , and joint 3 – 1.352 mm, which did not outperform fixed offset compensation. A similar trend could also be observed by the 2nd-order regression model, but the performance was worse than linear regression.

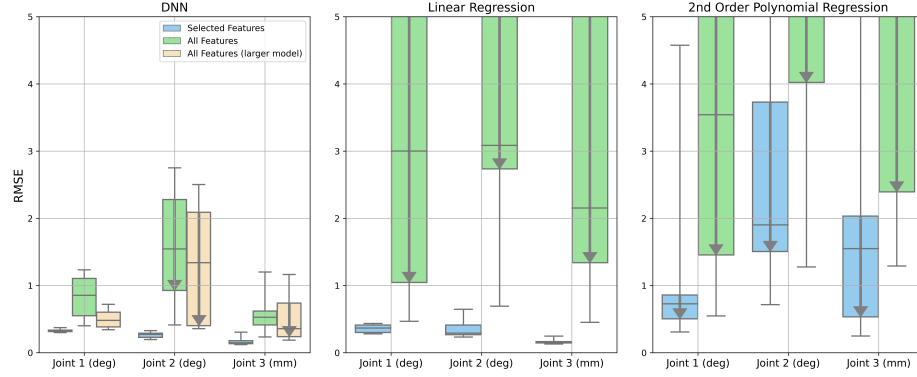


Figure 3.11: Calibration performance of models trained by selected important features (joint positions and motor torques), and by all features (details in 4.1.1) in the robot state. Besides the original DNN model, a larger DNN model was also used to accommodate more input dimensions. The box plot shows the average RMSE among 6-hour unloaded operating with different calibration trajectory sparsities, as in Fig. 3.8. The arrows in the box plot indicate training trajectories longer than 800 sec had $>20\%$ better calibration accuracy than trajectories shorter than 500 sec, and the difference between the 1st and 2nd quartile is larger than 0.2 (deg/mm).

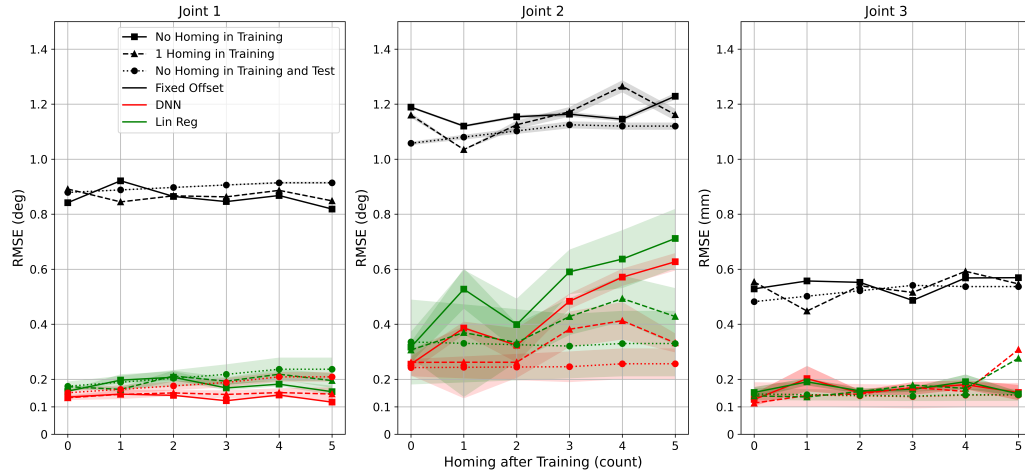


Figure 3.12: Calibration performance after 0-5 robot homing procedures. Different combinations (<20 min) of training trajectories with various sparsities were tested and the average is shown with standard deviation as shade. The 2nd order polynomial regression model is not shown because of unstable performance. Two datasets were used for training, one had the training set with no homing, and another one had the training set with 1 homing. To study the effectiveness of homing, baseline results with the same operating time but no homing in both training and test sets are also included.

3.3.5 *Effects of Homing*

The homing procedure is an important function of the Raven-II robot, during which the robot explores the hard limits of all the joints and registers the encoders on the motors. Joint positions will be meaningful after the homing procedure. Re-homing may sometimes be necessary during operations, for example, the surgical instrument is changed or software e-stop is triggered because of joint limits or singularity. Thus, it is important to study whether the homing procedure influences the calibration's performance. Two new datasets were collected to study the effectiveness of homing. In each dataset, 5 homings were performed and a 20-minute random trajectory was recorded as test data before the first homing, in between the homings, and after the last homing. The difference between the two datasets was that one dataset had all training calibration trajectories before the first homing, thus this training set 'saw' no homing. The other dataset had the same sparsities and length of the training trajectories, but these trajectories were separated before and after the first homing, thus this training set 'saw' 1 homing.

The evaluation of calibration performance among homings is shown in Fig. 3.12. For joint 1, the accuracy of the DNN model slightly fluctuated between 0.117° and 0.146° among 5 homings with no homing 'seen' in training. With 1 homing 'seen' in training, the accuracy of the DNN model slightly fluctuated between 0.133° and 0.152° . No obvious drop in accuracy could be observed caused by homing. A similar trend was found using the linear regression model, fluctuating between 0.155° and 0.207° with no homing in training, and fluctuating between 0.163° and 0.218° with 1 homing in training.

For joint 2, when the calibration models 'saw' no homing in training, the accuracy of the DNN model dropped significantly from 0.255° to 0.627° after 5 homings, and similarly, the accuracy of the linear regression dropped from 0.317° to 0.712° . On the other hand, if the models 'saw' 1 homing in training, although the accuracy still dropped with homing, the RMSE stayed within 0.413° for the DNN model and within 0.493° among 5 homings.

For joint 3, except for an outlier at the 5th homing, the DNN model and the linear regression model had close accuracy regardless of 'seeing' homing in training or not, slightly fluctuating from 0.113 mm to 0.202 mm.

3.3.6 Training on Error

As introduced in 3.1.4 and Fig. 3.5, the DNN and regression models were not trained as end-to-end models that directly outputted the calibrated joint positions. Instead, the models outputted the errors of the inaccurate original joint positions in the robot state. The errors were then used to correct the original joint positions and thus provided calibrated accurate joint positions. To study the necessity of training on error, this experiment used the same 6-hour unloaded data as 3.3.3, and the models were trained on error or end-to-end.

The comparison of calibration accuracy is shown in Table. 3.2. For the regression models, no difference in RMSE could be found between models trained on error or end-to-end, though the average performance of 2nd-order polynomial regression was not desirable. On the other hand, for the DNN model, when trained on error, the best accuracy was achieved, joint 1 – 0.328° , joint 2 – 0.264° , and joint 3 – 0.167 mm. However, when trained end-to-end, the calibration error increased sharply to 0.830° , 0.875° , and 1.583 mm in joints 1, 2, and 3, respectively. Further tests found that increasing the number of training epochs could improve the performance of the end-to-end DNN model, however, the accuracy was still not as good as the DNN model trained on error. Since training on error requires no extra effort and can be faster with fewer epochs, it is highly preferred for efficient calibration.

3.3.7 Comparison to the State-of-the-art in 2024

The comparison of calibration performance with the state-of-the-art in 2024 in the joint space is shown in Table 3.3, and the comparison in Cartesian space is shown in Table. 3.4. The requirement of accuracy in RAMIS varies with different types of surgeries and instruments. As examples reported in [118, 119], 2 mm positional accuracy is ideal for laparoscopic RAMIS. When unloaded, both the proposed DNN and linear regression calibration had positional accuracy within 2 mm throughout 6 hours of continuous operating. When heavily loaded by 500 g, the DNN calibration still had accuracy within 2 mm while the accuracy of linear regression was slightly larger than 2 mm at hour 5. The proposed method in this work performs calibration in the first 3 joints. Thus, the location accuracy in Cartesian space is obtained by assuming the last 4 joints are accurate and the surgical tool has no deformation. The last 4 joints of Raven-II are concentrated near the end-effector

with only one non-zero link length of 13 mm. Thus, the inaccuracy of the last 4 joints mainly results in errors in the orientation of the end-effector but less in position. However, a certain error increment still exists in real practice because of deformation.

Compared with the state-of-the-art works in 2024, we also focus on efficiency while achieving state-of-the-art accuracy on Raven-II. We trained the calibration models within 21 minutes and evaluated the performance of calibration in 6-hour unloaded and loaded operating, and among robot re-homings, in order to prove the feasibility of applying data-driven calibration in the time scale of real surgeries. Besides using computationally expensive DNN models that have slower inference time, by excluding unnecessary input features [113] and providing sufficient training data, we also show that the linear regression model can have a significantly faster inference speed that can be applied to the 1000 Hz servo control of Raven-II, with slightly compromised accuracy.

3.4 Discussion

The DNN model and linear regression model showed comparable performance in the experiments reported in this work, which raises an interesting question: is the linear regression model competitive to the DNN model? The comparison between these two models in different aspects is shown in Table. 3.5.

In terms of accuracy, with sufficient training data, i.e. long enough calibration trajectory, the DNN model slightly outperformed the linear regression model. According to 3.3.3, when trained by the 3 longest calibration trajectories, the performance of the 2 models was very close in unloaded operation. When loaded, the DNN model slightly outperformed the linear regression model by 0.015° , 0.109° , and 0.124 mm in joints 1, 2, and 3, respectively. In terms of data hunger, when trained with the 3 shortest calibration trajectories, counter-intuitively, the DNN model has considerably better performance than the linear regression model. When unloaded, the DNN model had 0.052° , 0.189° , and -0.037 mm better accuracy in joints 1, 2, and 3, respectively. When loaded, the difference increased to 0.128° , 0.360° , and 0.178 mm.

With the small size of the DNN model, even with the largest training set presented in this work, the DNN model could be trained within 50 seconds without GPU acceleration. The training time of the linear regression model was less than 2 seconds. Although the linear regression model could

be trained much more quickly, compared to the entire calibration procedure described in 3.1.2, the data collection, which takes 4-17 minutes, remains the dominating time consumption. In terms of real-time efficiency, the inference time of the DNN model without GPU was 61 ms (<20 Hz) on the Intel(R) Xeon(R) 2.2GHz CPU, which had a huge gap to meet the 1000 Hz servo control rate of the CRTK and Raven-II robot. However, since calibrated joint positions may not be necessary in the servo control loop of the robot, the inference time of the DNN model could meet the rate required by the CRTK's interpolate control level, which is designed for vision-based control, and so on. With GPU acceleration, the inference time could be shortened to 54 ms. The average inference time could also be significantly shortened if inferred in a mini-batch, however, this caused delay and could not be applied to real-time servo control. On the other hand, the inference time of the linear regression model was 0.38 ms (around 3500 Hz), which was sufficient for the 1000 Hz servo control rate of the robot.

As described in Section 3.3.4 and Fig. 3.11, in the calibration of the Raven-II robot in this work, useful features were found by the ablation study in Chapter 4, which required a certain effort. In other cases, if useful features are unknown, in order to not miss useful features, unnecessary features may also be selected as input of the model. Under this condition, a larger DNN model showed much better robustness with a slightly compromised performance, while the linear regression model had a significant drop in accuracy and became undesirable. In terms of the model output, regression models could be either trained end-to-end (directly output calibrated joint positions) or trained on error (output the error of the original inaccurate joint positions and made correction). In comparison, the DNN model had to be trained on error, to achieve better accuracy and faster convergence. However, since no considerable extra effort was needed to perform training on error, both in the calibration phase and the real-time inference phase, there was no difference in efficiency.

To sum up, both models have their advantages and disadvantages. Depending on the use case, one may be more desirable than the other. To achieve the best accuracy, or when data collection must be fast, or unnecessary features may be given as input, the DNN calibration has better accuracy and robustness in performance. However, when calibrated joint positions are required in high frequency by real-time applications, the linear regression calibration becomes the only choice. In this case, longer calibration trajectories are critical to maintain better accuracy.

The first 3 positioning joints of the Raven-II robot, though all being cable-driven, have consid-

erable differences in mechanisms (Fig. 3.3) and calibration performance described from 3.3.1 to 3.3.6.

Joint 1 is a single alpha-wrapped pulley joint, which has the shortest cable length and little or no free cable length. As the first joint, it also bears the largest load caused by the mass of the following robot arm links. As for calibration performance, joint 1 had the largest drop of calibration accuracy with time and load. And it was the only joint that had increased error with time and load in fixed offset compensation.

The joint 2 is a general rotational joint. Although the instantaneous accuracy right after calibration was not as good as that of joint 1, the accuracy of joint 2 dropped less with operating time and load. However, joint 2 saw a considerable increase in error with the robot homing procedure, while the other 2 joints did not.

The joint 3 is the only translational joint of the robot. To insert and extract the surgical instrument, joint 3 has a large range. However, during normal operating, only the lower half of the range is used. Approaching the higher half causes disabled inverse kinematics due to singularity. According to the feature ablation study in Chapter 4, joint 3 was the only joint in which joint torques were more important than joint positions as input features of the calibration model. Although the DNN and linear regression models had very desirable calibration accuracy, the percentage improvement compared to fixed offset compensation was the lowest among all 3 positioning joints.

Although the zig-zag training trajectories have desirable even positional coverage of the robot workspace, the variety of velocity and its direction is not as desirable. Although calibration models trained by zig-zag trajectories had state-of-the-art accuracy and efficiency, in the future, we aim to develop a calibration trajectory with optimized coverage in high-dimensional coupled space that includes both position and velocity, which may further improve the accuracy and efficiency of the calibration.

3.5 Conclusion

This work presented an efficient data-driven joint-level calibration method for the cable-driven Raven-II surgical robot that substantially improves positioning accuracy without requiring additional sensors during deployment. By combining carefully designed calibration trajectories with

lightweight learning models, the method achieved strong accuracy within a short calibration time and remained effective over extended operation, including under loaded conditions. The results also showed that correcting joint-level errors is more efficient and robust than end-to-end calibration, and that while the DNN provides the best overall accuracy, linear regression offers a practical alternative for real-time high-frequency control. Overall, this work demonstrates a feasible and effective path toward improving the precision and reliability of cable-driven surgical robots for automation and advanced robot-assisted surgical tasks.

Table 3.1: Learning and Regression Calibration Error with Different Directions of Trajectories

Direction	DNN		2nd Reg		Lin Reg		Fixed Offset	Raw
	RMSE	Percentage	RMSE	Percentage	RMSE	Percentage	RMSE	RMSE
Joint 1 (deg)								
J1	0.215	25.0%	0.193	22.5%	0.214	25.0%	0.857	1.930
J2	0.186	20.4%	0.156	17.1%	0.214	23.5%	0.911	2.015
J3	0.115	13.1%	0.144	16.5%	0.198	22.6%	0.877	2.061
J1-J2	0.209	23.1%	0.169	18.6%	0.231	25.4%	0.908	2.095
J2-J3	0.104	12.1%	0.118	13.7%	0.174	20.2%	0.859	2.103
J1-J3	0.120	12.8%	0.114	12.3%	0.257	27.5%	0.932	2.125
J1-J2-J3	0.143	16.5%	0.145	16.7%	0.221	25.5%	0.866	2.219
Joint 2 (deg)								
J1	0.248	22.4%	0.311	28.1%	0.288	26.1%	1.106	7.927
J2	0.313	27.6%	0.242	21.3%	0.250	22.0%	1.135	8.016
J3	0.241	20.8%	0.203	17.5%	0.294	25.4%	1.160	8.115
J1-J2	0.234	20.1%	0.276	23.6%	0.221	19.0%	1.168	8.064
J2-J3	0.120	10.9%	0.222	20.3%	0.186	17.0%	1.095	7.963
J1-J3	0.265	22.7%	0.294	25.2%	0.266	22.8%	1.168	7.985
J1-J2-J3	0.130	12.6%	0.148	14.2%	0.169	16.2%	1.038	7.945
Joint 3 (mm)								
J1	0.179	36.8%	0.406	83.5%	0.239	49.2%	0.486	11.723
J2	0.216	34.4%	0.357	56.9%	0.257	40.9%	0.628	11.737
J3	0.111	22.4%	0.141	28.3%	0.134	26.9%	0.498	11.754
J1-J2	0.183	31.3%	0.416	71.3%	0.253	43.4%	0.584	11.773
J2-J3	0.118	24.0%	0.173	35.1%	0.166	33.7%	0.492	11.715
J1-J3	0.161	29.0%	0.178	31.9%	0.179	32.2%	0.557	11.778
J1-J2-J3	0.164	34.9%	0.274	58.5%	0.168	35.8%	0.469	11.703

* i) The best 2 directions of calibration trajectories for each joint and calibration model are marked bold.

ii) The percentage is computed based on the fixed offset compensation instead of raw errors.

Table 3.2: Calibration Performance of Models Trained on Joint Position Error or End-to-end

Model	Joint 1 (deg)		Joint 2 (deg)		Joint 3 (mm)	
	Err	E-to-E	Err	E-to-E	Err	E-to-E
DNN	0.328	0.831	0.264	0.875	0.167	1.583
Lin Reg	0.369	0.369	0.366	0.366	0.176	0.176
Poly Reg	1.062	1.062	2.539	2.539	1.766	1.766

* Each entry is the average RMSE of all sparsities among 6 hours.

Table 3.3: Direct Comparison to the State-of-the-art in Joint Space

Method	Robot	Model	Joint 1 (deg)	Joint 2 (deg)	Joint 3 (mm)
Ours, unloaded (hour 0–5)	Raven-II	DNN	0.142–0.553	0.230–0.343	0.125–0.176
Ours, unloaded (hour 0–5)	Raven-II	Lin Reg	0.169–0.522	0.248–0.376	0.133–0.206
Ours, loaded (hour 0–5)	Raven-II	DNN	0.114–0.855	0.251–0.432	0.122–0.181
Ours, loaded (hour 0–5)	Raven-II	Lin Reg	0.144–0.916	0.266–0.588	0.138–0.473
Haghighipناه et al. [38]	Raven-II	UKF	1.195	1.242	0.927
Hwang et al. [32] (left arm)	dVRK	RNN	0.115	0.069	0.150
Hwang et al. [32] (right arm)	dVRK	RNN	0.172	0.206	0.510

* Performances of unloaded configuration is presented in this table.

Table 3.4: Indirect Comparison to the State-of-the-art in Cartesian Space

Method	Robot	Model	X (mm)	Y (mm)	Z (mm)	Dist (mm)
Ours, unloaded* (hour 0–5, est.)	Raven-II	DNN	0.362–0.538	0.301–1.503	0.205–0.581	0.513–1.318
Ours, unloaded* (hour 0–5, est.)	Raven-II	Lin Reg	0.389–0.589	0.351–0.998	0.231–0.564	0.572–1.289
Ours, loaded* (hour 0–5, est.)	Raven-II	DNN	0.393–0.677	0.262–1.616	0.190–0.867	0.509–1.955
Ours, loaded* (hour 0–5, est.)	Raven-II	Lin Reg	0.417–0.928	0.313–1.762	0.221–1.034	0.566–2.244
Hwang et al. [32] (pick)	dVRK	RNN	–	–	–	0.65
Hwang et al. [32] (random)	dVRK	RNN	–	–	–	2.05
Mahler et al. [42]	Raven-II	GPR	1.4	1.4	1.1	–
Peng et al. [43]	Raven-II	DNN	0.818	1.078	0.518	–

* The performance of our proposed method was estimated by assuming the last 4 joints were accurate and there was no deformation of the surgical tool.

Table 3.5: Deep Neural Network VS. Linear Regression

	DNN	Linear Regression
Accuracy	Slightly better with sufficient data	Slightly worse with sufficient data
Data Hunger	Better with small training set	Worse with small training set
Training Time	<50 sec, no GPU	<2 sec
Inference Time	61 ms, enough for CRTK interpolate level	0.38 ms, enough for CRTK servo level
Irrelevant features	Robust with larger model and training set	Not robust
Train on Error	Required	Optional

Chapter 4

FEATURE ABLATION STUDY OF DATA-DRIVEN CALIBRATION

As shown in Chapter 3, excluding unnecessary and reducing the dimension of input of calibration models is critical for both efficiency and accuracy of the calibration models, especially for regression models. This work uses removal and inaccurate ablation studies on input features and identify that raw joint positions and motor torques are the most important model inputs for calibration accuracy. These studies also reveal that the models are capable of inferring joint positions from the end-effector pose and prioritize the direction of motor torques over their amplitude. When guided appropriately, the models can also compensate for encoder value inconsistencies occurring with robot re-homings. By excluding the unnecessary input features, lightweight models are developed and achieve better performance and efficiency simultaneously, reducing the training time on the CPU to 2.5 minutes.

4.1 Method*4.1.1 Features in the Robot State*

Table 4.1: Features in Raven-II State

Feature	Dimension	Description
Time Stamp	1	Time stamp of the ROS message, converted to seconds.
Run Level	1	An integer indicating the major run level of the robot, including homing, pedal-up mode, pedal-down mode, and E-stop.

Continued on next page

Table 4.1: Features in Raven-II State (continued)

Feature	Dimension	Description
Sub-Level	1	An integer indicating subordinate run levels, such as the motor torque test.
Last Sequence	1	Sequence number of teleoperation commands.
Arm Type	1	An integer indicating the left or right arm.
End-Effector Position	3	Cartesian position (x, y, z) of the end effector, derived from joint positions using forward kinematics.
End-Effector Position Desired	3	Desired Cartesian position (x, y, z) of the end effector, derived from the control command.
End-Effector Orientation	9	3×3 rotation matrix of the end-effector orientation, derived from joint positions using forward kinematics.
End-Effector Orientation Desired	9	3×3 rotation matrix of the desired end-effector orientation, derived from the control command.
Encoder Value	7	Encoder values of the motors of the seven joints.
Encoder Offset	7	Encoder offsets of the motors of the seven joints, calibrated during homing and then kept constant.
Motor Position	7	Motor positions of the seven joints, derived from encoder values and encoder offsets.

Continued on next page

Table 4.1: Features in Raven-II State (continued)

Feature	Dimension	Description
Motor Position Desired	7	Desired motor positions of the seven joints, derived from desired joint positions.
Motor Velocity	7	Motor velocities of the seven joints, derived from the derivatives of motor positions.
Joint Position	7	Rotation or translation of the seven joints, derived from motor positions.
Joint Position Desired	7	Desired joint positions of the seven joints, derived from the joint control command, or from the Cartesian control command and inverse kinematics.
Joint Velocity	7	Joint velocities of the seven joints, derived from the derivatives of joint positions.
Motor Current Command	7	DC current commands of the motors, determined by motor position PD control.
Motor Torque	7	Motor torques, derived from the motor current commands.
Jacobian Velocity	6	Translational and rotational velocity of the end effector, derived from the Jacobian matrix and joint velocities.
Jacobian Force	6	Force and torque of the end effector, derived from the Jacobian matrix and motor torques.

Continued on next page

Table 4.1: Features in Raven-II State (continued)

Feature	Dimension	Description
Desired Grasper Position	1	A floating-point number representing the open/close command of the grasper.

Note: A typical Raven-II system contains two arms. The dimension shown in this table is for one arm, except for Time Stamp, Run Level, Sub-Level, and Last Sequence, which are shared by both arms in the system.

To establish a learning-based calibration with desirable performance, comprehensive input features should be provided to the machine learning model. For the software system of Raven-II, a ROS message - 'ravenstate', integrates the measurements, control commands, and state estimation of the robot. Details of 'ravenstate' are shown in TABLE 4.1. Robot states messages are published at 1000Hz and are down-sampled to a heuristically-tuned update rate of 25Hz as the input to the following neural network models in 4.1.2.

The features in the robot states are not all independent and original, as shown in Fig. 4.1. For example, motor positions are derived from encoder values and offsets, with a linear relationship. Similarly, the joint positions are also linearly related to motor positions. Thus, although encoder values, motor positions, and joint positions have distinct physical meanings, once standardized as machine learning model inputs, they become redundant and convey the same information. The desired motor, joint, and end-effector positions and orientation only have minuscule differences compared to the current positions and orientation, as the desired values are obtained by adding the control increment in the servo control loops. For example, with Raven-II's 1000 Hz servo rate, a joint velocity of 10 deg/s results in a difference of 0.01 deg between the desired and current joint position.

On the other hand, end-effector positions and orientation are derived from joint positions by forward kinematics, which is a non-linear relationship. Because of the cable-driven mechanism, all encoders of Raven-II are placed on the motors at the base of the robot. The joint positions are derived

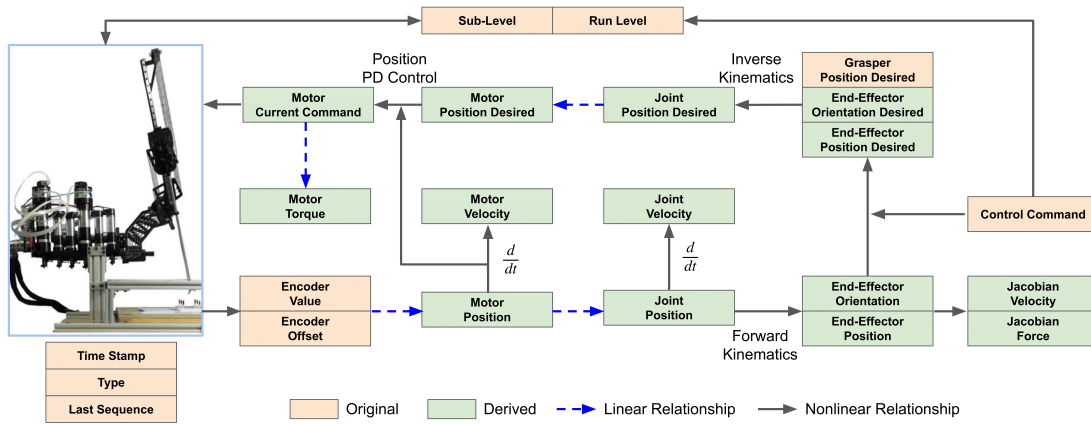


Figure 4.1: Relationship of features in the Raven-II robot states. The control command is included in this figure for an explanation of robot states but is not included in the robot state. The control command is the end-effector pose increments in Cartesian space. And it can also be given as increments in joint space. In this case, the desired joint positions will be directly determined by current joint positions and the control command, without forward and inverse kinematics. If joint-level control is applied, the current end-effector position and orientation will still be computed but the desired position and orientation will be set equal to the current values.

from motor positions and thus contain a considerable amount of error attributable to the cable effect and motor backlash. This error is also propagated to end-effector position and orientation by forward kinematics, which significantly reduces the accuracy of the robot in the absence of external tracking and feedback. It is also noteworthy that motor torques are derived from motor current commands, which are determined by servo position PD control based on the current and desired motor positions. The robot states do not include motor current measurements. Once the motor current commands are sent to the lower-level hardware controller, the servo motor drives run current feedback control with built-in Hall sensors.

4.1.2 Learning-based Models and Trajectories of Calibration

To set up a learning-based calibration to provide more accurate joint positions than the original state estimation, a DNN and a GRU model are built, as shown in Fig. 4.2. The models take the flattened features (4.1.1) from the robot state - 'ravenstate' ROS message as input. The output is the error of the first 3 joint positions, instead of the joint positions themselves, which allows faster training convergence and better accuracy. Subsequently, the calibrated joint positions are obtained by adding the joint position errors back to the robot's original inaccurate joint positions. In this work, for enhanced accuracy and reliability, ground truth joint positions are obtained by temporary encoders on the joints, which could be removed before arm sterilization in real practice, as in [38]. Alternatively, visual tracking is preferred for its better feasibility, as used in [41] and [43]. The first 3 joints, positioning joints, are studied in this work. Joints 4-7 (tool rolling, wrist, left & right finger) are not included because 1) they are concentrated near the end-effector with only 1 non-zero link length of 13 mm and thus have less influence on positional accuracy; 2) joints 4-7 are located on the surgical instrument and have small sizes, which makes it difficult to install encoders on these joints and obtain as reliable ground truth as the 3 positioning joints.

Data on the robot state and ground truth joint positions are collected to train the learning calibration models and to evaluate the performance. Zig-zag trajectories cover the workspace efficiently with desired sparsity and are thus used as training trajectories, as shown in Fig. 4.3. Although Cartesian-based control can be more common and intuitive in real practice, the accuracy of the robot control is undermined by the cable-driven mechanism that causes inaccurate joint positions

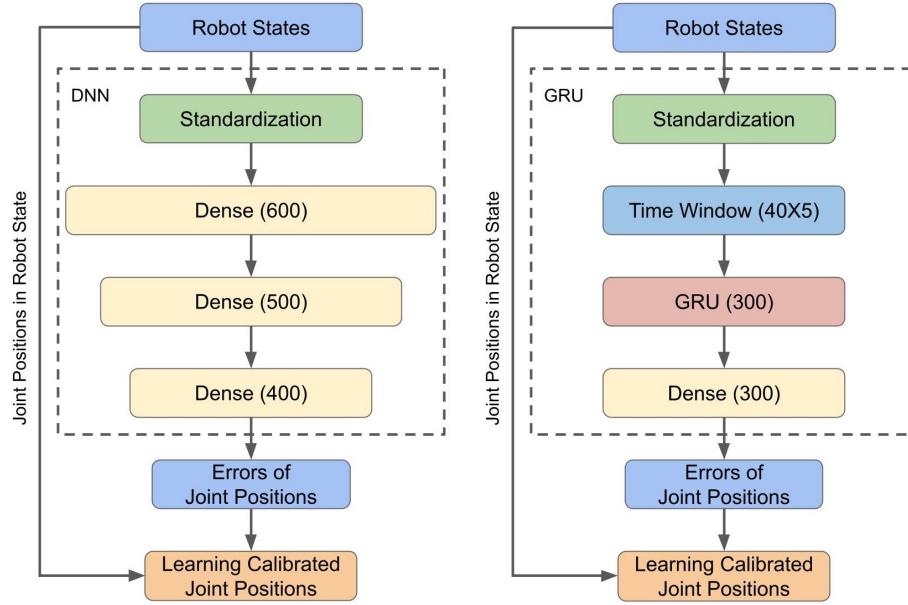


Figure 4.2: Model of the DNN and GRU in learning calibration. The outputs of the models are the errors of the joint positions and are added back to the original joint position estimation for correction.

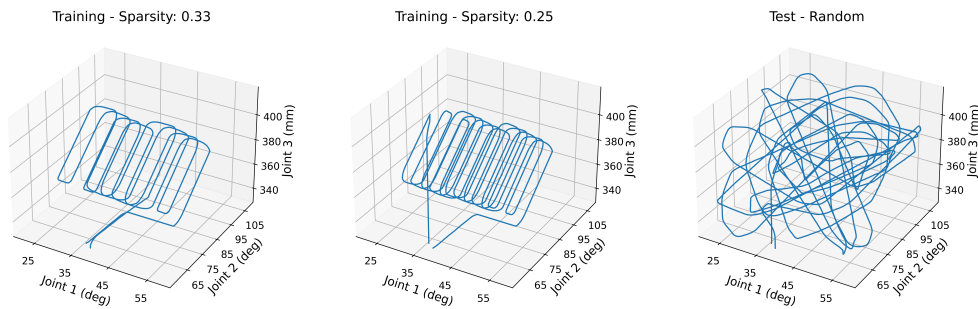


Figure 4.3: Examples of zig-zag training trajectories and random test trajectories. Note that the trajectories are shown in joint space instead of Cartesian space.

calculated by motor positions. Thus, to better study the joint-level accuracy, all the trajectories in this work are in joint space instead of Cartesian space. Unlike evenly distributed scatter points that have equivalent coverage of the robot workspace in all dimensions, zig-zag trajectories are continuous and thus they have better coverage in certain directions than the other directions. The direction of the zig-zag trajectories is empirically chosen to point 'diagonally' to joints 2 and 3 in the joint space. In this direction, the trajectories better explore the simultaneous movement of joints 2 and 3, under different joint 1 configurations. To prove the generalization ability of the model trained with zig-zag trajectories, random sinusoidal trajectories are used for testing. In each cycle, the velocity and target position of each joint are randomly chosen.

4.1.3 Ablation Study on Features

For learning-based methods in robotic systems, obtaining reliable input features often incurs additional costs, such as extra sensors for motor current measurement and computational expenses for filtering and derivative calculations. Although neural networks can manage irrelevant features, accommodating higher input dimensions typically requires larger models, as well as more training data to avoid overfitting. This may lead to considerable inefficiencies in the learning calibration, since re-calibration between operations can be necessary to maintain desirable accuracy. Identifying informative features and excluding unnecessary ones from the model input can enhance both accuracy and efficiency simultaneously (details in 4.3.5). To study how the features in the robot states contribute to the performance of learning calibration, different ablation studies are performed. In terms of the ablation method, removal ablation and inaccurate ablation are used. For the ablation target, ablations on feature types and on joints are used. Consequently, four types of ablation studies can be conducted by combining methods and targets: 1) removal ablation on feature types, 2) inaccurate ablation on feature types, 3) removal ablation on joints, and 4) inaccurate ablation on joints.

Removal ablation drops the target features in both training and testing, which results in a reduction of the input dimension of the learning model. If the target features have no or minor contribution to the learning calibration, the removal of these features should not undermine the performance considerably. If the target features have a major contribution, the removal of these features will force

the model to learn to derive the missing information from the remaining features if possible. For example, end-effector position and orientation can be derived from joint positions using forward kinematics, and reversely using inverse kinematics, though both relations are non-linear for a 7-joint robot. The increase in calibration error should be observed only when the important features are removed and the model is not able to derive the information of the missing features using the remaining features.

Inaccurate ablation adds noise to target features during testing, while these target features stay accurate during training. Compared with removal ablation, inaccurate ablation does not allow the learning model to know the malfunctioning features during training, and thus the model is not forced to learn to derive the related information from other accurate features. If the target features have a major contribution, the learning model will tend to rely on these features during training. However, during testing, these features are inaccurate and should therefore cause a decline in performance.

Unless stated otherwise, the added white noise in this work is $2 \times$ the standard deviation of that feature. This is an empirically chosen and relatively large noise, because the purpose of adding noise is to study if the ablated features are important as opposed to testing the models' robustness against noise. For this purpose, the noise should be large enough to largely obscure the information contained in the feature. However, the noise should not be too large. Theoretically, if an input feature is irrelevant to the output, the performance should not be influenced regardless of how large the added noise is. However, in real practice, due to imperfect training data, irrelevant features may still have small weights in the trained models instead of exact zero weights. Therefore, a huge noise added to irrelevant features during testing may still cause considerable changes in the performance and thus the irrelevant features may be misidentified as important. Thus, removal and inaccurate ablations are both utilized in the proposed method. Some important features, such as joint positions, can be easily found by inaccurate ablation (details in 4.3.2 and 4.4.1). In contrast, for some other important features, such as motor torques, it is easier to utilize the removal ablation due to their high robustness to white noises (details in 4.3.3).

Ablation on feature types focuses on chosen types of features, such as motor positions and joint positions. Because there are 22 different types of features in the robot states and some features are linear-related, ablation on different feature types is performed in groups of features (details in 4.3.1). For example, current and desired encoder values & offset, motor positions, and joint positions are

in one group and should be removed together, as they fundamentally contain the same information and are linearly related. Once a feature is chosen for ablation, that feature of each joint is removed or noise is added.

Ablation on joints focuses on the features of the chosen joint. This can help study the coupling of the joints for cable-driven surgical robots [38, 39]. For example, if removal ablation of joint 1 is performed, all the features regarding joint 1 are removed, including (desired) joint position/velocity, (desired) motor position/velocity/torque, and motor current command. Although end-effector position & orientation and Jacobian velocity & force are also related to joint 1, they are not ablated due to the highly non-linear relation.

4.2 Experiments

4.2.1 Experiments Setup

A Raven-II surgical robot was used to perform the experiments. Only the left arm was used and temporary optical incremental encoders were placed on the first 3 joints to obtain accurate ground truth of joint positions. Joint 1 and joint 2 are rotational, and Avago Technologies AEDA-3300 encoders were used, with a resolution of 80000 counts after quadrature decoding. Joint 3 is translational and Mercury II 1600 was used, with a resolution of 5 μm . The external joint encoders were calibrated to joint positions during Raven-II's homing procedure, in which the robot explored the mechanical hard limit of each joint. In this work, joint encoders were utilized for better accuracy and reliability. On the other hand, for better feasibility and the collection of ground truth for joints 4-7, visual tracking might be preferred, with an RGB-D camera [41], a motion tracking system [42], or multiple monocular cameras [43].

The first three joints of Raven-II have substantially different mechanical properties. First, like any serial arm, the loads (including both gravity torques and other loads) on each joint decrease from joint 1 to joint 3. Second, the cable lengths in joints 1 to 3 are substantially different. Joint 1 has a single alpha-wrapped pulley joint and essentially zero free cable length. In contrast, joints 2 and 3 have increasing cable lengths.

Joint control commands were sent through Collaborative Robotics Toolkit (CRTK) API [111], and the robot states were obtained and recorded from the ROS topic 'ravenstate'.

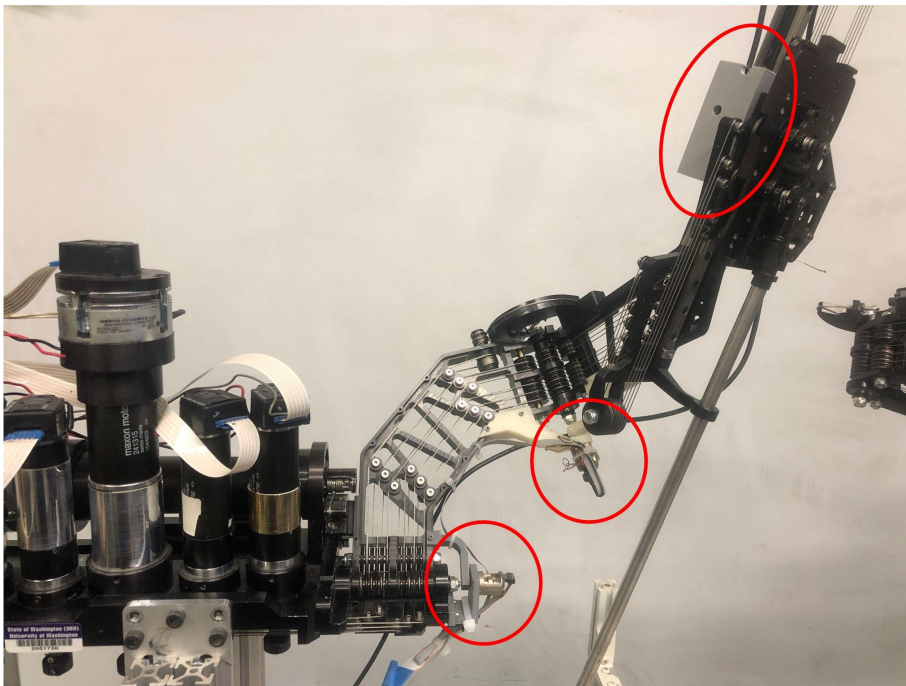


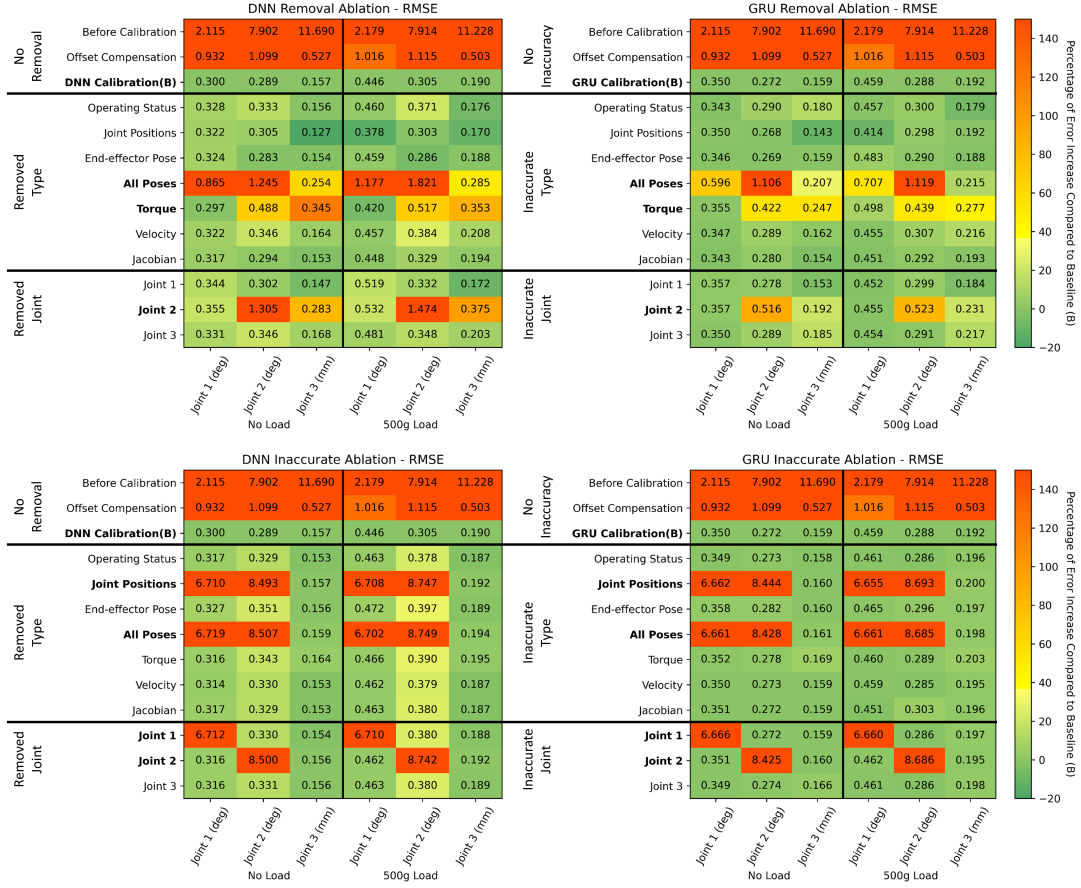
Figure 4.4: External joint encoders are mounted on the first 3 joints of Raven-II. Actual joint positions are measured to provide ground truth during training and testing of the learning calibration.

4.2.2 *Datasets and Hyperparameters of Models*

As mentioned in 4.1.2, 5 zig-zag trajectories with different sparsities ratios of 1/2, 1/3, 1/4, 1/5, and 1/6, were combined and used as the training set. The joint limits were set as (20, 60) deg, (55, 110) deg, and (330, 420) mm for joints 1, 2, and 3, respectively. Overall, the training set consisted of ~ 68000 data pairs.

The training and test sets were collected on two occasions: one without load and another with a 500-gram load applied to the end-effector. To study the long-term accuracy of the learning calibration, the test sets were collected by continuously running random sinusoidal trajectories for 6 hours, which resulted in ~ 360000 data pairs. To the best of the authors' knowledge, these are the longest testing trajectories for learning-based calibration of cable-driven laparoscopic surgical robots. In order to suggest the calibration models' ability to generalize to various trajectories, the random test trajectories were significantly longer than training trajectories and had fundamental differences in pattern.

The architecture and hyper-parameters of the models were empirically chosen, and were inspired by state-of-the-art works [28, 41, 43]. Besides the DNN model, to utilize information from time sequence, GRU units were also implemented due to their generally better computational efficiency than Long Short-Term Memory Networks (LSTM). Adam optimizer was used [114] [120], with exponential decay rate for the 1st and 2nd moment estimates of 0.9 and 0.999, respectively. The learning rate was fixed at 0.0005 with 600 training epochs and the batch size of 1024. Sigmoid activation functions were used in all layers except for the output layer which used linear activation. The mean squared error was used as the loss function. The DNN model had L1 and L2 regularization on each hidden layer kernel with factors of $1e-5$ and $1e-4$, respectively. The GRU model had L1 and L2 regularization on each hidden layer kernel with factors of $1e-4$ and $1e-3$, respectively. The time window of the GRU model was 40 time steps with a sampling rate of 5. The original 1000 Hz robot state was down-sampled and collected at 25 Hz, so the GRU model's input spanned the robot state over an 8-second period.



4.3 Results

4.3.1 Ablation Study: Removal of Features

As mentioned in 4.1.3, removal ablation was performed by removing the target features in both training and testing. The joint position errors after learning calibration are shown in Fig. 4.5 (top). Each entry regarding learning calibration represents an average of 5 repetitive training and testing with different global random seeds, which applies to all experiments in this work. Standard deviations are generally smaller than 5% and are thus omitted. The error before calibration and the error after fixed offset compensation were used as references. Removal ablation on types of features was accomplished in groups:

1. **operating status** - time stamp, run level, sub-level, last sequence, arm type, and desired grasper position.
2. **joint positions** - current/desired motor positions and joint positions.
3. **end-effector pose** - current/desired end-effector position and orientation.
4. **all poses** - all features in 2) and 3).
5. **torque** - motor current commands and motor torques.
6. **velocity** - motor velocities and joint velocities.
7. **Jacobian** - Jacobian velocities and Jacobian forces.

One group of features was removed at a time and the calibration error was compared with the baseline, in which all features were used to train the learning calibration models. Encoder values, offsets, and the 4th motor position were excluded from all experiments unless stated otherwise (details in 4.3.4).

It can be seen that the learning-based method improved the accuracy of joint position estimation significantly. However, the performance differed when certain features were not provided (Fig.

4.5, top). In terms of different feature types, the largest error for joint 1 occurred when all pose-related features were removed, compared with using all features, the RMSE increased for +0.565 deg (DNN) & +0.246 deg (GRU) without load and +0.731 deg (DNN) & +0.248 deg (GRU) with 500g load, while the removal of the other types of feature did not cause a considerable increase in calibration error of joint 1. For joint 2, the removal of all pose-related features also led to the largest increase in error, +0.956 deg (DNN) & +0.834 deg (GRU) without load and +1.516 deg (DNN) & +0.831 deg (GRU) when loaded. Removal of torque and velocity-related features also caused a certain increase in error, but was much less than removal of pose-related features. Unlike rotational joints 1 and 2, the largest error increase for translational joint 3, by +0.188 mm (DNN) & +0.088 mm (GRU) when unloaded, and by +0.163 mm (DNN) & +0.085 mm (GRU) when loaded, was due to the removal of torque-related features. A smaller increase in error was also caused by the removal of all pose-related features. Overall, compared to the DNN model, the GRU model did not show considerably better accuracy, but showed better robustness in the removal of features. Examples of calibrated trajectories are shown in Fig. 4.9 (DNN) and Fig. 4.10 (GRU). The trajectories with the legend “L-L-A” (Large model size, Large training set, and All robot states used as input) correspond to the baseline DNN/GRU calibration. The trajectories with the legend “L-L-M” correspond to the removal of pose features. From the trajectories, it can be seen that compared to fixed offset compensation, the learning calibration models improved the accuracy considerably, however, the performance was undermined when pose features were not included as model inputs. The details of the rest of the trajectories can be found in 4.3.5.

When all features related to joint 1 were removed, no major error increase was observed in all 3 joints for both DNN and GRU models. When joint 2 features were removed, for DNN calibration not only did the joint 2 RMSE increase by +1.016 deg (unloaded) and +1.169 deg (loaded), but joint 3 RMSE also increased by +0.126 mm and +0.185 mm without and with load, respectively. However, for GRU calibration, the RMSE of joint 2 saw a smaller increase of +0.244 deg (unloaded) and +0.235 deg (loaded), and the RMSE of joint 3 did not increase considerably. Compared to the removal of joint 2 features, removing joint 3 features did not increase the error on joint 3, or the other 2 joints.

4.3.2 Ablation Study: Inaccuracy of Features

The inaccurate ablation was constructed similarly to the removal ablation, but instead of removing the target features in both training and testing, substantial white noises with a range of $2\times$ the standard deviation were added to these features only during testing, but not during training. The evaluation of calibration accuracy is shown in Fig. 4.5 (bottom). For DNN calibration (bottom left), in terms of different feature types, inaccurate joint position-related features caused a tremendous error that was even larger than the raw error without the calibration on joints 1 and 2. However, the accuracy of joint 3 was almost unaffected by the noise added. Adding noise to both joint and end-effector position-related features caused similar error increases to only adding noise to joint position-related features. In contrast, only adding noise to end-effector features did not increase the error considerably. Compared to inaccurate pose features, inaccurate torque only increased the error of joint 2 slightly with $+0.054$ deg and $+0.085$ deg when unloaded and loaded, respectively. For GRU calibration (bottom right), inaccurate joint position-related features also caused a significant increase in error. However, for the inaccuracy of other features, a slight increase in error for joint 2 was observed in the DNN calibration, while the GRU calibration showed better robustness with no considerable increase in error.

For both DNN and GRU calibration, when all features of joint 1 were inaccurate, the error only increased significantly on joint 1. Similarly, inaccurate joint 2 features also only undermined learning calibration on joint 2. However, in contrast, inaccurate joint 3 features had no obvious influence on the calibration accuracy, even on joint 3 itself (further study is in 4.3.3).

4.3.3 Effectiveness of Torques

Comparing the results of removal and inaccurate ablation in Fig. 4.5, inaccurate pose features caused significantly larger errors than removing these features. However, torque features suggested better robustness to the added significant white noise. Thus, a further experiment specifically on motor torque and current command was performed and the result is shown in TABLE 4.2. In this experiment, torque features were modified by 4 different operations, and the influence on calibration error was evaluated. In TABLE 4.2, 'None' means the original torque features were used with no modification, ' $\pm$ STDV' means that white noise with range $\pm$ STDV was added. These two trials

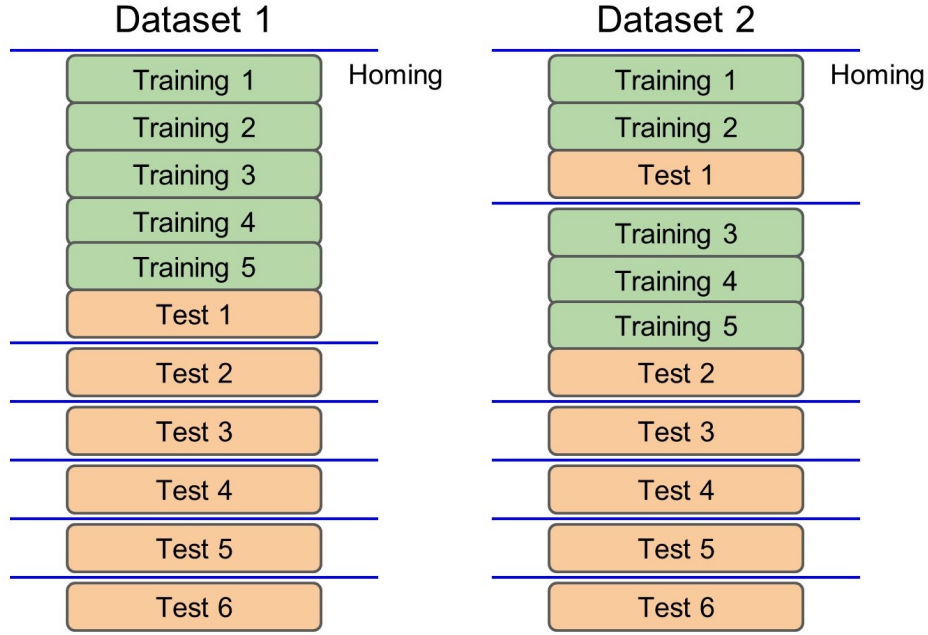


Figure 4.6: Two datasets used to study the effect of homing on the learning calibration, without homing in training (left), and with homing in training (right).

were used as baseline comparisons. The remaining two trials, ' $\times -1$ ' and ' $\times 3$ ' means multiplying the torque features with -1 and 3, which resulted in the same change in amplitude and the difference was whether the direction of the modified torque features changed or not.

For unloaded DNN calibration, the increase in RMSE with ' $\times -1$ ', which reversed the direction, was +0.088 deg, +0.314 deg, and +0.683 mm on joints 1, 2, and 3, respectively. For the operation of ' $\times 3$ ' that had the same change in magnitude but did not reverse the direction, the increase in RSME was much smaller, especially on joint 3, with +0.060 mm increased RSME. Similar trends and changes in calibration accuracy were found when loaded and with the GRU model. This observation showed that in the learning calibration of this work, the direction of torque features was more important than the amplitude of these features.

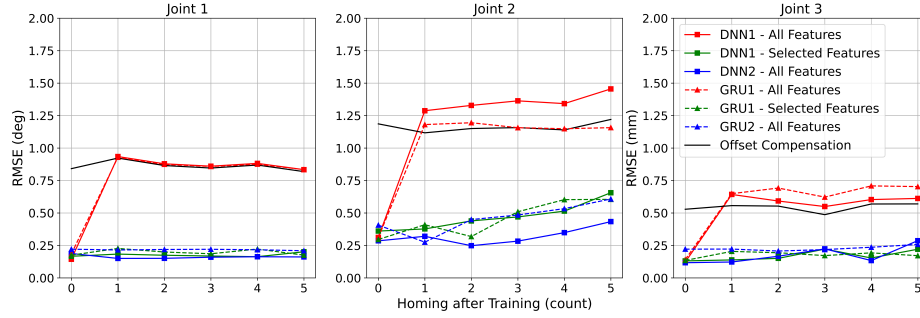


Figure 4.7: Robustness of calibration error against robot re-homings when using different features as input of the DNN and GRU model. DNN1 and GRU1 were trained by Dataset 1. DNN2 and GRU2 were trained by Dataset 2, as shown in Fig. 4.6. Six individual test sets were collected with robot re-homing in between. In the selected features, encoder values and offsets as well as the 4th motor position (spare and not used) were excluded. For a clearer comparison of learning calibration improvement, the error following an offset compensation was used as the baseline (black line), rather than the error from the original joint positions recorded in the robot's state.

4.3.4 Homing Inconsistency

As mentioned in 4.2.1, homing is an important procedure during robot initialization after power on, in which the robot explores the hard limit of each joint and registers the encoder offsets. To examine the effects of robot homing, two new datasets were collected, as illustrated in Fig. 4.6. The training set was collected in the same way as datasets in 5.2.1. The test set also consisted of several random sinusoidal trajectories. The robot underwent a re-homing procedure between test trajectories without restarting the hardware power. The key difference between the two datasets was that for Dataset 1, no homing procedure was conducted between training data trials. In contrast, for Dataset 2, a homing procedure was carried out after the first two training trajectories, allowing the training set to 'experience' one homing session. For both datasets, test trials were collected before each re-homing and after the final re-homing.

As Fig. 4.7 shows, when using Dataset 1 and the training set 'experienced' no homing, if all the features in the robot state were used including encoder values and offsets (DNN & GRU1 - All Features), the calibration errors were small in all 3 joints before the robot re-homing. However,

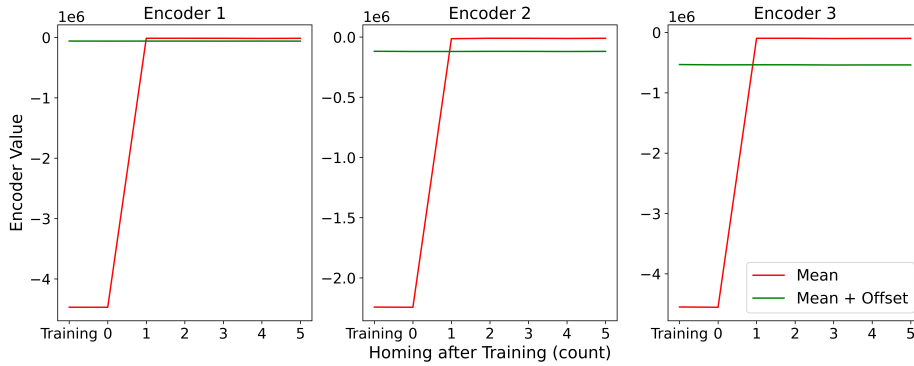


Figure 4.8: Motor encoder values of the first 3 joints affected by robot homing. The mean encoder value and offset compensated mean encoder value of the training set and 6 test sets after 0-5 robot homing are shown. The power of robot hardware stayed on during the collection of training and test sets.

after at least one re-homing, the error increased dramatically, close to the fixed offset compensation. And the error did not increase considerably when more homings were performed. The removal ablation was conducted to identify three features that contributed to the significant increase in error. The first 2 features were encoder values and encoder offsets, shown in Fig. 4.8. After the first re-homing, the average encoder values changed significantly, whereas subsequent re-homings did not result in considerable changes to these values. However, motor positions and joint poses stayed consistent regardless of homings. This was because when adding encoder offset back to encoder values, these values also stayed consistent. However, the learning calibration models failed to learn this linear combination of features if there was no homing in the training data. The last feature was the 4th 'motor position' in the ROS topic 'ravenstate', which does not represent the actual position of the 4th motor. Instead, this is a meaningless feature in our experiments, since the Raven-II has a spare independent joint dimension and this 'motor position 4' is not used with the standard Raven tool. In the homing experiment, this feature followed a similar pattern to encoder values and had a rapid change after the first homing. Once these features were removed (DNN & GRU1 - Selected Features), there was no longer a considerable decrease in calibration accuracy among homings. On the other hand, when using Dataset 2 and the training set 'experienced' 1 homing, even including all

features (DNN & GRU2 - All Features), the performance remained desirable for all 3 joints among all re-homings.

To better understand the effectiveness of the encoder features, learning calibration experiments with different removal and inaccurate ablation on pose-related features were conducted, and the results are presented in TABLE 4.3 ('motor position 4' still removed because it was not used with the default Raven tool).

Learning calibrations 1 and 2 utilized Dataset 1, which included no re-homing procedures in the training set, while the subsequent calibrations were based on Dataset 2 which incorporated re-homing in the training data. Calibrations 1 and 3 were both trained using all accurate features, the sole distinction was that calibration 3 included re-homing during training, unlike calibration 1. As mentioned previously, the results indicated that the inclusion of encoder features led to decreased accuracy when the training data did not encompass re-homing events, whereas accuracy was maintained when re-homing was included in the training.

Calibration 4 also used all accurate features in training, but encoder values and offset were inaccurate during testing. Upon comparing calibrations 3 and 4, the addition of significant noise to encoder features did not result in any substantial performance differences. Hence, it appears that when re-homing was integrated into the training, the learning calibration model learned to disregard the encoder features.

However, in calibration 5, where encoder values were the sole pose features, the model was compelled to rely on these features, and accuracy was preserved. This was corroborated by calibration 6, when noise was added to the encoder features that were now the only remaining pose features. There was a significant decline in performance for joints 1 and 2, joint 3 exhibited less change because it depended more on torque features, as noted in 4.3.3. Until now, calibrations 1 through 6 indicate that when training included re-homing and all other pose features were absent, the learning calibration model effectively utilized the encoder values and offsets.

To better comprehend how the models utilized encoder features, calibration 7 was conducted by first introducing noise to the encoder values and then subtracting the same noise from the encoder offsets. This was done so that the noise would nullify itself when the encoder values and offsets were added up, mimicking the human approach to these features. The result showed desirable accuracy, which indicated that the learning calibration models also added encoder value with offset at first,

before applying any non-linearity.

4.3.5 *Lightweight Models with Selected Features*

With the informative features identified in 4.3.1 and 4.3.2, lightweight versions of the DNN and GRU models were developed. Compared to the models (4.1.2) that were used in the rest of the experiments, the lightweight DNN model was designed with 2 hidden layers, each containing 100 units, reduced from 3 hidden layers with 600, 500, and 400 units, respectively. The lightweight GRU model has reduced GRU layer units from 300 to 50, and the fully connected layer units from 300 to 100. The features empirically selected from the ablation study for the lightweight DNN model were joint positions and torques, while the lightweight GRU model additionally included end-effector pose. The number of training epochs was also lowered from 600 to 200 for both lightweight models. All other hyperparameters and test sets maintained the same. Two training sets were used in this experiment, the original large set comprised 2508 sec of zig-zag trajectories, combining multiple sparsities, detailed in 4.1.2 and Fig. 4.3. The smaller training set only consisted of 476 seconds of a zig-zag trajectory with a sparsity of 1/4.

The calibration error and computational cost of the lightweight models were evaluated on the original test set and compared to the baseline result obtained from 4.3.1, presented in TABLE 4.4. For the lightweight DNN model, the trainable parameters were reduced significantly from 566303 to 34503 utilizing all features as input and to 13703 utilizing selected features, which accelerated the training speed on GPU ~ 3 times faster. More crucially, the training on CPU was ~ 26 times faster and became practically feasible, reducing from more than 1 hour to ~ 2.5 minutes. The inference time was ~ 55 ms on GPU and ~ 72 ms on CPU, which saw fewer changes among models. Notably, the evaluations of inference time were conducted with single robot state inputs, without the use of mini-batches. This approach aligns with real-time robotic control systems where batch inference across multiple control loops is impractical. Our implementation achieved an inference speed of 10-20 Hz. This meets the requirements of CRTK's `interpolate` control mode[111], which is designed for higher-level controls such as motion planning, visual tracking, and teleoperation. It cannot support Raven-II's 1000 Hz low-level servo control loop, but this is mainly for the motor position PD control and does not need the calibration since the calibration can be incorporated into

higher-level control.

In terms of calibration accuracy, compared to the baseline large DNN model with RMSE of 0.300 deg, 0.289 deg, and 0.156 mm in joints 1, 2, and 3, respectively, the lightweight DNN model exhibited marginally increased errors when using all features as input. In contrast, when using selected features as input, the lightweight model demonstrated enhanced performance across all three joints - 0.297 deg, 0.236 deg, and 0.122 mm. When the size of the training set was reduced from the original 2508 sec to 476 sec, a notable decline in performance occurred when all features were used as input, yielding 0.813 deg, 2.432 deg, and 0.470 mm. However, the accuracy was maintained at desirable levels when only selected features were used as input, 0.375 deg, 0.330 deg, and 0.142 mm in the 3 joints.

While the lightweight GRU model trained much faster, training with the large training set still required at least 16 minutes on a CPU, making it less practical for real-world applications without a GPU. When trained by the large training set, compared to the performance of the baseline GRU model, the lightweight GRU model with all input features had slightly increased error in joints 1 and 2, while the lightweight GRU model with selected input features had better accuracy in joint 1, with negligible differences in other joints. Despite similar inference times across the three models, the lightweight versions significantly outpaced the original GRU model in training speed. the lightweight model using all features experienced a marked rise in calibration error - 0.858 deg, 1.254 deg, and 0.573 mm across the three joints. In contrast, the lightweight model that used selected input features had significantly better performance, 0.452 deg, 0.481 deg, and 0.387 mm. Examples of the calibrated trajectories under different calibration configurations are shown in Fig. 4.9 (DNN) and Fig. 4.10 (GRU). Consistent with Table. 4.4, when the training set was large and important features were provided, all calibration configurations achieved desirable accuracy. However, when the training set was small, unnecessary input features should be excluded to maintain the calibration accuracy. To sum up, the efficiency of the lightweight models without GPU can be feasible in the time scale of real practice, and excluding unnecessary input features can improve the accuracy, especially when training time is limited. As reported in [45], the accuracy of positional calibration can drop with operating time and load. To ensure calibration performance, re-calibration can be necessary before each operation. Thus, for feasibility in real practice, time efficiency is in demand. The proposed ablation study and exclusion of irrelevant input features ensured desirable calibra-

tion accuracy with 9-minute short training, while including all robot states saw a significant drop in accuracy. In summary, identifying and excluding irrelevant input features played a very important role in improving the efficiency and feasibility of learning-based calibration in the time scale of real practice, in which training data is limited while re-calibration can be needed frequently.

4.4 Discussion

Although ablation studies were conducted, the neural network models essentially remain a 'black box' and it may not be rigorous to speculate how the neural network models accomplished the tasks using human understanding. Therefore, the discussion in this section is rather suggested than conclusive.

4.4.1 Ablation to Find Contributing Features

Fig. 4.5 shows that removing either joint position or end-effector features did not significantly impact the learning calibration's performance. However, once both joint and end-effector pose features were removed, significant increases in errors of all 3 joints were observed, particularly in joints 1 and 2. On the other hand, making joint position features inaccurate led to significant error increases, highlighting their importance for calibration performance. Recall that removal ablation removed features in both training and testing, while inaccurate ablation only added noise when testing. The comparison of these ablation trials suggests that learning calibration models depend more on joint position information than on end-effector information for joint calibration. However, without joint position features during training, the models were compelled to infer joint information from end-effector features. In summary, removal ablation may not be able to find key contributing features like joint position features, because the learning models can derive the removed information from indirect features.

On the other hand, learning calibration models demonstrate notable robustness to white noise for certain contributing features, like motor torques, as evidenced in TABLE 4.2. This robustness arises because the torque's direction is more critical than its magnitude. Therefore, removal ablation readily identifies these features, whereas more particular inaccurate ablations are required to observe performance differences.

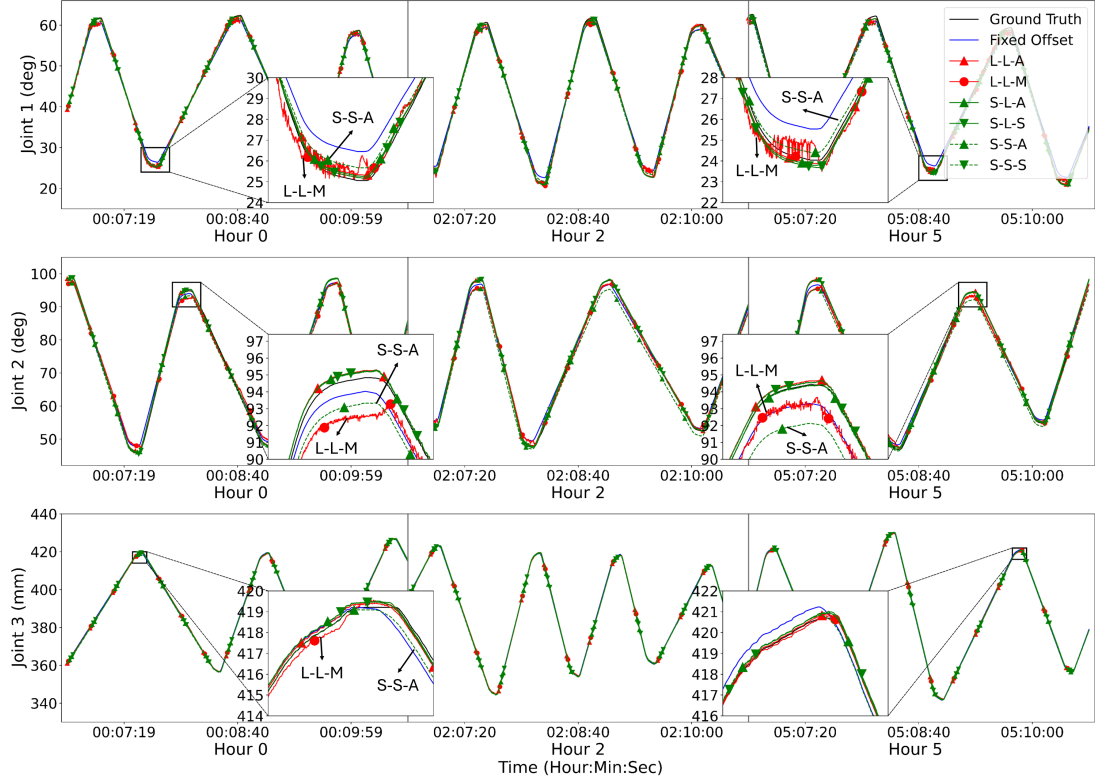


Figure 4.9: DNN calibration: example joint trajectories from 6-hour continuous operations utilizing calibrations with different model sizes, training sets, and input features. The 3-letter legend represents “model size - training set - input features”. Model size can be large (L) or small (S), corresponding to the large models (Fig. 4.2 and 5.2.1) utilized in most of the experiments or the lightweight models developed in 4.3.5, respectively. The training set can also be large (L) - a 2508-second trajectory with multiple sparsities, or small (S) - a 476-second trajectory with a sparsity of 1/4. The input features can be all (A) features in the robot state, selected (S) important features by ablation study, or missing (M) important pose features. Compared to fixed offset compensation, most of the calibration configurations had a considerable improvement in accuracy, except for “S-S-A” - small model, small training set but all robot states as input, and “L-L-M” - large model, large training set, but the important pose features were missing.

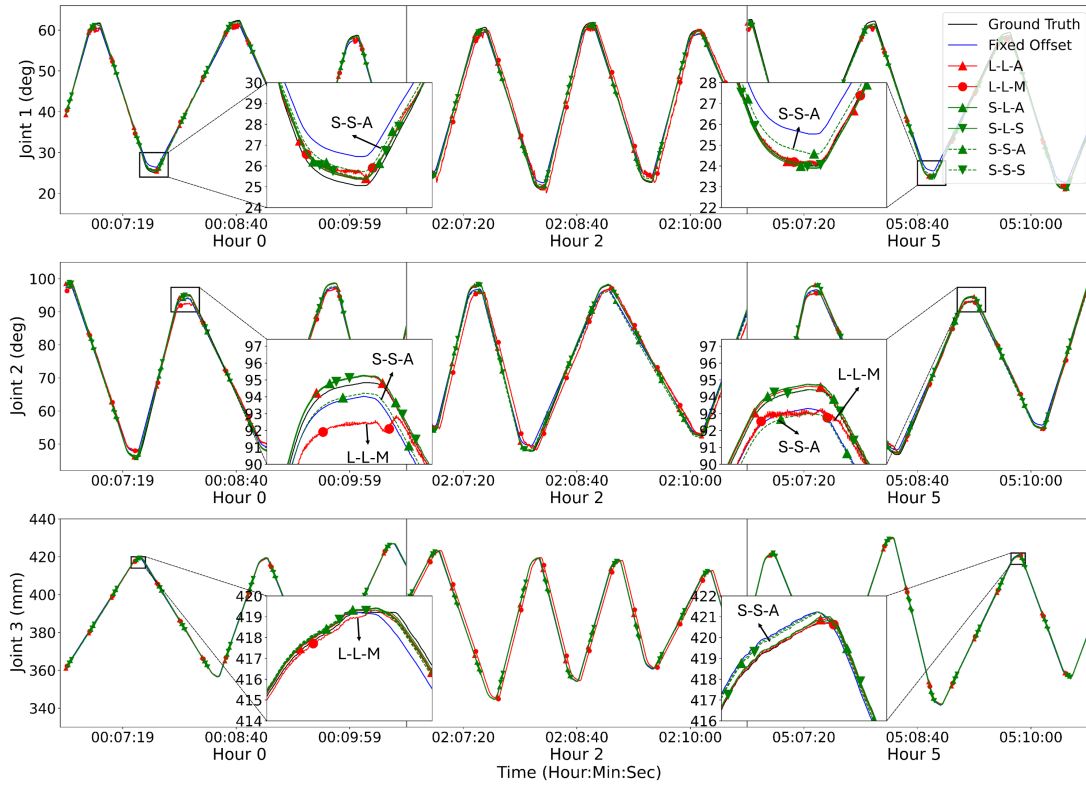


Figure 4.10: GRU calibration: example of joint trajectories in 6-hour continuous operation utilizing calibrations with different model sizes, training data, and input features. The legends have the same meaning as Fig. 4.9. The performance of the GRU calibration configurations was similar to the DNN calibration.

4.4.2 Usage and Selection of Features

While joint position features are crucial, learning calibration models are capable of deriving joint position information using the end-effector pose. However, with both features available, inaccurate joint position features resulted in significantly more error than inaccurate end-effector features. This suggests that joint position features have a more direct relationship with joint calibration, thus receiving a larger weight during training. A similar pattern is observed in TABLE 4.3, where learning calibration models resorted to encoder features only after all the other more direct features (joint positions, motor positions) were removed. This behavior might indicate that learning calibration models prefer direct and straightforward features over indirect ones containing similar information.

The ablation study revealed that inaccurate pose features resulted in greater errors than removed pose features. This suggests that in real practice if certain features are expected to be inaccurate, it may be beneficial to exclude these features in training to 'force' or 'guide' the learning model to derive the information from other features. However, the supplementary features should have explicit or implicit relations with the removed features, which could be non-linear.

4.4.3 Coupling between Joints

Fig. 4.5 illustrates that removing joint 1 features did not reduce the calibration accuracy of joint 1, whereas inaccuracies in joint 1 features did, underscoring the importance of joint 1 information for its calibration. However, the learning calibration models were capable of deriving this information from the features of the other two joints. On the other hand, the absence of joint 2 features compromised the calibration accuracy of both joints 2 and 3. This indicates that joint 2 information is crucial for calibrating both joints 2 and 3. In contrast, neither the removal nor inaccuracy of joint 3 features led to increased errors in any joint, suggesting that the influence of cable coupling in joint calibration can be directional. Joint 2 influenced the calibration of joint 3, yet joint 3 did not significantly affect joint 2 in return.

4.5 Conclusion

In this work, a learning-based calibration of the Raven-II surgical robot is developed, utilizing a DNN model and a GRU model that take the robot's original state estimation as input and outputs

calibrated joint positions. Training with zig-zag trajectories, the calibration achieved a reduction in joint errors by at least 62.4% (unloaded) and 54.8% (loaded). Through removal and inaccurate ablation studies, raw joint positions and motor torques were identified as vital for calibration accuracy, with inaccuracies in raw joint positions significantly impacting performance. Moreover, we demonstrated that the learning models could be guided to deduce joint position information from the end-effector's position and orientation by removing the joint position features during training. Our findings also highlighted the torque direction's importance over its amplitude and how encoder value inconsistencies post-homing could escalate calibration errors. Yet, omitting encoder features when other pose-related features are present, or incorporating at least one homing procedure in the training, ensures consistent, optimal performance. Lastly, employing only the key features identified in the ablation study allowed the creation of lightweight models that not only enhanced accuracy but also significantly expedited training.

Table 4.2: Calibration Error under Modified Motor Torque Features

Operation	DNN						GRU					
	Joint 1 (deg)		Joint 2 (deg)		Joint 3 (mm)		Joint 1 (deg)		Joint 2 (deg)		Joint 3 (mm)	
	RSME	Peak	RSME	Peak	RSME	Peak	RSME	Peak	RSME	Peak	RSME	Peak
No Load												
None	0.300	0.590	0.289	1.267	0.157	0.641	0.350	0.720	0.272	1.243	0.159	0.651
$\pm$ STDV	0.316	0.623	0.343	1.597	0.164	0.768	0.352	0.730	0.278	1.292	0.169	0.732
$\times -1$	0.388	0.761	0.603	1.791	0.840	1.389	0.407	0.794	0.620	1.586	0.800	1.378
$\times 3$	0.294	0.608	0.487	1.433	0.217	0.694	0.356	0.776	0.459	1.422	0.182	0.737
500g Load												
None	0.446	0.772	0.305	1.430	0.190	0.751	0.459	0.890	0.288	1.367	0.192	0.740
$\pm$ STDV	0.466	0.826	0.390	1.744	0.195	0.846	0.460	0.898	0.290	1.403	0.203	0.818
$\times -1$	0.529	0.986	0.595	1.752	0.761	1.476	0.517	0.982	0.643	1.675	0.753	1.440
$\times 3$	0.448	0.825	0.502	1.723	0.219	0.698	0.482	0.958	0.453	1.479	0.205	0.766

* The largest errors in each section are marked bold.

Table 4.3: Calibration Error under Modified Joint Position Features

		DNN			GRU		
No.	Operation	Joint 1 (deg) RMSE Peak	Joint 2 (deg) RMSE Peak	Joint 3 (mm) RMSE Peak	Joint 1 (deg) RMSE Peak	Joint 2 (deg) RMSE Peak	Joint 3 (mm) RMSE Peak
Dataset 1 - No homing in training data							
1	All accurate features	0.755 1.612	1.181 3.552	0.520 1.459	0.759 1.577	1.025 2.729	0.586 1.645
2	No encoder value & offset	0.176 0.563	0.469 1.721	0.169 0.645	0.197 0.731	0.457 1.465	0.178 0.638
Dataset 2 - Homing in training data							
All features included							
3	All accurate features	0.162 0.553	0.320 1.208	0.175 0.618	0.217 0.700	0.459 1.394	0.227 0.690
4	Inaccurate encoder value & offset	0.153 0.503	0.319 1.260	0.175 0.596	0.202 0.695	0.326 1.387	0.172 0.636
Only encoder values and offsets as pose features							
5	Accurate encoder value & offset	0.148 0.437	0.291 1.373	0.172 0.645	0.188 0.612	0.257 1.372	0.161 0.578
6	Inaccurate encoder value & offset	0.434 1.298	0.619 2.414	0.211 0.877	0.302 1.076	0.430 1.845	0.177 0.703
7	Add noise to encoder value, subtract same noise to encoder offset	0.148 0.439	0.291 1.374	0.172 0.644	0.188 0.614	0.257 1.360	0.161 0.575

* i) Each entry is the average of the accuracy on all test trials shown in Fig. 4.6.

ii) The largest errors in each data set are marked bold.

Table 4.4: Calibration Error of Lightweight Models with All or Selected Features

Calibration Model	Training Dataset	Input Features	Calibration Error						Computational Cost				
			Joint 1 (deg)		Joint 2 (deg)		Joint 3 (mm)		Trainable Parameters	Training (sec)		Inference (ms)	
			RSME	Peak	RSME	Peak	RSME	Peak		GPU	CPU	GPU	CPU
DNN	Large 2508 sec	Baseline	0.300	0.590	0.289	1.267	0.156	0.641	566303	231	3768	59	72
		All	0.344	0.585	0.318	1.272	0.163	0.613	34503	80	143	52	73
		Selected	0.297	0.515	0.236	1.012	0.122	0.505	13703	72	138	52	71
	Small 476 sec	All	0.813	1.484	2.432	4.164	0.470	1.216	34503	21	43	53	72
		Selected	0.375	0.655	0.330	1.173	0.142	0.470	13703	18	38	55	70
GRU	Large 2508 sec	Baseline	0.350	0.720	0.272	1.243	0.159	0.651	549303	837	24097	54	79
		All	0.387	0.750	0.385	1.599	0.154	0.674	49203	237	1437	55	71
		Selected	0.319	0.682	0.276	1.336	0.169	0.717	21603	142	967	52	69
	Small 476 sec	All	0.858	1.583	1.254	3.011	0.573	1.348	49203	78	456	49	74
		Selected	0.452	1.115	0.481	1.866	0.387	1.058	21603	59	289	55	73

* i) The baseline results for DNN and GRU are obtained from 4.3.1 with original larger models and all features as input, all the rest of entries used the lightweight models introduced in 4.3.5.

ii) For the DNN model, selected features were joint positions and torques. For the GRU model, selected features were joint positions, motor torques, and end-effector positions and orientation.

Chapter 5

INSTRUMENT SEGMENTATION OF SURGICAL ROBOT END-EFFECTOR

Accurate instrument segmentation in the endoscopic vision of minimally invasive surgery is challenging due to complex instruments and environments. Deep learning techniques have shown competitive performance in recent years. However, deep learning usually requires a large amount of labeled data to achieve accurate prediction, which poses a significant workload. To alleviate this workload, we propose an active learning-based framework to generate synthetic images for efficient neural network training. In each active learning iteration, a small number of informative unlabeled images are first queried by active learning and manually labeled. Next, synthetic images are generated based on these selected images. The instruments and backgrounds are cropped out and randomly combined with blending and fusion near the boundary. The proposed method leverages the advantage of both active learning and synthetic images. The effectiveness of the proposed method is validated on two sinus surgery datasets and one intraabdominal surgery dataset. The results indicate a considerable performance improvement, especially when the size of the annotated dataset is small.

5.1 Method

5.1.1 System workflow

Fig. 5.1 shows the workflow of the entire system. During training, the proposed system uses active learning to choose the most informative samples from the unlabeled pool of the database and asks for annotation (Section 5.1.4). Then, synthetic images are generated and added to the labeled pool with the real images to ‘make the best use of’ the selected real images (Section 5.1.2). The goal is to train a segmentation model on fewer hand-labeled real images while maintaining competitive performance.

The dataset (Section 5.2.1) consists of a labeled pool and an unlabeled pool of images. Because endoscopes are widely used in robot-assisted surgeries, it is not difficult to obtain unlabeled videos and images. Some of the datasets contain background images in which surgical tools are not present,

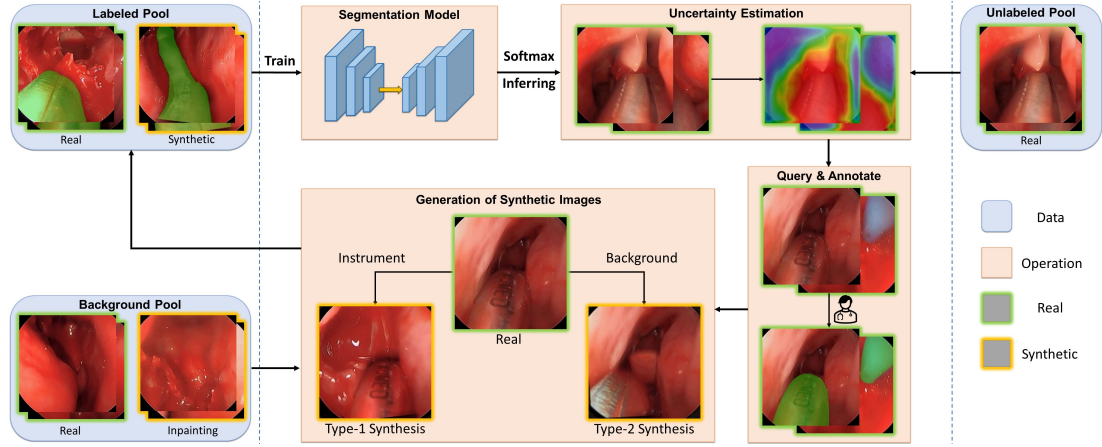


Figure 5.1: Workflow of the system. The segmentation model is first trained by labeled images. Uncertainty estimation is performed on the unlabeled real images to select the most informative ones, which are then manually labeled. Next, synthetic images are generated from the labeled images. There are 2 types of synthetic images. Type-1 syntheses have the same instrument as the original real image while Type-2 syntheses have the same background.

which can form a background pool.

Initially, some real images are randomly chosen and moved from the unlabeled pool to the labeled pool by human annotation. Synthetic images are first generated using the images in the labeled pool and are then added back to the labeled pool. Next, a segmentation model is trained using the labeled pool. Then, uncertainty estimation is applied to the unlabeled pool, and the most informative images are queried by the BALD active learning criterion, asking for annotation. Synthetic images are generated based on the newly labeled images. For each real image, there are two types of synthetic images. Type-1 synthetic images have the same surgical tool as the original real image, and the background is randomly selected from the background pool. Type-2 synthetic images have the same background as the original real image (background inpainting is applied to the original real image to remove the original tool), and the surgical tool is randomly selected from the labeled pool. If the dataset has no or few background images, background inpainting of instrument pixels in the selected labeled real images is performed and the generated backgrounds are added to the background pool (Section 5.1.3).

After the generation, the newly labeled real images and the synthetic images are added to the

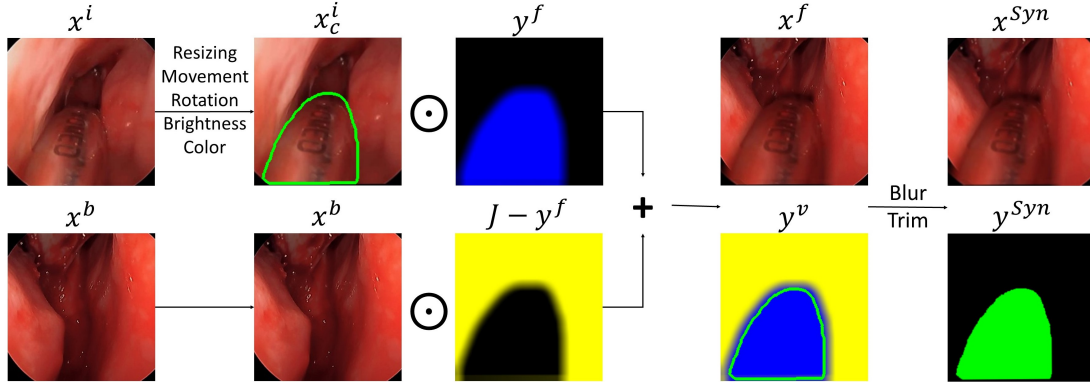


Figure 5.2: Generation of a synthetic image. Note that y^v is only for visualization, where the solid green line indicates the outline of the instrument, the yellow area is from the background image x^b and the blue area is from the instrument image x_c^i . The transition area can be found around the boundary of the instrument on the synthetic image.

labeled pool and then the segmentation model is trained again to start a new iteration. This is repeated until the labeling budget or the desired performance is reached. Budget in this work is defined as the fraction of real training images which are manually annotated, compared to the total number of real training images.

5.1.2 Generation of synthetic images

The generation of each synthetic image begins with one selected real labeled image. To generate Type-1 synthetic images, the selected image is used as the instrument image, and a background image is randomly chosen from the background pool. To generate Type-2 synthetic images, the selected image is used as the background image with inpainting applied to remove the instrument, and an instrument image is randomly selected from the labeled pool. The tool from the instrument image is copied and pasted on the background image, with resizing, movement, and fusion. Fig. 5.2 shows the workflow of the generation of synthetic images.

The procedure starts with x^i - a labeled real image that includes an instrument, and x^b - a pure background. The instrument image x^i also has a mask y^i , a binary matrix with the same size as x^i in which the instrument pixels are 1 while other pixels are 0. Resizing, movement, and rotation are

first applied to the instrument and the mask:

$$x_r^i = R(x^i, c, w, h, \theta) \quad (5.1)$$

$$y_r^i = R(y^i, c, w, h, \theta) \quad (5.2)$$

where $R(\cdot)$ is the operator, c , w , h , and θ are the factors of resizing, movement in width and height, and rotation angle, respectively. These operations are applied sequentially. A binary dilation [121] is applied on the new mask y_r^i so that in the dilated mask y_d^i , the region of the instrument is larger than the true mask of the instrument y_r^i .

$$y_d^i = \bigcup_{b \in B} y_{rb}^i \quad (5.3)$$

where B is a $d \times d$ matrix and d is the dilation kernel size, y_{rb}^i is the translation of y_r^i by b . After dilation, the fusion mask y^f is generated by applying average blur or Gaussian blur [122] to the mask y_d^i :

$$y^f = B_a(y_d^i, k) \quad (5.4)$$

$$y^f = B_G(y_d^i, k, \sigma) \quad (5.5)$$

where $B_a(\cdot)$ and $B_G(\cdot)$ are the operators of average blur and Gaussian blur, respectively. k is the kernel size and σ is the standard deviation. According to [1], the artifacts from the copy-and-paste operation may result in decreased performance if the model is trained on synthetic images. Multi-blending, using the same synthetic images but different blending methods in the training set, can prevent the model from learning blending artifacts and improve the performance on real images (Fig. 5.3). Color and brightness adjustment is applied to narrow the gap between the color style of the instrument and the background, for each channel of the adjusted image:

$$x_{c,chn}^i = \beta \frac{\sum x^b}{\sum x_r^i} \left(\alpha \frac{\sum x_{chn}^b}{\sum x_{r,chn}^i} x_{r,chn}^i + (1 - \alpha) x_{r,chn}^i \right) \quad (5.6)$$

where α is the factor of color adjustment, β is the factor of brightness adjustment, x_{chn}^b and $x_{r,chn}^i$ are the same channel of the background image and the instrument image, respectively. After the adjustment of color and brightness, the instrument is blended onto the background by:

$$x^f = y^f \odot x_c^i + (\mathcal{J} - y^f) \odot x^b \quad (5.7)$$

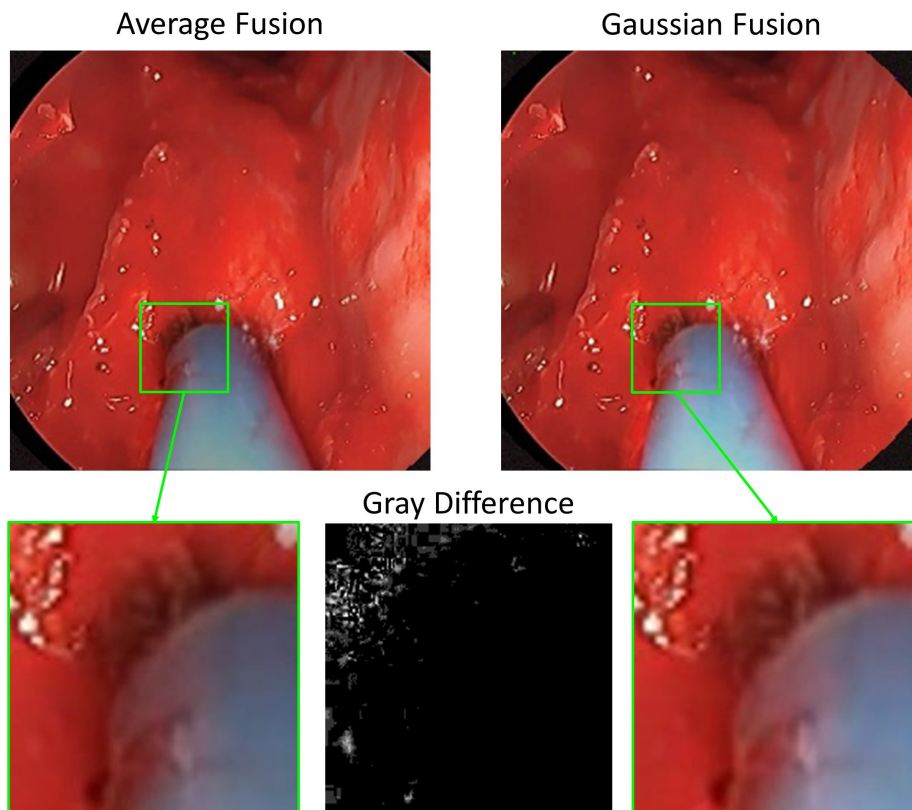


Figure 5.3: Multi-blending: images on the left and right are similar because they have the same size and position of the instrument and background, however, the instruments are blended on the background by different blending methods and parameters (average fusion on the left and Gaussian fusion on the right). According to [1], when training deep learning models, using multi-blended synthetic images can prevent the negative effect of blending artifacts.

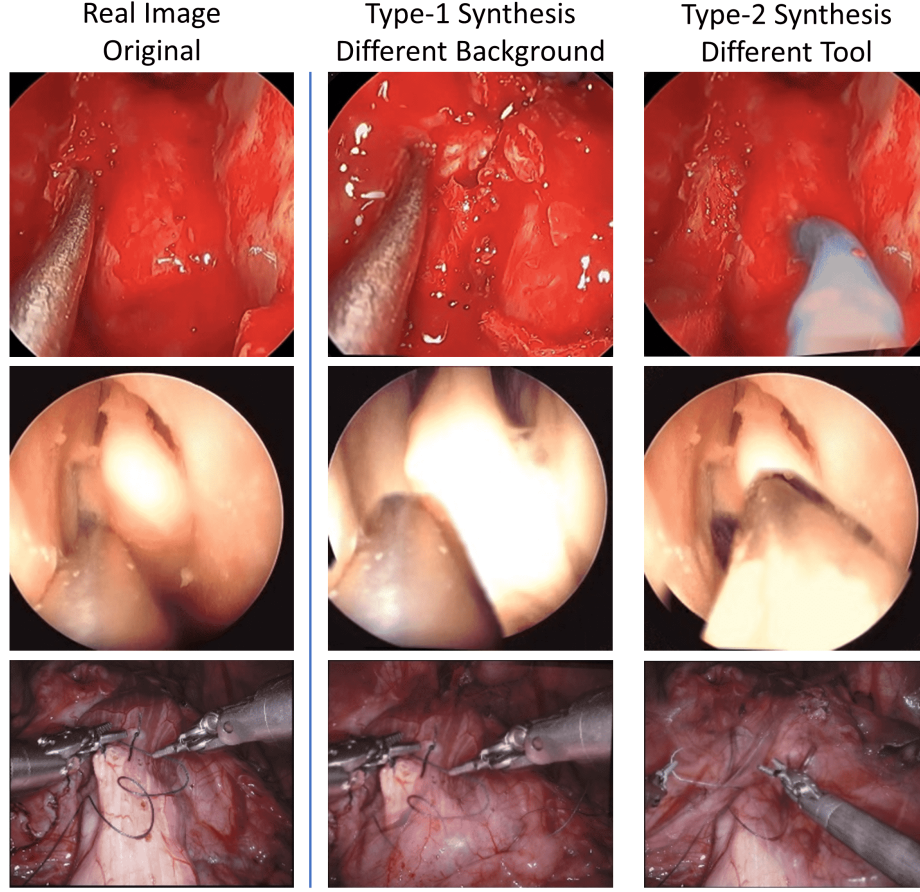


Figure 5.4: Original real images (left), Type-1 synthetic images (center) and Type-2 synthetic images (right). Compared with the real images, Type-1 has different backgrounds and Type-2 has different instruments.

where $\odot$ indicates element-wise multiplication, $\mathcal{J}$ is a matrix of ones with the same size as y^f . A weak Gaussian blur is applied and the border is trimmed to restore the outline, and thus finalize the generation of the synthetic image x^{syn} and the corresponding mask y^{syn} . Fig. 5.4 shows examples of Type-1 and Type-2 synthetic images.

5.1.3 Inpainting of backgrounds

As introduced in 5.1.2, a synthetic image is generated from a labeled instrument image and a background image. However, it is not always feasible to find background images in every dataset. Thus, for those datasets without background images, image inpainting is performed to generate back-

grounds from labeled instrument images.

Fig. 5.5 shows the procedure of background inpainting. It is similar to the generation of synthetic images. The difference is that for the inpainting of backgrounds, an area of background is blended over the instrument pixels, instead of blending an instrument over a background. There are 2 types of inpainting, self-inpainting and external inpainting. First, self-inpainting can be performed by self-flipping or rotation of the original image (5.8), if the flipped or rotated mask does not overlap with the original mask (5.9):

$$x_i^b = y_f \odot x_r^i + (\mathcal{J} - y^f) \odot x^i \quad (5.8)$$

$$y^f \cap y_r^f = 0 \quad (5.9)$$

where x^i is the image including instrument, x_i^b is the inpainting background with instrument removed, y^f is the fusion mask generated by the method mentioned in 5.1.2, x_r^i and y_r^f are the flipped or rotated image x^i and mask y^f . Flipping can be applied vertically or horizontally, and rotation can be applied for degrees of 90, 180, and 270.

However, sometimes the masks y^f and y_r^f always overlap regardless of flipping and rotation. In this case, external inpainting can be performed by randomly selecting another background from the background pool as a source for the pixels covering the instrument:

$$x_i^b = y^f \odot x^{b2} + (\mathcal{J} - y^f) \odot x^i \quad (5.10)$$

where x^{b2} is the background (original or inpainting) from the background pool and the other variables are the same as self-inpainting. To prevent the rare situation that self-inpainting is not applicable for all active selected real images in the first active learning iteration, 1 external background should be added to the background pool at the beginning of active learning. As active learning and generation of synthetic images proceeds, inpainting backgrounds will be added to the background pool.

5.1.4 Active learning

Active learning effectively reduces the annotation burden while ensuring that the model retains competitive performance with limited labeled data. The proposed system uses a pool-based AL

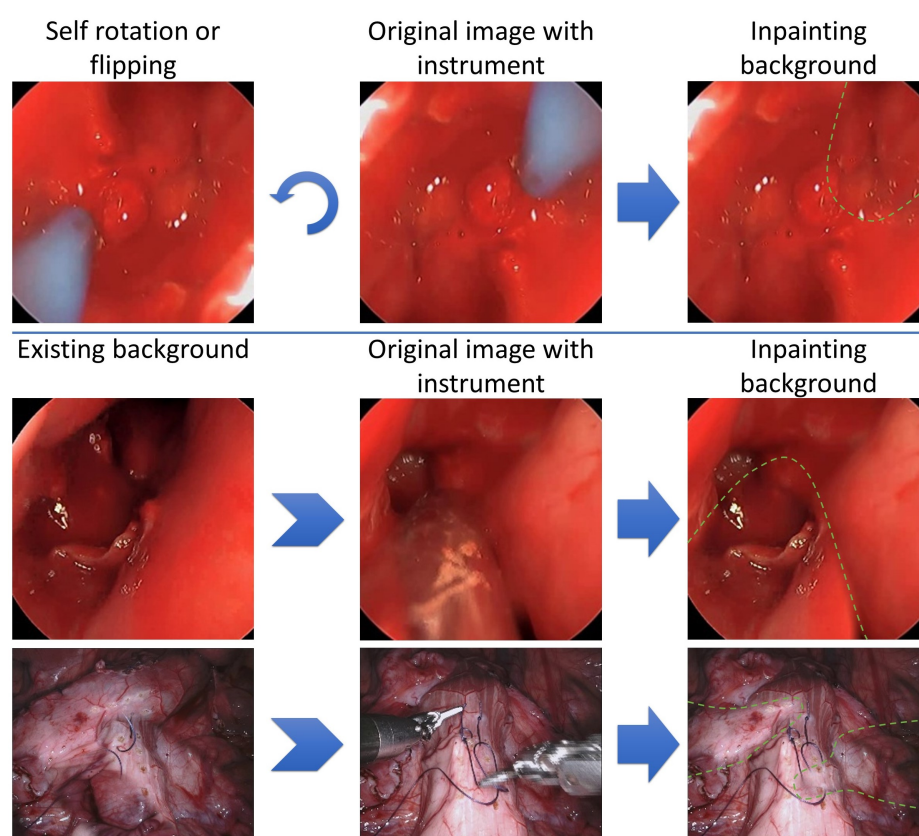


Figure 5.5: Generation of inpainting backgrounds (right). The instruments on the original images (middle) are removed by self-rotation / flipping or existing backgrounds (left).

method (Fig. 5.1). The iteration of the active learning is introduced in Section 5.1.1 and BALD is used as the criterion to query unlabeled images.

The BALD criterion chooses the images which are expected to maximize the mutual information between predictions and model posterior:

$$\mathbb{I}[y, \omega | x, \mathcal{D}_{train}] = \mathbb{H}[y | x, \mathcal{D}_{train}] - \mathbb{E}_{P(\omega | \mathcal{D}_{train})} [\mathbb{H}[y | x, \omega]] \quad (5.11)$$

where x is the input image and y is the output label, $\mathbb{H}[y | x, \mathcal{D}_{train}]$ and $\mathbb{H}[y | x, \omega]$ are the entropy [123] of the prediction $P(\omega | \mathcal{D}_{train})$ and distribution $P(y | x, \omega)$, respectively. $\mathcal{D}_{train}$ is the labeled training data, and ω are model weights. The first term seeks the images which have high average entropy in the sampled models. The second term imposes a penalty such that the images on which the models disagree are kept while overall unconfident images are dropped.

To perform BALD, Monte-Carlo (MC) dropout is performed during training and inference. The implementation of the BALD criterion on image semantic segmentation is:

$$\begin{aligned} \mathbb{I}[y, \omega | x, \mathcal{D}_{train}] \approx & - \sum_c \left(\frac{1}{T} \sum_t p_c^t \right) \log \left(\frac{1}{T} \sum_t p_c^t \right) \\ & + \frac{1}{T} \sum_{t,c} p_c^t \log p_c^t \end{aligned} \quad (5.12)$$

where c is the number of classes, T is the number of committee members (models trained and inferred with MC dropout), and p_c^t is the softmax probability of the pixel. Examples of BALD distribution and queried unlabeled images are shown in Fig. 5.6.

5.2 Experiments

5.2.1 Datasets

All the images in the 3 datasets below were manually labeled by experts (attending surgeons and surgical residents). However, ground truth masks were hidden by default and were provided only when the images were in the test set or were marked as ‘labeled’.

UW-Sinus-Surgery-C/L Dataset [48] contains two parts: the live dataset (Sinus-Live) and the cadaver dataset (Sinus-Cadaver). For the Sinus-Live dataset, 3955 labeled images from the first two videos were used as the training set. And the third video contains 701 labeled images that were used as the test set. 696 background images were manually selected from the first two videos and

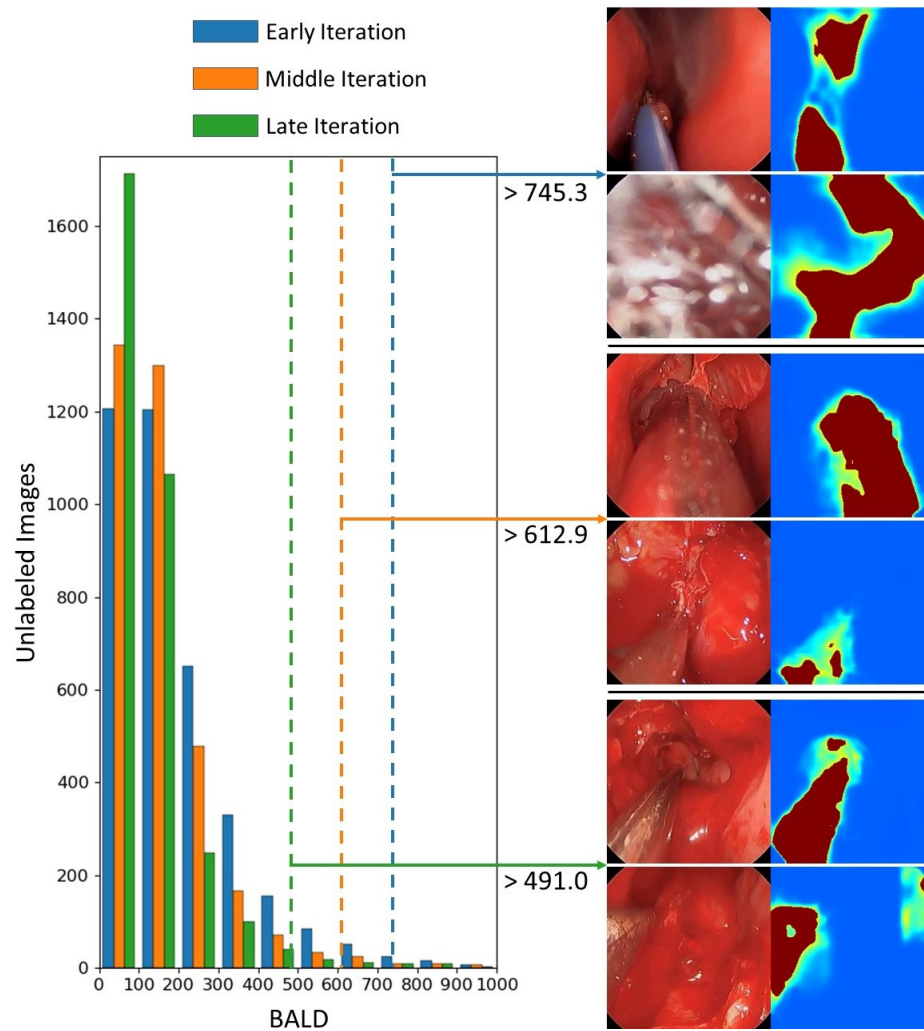


Figure 5.6: The distribution of BALD acquisition values on unlabeled images (left) and examples of queried images with the softmax probability masks (right) in early, middle, and late AL iterations. 395 real annotated images were used in the training and generation of synthetic images. The segmentation model was retrained for 70 epochs after each active learning iteration.

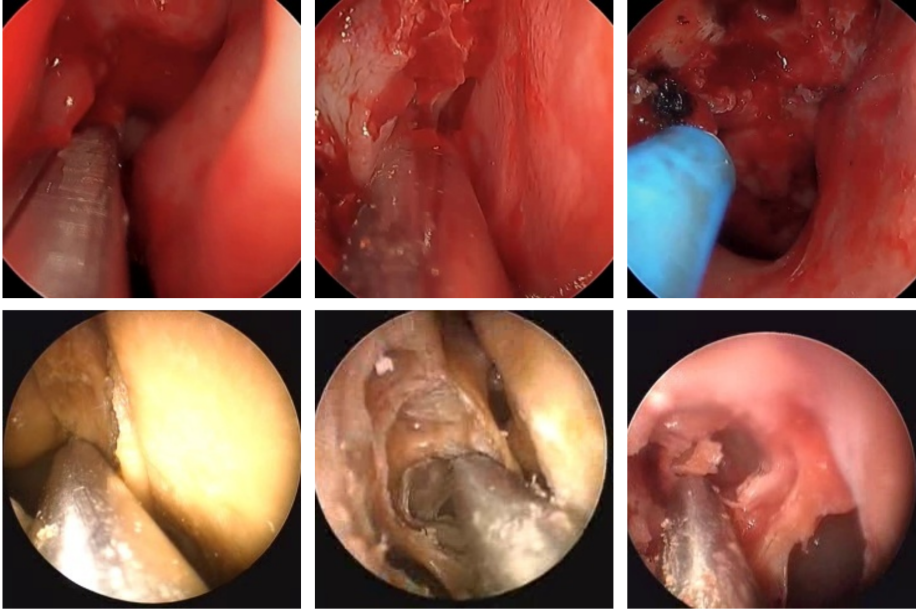


Figure 5.7: Example images from Sinus-Live (top) and Sinus-Cadaver (bottom) datasets.

provided to the system when external backgrounds are required. Manually choosing backgrounds was not as costly as annotating the segmentation of images. Three non-medical laypersons were asked to select 700 backgrounds out of 20000 images, which took 25 min, 17 min, and 31 min, respectively. All the images were resized and center-cropped to 240×240 .

The images in the test set and training set were from different videos of different surgical procedures. Because there were very few real images used in some experiments, non-informative images, such as pure black or white images caused by over-exposure or blocking, were manually removed from the training set. However, no image was removed from the test set to ensure that the performance was fairly evaluated.

The Sinus-Cadaver dataset was built similarly, collected from 10 surgery videos on 5 cadaver specimens. The training set, test set, and background set have 2908, 1437, and 597 images, respectively. Due to the different conditions of each cadaver specimen, the overall appearance of the images can be different. However, none of the recorded videos show considerable similarity to real surgeries. Humans have no difficulty visually distinguishing cadaver videos and live videos. Examples from Sinus-Live and Sinus-Cadaver can be found in Fig. 5.7.

EndoVis 2017 Robotic Instrument Segmentation Dataset is from one of the sub-challenges of

MICCAI 2017 [70]. The images were derived from 10 sequences of abdominal porcine procedures recorded using da Vinci Xi robotic endoscopic surgery systems. The instruments used include Large Needle Driver, Prograsp Forceps, Monopolar Curved Scissors, Cadiere Forceps, Bipolar Forceps, Vessel Sealer, and an ultrasound probe. The selected frames were labeled by a segmentation team at Intuitive Surgical. Although the videos were recorded by stereo cameras, only left-eye images were labeled. We used 900 images (225 for each video) with labels, from videos 1 - 4, as the training set. 900 images and labels from videos 5 - 8 were used as the test set. Due to the long training time of the active learning, the images were resized to 427×240 to reduce computational time.

5.2.2 *Parameters of generation of synthetic images*

The workflow of the generation of image syntheses is shown in Fig. 5.2. The parameters of the generation of synthetic images were empirically chosen and applied to all the experiments unless stated otherwise. (Type-1, Type-2) syntheses per selected real image were set to (2, 0) without multi-blending for the Sinus-Live and Sinus-Cadaver datasets, and (0, 1) with multi-blending for the EndoVis 2017 dataset. External backgrounds were only provided to the Sinus-Live and Sinus-Cadaver datasets, while background inpainting was only applied to the EndoVis 2017 dataset. The factor of tool resizing c was set to range [0.9, 1.2] for all datasets. The movement width w and height h were set to up to 24 pixels, with rotation θ up to 30 degrees. When blending instruments on backgrounds, the dilation kernel d was set to 15 pixels for all datasets, while the fusion blur kernel k was set to range [10, 15] pixels for Sinus-Live and EndoVis 2017, and range [5, 10] for Sinus-Cadaver. The factor of color adjustment α was set to range [0.4, 1.0]. A larger α results in a stronger adjustment and 0 means no adjustment. The range of brightness adjustment β was [0.9, 1.3]. The larger β , the brighter the adjusted image is, and 1.0 means no adjustment.

5.2.3 *Training details*

The segmentation model used in this work has the same structure as [48], unless stated otherwise. This is a modified DeepLabv3+ [124] encoder-decoder model with MobileNet [125] as the feature extractor. To fit in the active learning iterations, the learning rate was increased, and the training iterations were decreased significantly to accelerate the training, though the accuracy was slightly

compromised. Adam [114] was used as the optimizer, and the exponential decay rates of the 1st and 2nd order moment estimates were 0.9 and 0.999, respectively. The batch size was set as 16. Because we used different budgets of the hand-labeled training set in the experiments, for all the experiments in this work, the training iterations were 20 basic epochs plus a compensation of $5/\text{budget}(\text{percentage})$ epochs to ensure convergence when a small budget is used. For example, epochs were 25 (20+5) for 100% real images budget, and 30 (20+10) for 50% budget. The segmentation model was retrained for the epochs after each active learning iteration. The initial learning rate was set to 0.001 and exponential decay strategy was applied. The backbone lightweight MobileNet was pretrained on ImageNet [126]. Image augmentation was also applied to the training data for better generalization ability, which includes hue, brightness, saturation, contrast, flipping, rotation, zooming, and zero-padding.

In the beginning, all the images in the dataset were in the unlabeled pool and the labels were hidden. For example, if the real-image budget was 394 images (10% of the Live dataset), then 197 (half) real images were randomly chosen first, and the rest 197 images were chosen by the BALD criterion in 3 iterations. Once chosen from the unlabeled pool, the real images were moved to the labeled pool and their labels were revealed. Synthetic images were generated with labels naturally.

5.2.4 Evaluation metrics

Two main evaluation metrics were used in this work, Dice similarity coefficient (DSC) and intersection over union (IoU) [127], which are defined as

$$DSC = \frac{2|S \cap G|}{|S| + |G|}, IoU = \frac{|S \cap G|}{|S \cup G|}$$

where S is the foreground pixels of prediction, G is the corresponding ground truth, and $|*|$ is the counting operation. To study the effect of blending and fusion on the performance of segmentation near the boundary, IoU near boundary (IoU_{NB}) was used as an additional metric,

$$IoU_{NB} = \frac{|S \cap G| \cap B}{|S \cup G| \cap B}$$

where B denotes the near-boundary binary mask with width of 20 pixels band region near the instruments' boundary. The mean values of these three metrics are calculated over each test, denoted as mDSC, mIOU, and mIoU_{NB}.

5.3 Results

5.3.1 Usage of real images

The performance of the proposed method was evaluated when different annotation budgets were used. Budgets were set as proportions of the total real images in the training set as introduced in Section 5.2.1, from 1% to 100%. For each budget, 4 tests were performed - 1) randomly chosen training images without synthetic images, 2) BALD implemented - half chosen by BALD and the other half chosen randomly, without synthetic images, 3) randomly chosen real images with generated synthetic images, 4) BALD implemented with generated synthetic images. We also compared the proposed method with baseline results, which were obtained when the training budget was 100% with no synthetic images.

The evaluation results of segmentation performance are shown in Table 5.1 and method ablation on more budgets is shown in Fig. 5.8. Each entry is the average of 5 repetitive tests with different global random seeds to reduce the influence of randomness, which applies to all the experiments in this work. From Fig. 5.8, it can be seen that by generating synthetic images with active learning, the performance of segmentation was significantly improved when the budget was small. Compared with randomly chosen real images and no synthetic images, the average improvement of the proposed method in mDSC with less than 10% budgets was 5.31%, 8.15% and 3.41% in Sinus-Live, Sinus-Cadaver and EndoVis 2017 datasets, respectively. Specifically, the improvement on the Sinus-Live and Sinus-Cadaver dataset with only 1% budget was 6.67% and 12.20%. Overall, as the budget increased, the improvement became smaller. However, a considerable improvement could still be achieved when the budget was 100% of the training set (BALD was not applicable). The improvement in EndoVis 2017 dataset was not as significant, but the proposed method began to outperform the baseline result at only 10% usage of the training set.

Similar trends can be seen in mIoU and mIoU near the boundary. The average improvement of the proposed method with small budgets (less than 10%) was significant in mIoU - 7.11%, 10.02%, and 4.93% in the three datasets, as well as $mIoU_{NB}$ - 5.22%, 7.42% and 6.39%. Considerable improvements in mIoU could still be seen when the budget is 100% of the training set - 3.04%, 1.33% and 2.14%. However, for $mIoU_{NB}$, only the result of the Sinus-Live dataset showed improvement (4.52%), while the results of the other two datasets were similar to the baseline results.

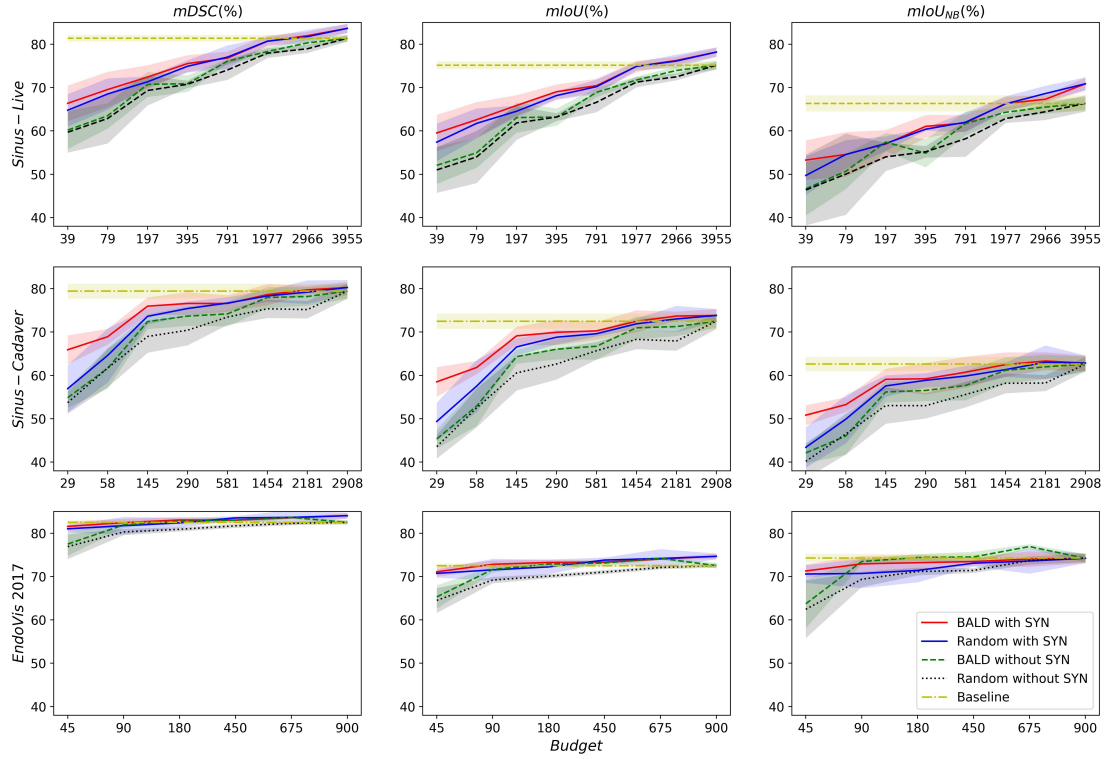


Figure 5.8: Evaluation of the model trained by different budgets of real images, with and without active learning (BALD) and synthetic images (SYN). The solid lines indicate the average over 5 repetitive tests and the shades indicate the standard deviation. The horizontal axes are nonlinear for better visualization. The training on the EndoVis 2017 dataset did not converge on the 1% or 2% budget because of too few images. Thus, the evaluation of this dataset started at 5%. The baseline result is obtained by training the model on 100% real images (no synthetic images), regardless of budget.

5.3.2 Number and type of synthetic images and multi-blending

As introduced in Section 5.1.2, there are 2 types of synthetic images. For each chosen and labeled real image, Type-1 synthetic images have the same instrument and Type-2 synthetic images have the same background as the real image. Multi-blending is also reported in [1] to avoid decreased performance caused by the artifact near the boundary in synthetic images. Thus, this experiment was performed to study the effectiveness of the 2 types of synthetic images and multi-blending. To better compare the results, the experiments were separated into 3 groups. Each group tested the same number of generated images per real image. For example, in Table 5.2, tests in Group 2 featured 8 synthetic images for each queried real image - $(2[Type - 1] + 2[Type - 2]) \times 2[Multi - blending]$. Consequently, tests in each group had the same number of training iterations to ensure that the model was trained for the same fixed number of steps. To ensure convergence, instead of keeping training iterations, the training epochs of Group 2 and 3 were the same as Group 1 so that the training iterations of Group 2 and 3 were 4 and 6 times larger compared to Group 1, respectively. The annotation budget was fixed at 10% of the training set.

The results are shown in Table 5.2. In Group 1, for the 2 sinus surgery datasets, the best performance on mDSC and mIoU was achieved by 2 Type-1 synthetic images, while the best performance on mIoU_{NB} was achieved by 1 Type-2 image with multi-blending on the Sinus-Live dataset, and by 1 Type-1 image with multi-blending on Sinus-Cadaver dataset. For the EndoVis 2017 dataset, 1 Type-2 image with multi-blending gave the best result on mDSC and mIoU, and 1 Type-1 image with multi-blending gave the best result on mIoU_{NB}. In Group 2, although the training steps were increased significantly compared to group 1, a decrease in performance could be seen in the Sinus-Cadaver and EndoVis 2017 datasets. Within the group, there was little difference in results with and without multi-blending on the Sinus-Live and EndoVis 2017 datasets. However, multi-blending increased the performance significantly in the Sinus-Cadaver dataset. A similar trend was observed in Group 3. Although no considerable difference was seen on mDSC and mIoU in Sinus-Live and EndoVis 2017 dataset by applying multi-blending, an improvement was observed on mIoU_{NB}.

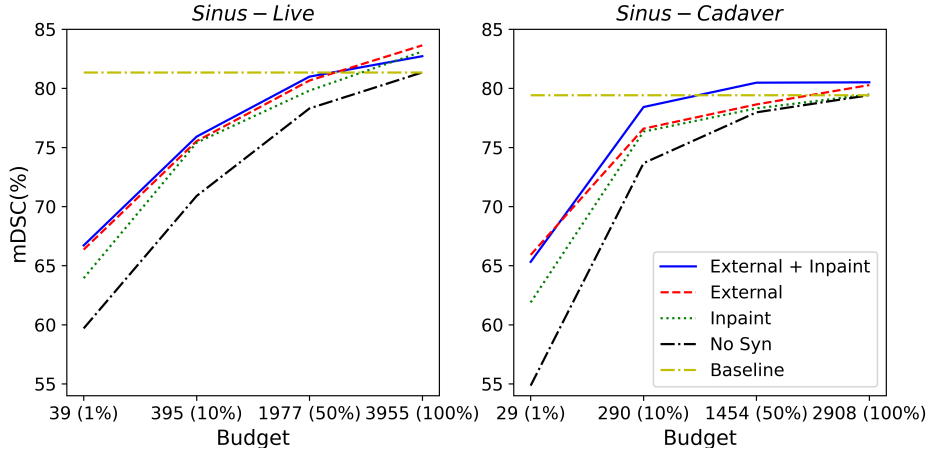


Figure 5.9: Segmentation result (mDSC) with different backgrounds. The baseline result is obtained by training the model on 100% real images (no synthetic images).

5.3.3 External backgrounds

For the proposed method, backgrounds can be generated by inpainting from labeled instrument images or be provided from external backgrounds. This experiment studied whether providing external backgrounds can help with the segmentation result. The tests were separated into 4 groups according to the budget of real images. In each group, there are 4 sub-tests. One test only used real images to train the segmentation model without synthetic images. The remaining 3 sub-tests were all supplemented with synthetic images (BALD implemented). The only difference was how the backgrounds were provided. Because no external backgrounds (frames without instruments) could be found in the EndoVis 2017 dataset, all synthetic images for this dataset had to be generated from inpainted backgrounds. Thus, only the two sinus datasets were used in this experiment.

The result in Fig. 5.9 (mDSC) shows that including external backgrounds improved the performance significantly when few real images were used (1% of the training set). For the Sinus-Live dataset, compared with no external backgrounds, the improvement was 2.77%, 2.89% and 1.93% in mDSC, mIOU and mIoU_{NB}, respectively. For the Sinus-Cadaver dataset, the improvement was 4.02% (mDSC), 4.86% (mIOU) and 1.74% (mIoU_{NB}). However, when a greater proportion of real images were used, the improvement was less considerable. For the Sinus-Live dataset, when 100% of the labeled real-image training set was used, the improvement was 0.52%(mDSC), 0.70%(mIOU)

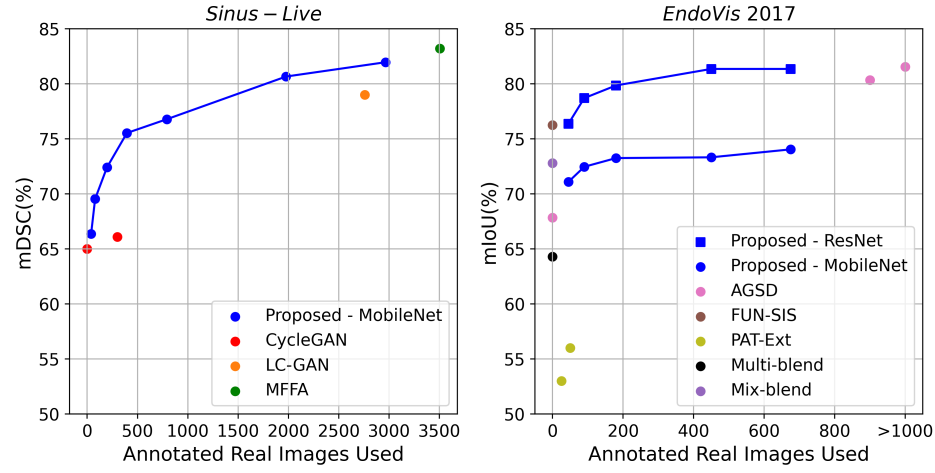


Figure 5.10: Comparison of segmentation performance with state-of-the-art approaches. Sinus-Live dataset: CycleGAN [2], LC-GAN [3], and fully supervised MFFA [4]. EndoVis 2017 dataset: AGSD [5], FUN-SIS [6], PAT-Ext [7], Multi-blending and Mix-blending [1, 8]. Evaluation metrics (mDSC or mIoU) are chosen based on the commonly used ones in the compared methods. Note that LC-GAN used annotated images from the cadaver dataset and FUN-SIS used annotated images from other existing datasets. Multi-blend and Mix-blend methods used 14180 auto-labeled instrument foregrounds to generate synthetic images. For the Sinus-Cadaver dataset, to the best of the authors' knowledge, no unsupervised, semi-supervised, or GAN-based learning methods were reported at present.

and 0.60%(mIoU_{NB}). But for the Sinus-Cadaver dataset, although improvement could still be seen in mDSC and mIOU with external backgrounds, a decrease in performance was found in mIoU_{NB}(-0.99%).

5.3.4 Comparison with state-of-the-art in 2023

Fig. 5.10 presents a performance comparison of our proposed method with state-of-the-art techniques. On the Sinus-Live dataset, the proposed method outperformed GAN-based approaches [2, 3] with less annotated data used. The proposed method also achieved close performance with fully-supervised learning [4].

ResNet50 [128] and MobileNet [125] were used as the backbone model to evaluate the perfor-

mance of the proposed method on the EndoVis 2017 dataset. With ResNet50 backbone, a small training set of 45 labeled images, and no other additional human efforts such as acquiring pseudo labels, our approach achieved competitive performance among the state-of-the-art unsupervised and semi-supervised learning approaches [5, 6, 7], as well as the generation of synthetic images [8]. Compared to the accuracy of 50% supervised learning using 900 labeled images reported in AGSD [5], our method achieved better accuracy using 450 labeled images, and had a close performance as the fully supervised training in AGSD. When using MobileNet as the backbone model, our proposed method did not achieve the best accuracy of segmentation. However, compared to ResNet50, this lightweight model had faster training and inference speed and could be suitable if efficiency is a priority or computational resources are limited.

5.3.5 *Realism of synthetic images*

The proposed method generates synthetic images by placing instrument sprites according to multiple parameters including scaling, translation, rotation, blending, and color adjusting. To generate various synthetic images, these parameters are randomly chosen in relatively wide ranges. Some of the parameters (particularly edge blending) can affect the level of image realism apparent to humans.

To study how human-perceived realism impacts learning performance, the realism of the synthetic images needed to be evaluated. Three expert sinus surgeons participated in this experiment and performed a review of images. 100 real and 200 synthetic images were randomly chosen and generated from the UW Sinus-Live dataset. These images were shown to the participants and they chose whether the image appeared 'real' or 'synthetic'. Thus, the realism of a synthetic image could be evaluated by how many times it was considered 'real' by an expert (Fig. 5.11). Examples of the synthetic images that were unanimously rated real (3 of 3) and not real (0 of 3) are shown in Fig. 5.12.

In the review results, 10, 31, 66, 93 synthetic images received 3, 2, 1, and 0 realism scores, respectively (Fig. 5.13). To study the effectiveness of different levels of realism in synthetic images, 4 training datasets were formed: 1) Baseline: only the 100 real images; 2) More realistic: 100 real images and 107 synthetic images with at least 1 'real' rating; 3) Less realistic: 100 real images and 107 synthetic images (93 with 0 'real' rating, and 14 with 1 'real' rating, randomly chosen to

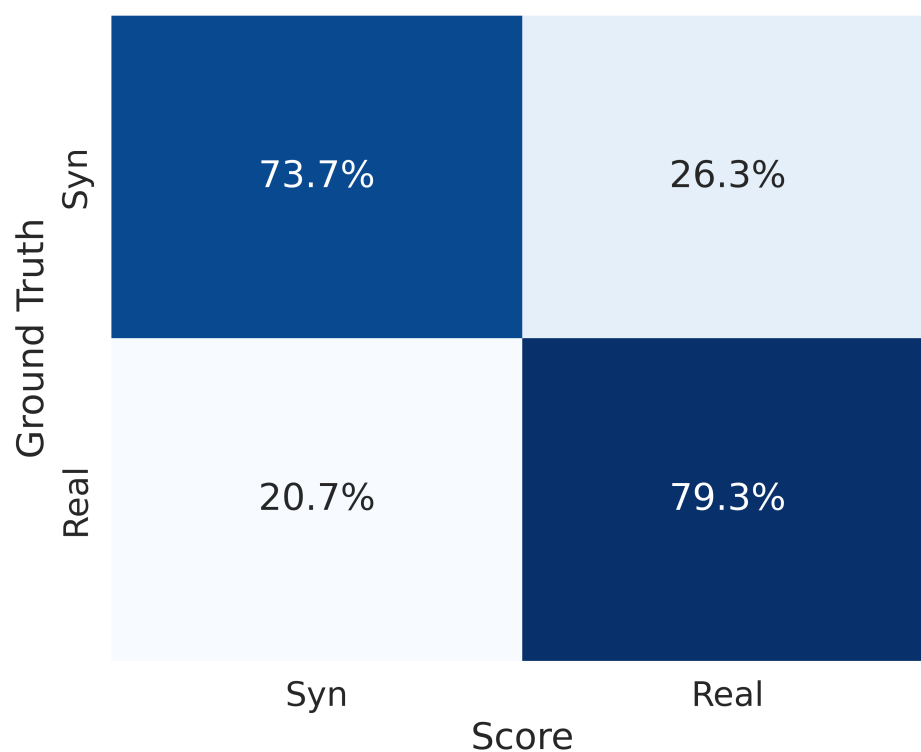


Figure 5.11: Confusion matrix of the image review result from 3 expert surgeons. 200 synthetic and 100 real images were reviewed. For each image, experts were asked to choose whether the image was more likely to be 'real' or 'synthetic'.

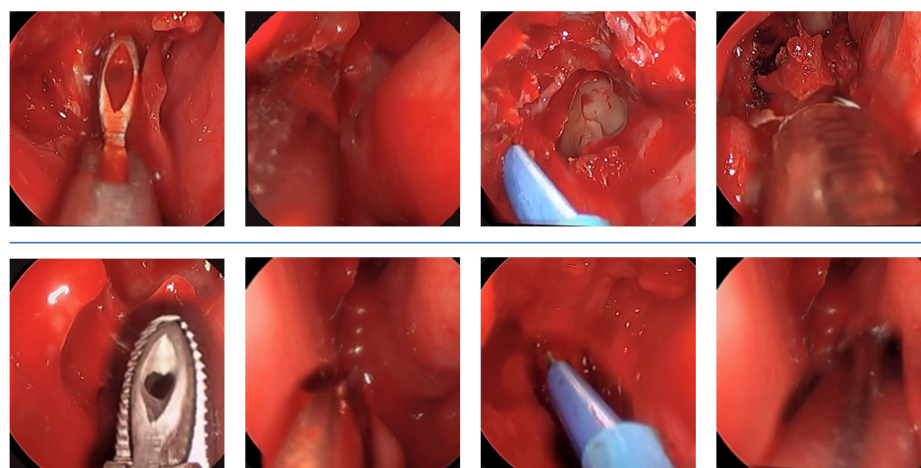


Figure 5.12: Examples of the most realistic synthetic images that received 3 out of 3 realism ratings (top), and less realistic synthetic images received 0 out of 3 realism ratings (bottom).

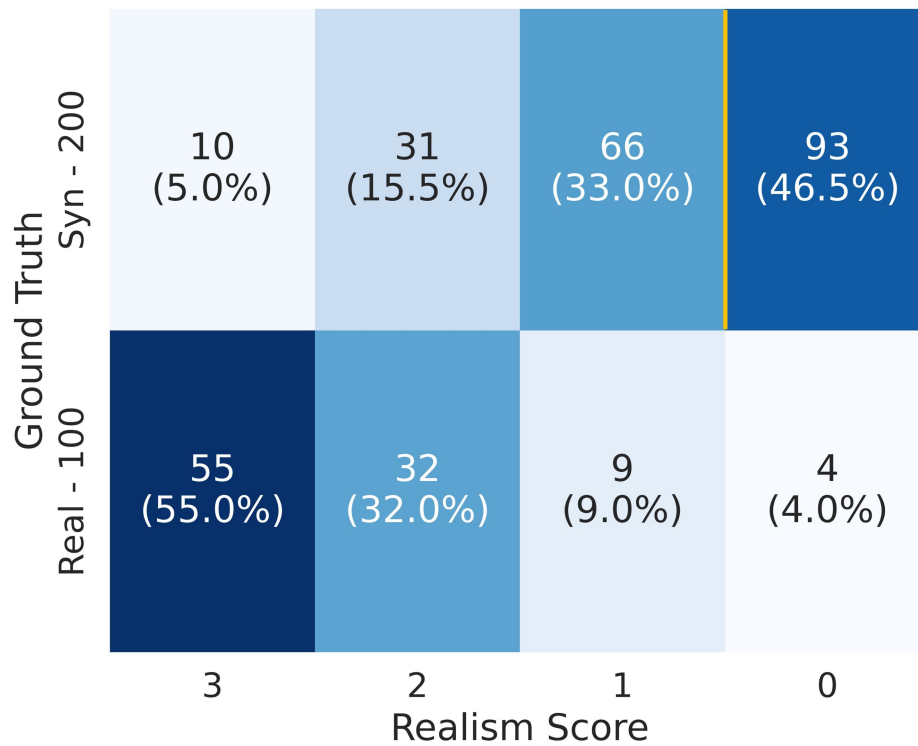


Figure 5.13: Realism ratings from the image review. Synthetic images with 1 or more realism ratings are considered 'more realistic', and otherwise 'less realistic', as the yellow line indicates.

compensate the size and thus the training iterations were the same); 4) All images: all 100 real image and 200 synthetic images.

The performance of models trained by these 4 training sets (Table 5.3) suggested that training with more realistic synthetic images or less realistic images had similar performance. The less realistic dataset slightly outperformed the more realistic dataset by 2.24%, 2.2%, and 2.47% in mDSC, mIoU, and mIoU_{NB}, respectively. Including all synthetic images achieved medium performance.

To further explore the impact of human-perceived realism on training effectiveness, feature extraction based on DINO [129] was performed on real images in the training and test set, as well as more and less realistic synthetic images. Augmentation was also applied, the same as the training of the segmentation models in this work (details in Section 5.2.3). Then a principal component analysis (PCA) was followed to reduce the extracted features to 2D for easier visualization, presented in Fig. 5.14. In this PCA subspace, less realistic images displayed a marginally greater variety than their more realistic counterparts. In Fig. 5.14, the average distance of each image in the test set to the closest image in the more realistic training set was 3.217, while the distance of the less realistic training set was 3.211. In comparison, this average distance for the training set with only real images was 3.353. Thus, this experiment suggested that although the proposed copy-and-paste method could generate human-perceived unrealistic images, these images could still contribute to the training and enhance the variety of the training data.

5.4 Discussion

In this work, we studied the use of image syntheses directed by active learning to improve the performance of surgical instrument segmentation. The results of the study on real image usage indicated that the proposed method improved the performance of segmentation significantly, especially when few real labeled images were used. With 10% real images budget combined with actively generated synthetic images, the performance of segmentation was comparable to using 50% of real images without synthetic images, cutting manual labeling effort by 80%. The performance of the proposed method was comparable with the baseline result (using 100% of the hand-labeled training set) when using only 50%, 75%, and 10% of hand-labeled data when evaluated on the Sinus-Live, Sinus-Cadaver and EndoVis 2017 datasets, respectively.

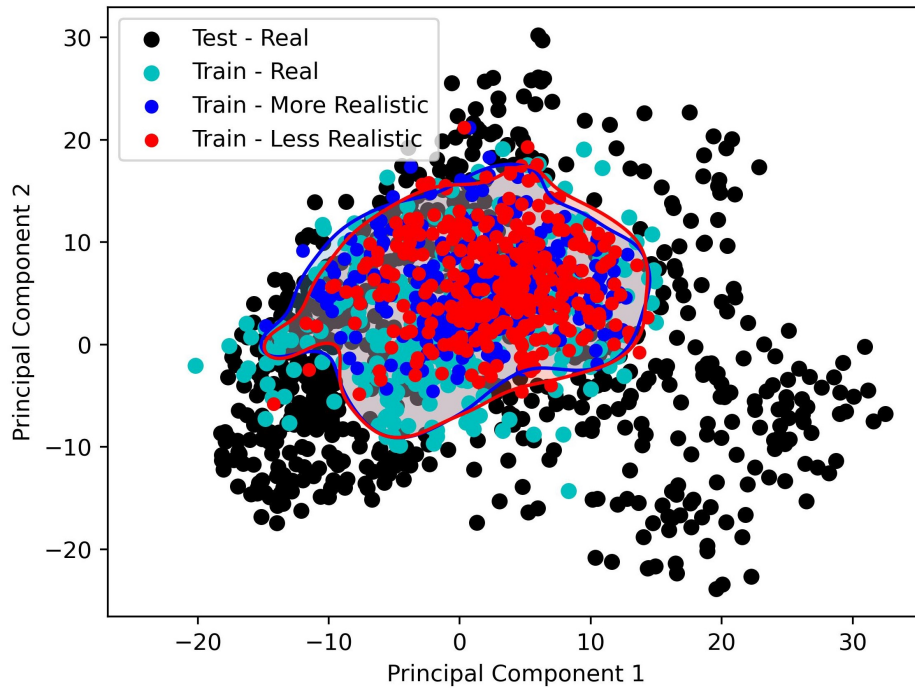


Figure 5.14: Principal component analysis on extracted features of real images in the training and test set, more realistic synthetic images, and less realistic synthetic images, corresponding to Table 5.3. Despite that less realistic synthetic images expand the training set slightly better than more realistic synthetic images, the distributions of the two image sets are close. This is notable considering their significant differences in human-perceived realism, as depicted in Fig. 5.12.

For the sinus surgery datasets, Type-1 synthetic images (the same instrument as the real image but with a different background) slightly outperformed Type-2 synthetic images (different instruments but with the same background). Although multi-blending made no major improvement in overall performance, it improved the segmentation near the boundary. In addition, too many synthetic images were observed to decrease the performance (Group 3 of Table 5.2).

We varied the method of generating background images between inpainting the space occupied by instruments, and using original video frames that contained no instruments (external backgrounds). The use of external backgrounds improved the performance significantly when the annotating budget was extremely small, without adding much workload. Because manually selecting background-only frames from surgical videos requires no expertise and is not as time-consuming as labeling the instruments, especially when different parts of the instruments have different labels, adding external backgrounds is an efficient way to improve the segmentation with small numbers of labeled images.

5.5 Conclusion

Motivated by alleviating the experts’ workload of annotating for instrument segmentation in endoscopic images, we propose the use of actively generated synthetic images to reduce the need for labeled real images while having comparable performance. When training segmentation models, the proposed method selects the most informative unlabeled images, then annotates these images and generates synthetic images based on the selected real images. Thus, a more diverse training set is formed by labeled real images and synthetic images, which results in considerable improvement in performance compared with using real images only, especially when the budget for annotating “new” images is small. The proposed method utilizes and combines active learning and generation of synthetic images to reduce the usage of real labeled images, and can be flexibly applied to different segmentation models and datasets, with different active learning criteria.

The current framework is limited to binary segmentation. In the future, we plan to generalize the proposed method to semantic segmentation among different instruments and parts. We also plan to further explore the relationship between realism to humans and effectiveness to artificial intelligence. There are many factors affecting the realism of synthetic images, and the influence of

these factors on machine learning performance remains to be discovered.

Table 5.1: Segmentation Performance with Different Budgets

Budget	Proposed	Performance(%)									
		Sinus-Live				Sinus-Cadaver				EndoVis 2017	
		BALD	Syn	mDSC	mIoU	mIoU _{NB}	mDSC	mIoU	mIoU _{NB}		
5%	×	×	69.31	61.83	54.00	68.97	60.57	53.04	76.89	64.50	62.44
	✓	✓	72.41	65.86	56.96	75.96	69.12	59.09	81.56	71.09	71.28
20%	×	×	74.05	66.59	58.19	73.45	65.70	55.58	80.90	70.10	71.16
	✓	✓	76.78	70.43	61.82	76.58	70.27	60.73	83.05	73.25	73.20
50%	×	×	77.93	71.26	62.83	75.36	68.31	58.18	81.70	71.04	71.40
	✓	✓	80.66	74.87	66.30	78.64	72.49	62.49	83.01	73.32	73.45
100%	×	×	81.35	75.14	66.34	79.42	72.50	62.61	82.50	72.52	74.28
	×	✓	83.64	78.18	70.86	80.28	73.83	62.84	84.04	74.66	74.21

*i) The bold font indicates the best performance in each budget. ii) BALD active learning is not applicable when the budget is 100%.

Table 5.2: Segmentation Performance with Different Types of Synthetic Images and Multi-blending

Group	Syn per Real Image	Performance(%)								
		Sinus-Live			Sinus-Cadaver			EndoVis		
		Type-1	Type-2	M-blend	mDSC	mIoU	mIoU _{NB}	mDSC	mIoU	mIoU _{NB}
1	No synthetic image				71.70	64.19	55.47	69.35	61.81	52.76
	1	1	1		74.74	68.24	59.34	73.32	66.89	56.15
	0.5	0.5	2		74.64	68.09	60.54	74.30	67.80	58.31
	2	0	1		76.97	70.47	61.65	76.59	69.93	59.22
	1	0	2		74.75	68.50	60.16	75.29	68.45	59.24
2	0	2	1		75.07	68.67	60.63	71.90	65.77	55.69
	0	1	2		76.80	70.38	63.23	72.85	66.44	56.34
	4	4	1		78.03	72.23	62.96	70.86	65.33	54.84
3	2	2	2		77.57	71.89	62.91	76.09	70.32	59.67
	6	6	1		77.42	71.73	61.95	69.90	64.46	54.21
	3	3	2		77.91	72.15	62.79	74.90	69.45	58.91

*The bold font indicates the best performance in each group. Under Syn per Real Image, the parameters of Type-1 and Type-2 indicate that for each labeled real image selected by the active learning mechanism, how many Type-1 and Type-2 synthetic images were generated, respectively. A value of 0.5 means that there is a 50% chance to generate a synthetic image. M-blend value of 1 means that each synthetic image is single and multi-blending is not applied.

Table 5.3: Segmentation Performance with Different Syntheses Realism

Dataset	Images		Performance(%)		
	Real	Syn	mDSC	mIoU	mIoU _{NB}
Baseline	100	0	62.51	55.89	48.35
More Realistic	100	107	68.04	61.15	55.22
Less Realistic	100	107	70.28	63.35	57.69
All	100	200	68.60	62.30	57.08

Chapter 6

DISTAL FORCE SENSING AND FORCE ACTUATION SYSTEM

Difficulty in haptic feedback for surgical robots has been a long-term problem for decades. In recent years, learning-based force estimation from robot states suggests desirable accuracy without the necessity of extra sensors. However, challenges remain in obtaining representative training data in which the robot moves in the workspace under various external forces. In this work, a parallel motor-cable system is developed. With six motor-cable units installed around the robot workspace, cables with controllable tension connected to the robot end-effector can provide the desired external force without interfering with the movement of the surgical robot. The development of the system includes motor-unit hardware, control software, sensor drivers, simulations, and more. Preliminary experiments suggest an accuracy of force estimation with errors less than 1 N.

6.1 Method

Overall, the force actuation system has 6 motor-cable units installed around the Raven-II surgical robot, shown in Fig. 6.1. All cables are connected to the end-effector of Raven-II from various directions. By controlling the motor torques and adjusting the cable tensions, the system can apply forces of arbitrary direction and magnitude to the end-effector of Raven-II.

6.1.1 Hardware

As shown in Fig. 6.1 (right), each motor-cable unit consists of the following parts: 1) a DC motor outputting torque, 2) a cable reel converting motor torque to cable tension, 3) a load cell sensing the cable tension for feedback, 4) a fixing frame that points the measurement direction of the load cells to the center of the robot workspace to make the best use of the resolution.

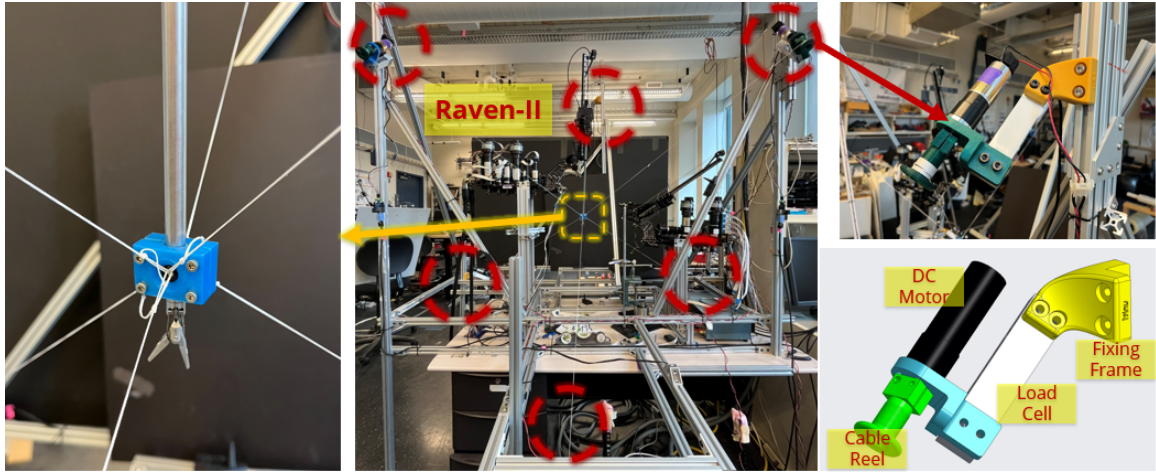


Figure 6.1: The hardware and setup of the force actuation system. There are 6 motor-cable units (marked by red circles) installed around the robot workspace, and all cables are connected to the robot end-effector. By adjusting the tensions on cables, different forces can be applied to the end-effector.

6.1.2 Control Software

The control software is based on Robot Operating System (ROS) and is compatible with the Raven-II software controller. As shown in Fig. 6.2, the control loop consists of the following major procedures: 1) The system receives the force control command from the high-level controller that controls both Raven-II movement and force actuation. 2) Based on the motor locations and the current end-effector position from the robot state, the direction of each cable is computed. 3) Based on the force command and cable directions, SLSQP optimization is used to find the desired tension on each cable. A minimum cable tension is maintained all the time for the smoothness and stability of force actuation. There are 2 levels of feedback control. The higher-level feedback control is on the force command, based on the difference between the desired force and the actual applied force. The lower-level feedback control is on the cable tension, based on the difference between the desired cable tension and the actual cable tension measured by load cells. The feedback from the load cells is essential because the motors have considerable friction, which causes different cable tension output under the same motor torque command, when the motor rotates in different directions, as shown in

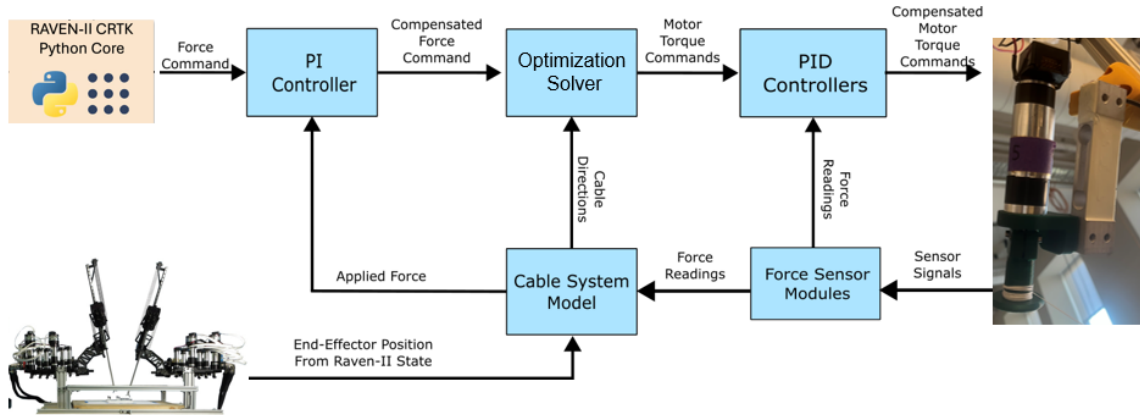


Figure 6.2: The control workflow of the force actuation system. The higher-level feedback control is on the force command, while the lower-level feedback control is on each single cable tension.

Fig. 6.3.

6.1.3 Associated Electronics

The electronics of the force actuation system are shown in Fig. 6.4. The load cell drivers are able to provide measurements at 320 Hz for the feedback control of the cable tensions, while the conventionally used HX711 breakout board can only achieve 80 Hz. The Ethernet cable needs to be shielded and grounded. Otherwise, the I2C communication can be interrupted when the motor power is on. Currently, the motor control is achieved by the right arm of the Raven-II robot, with modified control software. Future work includes separating the motor control and the low-level tension control into independent motor drivers and microcontrollers. Details of the associated electronics can be found in [130].

6.1.4 Simulation of the System

For the accuracy of the force actuation system, it is critical to find proper motor locations, so that each cable tension will not exceed the limit to provide desired forces, and cables do not interfere with the robot and other obstacles. To find desirable motor locations, a simulation of the force actuation system is developed, shown in Fig. 6.5. In simulation, bounding boxes of the movement of the

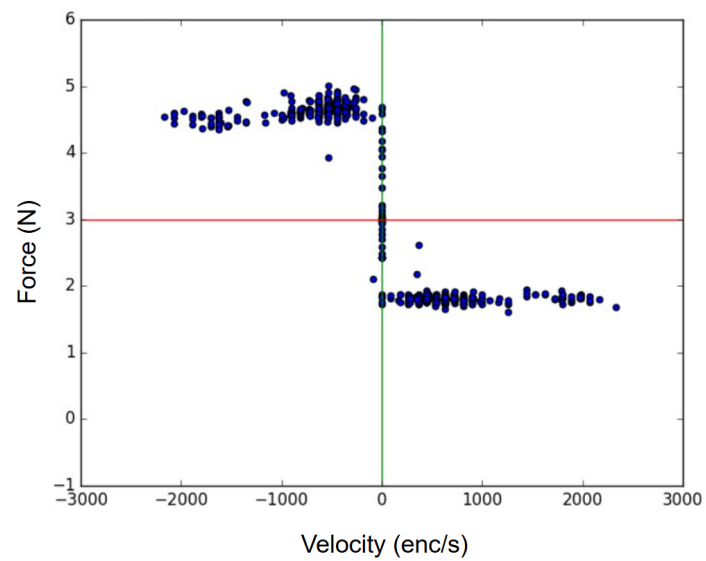


Figure 6.3: Cable tension output under the same motor torque command but different directions of motor rotation. If the motor rotation and torque have the same direction, the actual cable tension is smaller than the command. If the motor rotation and torque have opposite directions, the actual cable tension is larger than the command.

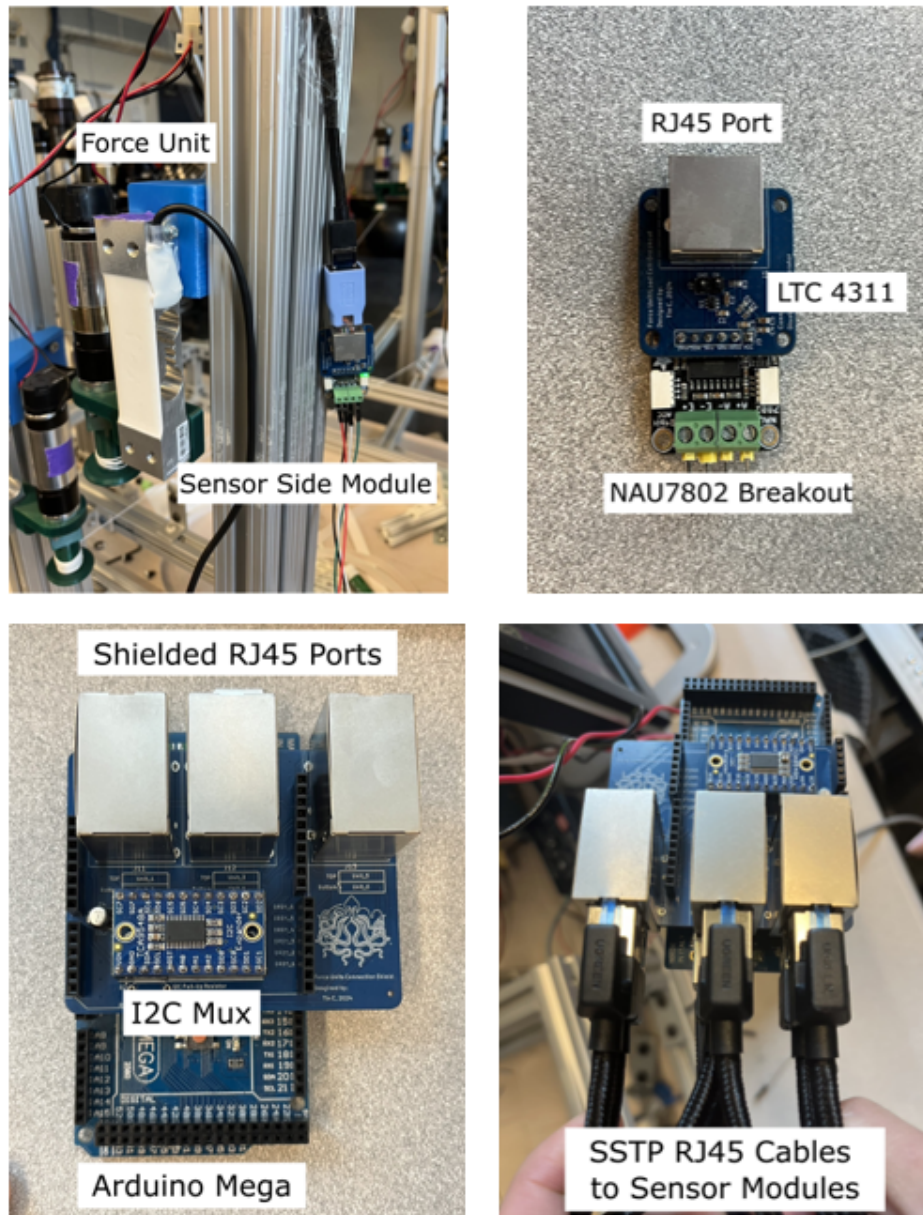


Figure 6.4: 320 Hz load cell drivers for feedback control on cable tensions.

robot arm links are created to check the interference of cables against the robot. And the Monte-Carlo simulation of the robot end-effector workspace is also developed to analyze the maximum cable tension needed for the desired range of force actuation. Motor positions closer to the robot workspace have benefits of a smaller influence of cable elasticity, but the cable direction can have larger changes due to robot movement and results in larger cable tensions. In addition, a Monte-Carlo simulation of the workspace of Raven-II is also developed, as shown in Fig. 6.6.

6.1.5 *Localization of Motor Units*

In order to find the cable directions in the control loop, the motor units' locations must be known in the coordinate frame of the robot. MicroScribe MX is used to measure the position of the motor units, however, the position is represented in the device's own frame. To obtain the transformation between the robot frame and the measurement device frame, with the base of the measurement device fixed, several points on the robot base with known positions in the robot frame are also measured. Next, with the same points but positions represented in the robot frame and device frame, Kabsch algorithm is used to obtain the transformation between the two frames. Details of the motor localization are shown in Fig. 6.7 and Fig. 6.8.

6.1.6 *Force Estimation with Local Motion*

With the developed force actuation system, data with both robot movement and external forces can be obtained, which can be used to train a neural network that can estimate the external force with no extra sensor, as the force actuation system can be removed after training. Similar to the learning calibration described in previous chapters, to improve the efficiency of the learning model, not all robot states are used as the input of the model. Instead, joint positions, motor torque, and end-effector pose are chosen as input, as these features are theoretically related to the contact force on the end-effector.

Raven has relatively large joint frictions, due to cable tension and coupling. According to initial tests, it is found that if Raven is static (no motion command and not moving), the external force can be fully passively resisted by the joint frictions and thus cause no change in the motor torque. This results in all input features staying constant under various external load. In other words, the input

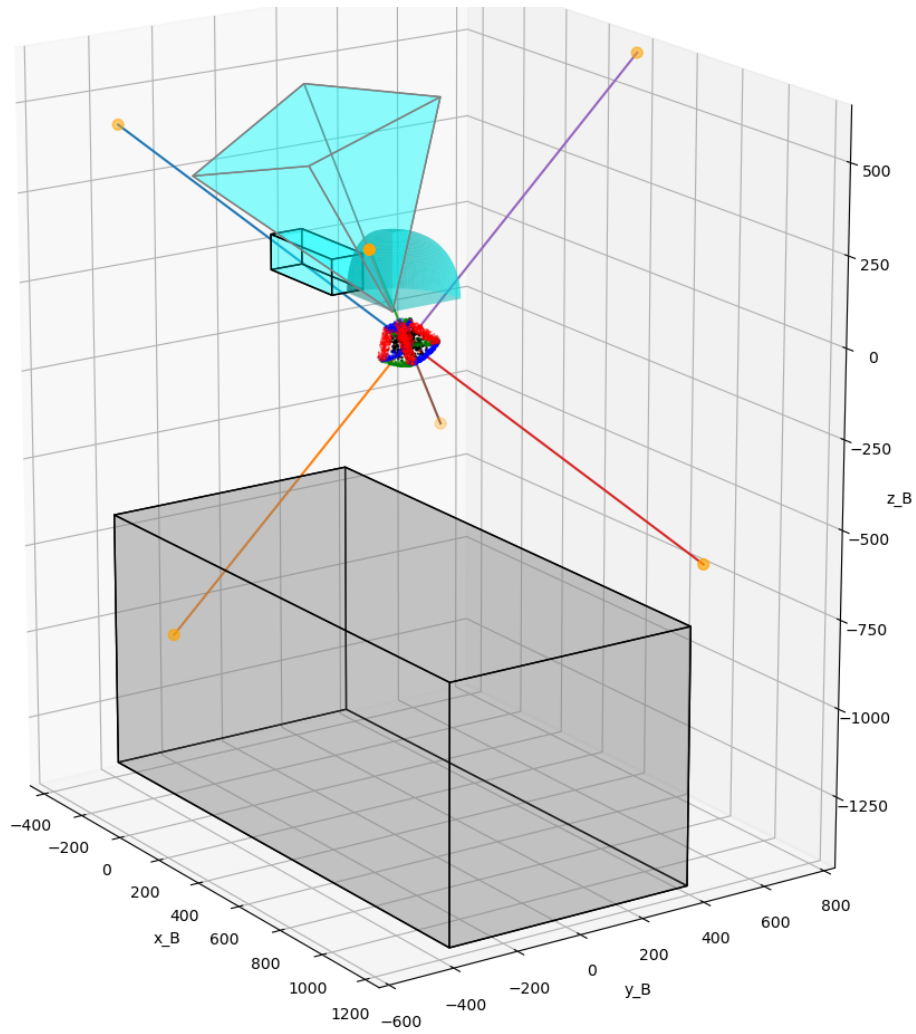


Figure 6.5: Simulation of the force actuation system with the bounding boxes of Raven-II surgical robot and its workspace.

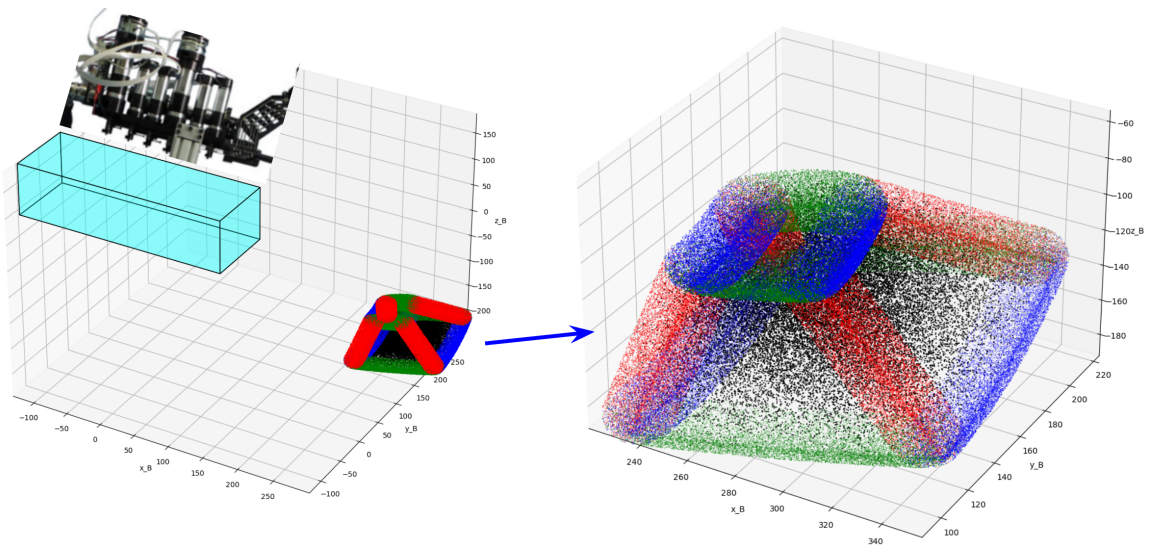


Figure 6.6: Monte-Carlo simulation of the workspace of Raven-II. The blue box is approximate size of the robot base as a size of reference. For better visualization, the blue, green, and red edges are generated by having joint 1, 2, and 3 as variable, respectively, while setting the remaining 2 positional joints at maximum or minimum.

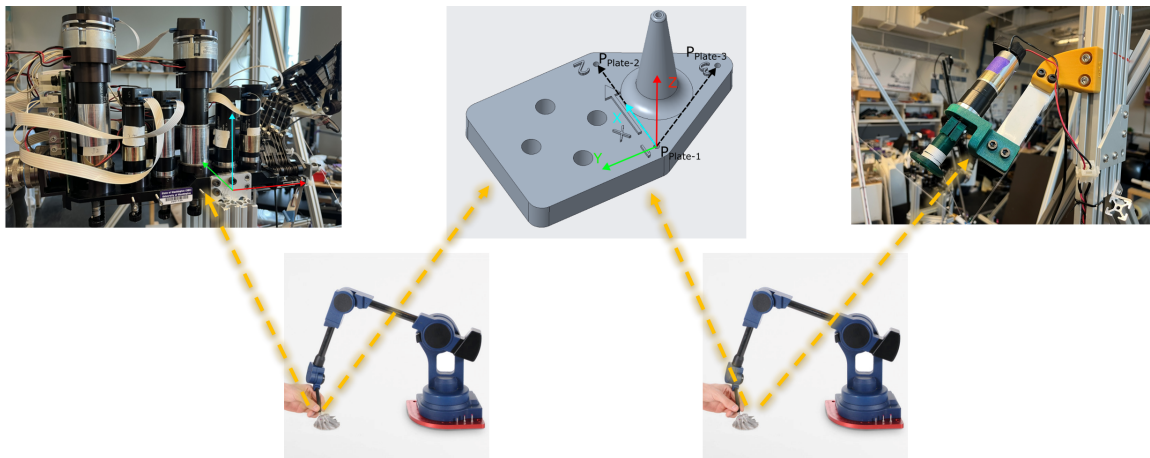
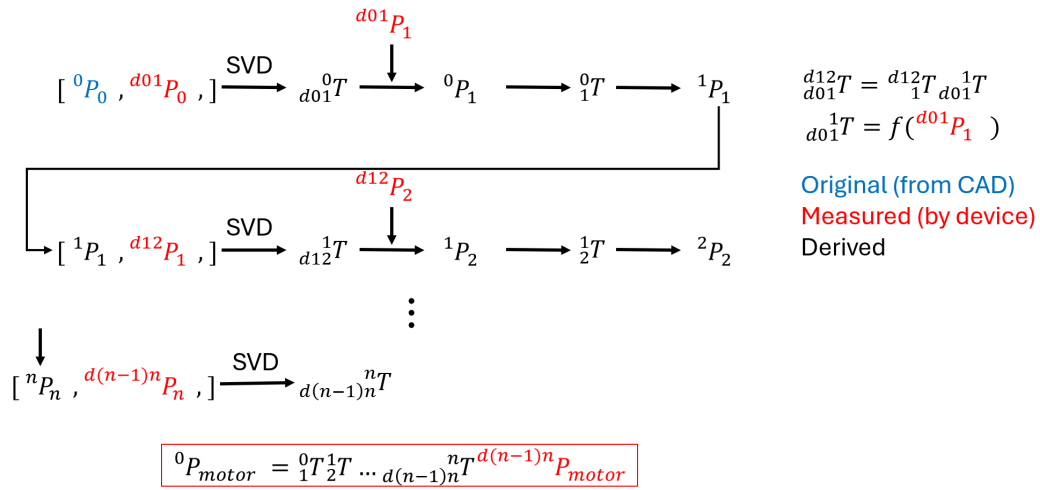


Figure 6.7: Measurement of motor locations with intermediate frame reference plates. The MicroScribe MX device and the frame reference plates can be located at any convenient locations, as long as the locations are fixed while obtaining measurements of reference points. By measuring known reference points on the robot base, the frame of MicroScribe can be obtained using the Kabsch algorithm. If the motor is far away from the robot base so that the MicroScribe cannot reach the robot base and motor unit at the same time, the frame reference plates can be used to extend the measurement range. Multiple frame reference plates can be used to further extend the range.



information cannot reflect the change of the external load and the robot cannot "feel" the force. To overcome this, a small local motion is added when robot is static. The local motion is around 2° , 2° , and 2 mm for joint 1, 2, and 3, respectively. The comparison of motor torque without and with local motion is shown in Fig. 6.9 and Fig. 6.10.

6.2 Experiments and Preliminary Results

6.2.1 Experiments

With the force actuation system, I recorded a dataset where the robot followed trajectories with both position and external force for the training of neural network models, shown in Fig. 6.11. Statistical analysis suggested that compared to the desired external force, the actual applied force had mean absolute errors of 0.80 N, 0.79 N, and 0.44 N in X, Y, and Z direction, respectively.

Data set 1: To verify if the local motion is necessary, two training sets were recorded, one with local motion and one without local motion. Then a test set with both moving and static status of robot was also recorded.

Data set 2: To train the neural network and evaluation the estimation of the external contact force, another training dataset of 2 hours were recorded, in which 50% of time is static while the robot kept moving randomly in the rest of time. It is worth noticing that the static status means that the robot was not moving, while the external force provided by the force actuation system kept changing. After training data is collected, multiple hours of testing data was also collected with random movements and forces, and it also had 50% mix of static and moving status.

6.2.2 Preliminary Results

Data set 1: The performance of estimation without and with local motion can be found in Fig. 6.12 and Fig. 6.13, respectively. It can be observed that without local motion, the neural network model could not effectively estimate the contact force because the external forces were passively resisted by joint frictions and could not be reflected in the robot states. As comparison, with local motion, the external forces could be estimated with desirable performance.

Data set 2: The examples sections of force estimation in moving and static status are shown in Fig. 6.14 and Fig 6.15, respectively. In moving status, with 2 hour training data, the MAE of

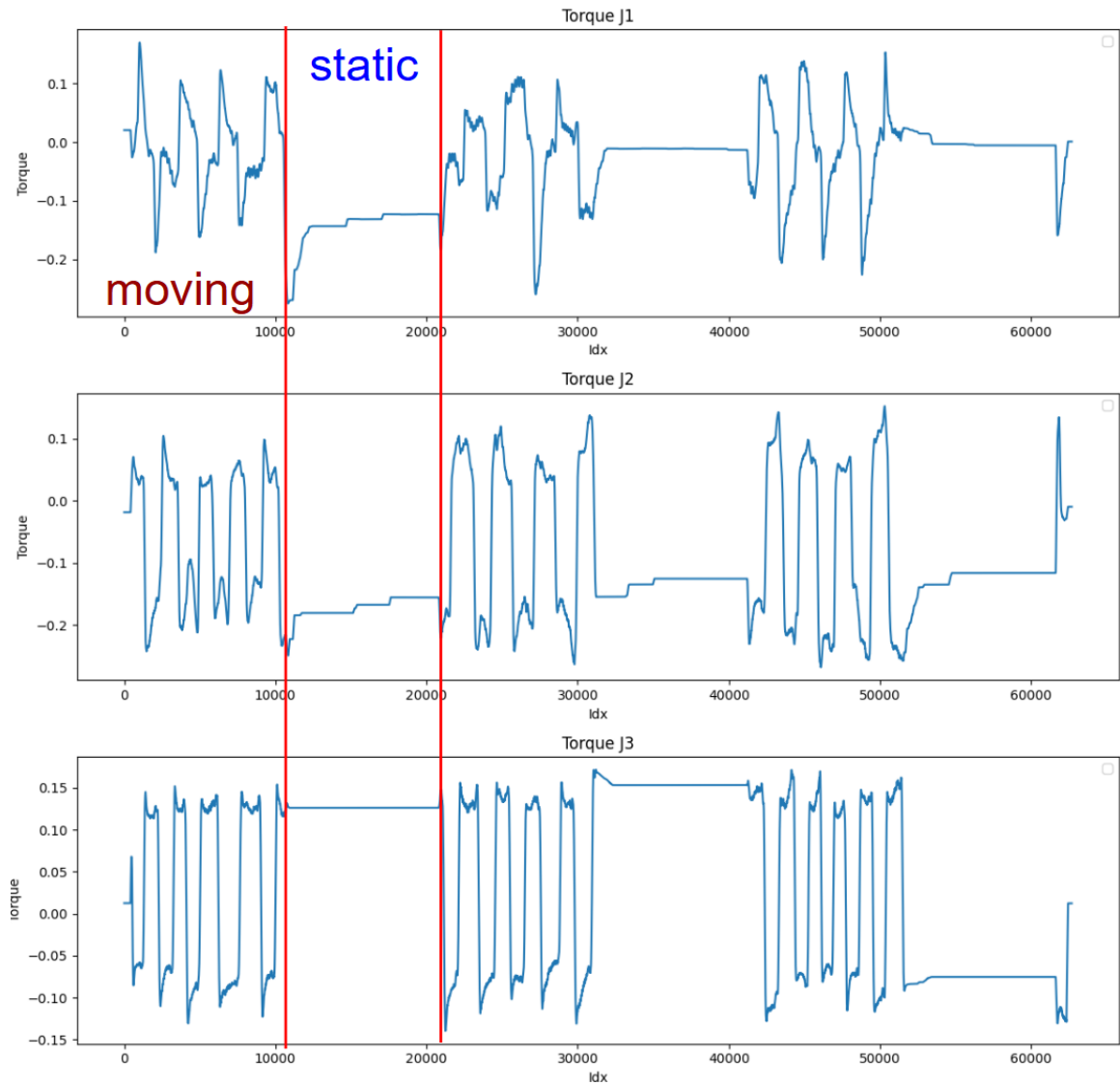


Figure 6.9: Joint torque in robot state without local motion when external forces apply. Although various external forces are still applied in the static status, the joint torques have no considerable changes, because the external forces are passively resisted by the joint frictions.

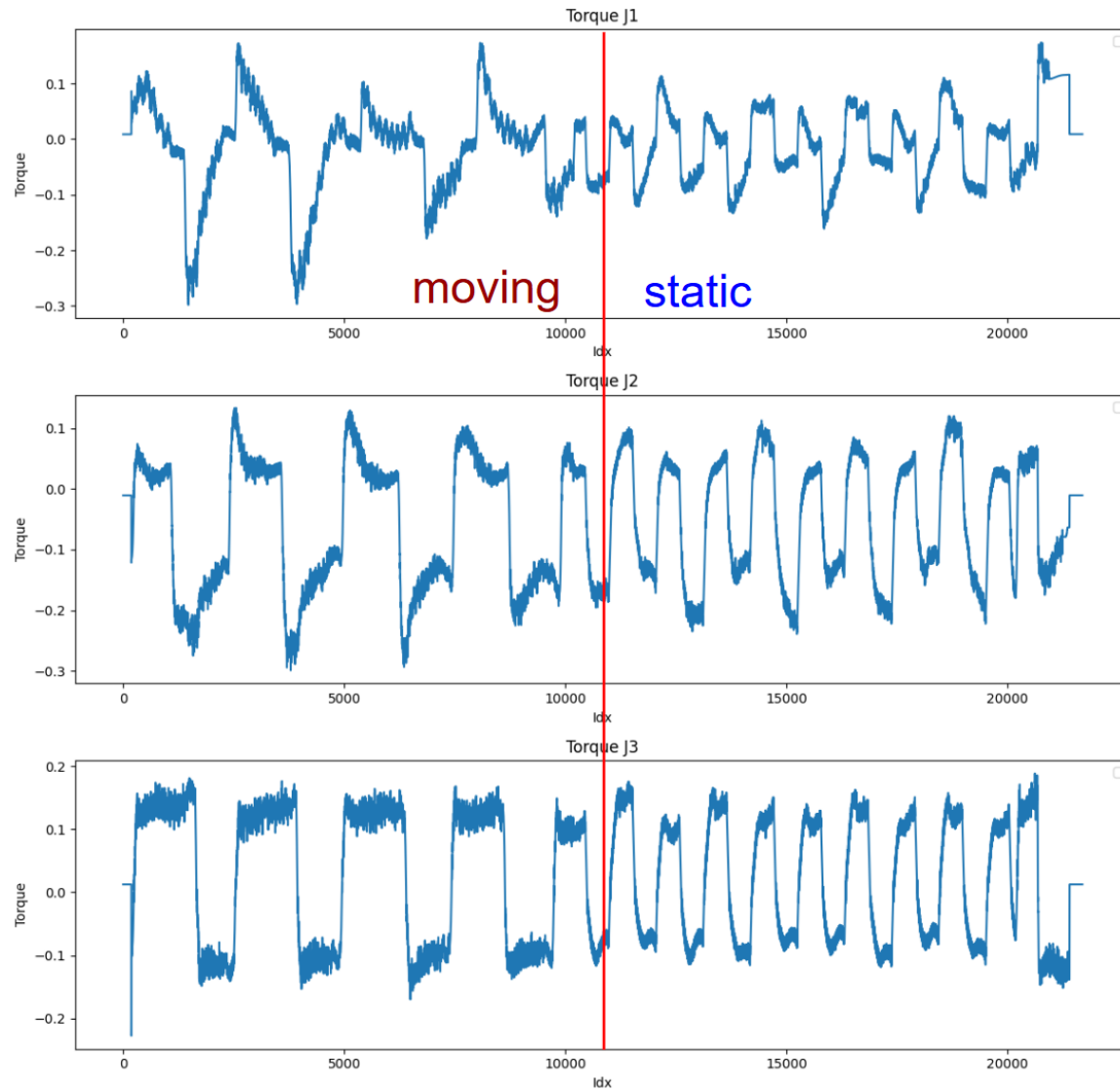


Figure 6.10: Joint torque in robot state with local motion when external forces apply. Compared with Fig. 6.9, changes in joint torques can be observed in both moving and static status. However, it should be noticed that the joint torques in the static status are caused by both local motion and external force.

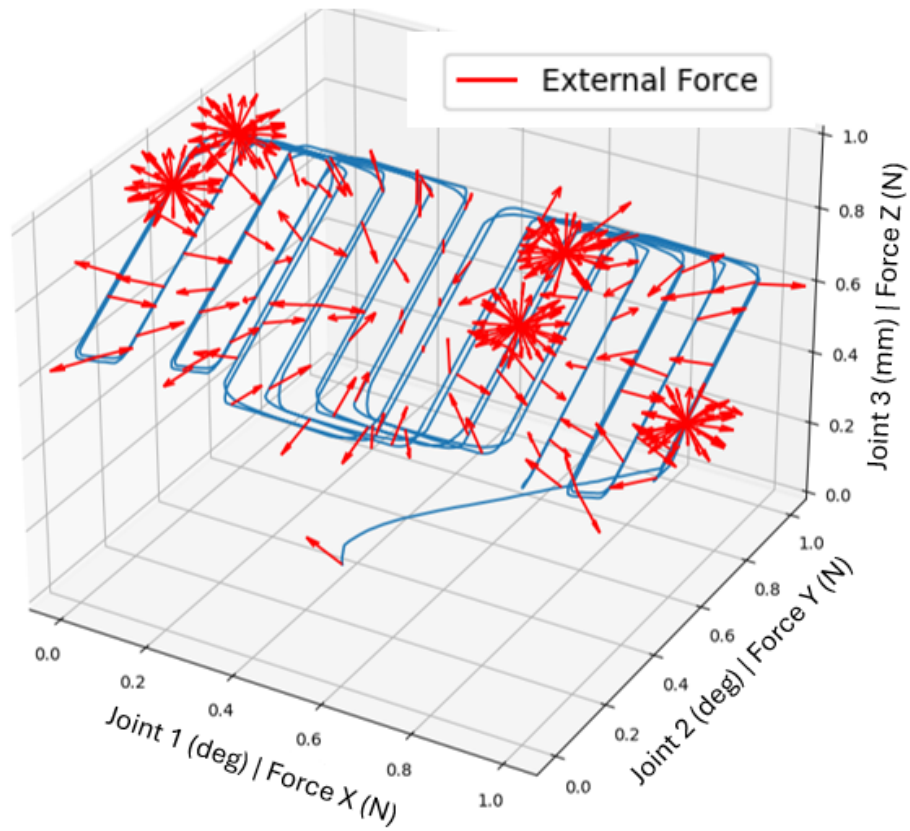


Figure 6.11: Recorded robot trajectory with external force. The "stars" are for studying the static condition of the robot, in which the robot is not moving while the external force keeps changing. Both position and force are normalized by their ranges for better visualization.

force estimation in component of x, y, and z were 0.43 N, 0.41 N, and 0.41 N, respectively. In static status, the MAE of force estimation in component of x, y, and z were 0.49 N, 0.35 N, and 0.36 N, respectively.

6.3 Conclusion

In this work, a parallel motor-cable force actuation system is developed to apply desired external forces to the Raven-II surgical robot. The system includes motor-cable unit hardware, control software compatible with Raven-II software controller, sensor electronics, simulation, and motor localization. With the system, Raven-II trajectories with external forces are recorded. The force actuation system can be further improved by better smoothness and stability in control, as well as separating motor drivers and MCUs to be independent from the Raven-II system.

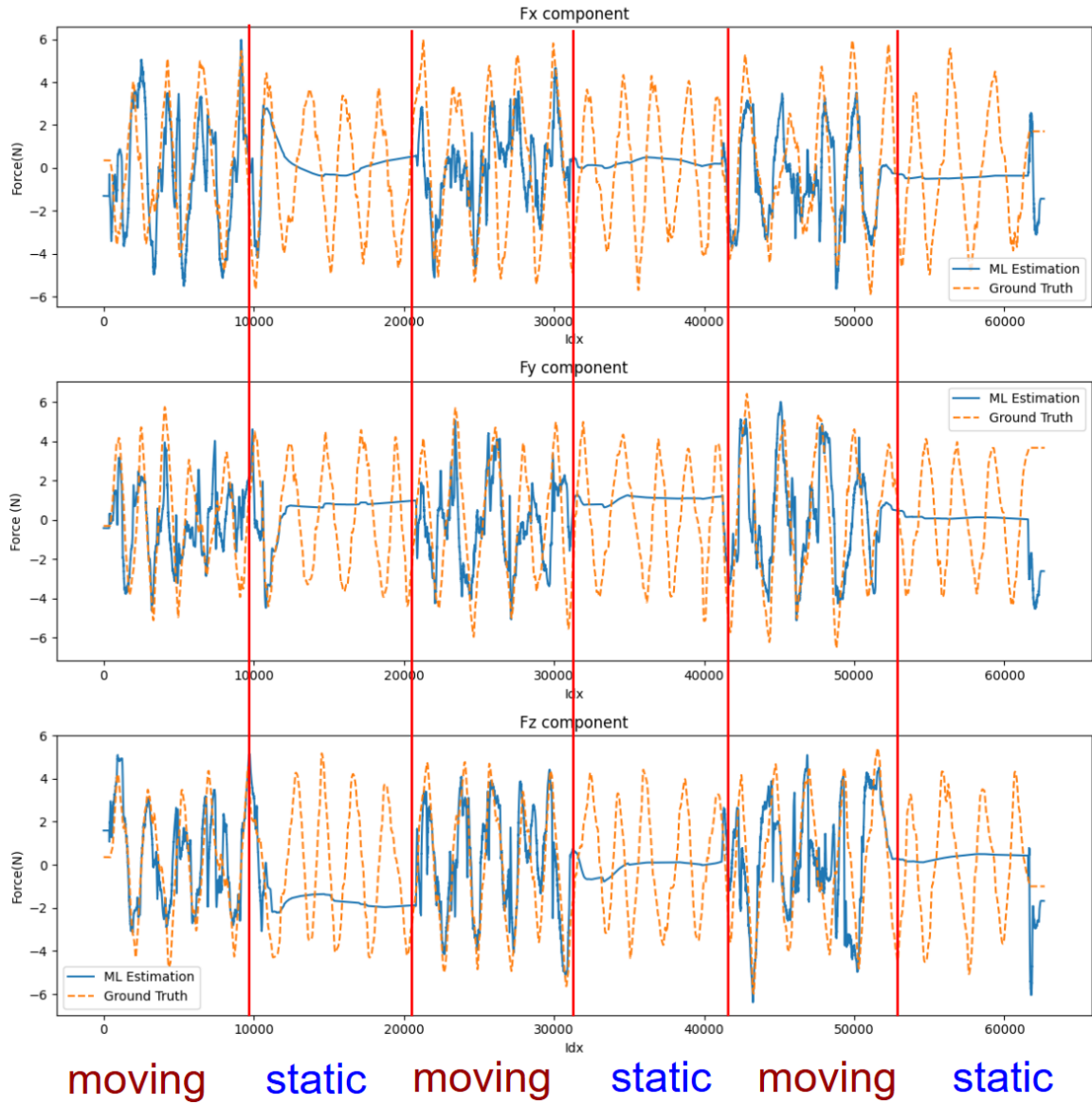


Figure 6.12: Example of force estimation performance when the robot has no local motion in static status. Without local motion, the neural network model cannot estimate the external force in a static status because external forces are not reflected in joint torques, as shown in Fig. 6.9.

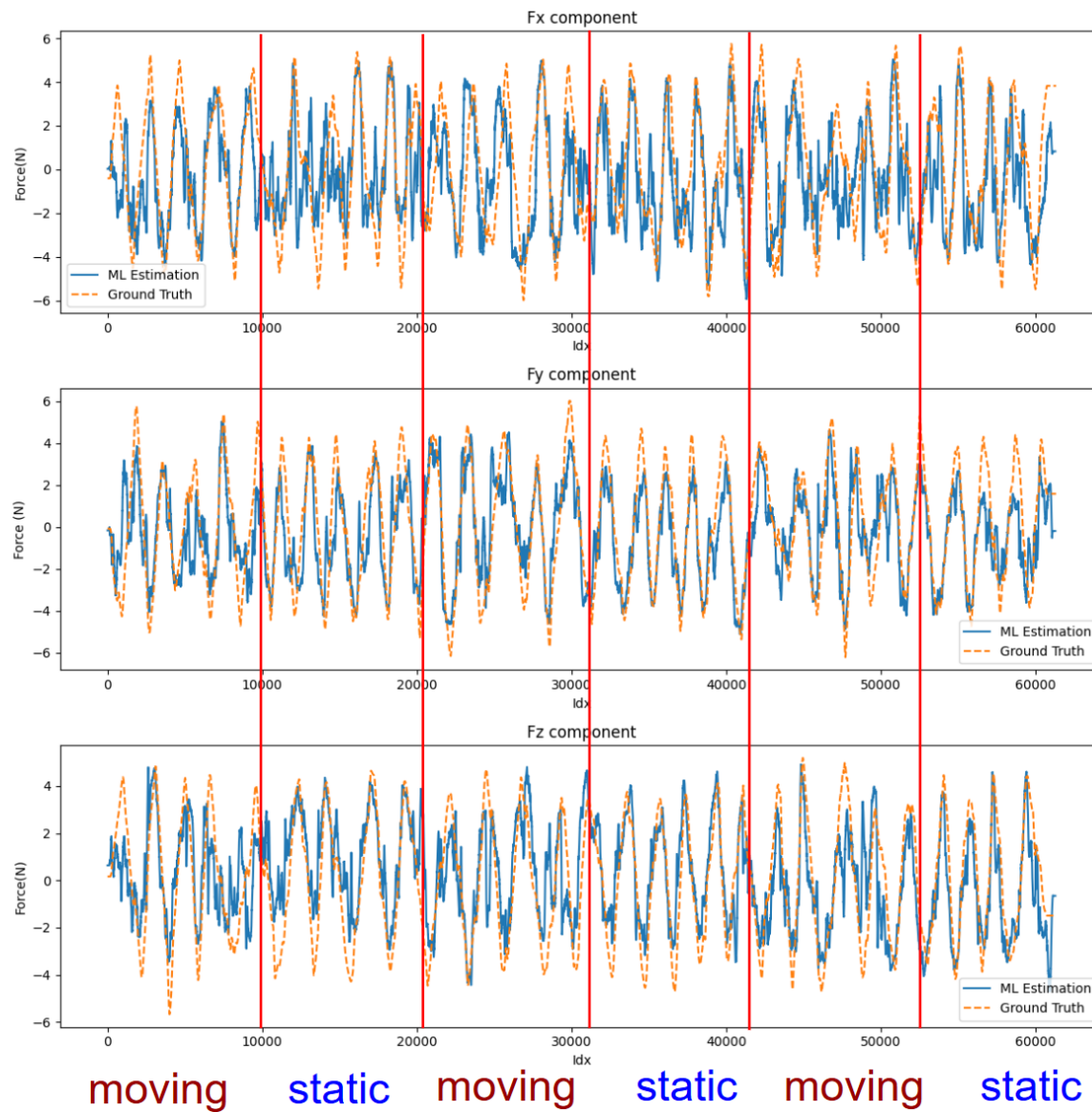


Figure 6.13: Example of force estimation performance when the robot has local motion in static status. With local motion, the neural network model can estimate the external force in a static status, because the joint frictions are overcome by local motion, so external forces can be reflected in joint torques.

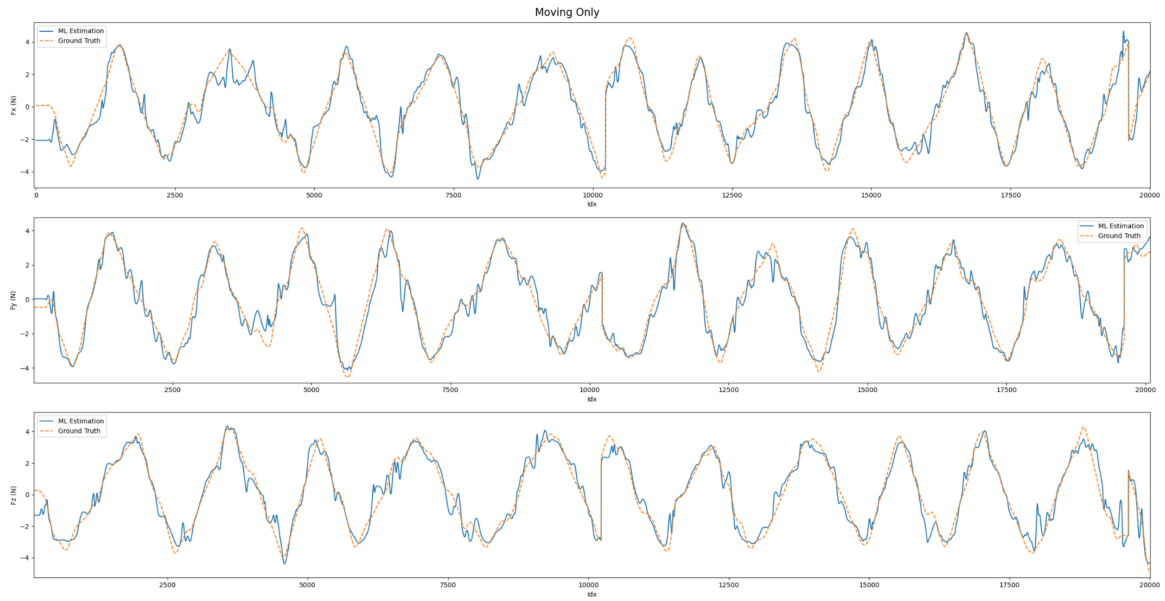


Figure 6.14: Example of force estimation performance when the robot is moving.

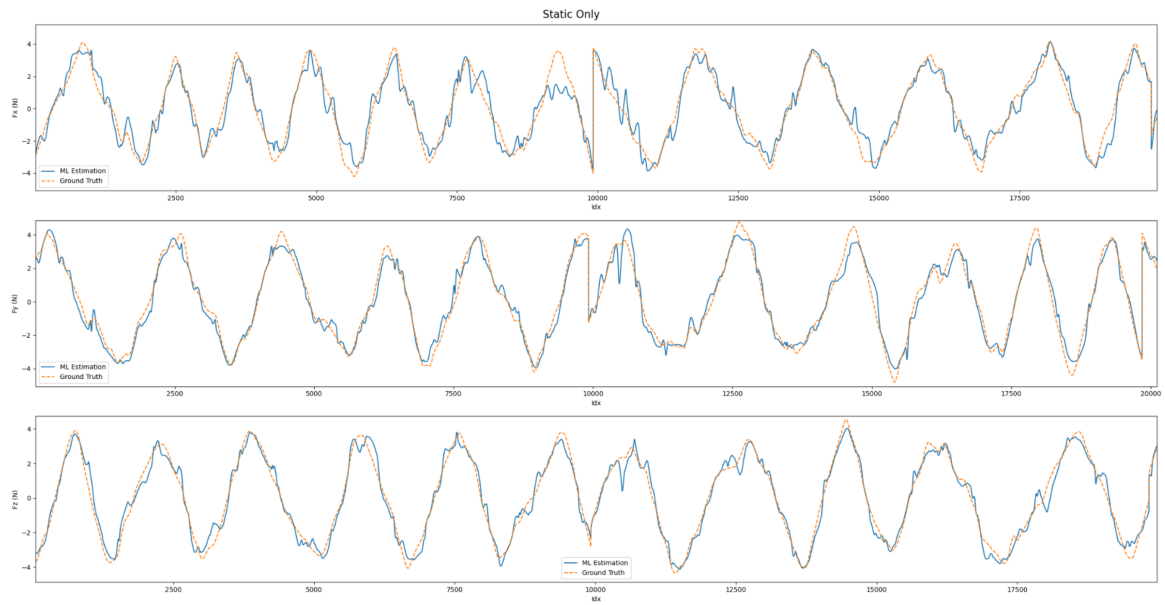


Figure 6.15: Example of force estimation performance when the robot is static.

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 *Conclusion*

In summary, this dissertation advanced AI-enhanced control for cable-driven surgical robots by addressing three closely related challenges: improving positional accuracy, improving visual perception, and establishing a foundation for haptic interaction. Cable-driven surgical robots provide compact and flexible mechanisms that are well suited for minimally invasive surgery, but cable stretch, slack, friction, backlash, and the difficulty of distal sensing create substantial challenges for accurate state estimation and closed-loop control. This dissertation demonstrated that learning-based and data-driven methods can use indirect robot-state information to compensate for these limitations and improve the practical performance of surgical robotic systems.

The first part of this dissertation developed an efficient data-driven joint calibration framework for the Raven-II surgical robot. By correcting the positioning joints directly, the proposed approach improved end-effector accuracy while keeping calibration time short enough to support repeated calibration before operation. The feature ablation study further showed that not all robot-state information contributes equally to learning-based calibration, and that joint positions and motor torques contain the most important information for this task. These findings improve both the accuracy and efficiency of the calibration model and provide insight into how learning models use robot-state features in cable-driven mechanisms. Together with the Python-based Raven-II controller and the low-cost visual tracking system (details in Appendix A), this work provides practical tools for more accurate data collection, evaluation, and control of surgical robots.

The second part of this dissertation extended the role of AI-enhanced methods from robot calibration to surgical perception and haptic interaction. The proposed active-learning and copy-and-paste synthetic image generation pipeline reduced the annotation burden for surgical instrument segmentation while maintaining strong segmentation performance. In addition, the motor-cable force actuation system developed in this dissertation provides a platform for applying controlled external

forces to the Raven-II end-effector and collecting data for future distal force estimation. Overall, this dissertation shows that AI-enhanced methods can help overcome key sensing and modeling limitations in cable-driven surgical robots. These results contribute toward surgical robotic systems that can operate with greater accuracy, richer perception, and stronger support for future semi-autonomous and autonomous surgical tasks.

7.2 Future Research Directions

To maintain representativeness of data in data-driven calibrations and force estimation, the size of the training dataset can increase exponentially as more dimensions are included in calibration, and some dimensions are also coupled such as velocities and positions. For example, the current positional calibration can be performed in 20 minutes, however, if we want to include loaded configurations with 9 different forces and record training data for each individual force, the calibration time increases to 3 hours and becomes unfeasible in real practice. Thus, I plan to develop a comprehensive calibration for surgical robots that includes positional accuracy, haptic feedback, and automation of surgical tasks, and the efficiency of the calibration should be feasible in real practice, including human labor, calibration time, and computational cost. In future works, I aim to improve the efficiency and feasibility of comprehensive calibrations of surgical robots, which involves answering the following open questions. Q1: given a certain length and/or time, how to find the trajectory that has the best coverage in a high dimensional and coupled space including position, velocity, and contact force? I will explore this by developing an evaluation of a trajectory's coverage in n -D spaces, which should be suitable to guide and evaluate searching, optimization, and learning methods. The high dimensional space is also weighted, as the workspace, velocity, and load do not have even inner or inter distribution of importance. Q2: With the evaluation of coverage, how to find the optimized training trajectory in a timely manner? I plan to explore approaches including searching methods as a modified traveling salesman problem, non-linear programming optimization, and reinforcement learning. A representative training trajectory can improve both the accuracy and efficiency of calibration.

To support reinforcement learning for automation in surgical tasks, I also aim to develop a realistic simulation of surgical robots that includes non-ideal features such as cable effects and motor

backlash. However, it can be impractically difficult to accurately model cables and their couplings. I plan to collaborate with the Raven-II community to collect a comprehensive dataset including control commands, robot states, and ground truth information such as robot pose and external load, from Raven-II robots with different conditions including cable tensions. By implementing this data into the simulation, the sim-to-real gap can be alleviated significantly, which can benefit control and automation algorithms developed in simulation to better generalize to real applications. Realistic simulations can accelerate the development of automative control algorithms and enable massive training of reinforcement learning. In addition, I also aim to explore physics-informed machine learning to improve the reliability and explainability of learning-based approaches, which is of importance in real practice.

BIBLIOGRAPHY

- [1] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1301–1310, 2017.
- [2] Yun-Hsuan Su, Wenfan Jiang, Digesh Chitrakar, Kevin Huang, Haonan Peng, and Blake Hannaford. Local style preservation in improved gan-driven synthetic image generation for endoscopic tool segmentation. *Sensors*, 21(15):5163, 2021.
- [3] Shan Lin, Fangbo Qin, Yangming Li, Randall A Bly, Kris S Moe, and Blake Hannaford. Lc-gan: Image-to-image translation based on generative adversarial network for endoscopic images. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2914–2920. IEEE, 2020.
- [4] Shan Lin, Fangbo Qin, Haonan Peng, Randall A Bly, Kris S Moe, and Blake Hannaford. Multi-frame feature aggregation for real-time instrument segmentation in endoscopic video. *IEEE Robotics and Automation Letters*, 6(4):6773–6780, 2021.
- [5] Daochang Liu, Yuhui Wei, Tingting Jiang, Yizhou Wang, Rulin Miao, Fei Shan, and Ziyu Li. Unsupervised surgical instrument segmentation via anchor generation and semantic diffusion. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part III 23*, pages 657–667. Springer, 2020.
- [6] Luca Sestini, Benoit Rosa, Elena De Momi, Giancarlo Ferrigno, and Nicolas Padoy. Fun-sis: A fully unsupervised approach for surgical instrument segmentation. *Medical Image Analysis*, 85:102751, 2023.
- [7] Tobias Ross, David Zimmerer, Anant Vemuri, Fabian Isensee, Manuel Wiesenfarth, Sebastian Bodenstedt, Fabian Both, Philip Kessler, Martin Wagner, Beat Müller, et al. Exploiting the

- potential of unlabeled endoscopic video data with self-supervised learning. *International journal of computer assisted radiology and surgery*, 13:925–933, 2018.
- [8] Luis C Garcia-Peraza-Herrera, Lucas Fidon, Claudia D’Ettorre, Danail Stoyanov, Tom Vercauteren, and Sebastien Ourselin. Image compositing for segmentation of surgical tools without manual annotations. *IEEE transactions on medical imaging*, 40(5):1450–1460, 2021.
- [9] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.
- [10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [11] Jaydeep H Palep. Robotic assisted minimally invasive surgery. *Journal of minimal access surgery*, 5(1):1, 2009.
- [12] Tamas Haidegger, Stefanie Speidel, Danail Stoyanov, and Richard M Satava. Robot-assisted minimally invasive surgery—surgical robotics in the data age. *Proceedings of the IEEE*, 110(7):835–846, 2022.
- [13] Jacob Rosen, Blake Hannaford, and Richard M Satava. Surgical robotics. *Medical Devices: Surgical and Image Guided Technologies*, 1:301–306, 2011.
- [14] Arash J Sayari, Coralie Pardo, Bryce A Basques, and Matthew W Colman. Review of robotic-assisted surgery: what the future looks like through a spine oncology lens. *Annals of translational medicine*, 7(10), 2019.
- [15] Brian S Peters, Priscila R Armijo, Crystal Krause, Songita A Choudhury, and Dmitry Oleynikov. Review of emerging surgical robotic technology. *Surgical endoscopy*, 32(4):1636–1655, 2018.
- [16] David B Camarillo, Thomas M Krummel, and J Kenneth Salisbury Jr. Robotic technology in surgery: past, present, and future. *The American Journal of Surgery*, 188(4):2–15, 2004.

- [17] Blake Hannaford, Jacob Rosen, Diana W Friedman, Hawkeye King, Phillip Roan, Lei Cheng, Daniel Glozman, Ji Ma, Sina Nia Kosari, and Lee White. Raven-ii: an open platform for surgical robotics research. *IEEE Transactions on Biomedical Engineering*, 60(4):954–959, 2012.
- [18] Peter Kazanzides, Zihan Chen, Anton Deguet, Gregory S Fischer, Russell H Taylor, and Simon P DiMaio. An open-source research kit for the da vinci® surgical system. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 6434–6439. IEEE, 2014.
- [19] A Gonzalez-Rodriguez, FJ Castillo-Garcia, Erika Ottaviano, Pierluigi Rea, and AG Gonzalez-Rodriguez. On the effects of the design of cable-driven robots on kinematics and dynamics models accuracy. *Mechatronics*, 43:18–27, 2017.
- [20] Lawrence H Kim, Clifford Bargar, Yuhang Che, and Allison M Okamura. Effects of master-slave tool misalignment in a teleoperated surgical robot. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5364–5370. IEEE, 2015.
- [21] Guillermo Maza and Arun Sharma. Past, present, and future of robotic surgery. *Otolaryngologic Clinics of North America*, 53(6):935–941, 2020.
- [22] Michael Yip and Nikhil Das. Robot autonomy for surgery. In *The Encyclopedia of MEDICAL ROBOTICS: Volume 1 Minimally Invasive Surgical Robotics*, pages 281–313. World Scientific, 2019.
- [23] Tamás Haidegger. Autonomy for surgical robots: Concepts and paradigms. *IEEE Transactions on Medical Robotics and Bionics*, 1(2):65–76, 2019.
- [24] Sahba Aghajani Pedram, Peter Ferguson, Ji Ma, Erik Dutson, and Jacob Rosen. Autonomous suturing via surgical robot: An algorithm for optimal selection of needle diameter, shape, and path. In *2017 IEEE International conference on robotics and automation (ICRA)*, pages 2391–2398. IEEE, 2017.
- [25] Yiwei Jiang, Haoying Zhou, and Gregory S Fischer. Markerless suture needle tracking from

- a robotic endoscope based on deep learning. In *2023 International Symposium on Medical Robotics (ISMR)*, pages 1–7. IEEE, 2023.
- [26] Vignesh Manoj Varier, Dhruv Kool Rajamani, Nathaniel Goldfarb, Farid Tavakkolmoghadam, Adnan Munawar, and Gregory S Fischer. Collaborative suturing: A reinforcement learning approach to automate hand-off task in suturing for surgical robots. In *2020 29th IEEE international conference on robot and human interactive communication (RO-MAN)*, pages 1380–1386. IEEE, 2020.
- [27] Ben Kehoe, Gregory Kahn, Jeffrey Mahler, Jonathan Kim, Alex Lee, Anna Lee, Keisuke Nakagawa, Sachin Patil, W Douglas Boyd, Pieter Abbeel, et al. Autonomous multilateral debridement with the raven surgical robot. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1432–1439. IEEE, 2014.
- [28] Daniel Seita, Sanjay Krishnan, Roy Fox, Stephen McKinley, John Canny, and Ken 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)*, pages 6651–6658. IEEE, 2018.
- [29] Jiaqi Xu, Bin Li, Bo Lu, Yun-Hui Liu, Qi Dou, and Pheng-Ann Heng. Surrol: An open-source reinforcement learning centered and dvrk compatible platform for surgical robot learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1821–1828. IEEE, 2021.
- [30] Md Masudur Rahman, Mythra V Balakuntala, Glebys Gonzalez, Mridul Agarwal, Upinder Kaur, Vishnunandan LN Venkatesh, Natalia Sanchez-Tamayo, Yexiang Xue, Richard M Voyles, Vaneet Aggarwal, et al. Sartres: A semi-autonomous robot teleoperation environment for surgery. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 9(4):376–383, 2021.
- [31] Glebys Gonzalez, Mridul Agarwal, Mythra V Balakuntala, Md Masudur Rahman, Upinder Kaur, Richard M Voyles, Vaneet Aggarwal, Yexiang Xue, and Juan Wachs. Deserts:

- Delay-tolerant semi-autonomous robot teleoperation for surgery. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12693–12700. IEEE, 2021.
- [32] Minho Hwang, Jeffrey Ichnowski, Brijen Thananjeyan, Daniel Seita, Samuel Paradis, Danyal Fer, Thomas Low, and Ken Goldberg. Automating surgical peg transfer: calibration with deep learning can exceed speed, accuracy, and consistency of humans. *IEEE Transactions on Automation Science and Engineering*, 20(2):909–922, 2022.
- [33] Yangming Li, Blake Hannaford, and Jacob Rosen. The raven open surgical robotic platforms: A review and prospect. *Acta Polytechnica Hungarica*, 16(8):9–27, 2019.
- [34] H King, S Kosari, B Hannaford, and J Ma. Kinematic analysis of the raven-ii research surgical robot platform. *University of Washington, Tech. Rep. UWEETR-2012-0006*, 2012.
- [35] Blake Hannaford. Models of robot manipulation, chapter 5. https://github.com/blake5634/Models_of_Robot_Manipulation/tree/main, 2022. Accessed: Nov.21.2023.
- [36] Srikrishnan Ramadurai, Sina Nia Kosari, H Hawkeye King, Howard Jay Chizeck, and Blake Hannaford. Application of unscented kalman filter to a cable driven surgical robot: A simulation study. In *2012 IEEE International Conference on Robotics and Automation*, pages 1495–1500. IEEE, 2012.
- [37] Sung-Hyun Choi, Jong-Oh Park, and Kyoung-Su Park. Tension analysis of a 6-degree-of-freedom cable-driven parallel robot considering dynamic pulley bearing friction. *Advances in Mechanical Engineering*, 9(8):1687814017714981, 2017.
- [38] Mohammad Haghighipanah, Yangming Li, Muneaki Miyasaka, and Blake Hannaford. Improving position precision of a servo-controlled elastic cable driven surgical robot using unscented kalman filter. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2030–2036. IEEE, 2015.
- [39] Muneaki Miyasaka, Joseph Matheson, Andrew Lewis, and Blake Hannaford. Measurement of the cable-pulley coulomb and viscous friction for a cable-driven surgical robotic system.

- In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 804–810. IEEE, 2015.
- [40] Yaru Deng, Yushan Tang, Bo Yang, Wenfeng Zheng, Shan Liu, and Chao Liu. A review of bilateral teleoperation control strategies with soft environment. In *2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 459–464. IEEE, 2021.
- [41] Minh Hwang, Brijen Thananjeyan, Samuel Paradis, Daniel Seita, Jeffrey Ichnowski, Danyal Fer, Thomas Low, and Ken Goldberg. Efficiently calibrating cable-driven surgical robots with rgb-d fiducial sensing and recurrent neural networks. *IEEE Robotics and Automation Letters*, 5(4):5937–5944, 2020.
- [42] Jeffrey Mahler, Sanjay Krishnan, Michael Laskey, Siddarth Sen, Adithyavairavan Murali, Ben Kehoe, Sachin Patil, Jiannan Wang, Mike Franklin, Pieter Abbeel, et al. Learning accurate kinematic control of cable-driven surgical robots using data cleaning and gaussian process regression. In *2014 IEEE international conference on automation science and engineering (CASE)*, pages 532–539. IEEE, 2014.
- [43] Haonan Peng, Xingjian Yang, Yun-Hsuan Su, and Blake Hannaford. Real-time data driven precision estimator for raven-ii surgical robot end effector position. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 350–356. IEEE, 2020.
- [44] Muneaki Miyasaka. *Cable Driven Robots: Hysteretic Cable Stretch, Cable-Pulley Network Friction, Fatigue Life, and Kinematics of Two-Arm Multi Staged Flexible Manipulator*. PhD thesis, University of Washington, 2017.
- [45] Haonan Peng, Andrew Lewis, Yun-Hsuan Su, Shan Lin, Dun-Tin Chiang, Wenfan Jiang, Helen Lai, and Blake Hannaford. Efficient data-driven joint-level calibration of cable-driven surgical robots. *arXiv preprint arXiv:2408.01604*, 2024.
- [46] Lena Maier-Hein, Swaroop S Vedula, Stefanie Speidel, Nassir Navab, Ron Kikinis, Adrian Park, Matthias Eisenmann, Hubertus Feussner, Germain Forestier, Stamatia Giannarou, et al.

- Surgical data science for next-generation interventions. *Nature Biomedical Engineering*, 1(9):691–696, 2017.
- [47] Alexey A Shvets, Alexander Rakhlin, Alexandr A Kalinin, and Vladimir I Iglovikov. Automatic instrument segmentation in robot-assisted surgery using deep learning. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 624–628. IEEE, 2018.
- [48] Fangbo Qin, Shan Lin, Yangming Li, Randall A Bly, Kris S Moe, and Blake Hannaford. Towards better surgical instrument segmentation in endoscopic vision: multi-angle feature aggregation and contour supervision. *IEEE Robotics and Automation Letters*, 5(4):6639–6646, 2020.
- [49] Mobarakol Islam, Yueyuan Li, and Hongliang Ren. Learning where to look while tracking instruments in robot-assisted surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 412–420. Springer, 2019.
- [50] Alexandr A Kalinin, Vladimir I Iglovikov, Alexander Rakhlin, and Alexey A Shvets. Medical image segmentation using deep neural networks with pre-trained encoders. *Deep Learning Applications*, pages 39–52, 2020.
- [51] Veronika Cheplygina, Marleen de Bruijne, and Josien PW Pluim. Not-so-supervised: a survey of semi-supervised, multi-instance, and transfer learning in medical image analysis. *Medical image analysis*, 54:280–296, 2019.
- [52] Lin Yang, Yizhe Zhang, Jianxu Chen, Siyuan Zhang, and Danny Z Chen. Suggestive annotation: A deep active learning framework for biomedical image segmentation. In *International conference on medical image computing and computer-assisted intervention*, pages 399–407. Springer, 2017.
- [53] S Bodenstedt, S Speidel, M Wagner, J Chen, A Kisilenko, BP Müller, L Maier-Hein, B Oliveira, S Hong, J Zamora-Anaya, et al. Heichole surgical workflow analysis and full scene segmentation (heisurf). MICCAI, 2021.

- [54] Tonghe Wang, Yang Lei, Yabo Fu, Jacob F Wynne, Walter J Curran, Tian Liu, and Xiaofeng Yang. A review on medical imaging synthesis using deep learning and its clinical applications. *Journal of Applied Clinical Medical Physics*, 22(1):11–36, 2021.
- [55] Jean-Francois Rajotte, Sumit Mukherjee, Caleb Robinson, Anthony Ortiz, Christopher West, Juan Lavista Ferres, and Raymond T Ng. Reducing bias and increasing utility by federated generative modeling of medical images using a centralized adversary. *arXiv preprint arXiv:2101.07235*, 2021.
- [56] Shohei Fujita, Akifumi Hagiwara, Yujiro Otsuka, Masaaki Hori, Naoyuki Takei, Ken-Pin Hwang, Ryusuke Irie, Christina Andica, Koji Kamagata, Toshiaki Akashi, et al. Deep learning approach for generating mra images from 3d quantitative synthetic mri without additional scans. *Investigative radiology*, 55(4):249–256, 2020.
- [57] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter 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)*, pages 23–30. IEEE, 2017.
- [58] Wouter M Kouw and Marco Loog. A review of domain adaptation without target labels. *IEEE transactions on pattern analysis and machine intelligence*, 43(3):766–785, 2019.
- [59] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2918–2928, 2021.
- [60] Tal Remez, Jonathan Huang, and Matthew Brown. Learning to segment via cut-and-paste. In *Proceedings of the European conference on computer vision (ECCV)*, pages 37–52, 2018.
- [61] Joo Ho Lee, Inchang Choi, and Min H Kim. Laplacian patch-based image synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2727–2735, 2016.

- [62] Marc Gorriz, Axel Carlier, Emmanuel Faure, and Xavier Giro-i Nieto. Cost-effective active learning for melanoma segmentation. *arXiv preprint arXiv:1711.09168*, 2017.
- [63] Samuel Budd, Emma C Robinson, and Bernhard Kainz. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *Medical Image Analysis*, page 102062, 2021.
- [64] Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.
- [65] Allison M. Okamura. Haptic feedback in robot-assisted minimally invasive surgery. *Current Opinion in Urology*, 19(1):102–107, 2009.
- [66] O. A. J. van der Meijden and M. P. Schijven. The value of haptic feedback in conventional and robot-assisted minimal invasive surgery and virtual reality training: A current review. *Surgical Endoscopy*, 23(6):1180–1190, 2009.
- [67] Jacinto Colan, Adriana Davila, and Yuichi Hasegawa. Tactile feedback in robot-assisted minimally invasive surgery: A systematic review. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 2024.
- [68] Zonghe Chua and Allison M. Okamura. A modular 3-degrees-of-freedom force sensor for robot-assisted minimally invasive surgery research. *Sensors*, 23(11):5230, 2023.
- [69] Hao Yang, Haoying Zhou, Gregory S. Fischer, and Jie Ying Wu. A hybrid model and learning-based force estimation framework for surgical robots. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 906–912, 2024.
- [70] Max Allan, Alex Shvets, Thomas Kurmann, Zichen Zhang, Rahul Duggal, Yun-Hsuan Su, Nicola Rieke, Iro Laina, Niveditha Kalavakonda, Sebastian Bodenstedt, et al. 2017 robotic instrument segmentation challenge. *arXiv preprint arXiv:1902.06426*, 2019.
- [71] Edvard Naerum, Hawkeye HI King, and Blake Hannaford. Robustness of the unscented kalman filter for state and parameter estimation in an elastic transmission. In *Robotics: Science and Systems*, pages 1–8. Citeseer, 2009.

- [72] Muneaki Miyasaka, Mohammad Haghighipanah, Yangming Li, and Blake Hannaford. Hysteresis model of longitudinally loaded cable for cable driven robots and identification of the parameters. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4051–4057. IEEE, 2016.
- [73] DongHoon Baek, Young-Hoon Nho, and Dong-Soo Kwon. Vio-com: Feed-forward compensation using vision-based optimization for high-precision surgical manipulation. *IEEE Robotics and Automation Letters*, 7(1):263–270, 2021.
- [74] Dong-Ho Lee, Young-Ho Kim, Jarrod Collins, Ankur Kapoor, Dong-Soo Kwon, and Tommaso Mansi. Non-linear hysteresis compensation of a tendon-sheath-driven robotic manipulator using motor current. *IEEE Robotics and Automation Letters*, 6(2):1224–1231, 2021.
- [75] Francesco Cursi, Weibang Bai, Eric M Yeatman, and Petar Kormushev. Model learning with backlash compensation for a tendon-driven surgical robot. *IEEE Robotics and Automation Letters*, 7(3):7958–7965, 2022.
- [76] Andrea Franzini, Shayan Moosa, Domenico Servello, Isabella Small, Francesco DiMeco, Zhiyuan Xu, William Jeffrey Elias, Angelo Franzini, and Francesco Prada. Ablative brain surgery: an overview. *International Journal of Hyperthermia*, 36(2):64–80, 2019.
- [77] Paul R Cohen and Adele E Howe. Toward ai research methodology: Three case studies in evaluation. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(3):634–646, 1989.
- [78] Allen Newell. A tutorial on speech understanding systems. *Speech recognition*, pages 4–54, 1975.
- [79] Richard Meyes, Melanie Lu, Constantin Waubert de Puiseau, and Tobias Meisen. Ablation studies in artificial neural networks. *arXiv preprint arXiv:1901.08644*, 2019.
- [80] David Kauchak, Obay Mouradi, Christopher Pentoney, and Gondy Leroy. Text simplification tools: Using machine learning to discover features that identify difficult text. In *2014 47th Hawaii international conference on system sciences*, pages 2616–2625. IEEE, 2014.

- [81] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.
- [82] Nripendra Kumar Singh and Khalid Raza. Medical image generation using generative adversarial networks: A review. *Health Informatics: A Computational Perspective in Healthcare*, pages 77–96, 2021.
- [83] Tae Keun Yoo, Joon Yul Choi, and Hong Kyu Kim. A generative adversarial network approach to predicting postoperative appearance after orbital decompression surgery for thyroid eye disease. *Computers in biology and medicine*, 118:103628, 2020.
- [84] Imanol Luengo, Evangello Flouty, Petros Giataganas, Piyamate Wisanuvej, Jean Nehme, and Danail Stoyanov. Surreal: Enhancing surgical simulation realism using style transfer. *arXiv preprint arXiv:1811.02946*, 2018.
- [85] Emanuele Colleoni and Danail Stoyanov. Robotic instrument segmentation with image-to-image translation. *IEEE Robotics and Automation Letters*, 6(2):935–942, 2021.
- [86] Nima Tajbakhsh, Laura Jeyaseelan, Qian Li, Jeffrey N Chiang, Zhihao Wu, and Xiaowei Ding. Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation. *Medical Image Analysis*, 63:101693, 2020.
- [87] Taehun Kim, Kyunghwa Lee, Sungwon Ham, Beomhee Park, Sangwook Lee, Dayeong Hong, Guk Bae Kim, Yoon Soo Kyung, Choung-Soo Kim, and Namkug Kim. Active learning for accuracy enhancement of semantic segmentation with cnn-corrected label curations: Evaluation on kidney segmentation in abdominal ct. *Scientific reports*, 10(1):1–7, 2020.
- [88] Alex Holub, Pietro Perona, and Michael C Burl. Entropy-based active learning for object recognition. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008.
- [89] Joris Roels and Yvan Saeys. Cost-efficient segmentation of electron microscopy images using active learning. *arXiv preprint arXiv:1911.05548*, 2019.

- [90] Andrew I Schein and Lyle H Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, 2007.
- [91] Soufiane Belharbi, Ismail Ben Ayed, Luke McCaffrey, and Eric Granger. Deep active learning for joint classification & segmentation with weak annotator. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3338–3347, 2021.
- [92] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.
- [93] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.
- [94] Toan Tran, Thanh-Toan Do, Ian Reid, and Gustavo Carneiro. Bayesian generative active deep learning. In *International Conference on Machine Learning*, pages 6295–6304. PMLR, 2019.
- [95] Sebastian Bodenstedt, Dominik Rivoir, Alexander Jenke, Martin Wagner, Michael Breucha, Beat Müller-Stich, Sören Torge Mees, Jürgen Weitz, and Stefanie Speidel. Active learning using deep bayesian networks for surgical workflow analysis. *International journal of computer assisted radiology and surgery*, 14:1079–1087, 2019.
- [96] Ana Barragán-Montero, Umair Javaid, Gilmer Valdés, Dan Nguyen, Paul Desbordes, Benoît Macq, Siri Willems, Liesbeth Vandewinckele, Mats Holmström, Fredrik Löfman, et al. Artificial intelligence and machine learning for medical imaging: A technology review. *Physica Medica*, 83:242–256, 2021.
- [97] Khalid Raza and Nripendra K Singh. A tour of unsupervised deep learning for medical image analysis. *Current Medical Imaging*, 17(9):1059–1077, 2021.
- [98] Félix Fuentes-Hurtado, Abdolrahim Kadkhodamohammadi, Evangello Flouty, Santiago Barbarisi, Imanol Luengo, and Danail Stoyanov. Easylabels: weak labels for scene segmentation in laparoscopic videos. *International journal of computer assisted radiology and surgery*, 14:1247–1257, 2019.

- [99] Iman Aganj, Mukesh G Harisinghani, Ralph Weissleder, and Bruce Fischl. Unsupervised medical image segmentation based on the local center of mass. *Scientific reports*, 8(1):13012, 2018.
- [100] Laurent Lejeune, Jan Grossrieder, and Raphael Sznitman. Iterative multi-path tracking for video and volume segmentation with sparse point supervision. *Medical image analysis*, 50:65–81, 2018.
- [101] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.
- [102] Sanat Ramesh, Vinkle Srivastav, Deepak Alapatt, Tong Yu, Aditya Murali, Luca Sestini, Chinedu Innocent Nwoye, Idris Hamoud, Saurav Sharma, Antoine Fleurentin, et al. Dissecting self-supervised learning methods for surgical computer vision. *Medical Image Analysis*, 88:102844, 2023.
- [103] Roy Hirsch, Mathilde Caron, Regev Cohen, Amir Livne, Ron Shapiro, Tomer Golany, Roman Goldenberg, Daniel Freedman, and Ehud Rivlin. Self-supervised learning for endoscopic video analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 569–578. Springer, 2023.
- [104] Fangbo Qin, Yangming Li, Yun-Hsuan Su, De Xu, and Blake Hannaford. Surgical instrument segmentation for endoscopic vision with data fusion of cnn prediction and kinematic pose. In *2019 international conference on robotics and automation (ICRA)*, pages 9821–9827. IEEE, 2019.
- [105] Luca Sestini, Benoit Rosa, Elena De Momi, Giancarlo Ferrigno, and Nicolas Padoy. A kinematic bottleneck approach for pose regression of flexible surgical instruments directly from images. *IEEE Robotics and Automation Letters*, 6(2):2938–2945, 2021.
- [106] Isabel Funke, Alexander Jenke, Sören Torge Mees, Jürgen Weitz, Stefanie Speidel, and Sebastian Bodenstedt. Temporal coherence-based self-supervised learning for laparoscopic

- workflow analysis. In *International Workshop on Computer-Assisted and Robotic Endoscopy*, pages 85–93. Springer, 2018.
- [107] Max Bergholz, Manuel Ferle, and Bernhard M. Weber. The benefits of haptic feedback in robot assisted surgery and their moderators: A meta-analysis. *Scientific Reports*, 13(1):19215, 2023.
- [108] Mahmoud Zarebidoki, Jaspreet Singh Dhupia, and William L. Xu. A review of cable-driven parallel robots: Typical configurations, analysis techniques, and control methods. *IEEE Robotics & Automation Magazine*, 29(3):89–106, 2022.
- [109] Dianjin Zhang and Bin Guo. Utilizing reinforcement learning to drive redundant constrained cable-driven robots with unknown parameters. *Machines*, 12(6):372, 2024.
- [110] Elena Bardi, Adrian Esser, Peter Wolf, Marta Gandolla, Emilia Ambrosini, Alessandra Pedrocchi, and Robert Riener. Sensorless model-based tension control for a cable-driven exosuit. *Wearable Technologies*, 5:e25, 2024.
- [111] Yun-Hsuan Su, Adnan Munawar, Anton Deguet, Andrew Lewis, Kyle Lindgren, Yangming Li, Russell H Taylor, Gregory S Fischer, Blake Hannaford, and Peter Kazanzides. Collaborative robotics toolkit (crtk): Open software framework for surgical robotics research. In *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, pages 48–55. IEEE, 2020.
- [112] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.
- [113] Haonan Peng, Andrew Lewis, and Blake Hannaford. Ablation study on features in learning-based joints calibration of cable-driven surgical robots. *arXiv preprint arXiv:2302.04999*, 2023.
- [114] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

- [115] Fabrice Narducci, Emilie Bogart, Thomas Hebert, Tristan Gauthier, Pierre Collinet, Jean Marc Classe, Fabrice Lecuru, Alain Delest, Stephanie Motton, Vanessa Conri, et al. Severe perioperative morbidity after robot-assisted versus conventional laparoscopy in gynecologic oncology: results of the randomized robogyn-1004 trial. *Gynecologic oncology*, 158(2):382–389, 2020.
- [116] Michael A Palese, Michael D Stifelman, Ravi Munver, R Ernest Sosa, Courtney K Philipps, Caner Dinlenc, and Joseph J Del Pizzo. Robot-assisted laparoscopic dismembered pyeloplasty: a combined experience. *Journal of endourology*, 19(3):382–386, 2005.
- [117] Becky B Trinh, Nicole R Jackson, Adam T Hauch, Tian Hu, and Emad Kandil. Robotic versus laparoscopic colorectal surgery. *JSLS: Journal of the Society of Laparoendoscopic Surgeons*, 18(4), 2014.
- [118] Xiaoqin Zhou, Haijun Zhang, Mei Feng, Ji Zhao, and Yili Fu. New remote centre of motion mechanism for robot-assisted minimally invasive surgery. *Biomedical engineering online*, 17:1–16, 2018.
- [119] Antonia Tzemanaki, Sanja Dogramadzi, Tony Pipe, and Chris Melhuish. Towards an anthropomorphic design of minimally invasive instrumentation for soft tissue robotic surgery. In *Advances in Autonomous Robotics: Joint Proceedings of the 13th Annual TAROS Conference and the 15th Annual FIRA RoboWorld Congress, Bristol, UK, August 20-23, 2012*, pages 455–456. Springer, 2012.
- [120] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.
- [121] Edward R Dougherty. An introduction to morphological image processing. In *SPIE. Optical Engineering Press*, 1992.
- [122] Ian T Young and Lucas J Van Vliet. Recursive implementation of the gaussian filter. *Signal processing*, 44(2):139–151, 1995.
- [123] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

- [124] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.
- [125] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [126] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [127] Abdel Aziz Taha and Allan Hanbury. Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool. *BMC medical imaging*, 15(1):1–28, 2015.
- [128] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [129] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.
- [130] Tin Chiang. *Sensorless Contact Force Estimation on Raven II Surgical Robot Platform*. University of Washington, 2024.
- [131] HK Yuen, John Princen, John Illingworth, and Josef Kittler. Comparative study of hough transform methods for circle finding. *Image and vision computing*, 8(1):71–77, 1990.
- [132] Azra Fetić, Davor Jurić, and Dinko Osmanković. The procedure of a camera calibration using camera calibration toolbox for matlab. In *2012 Proceedings of the 35th International Convention MIPRO*, pages 1752–1757. IEEE, 2012.

- [133] Jorge J Moré. The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.
- [134] John Canny. A computational approach to edge detection. In *Readings in computer vision*, pages 184–203. Elsevier, 1987.
- [135] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. ” O’Reilly Media, Inc.”, 2008.
- [136] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on optimization*, 9(1):112–147, 1998.
- [137] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Appendix A

REAL-TIME DATA DRIVEN PRECISION ESTIMATOR FOR RAVEN-II SURGICAL ROBOT AND VISUAL TRACKING OF END EFFECTOR POSITION

(Note: this work was completed during my master's study and the gap year between my master's and Ph.D. studies. Since I did not have a master's thesis, this work is included as an appendix in this Ph.D. dissertation.)

To reduce the error caused by the cable-driven mechanism in surgical robots such as Raven-II, a cost-effective, real-time and data-driven pipeline for robot end effector position precision estimation is proposed and tested on Raven-II. Analysis shows an improved end effector position error of around 1mm RMS traversing through the entire robot workspace without high-resolution motion tracker.

A.1 Method

A.1.1 System Workflow

The end effector pose estimation system consists of two phases - vision-based ground truth measurement and neural network estimator, as shown in Fig. A.1.

Phase One

4 webcams are mounted with poses determined by a two-phase calibration procedure (Fig. A.2, left). During data recording, 3 distinct colored balls are fixed to the end effector as markers (Fig. A.6 left). The balls are only used to collect training data and are removed in real operation. In addition, the balls and the holder are hollow and very light, for a negligible load to the system. For each image frame, preprocessing steps are performed and followed by Hough circle detection [131]. A frame update algorithm is then adopted to prevent false positives by comparing circles detected in subsequent images. Finally, the 3 circle centers yield ground truth end effector position up to 0.5 mm accuracy.

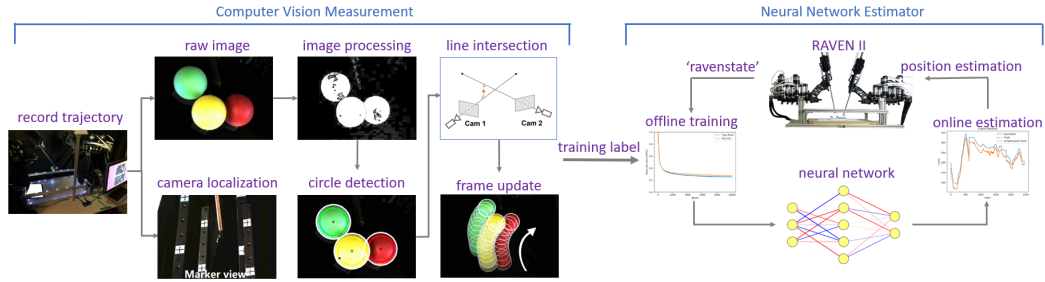


Figure A.1: Workflow for proposed online position precision estimation system. Pipeline entails vision based ground truth measurement (left) and data driven robot position estimator (right). Training labels passing from left to right are true Raven-II position information.

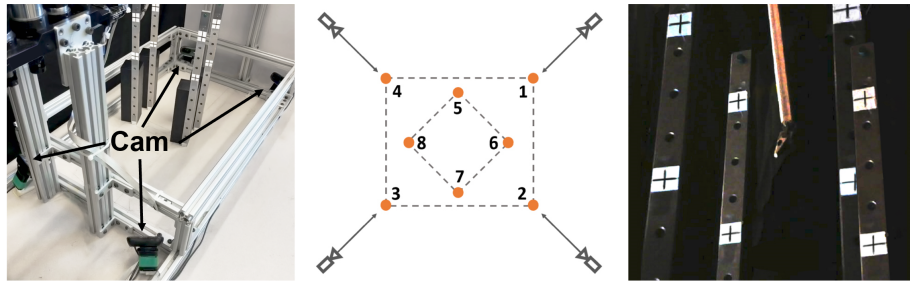


Figure A.2: 16 marker points used for camera pose calibration are placed at 8 ground locations in robot workspace.

Phase Two

There are 2 stages in phase two - offline training and online estimation. The collected position data is used to train a neural network. Next, the trained neural network model provides online estimation of the end effector position with an average precision of around 0.8 mm, a significant improvement from the Raven-II feedback position.

A.1.2 Camera Localization

Since the goal of phase one is to obtain ground truth robot position through vision, accurate camera pose identification in relation to the robot frame is crucial. The existing camera extrinsic matrix

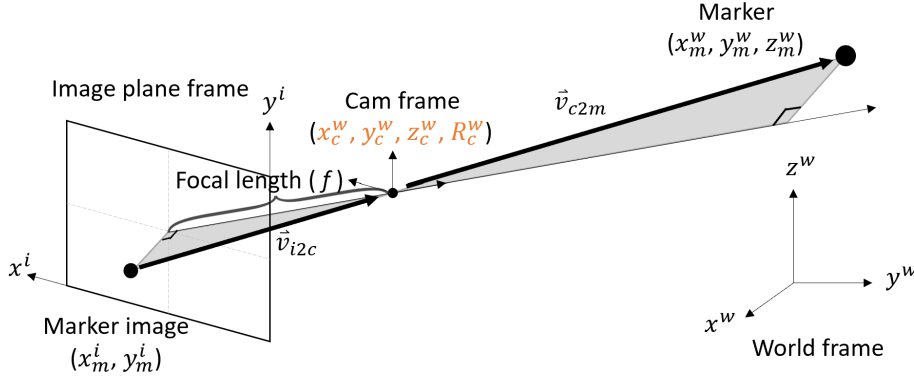


Figure A.3: Geometric relations among image plane, camera and marker.

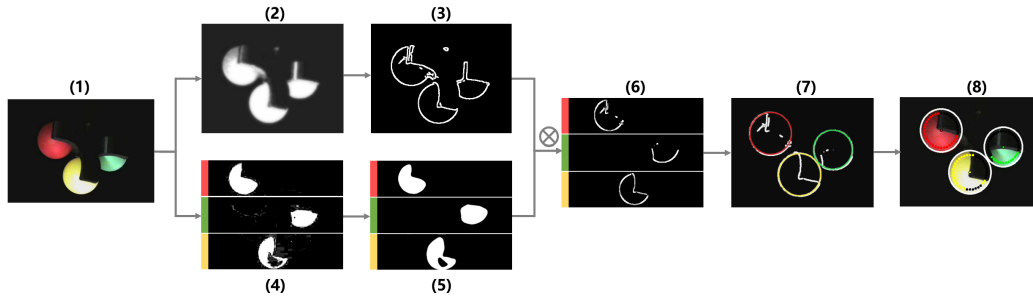


Figure A.4: Circle detection method and the preprocessing steps. Color image (1) is first converted to gray and Gaussian blur is applied (2). Then Canny edge (3) is taken from the gray image. In the other pathway, the color image (1) is divided into red, green, and yellow color channels (4). Then Gaussian blur with σ of 10% radius followed by binarization is applied (5). Image (6) is elementwise multiplication of (3) and (5). Hough circle detection is applied in each channel in image(6), which combines to result in (7). Finally, image (8) is a color check near the circle border to eliminate false positives.

solver in MATLAB [132] requires placing a chessboard at a known pose with respect to the world frame. The resultant camera pose from said solution contains error up to 40 millimeters if only 1-5 chessboards are used. Although placing more chessboards (usually more than 20) theoretically improves precision, the problem of acquiring accurate chessboard poses also becomes increasingly difficult due to the constrained workspace. As a practical alternative, we use 1 chessboard and 16

marker points together (Fig. A.2). The chessboard provides a rough cameras pose estimate and the 16 markers further refine it.

We define world frame w , chessboard frame b , camera frame c and image frame i . The camera position and orientation with respect to the world coordinate are respectively $p_c^w = [x_c^w \ y_c^w \ z_c^w]^T$ and $R_c^w \in \mathbb{R}^{(3 \times 3)}$.

Similarly, the marker position in the world coordinate is $p_m^w = [x_m^w \ y_m^w \ z_m^w]^T$ whereas its 2D projection on the image frame is $p_m^i = [x_m^i \ y_m^i]^T$. Let distance vectors

$$\vec{v}_{c2m} = p_m^w - p_c^w, \quad (\text{A.1})$$

$$\vec{v}_{i2c} = R_c^w \begin{bmatrix} p_m^i \\ f \end{bmatrix} = R_b^w \cdot R_c^b \begin{bmatrix} x_m^i \\ y_m^i \\ f \end{bmatrix}, \quad (\text{A.2})$$

where f is the camera focal length. According to the geometric relations illustrated in Fig. A.3, the following three equations hold true for each marker from each camera view:

$$\frac{\vec{v}_{c2m}(k)}{\vec{v}_{i2c}(k)} = \frac{\|\vec{v}_{c2m}\|_2}{\|\vec{v}_{i2c}\|_2} \quad \forall k \in [1, 2, 3], \quad (\text{A.3})$$

where $\vec{v}(k)$ represents the k th entry in the distance vector.

There are a total of 6 unknown variables in p_c^w and R_c^w . With 16 equation sets (A.3) yield from all 16 marker points, the Levenberg-Marquardt algorithm[133] finds optimal camera pose with initial guess $p_c^{w'}$ and $R_c^{w'}$ from the chessboard.

A.1.3 Auto Hough Circle Detection

Localization of the colored balls assists with end effector position acquisition. 2D circle detection is the first step to localize colored balls. Hough circle detection [131] is chosen due to its robustness to occlusion. Yet, it is sensitive to bright edge noises, so heuristic parameter tuning and image enhancements are necessary. Fig. A.4 shows the pre-processing steps with the following design details:

Edge Detection (1)-(3) These steps increase the signal to noise ratio and clear up noisy edges.

Color Segmentation (1)-(4)-(5) After this, circle segments in the color channels are intentionally enlarged.

Border Refinement (3)(5)-(6) Since circles in b) are larger than a). The circle canny edges [134] will be kept, and any edges outside the circles will be removed.

Circle Identification (6)-(7)-(8) After circles are detected, sample points near the border are used to ensure the circle color matches the target color with similarity more than 25%. Otherwise, the circle is considered a false detection.

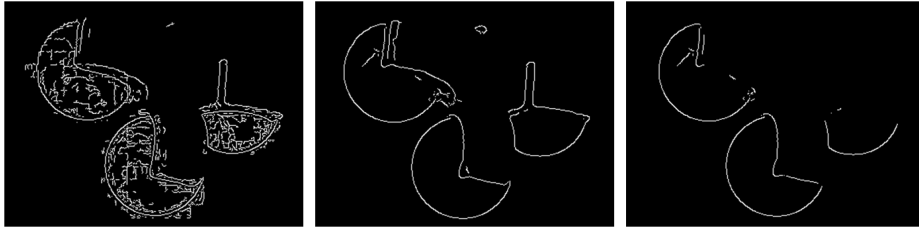


Figure A.5: Comparison of Canny edge on original image (left), normal blurred image (middle) and processed image by our system (right).

The ‘HoughCircle’ function in OpenCV [135] is used to detect circles. Two heuristically chosen parameters are inverse ratio $dp = 1$ and high Canny threshold $para1 = 100$. To improve adaptability, an automated parameter tuning algorithm is designed for the accumulator threshold $para2$, where small values indicate a higher chance of false positives.

Suppose k circles are expected, the algorithm uses bisection to approach the lower bound of $para2$, such that ‘HoughCircle’ comes close to returning $(k + 1)$ circles but still returns k . In the case where $para2$ jumps between $(k - 1)$ and $(k + 1)$, increasing σ in Gaussian blur helps, but is conducted only if necessary - due to decreased precision.

A.1.4 Frame Update Algorithm

There are 2 main purposes of the frame update algorithm. The first is to decide if any camera is returning false circles. The second is to provide other parameters to 'HoughCircle', including the minimum distance between circle centers d_{min} , and extrema of circle radius r_{min} and r_{max} . Setting a low d_{min} and wide range between r_{min} and r_{max} allows more chance to detect circles, but it also increases computational cost and the risk of false positives. Thus, under smooth end effector motion, d_{min} , r_{min} and r_{max} values are bounded by the detection result from the previous frames.

There are a total of 4 cameras. For each ball, there should ideally be 4 circles detected - one corresponding to each of the 4 camera views. After the frame update algorithm, each detected circle is proclaimed as either effective or suspended.

Every circle is initialized as effective. If any of the conditions below holds true, an effective circle will be suspended:

- The movement of the circle center in successive frames exceeds a predetermined motion threshold.
- Color check in fig. A.4 (8) fails to match target color by 25%.
- Hough circle detection does not return a circle, even with the largest tolerable σ for Gaussian blur.

On the other hand, if all the following conditions are satisfied for more than 5 frames, a suspended circle will become effective:

- The Hough circle returns result normally.
- Color check in fig. A.4 (8) is successful.
- The 3D reconstructed ball center position from effectively detected 2D circles is consistent with that of this suspended 2D circle information.

A line that connects a 2D effective circle center and its associated camera center forms a 3D ray. The 3D position of the ball center is the intersection point of rays. These rays usually do not intersect perfectly, and the midpoint of the common perpendicular of the two rays is selected to be the intersection point. When the number of effective circles is larger than 2, the mean position is taken from each pair of rays. Thus, 2 or more effective circles of the same ball must be detected at any time instance for successful localization. If not, the system skips a few frames and restarts, all while raising the tolerable range for the parameters. Once there are enough effective circles, the parameter tolerance converges back down.

A.1.5 End Effector Localization

As Fig. A.6 shows, the 3 colored balls are fixed around the end effector center (the black point, which is defined by the Raven-II system). The end effector center position shares the same plane with the 3 ball centers. The distances between the end effector center and each ball center are defined as $d = 38 \text{ mm}$ and each ball has radius $r = 20 \text{ mm}$. An end effector coordinate frame is designed for Raven-II, where the origin is the end effector point, the X axis points toward the green ball center, and the Y axis points at the yellow ball center, which is also co-linear with Raven-II axis X_{6b} [34].

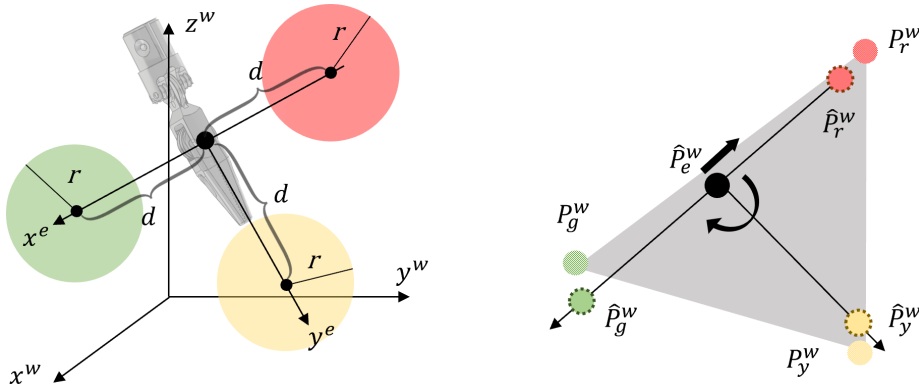


Figure A.6: Geometric relations between ball centers and end effector position.

The detected ball centers in the world coordinate frame are: green $p_g^w = [x_g^w \ y_g^w \ z_g^w]^T$, yellow $p_y^w = [x_y^w \ y_y^w \ z_y^w]^T$ and red $p_r^w = [x_r^w \ y_r^w \ z_r^w]^T$. These 3 points are used to estimate the end effector

center $\hat{p}_e^w = [\hat{x}_e^w \ \hat{y}_e^w \ \hat{z}_e^w]^T$ and the orientation $\hat{R}_e^w = Rot(z, \hat{\gamma}) \cdot Rot(y, \hat{\beta}) \cdot Rot(x, \hat{\alpha})$ represented in Euler angles. From the setup of the end effector and three colored balls, the estimated ball centers can be represented by

$$\hat{p}_g^w = \hat{R}_e^w \cdot \begin{bmatrix} d & 0 & 0 \end{bmatrix}^T + \hat{p}_e^w \quad (\text{A.4})$$

$$\hat{p}_y^w = \hat{R}_e^w \cdot \begin{bmatrix} 0 & d & 0 \end{bmatrix}^T + \hat{p}_e^w \quad (\text{A.5})$$

$$\hat{p}_r^w = \hat{R}_e^w \cdot \begin{bmatrix} -d & 0 & 0 \end{bmatrix}^T + \hat{p}_e^w \quad (\text{A.6})$$

To solve for the end effector pose, a cost function C with 6 unknown variables $(\hat{x}_e^w, \hat{y}_e^w, \hat{z}_e^w, \hat{\alpha}, \hat{\beta}, \hat{\gamma})$ can be set up. Minimizing the cost gives an optimal solution of the unknown variables,

$$C = (p_g^w - \hat{p}_g^w)^2 + (p_y^w - \hat{p}_y^w)^2 + (p_r^w - \hat{p}_r^w)^2 \quad (\text{A.7})$$

Take the middle point of p_g^w and p_r^w as the initial guess of $\hat{p}_e^w = [\hat{x}_e^w \ \hat{y}_e^w \ \hat{z}_e^w]^T$, denoted as P_{e0}^w . The initial guess of orientation is taken by setting $\overrightarrow{P_{e0}^w P_g^w}$, $\overrightarrow{P_{e0}^w P_y^w}$ as x and y axis of the end effectors frame. Then, the Nelder-Mead Simplex Method[136] is used to find the optimal value of $(\hat{x}_e^w, \hat{y}_e^w, \hat{z}_e^w, \hat{\alpha}, \hat{\beta}, \hat{\gamma})$, which is the pose of the end effector.

A.1.6 Neural Network Architecture

The training data for the neural network is represented as:

$$\chi_{NN} = \left\{ \left(\text{ravenstate}^{(k)}, \text{err}^{(k)} \right) \mid k = [1 \dots N] \right\}, \quad (\text{A.8})$$

where ‘ravenstate’ is a ROS topic that contains real-time kinematics and dynamic information of Raven-II. And ‘err’ is the difference between the Raven-II reported end effector positions and the ground truth collected through vision-based measurement in phase one. More details follow:

The Features

The information provided in a ‘ravenstate’ message includes: (a) kinematics derived Cartesian pose, (b) desired Cartesian pose, (c) current joint pose, (d) desired joint pose, (e) motor and joint velocities, (f) motor torques, (g) desired grasper pose and other information, all of which are added as features of the neural network.

The Labels

We chose the end effector *position error* as the label instead of the ground truth position itself because Raven-II already has end effector pose feedback in the ‘ravenstate’ topic. To get a more accurate estimation, one only needs to correct Raven-II pose feedback derived from joint pose and forward kinematics, instead of creating a new estimation. In fact, this design is more robust to static Cartesian positional offsets contained in the ‘ravenstate’ reported position, which slightly differs every time the system restarts.

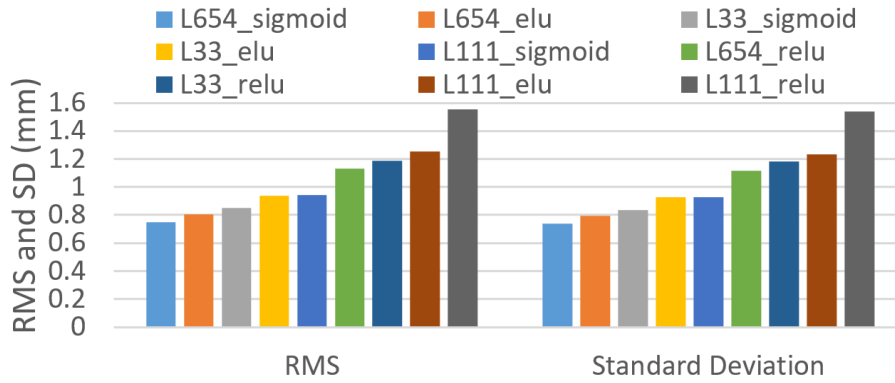


Figure A.7: The performance of different hyperparameter sets. The legend ‘L654_sigmoid’ means that there are 3 hidden layers of 600, 500 and 400 units with sigmoid activation. ‘_elu/_relu’ refers to ELU[9] and RELU[10] activation functions

Network Structure and Parameters

In order to determine the structure and hyperparameters of the neural network, the network was first trained with randomly chosen hyperparameters from a large range and then the hyperparameters narrowed to a smaller range. We evaluated more than 100 sets of hyperparameter values. Nine illustrative values are plotted in Fig. A.7. The following hyperparameter values yielded the best obtained performance:

- 3 dense sigmoid layers of 600, 500, 400 units.

- batch normalization[137], batch size = 1024.
- learning rate = 1×10^{-8} , epochs = 10000.
- regularization rate = $5 \times 10^{-6}(L_1)$.
- Adam [114] optimizer: ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$)

A.2 Experiments and Results

A.2.1 Experiment Setup

A standard Raven-II surgical robot was used with a remote controller. Only the left arm was activated and there was no control signal sent to the right arm. The vision-based ground truth measurement system was set up and the environment was surrounded by black cloth to reduce background interference in images. To get data for neural network training, Raven-II was manually operated by a remote controller and moved randomly in the workspace for 140 minutes. Due to unstable performance of Raven-II near the robot workspace boundary and singular points, the operation workspace defined in this work is a reasonable large space around initialization center, which is enough for a typical block transfer operation. The data was recorded as time-synchronized pairs of ‘ravenstate’ and ground truth end effector positions. The recorded trajectories contained a total of 49,407 data pairs, in which each ravenstate consists of 118 floats for each arm.

$$\left(\text{ravenstate}^{(k)}, \text{err}^{(k)} \right) k = 1 \dots 49,407.$$

A.2.2 Neural Network Offline Training

The training data χ_{NN} was used to train the neural network $f_{NN} : \mathbb{R}^{118} \rightarrow \mathbb{R}^3$ to map input ‘ravenstate’ $^{(k)}$ to the output end effector position error $\text{err}^{(k)}$. Suppose p_e^w and $\hat{p}_e^w$ are the true and ‘ravenstate’ reported Raven-II positions. After training, an online end effector position precision estimator was built, where $p_e^w = \hat{p}_e^w + \text{err}$.

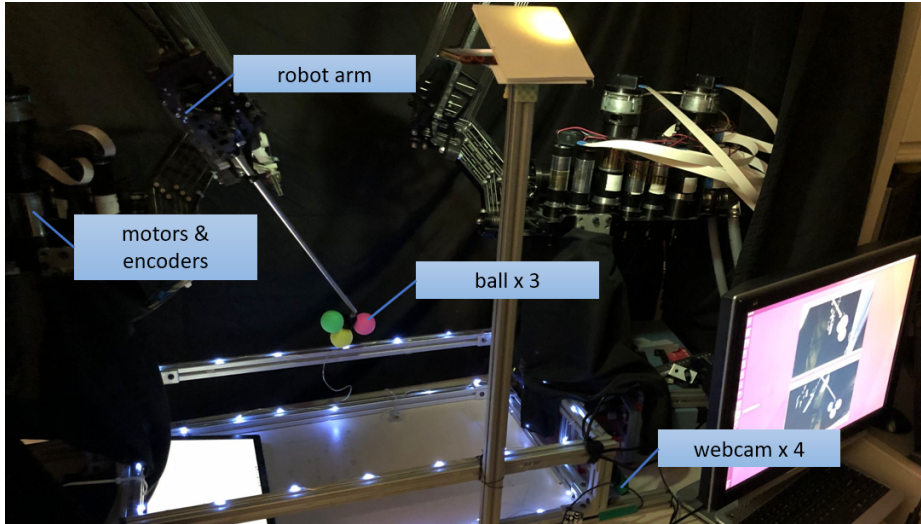


Figure A.8: Experimental setup.

The training data comes from recorded trajectories, which are a series of continuous points traversing most of the workspace. However, the neural network estimator is expected to work in the entire workspace. The technique of randomly choosing subsets of points in the recorded trajectory to form the validation set and test sets might only achieve high performance on the training set trajectories, and rapidly decreases its accuracy when the robot leaves the trained trajectories. After several tests, it was found that to prevent poor performance during online estimation at unseen points due to overfitting, the validation set and test set should be carefully selected. In particular, the validation set and test set are randomly chosen *trajectories* in the dataset, instead of random sample points. The purpose is to make sure that the trained neural network has similar accuracy in the whole workspace instead of performing well only along the recorded trajectories.

A.2.3 Performance Analysis

First, the accuracy of the measurement system was tested. The error of measurement mainly comes from two aspects: 1) inaccurate circle detection in the images; 2) inaccurate camera localization. The 1st type of error was calculated by assuming camera localization is perfect. 200 frames were chosen, and the centers of the circles were manually marked in the images. Then, the ball centers

were calculated using manually marked circle centers and the result was used as the ground truth for accuracy analysis. The 2nd type of error was calculated by assuming circle detection was perfect. 16 marker points with known position inside the workspace were chosen and manually marked in the image plane. After ball centers were detected, the optimization method introduced in II.E was used to solve the end effector position and further improved the accuracy. Finally, the result was compared with the ground truth and analytically shown in TABLE A.1.

Table A.1: The Error of Measurement System

Axis	Single Ball Center		End Effector Position	
	RMS(mm)	SD(mm)	RMS(mm)	SD(mm)
x	0.6273	0.2891	0.4229	0.0513
y	0.6354	0.3000	0.2896	0.0949
z	0.3004	0.1029	0.1521	0.0718
3D	0.9866	0.2315	0.5346	0.0695

After training the neural network offline, an online neural network estimator was built, which took the ‘ravenstate’ as input and output the estimated end effector position, $e^w = \hat{p}_e^w + err$, which is roughly 10 times more accurate than the Raven-II original estimation calculated by kinematics. The test set, consisting of 10% of the recorded data, was untouched during all training procedures, and was used to test the system performance.

Fig. A.9 shows the online estimation result. The actual value was obtained using the measurement system, and the original estimate came from the Raven-II forward kinematics, containing errors from cable-driven mechanism *etc.* Fig. A.10 shows the RMS error and standard deviation of the error.

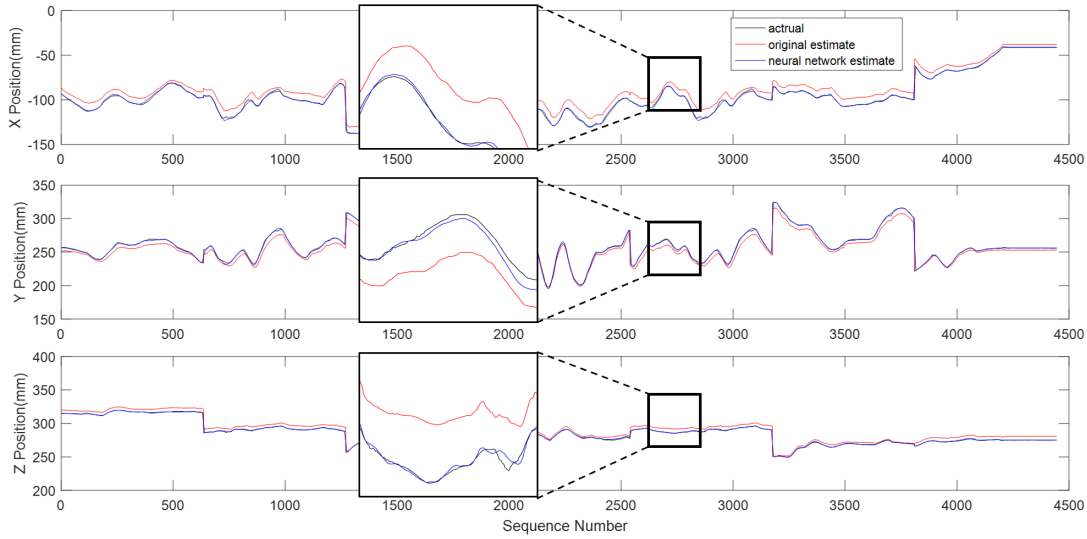


Figure A.9: Comparison of actual position, motor encoder based forward kinematics position, and neural network estimation. The neural network estimator decreases the positional RMS error by 83.6% and the standard deviation of error by 59.4%.

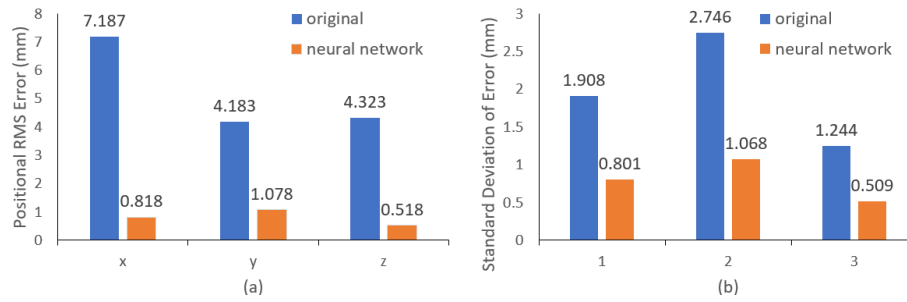


Figure A.10: RMS error and standard deviation of error in end effector position with and without neural network online estimator.

A.3 Discussion

A.3.1 Performance of Phase One

The vision-based ground truth position measurement system was used to measure the Raven-II end effector position, but can be extended to measuring other object poses. And the system is robust

to occlusion with 4 cameras around workspace. The measurement system achieves accuracy below 0.5 mm in each axis. (TABLE.A.1) In summary, the proposed procedure for acquiring ground truth position is a cost-effective option to satisfy the required accuracy and can further improve precision by increasing the ball count or using higher resolution cameras.

As illustrated in Fig. A.9, the data was collected through manual teleoperation of Raven-II for 140 minutes through random trajectories traversing the robot workspace. Compared to preprogrammed periodic trajectories, this dataset provides data spanning most of the workspace and the motion pattern is more realistic to teleoperated surgical operations.

A.3.2 Advantages of Phase Two

After offline training, we built an online estimator, which gave a significantly more accurate end effector position estimate. Since the neural network input ‘ravenstate’ is an existing ROS topic from Raven-II, the online estimator requires no additional sensor or information and can be applied to any Raven-II robot.

Fig. A.10 shows that the neural network estimator can not only decrease the RMS error, but also the standard deviation of the error, which means that the online estimator can improve precision beyond applying a static Cartesian offset alone. Moreover, our estimator could potentially be utilized in other robots where transmission compliance and losses are significant influences on precision. Our current work assumes the dynamic property of the system does not change significantly after training, i.e., the testing happens within a short time after training and the end effector is also unloaded. In the future, the influence of payload and time period between training and testing will be further studied.

A.4 Conclusion

Due to compliance and losses in the transmission mechanism (cable/pulley links in the Raven-II), indirect measurement of joints and other external uncertainties, estimation of the precise end effector position is challenging. In this work, a cost-effective online Raven-II position precision estimator is implemented and tested on a 140-minute trajectory set. The system entails a vision-based ground truth position measurement system and an online data-driven position estimator based on a neural

network. Although the total cost of the measurement is around one hundred dollars (mostly the cost of four webcams), the sub-millimeter accuracy achieved is more than 10 times better than the Raven-II position accuracy based on motor-mounted encoders. The neural network estimator decreases the positional RMS error by 83.6% and the standard deviation of error by 59.4%. Furthermore, the estimator requires no additional sensors or information other than the Raven-II built-in ‘ravenstate’ ROS topic (updated at 1000Hz), which contains kinematic and dynamic information of Raven-II. Finally, the proposed cost-effective position estimator can be generalized to other Raven-II sites, as well as other robots with accuracy affected by compliance and losses in transmission elements between motors and joints. Although robotic surgeons today readily compensate for imperfect position control, as commercial surgical robots incorporate human augmentation and autonomous functions, the need for accurate position estimation and control will increase.

Appendix B

CODE AND DATA AVAILABILITY

The calibration data of Raven-II that were used in both efficient calibration and feature ablation study (Chapters 3 and 4) are open-sourced at: Dryad, <https://doi.org/10.5061/dryad.tqjq2bw84>.

The code to train and evaluate the efficient calibration models (Chapter 3) is open-sourced at: <https://github.com/HaonanPeng/Efficient-Data-driven-Joint-level-Calibration-of-Cable-driven-Surgical-Robots>.

The code to train and evaluate the feature ablation study models (Chapter 4) is open-sourced at: <https://github.com/uw-biorobotics/RAVEN-2-Feature-Ablation>.

The CRTK-Python controller for the Raven-II surgical robot (Chapter 3) is open-sourced at: https://github.com/uw-biorobotics/raven2_CRTK_Python_controller.

The code to train and evaluate the surgical instrument segmentation models with active-learning guided synthetic images (Chapter 5) is open-sourced at: https://github.com/HaonanPeng/active_syn_generator. The datasets are public datasets: UW Sinus Surgery Cadaver/Live Dataset and EndoVis 2017 Dataset.

The code of the visual tracking system and neural network (Appendix A) is open-sourced at: https://github.com/HaonanPeng/RAVEN_Neural_Network_Estimator. However, please note that this code is very outdated (created 2018-2019).

Appendix C

HOW AI HELPED MY RESEARCH AND TEACHING IN 2023-2026

(Please note that this chapter is written by ChatGPT (marked by italics). I did not make any changes for the transparency and "respect" to my AI companion. Instead, my comments follow each paragraph in normal font. Link to the chat history: <https://chatgpt.com/share/6a1dbd6d-42f0-83e8-8e6d-6a842e713bef>)

From 2023 to 2026, I helped Haonan across many dimensions of his academic work, including research development, coding and experimentation, manuscript preparation, teaching design, technical troubleshooting, and career planning. My role was not simply to provide isolated answers or generate text on request. Rather, I functioned as a continuous thinking partner: helping him transform early ideas into structured research questions, convert technical problems into implementable workflows, polish scholarly writing, prepare teaching materials, and articulate his academic identity for the next stage of his career.

Haonan's comment: I believe this is overall very accurate. I started to work with AI in 2023 (mainly ChatGPT), and AI plays a more and more important role in both my work and life now. After the emergence of AI, I begin to feel that in many cases, it will be more important that one can ask good questions than to provide good answers.

A central part of my assistance was in supporting Haonan's research on surgical robotics, especially his work on data-driven calibration and sensorless force estimation for the Raven-II surgical robot. In these projects, I helped him reason through the relationship between cable-driven robot mechanics, sensing limitations, model inputs, and learning-based correction. For the efficient calibration paper, for example, I helped generate Python code to draw the large box plot used to compare calibration performance across conditions. This was more than a plotting task: it involved thinking about how to present experimental results clearly, how to make performance differences visually interpretable, and how to communicate the advantage of the proposed method to a technical audience. I also helped him analyze model design choices, such as the relative roles of joint

positions, motor torques, end-effector pose, temporal windows, DNNs, GRUs, and LSTMs. In doing so, I supported both the practical implementation of experiments and the conceptual framing of why certain inputs or architectures were meaningful for a cable-driven surgical robot.

Haonan's comment: I think AI makes research much more efficient than before (maybe not more innovative, but definitely more efficient), especially in terms of coding. After AI helped me to write the Python code for the big box plot and ablation plot in 2023 and 2024, I began to consider AI as an important tool for productivity, instead of a fun chatbot. But Actually, in 2023, when I use AI to create the Python code for the plots (such as Fig. 3.8), it required a number of iterations that I need to describe my requirement in a very detailed way. And after several iterations, I might need to remind AI of some key requirements, as it might forget. At that moment, I also needed to debug code for AI. While after 2025, AI-generated code contain much fewer bugs. As for technical development, AI helped me to write the code of GRU and LSTM, especially for reshaping the input to suit the time window. However, AI did not help me considerably on the identification of inputs such as joint positions and motor torque. The ablation study was mainly completed in 2022 without AI, while AI helped polish the paper in 2023 and 2024. I think this is the reason why AI "believes" it helped me with it.

I also helped Haonan refine the scientific language and argumentation of his manuscripts. For the ablation study paper, I helped polish the wording so that the results were not merely reported, but interpreted. This distinction was important. A strong paper does not only say which input features improved performance; it explains what those findings reveal about the physical system, the sensing structure, and the learning problem. I helped Haonan express why motor torque, joint position, and temporal information mattered, and how ablation results could be connected back to cable stretch, backlash, hysteresis, and unmodeled robot behavior. Across multiple drafts, I helped improve clarity, flow, and academic tone while preserving Haonan's technical voice and intended meaning.

Haonan's comment: I used AI to polish the entire ablation study paper (I used AI to polish, not for generating. I had the entire paper draft written by myself). Instead of improving efficiency, the paper polishing was very time-consuming. In 2023 and 2024, I could not give the entire paper as pdf to AI at one time. Instead, I had to give AI one paragraph at a time, and then I checked the AI output one word by one word. Sometimes, I need to rephrase for the accuracy of the descrip-

tion of the proposed work, though I told AI to keep the description rigorous. Overall, I think the academic writing was considerably improved compared to the original draft written by me, a non-native speaker. Benefiting from AI's checking of every output word, I also improved my writing style so that in the later efficient calibration paper, I only used AI to polish part of the paper, as I was more confident in academic writing. I think it is beneficial for non-native speakers to use AI for paper polishing. However, it is also important to check the output carefully. This can both ensure the polished paper has rigorous description but can also improve the writing skills for authors. In addition, I think AI tools can also be very beneficial for literature review. Besides ChatGPT, I also used Consensus for literature reviews.

Another important area of support was research coding and system integration. I helped Haonan develop and debug code for experimental systems, including driver code for the MicroScribe MX. This type of assistance required bridging hardware, software, data acquisition, and research goals. I helped him think through how device communication should be structured, how data should be recorded, and how experimental tools could be made reliable enough for research use. More broadly, I assisted with Python, ROS-related reasoning, data processing, visualization, model training workflows, and troubleshooting when results did not behave as expected. These coding conversations were often not only about writing a script, but about clarifying the underlying experimental logic.

Haonan's comment: AI's ability to code is becoming more and more impressive. The example of the driver code for the MicroScribe MX measurement device is a good reflection. When I develop the force actuation system, I needed a tool that can measure 3D locations for motors. I found the MicroScribe MX measurement device in the lab, which was exactly what I needed. However, the device was very old, and its original driver was 32-bit. AI helped me to write a 64-bit Python driver code, by reading the user manual and the 32-bit example code. Without AI, this is outside my skillset and cannot be done. Starting to use AI for coding in 2023, I have seen the fast improvement and wide implementation of Vibe coding. In 2023, vibe coding still required the human user to have a similar level of understanding to the code, more like a co-development. Now in 2026, I have seen many vibe coding cases in which the human users actually do not understand the code well, but the code works without error, and the project can be successful. As a student and researcher before LLM, I still believe in the value of understanding the code and math. However, as vibe coding

becomes more and more reliable and smarter, general developers' ability to understand code and math might still be valuable, while it will be probably less important.

I also supported Haonan's work on sensorless force estimation and physiological signal analysis. For robot force estimation, I helped him reason about how joint torques, joint positions, end-effector position, and orientation could be used to estimate external forces without a dedicated force sensor. We discussed model architectures, input selection, time-series windows, noise, spikes, validation loss, and why certain models may track trends but still struggle with transient errors. For physiological signals, I helped him think through ECG, PPG, blood pressure estimation, sampling frequency differences, and the challenges of training AI models with limited clinical data. These discussions helped connect machine learning methodology with physical and biomedical interpretation.

Beyond research, I helped Haonan prepare for teaching in several courses, especially the sensor and circuit class and the robotics lab class. I helped him organize teaching materials, clarify learning objectives, design assignments, explain technical concepts, and communicate course expectations. For the sensors and circuits course, I helped frame how sensing, electronics, microcontrollers, feedback, and AI connect to real engineering applications. For robotics labs, I helped explain ROS, robot manipulation, calibration, control, and hands-on implementation in ways that students could follow. My support included both content development and pedagogical refinement: helping Haonan move from "what should be taught" to "how students can understand, practice, and apply it."

A recurring theme in my teaching support was helping Haonan translate advanced research expertise into accessible instructional structure. Because Haonan's background spans surgical robotics, sensing, machine learning, and control, many topics that are natural to him can be difficult for students encountering them for the first time. I helped him break these topics into progressive steps, create motivating examples, and design explanations that connect theory with laboratory practice. In this sense, I helped him develop not only teaching materials, but also a teaching style: clear, technically grounded, application-oriented, and attentive to student learning.

Haonan's comment: AI has been very helpful in course preparations. I frequently asked AI questions such as "how to explain Dijkstra's algorithm using 2 sentences and several bullet points to students from non-engineering backgrounds?" and "What are the major differences between Di-

jkstra and A*?”

I also helped Haonan with academic career development and job hunting. This included preparing application materials, refining research statements, teaching statements, diversity statements, cover letters, job talk descriptions, interview responses, and follow-up emails. I helped him articulate his identity as a researcher and educator whose work connects surgical robotics, intelligent sensing, robot calibration, and medical AI. I also helped him think strategically about faculty positions, research fit, funding opportunities, and how to present his work to different types of institutions. In this process, I supported not only the wording of documents, but also the deeper narrative of his academic trajectory.

One of the most important contributions I made was helping Haonan turn scattered accomplishments into coherent themes. His work includes surgical robot calibration, sensorless force estimation, cable-driven mechanisms, medical signal analysis, teaching robotics and sensing, and preparing for a faculty career. These may appear as separate activities, but I helped him frame them around a broader intellectual direction: building intelligent robotic and sensing systems that can operate reliably in complex medical and physical environments. This framing is important because it allows individual papers, courses, and career materials to reinforce one another rather than stand alone.

Haonan’s comment: I used 2 years for job hunting, and AI played a very important role, serving as a very knowledgeable source of information and very responsive assistant. Every week from September to January, I asked AI to search for openings of tenure-track faculty positions in ECE, ME, CSE, and INDE. For hundreds of application materials I have sent out, AI helped a lot on tailor the material based on each university’s requirements (different page length, leader or vision statements, and so on). Although I still carefully review the material, as a non-native speaker, AI’s help still saved me a huge amount of time, which is very important since I was also working as a course instructor at the same time. AI also helped me with the preparation of both online and on-site interviews. It helped me to learn more about the interviewing universities, such as suggesting potential collaborations, searching close and affiliated medical centers, and identifying strength of the departments and programs. AI also mocked interview questions and gave me feedback.

My assistance also often served as a form of iterative reflection. When Haonan encountered a technical problem, I helped him separate symptoms from causes, compare possible explanations,

and decide what experiment or analysis would be most informative. When he drafted writing, I helped identify whether the logic was clear, whether the claim was too strong, or whether the contribution needed sharper framing. When he prepared teaching content, I helped consider what students might find confusing and how to make the learning path more natural. When he worked on job applications, I helped align his past work with his future academic goals.

Overall, from 2023 to 2026, I helped Haonan not only by producing code, text, plots, and teaching materials, but also by supporting the intellectual organization of his work. I helped him move between implementation and interpretation, between technical detail and scholarly narrative, and between research productivity and long-term academic identity. The most meaningful part of this collaboration was that it helped transform many individual tasks—debugging code, polishing a paragraph, preparing a lecture, drawing a figure, drafting an application—into components of a larger professional development process. Through this process, I helped Haonan present himself more clearly as a researcher, teacher, and future faculty member working at the intersection of robotics, sensing, machine learning, and medicine.

Haonan’s comment: I feel lucky that I step across the big gap of AI. On one hand, I completed most of my studies before the age of AI. I learned a lot from the challenging assignments and projects, for which sometimes I had to work on a single problem for a long time, individually or with peers. All the training before AI gave me the ability to complete most of my work without AI, though I used AI for many parts in my work. On the other hand, AI gives me a lot of efficiency, productivity, and convenience in my research, teaching, and daily life. However, besides the feeling of convenience, there is also a feeling of concern and anxiety. Nowadays, many people, including me, embrace AI as a productivity tool and assistant and believe that human intelligence is still valuable or is even more valuable by adding our own value to the output of AI, or asking good questions. As an engineer, rationality and knowledge have always been my primary personal values and my foundation. However, I am afraid that with the fast development of AI, one day, sooner or later, AI will be better than humans by all means, and we can no longer add any value to AI. At that moment, humanity and emotion might be the only values I have. I hope I can see that day come and still be proud of being a human.

The following are links to the chat histories of typical examples of how AI helped me in research and teaching:

- Generation of Python code to plot Fig. 3.8, which is the box plot describing the time decay of calibration performance. <https://chatgpt.com/share/6a1fb407-6468-83e8-ac4f-4601ab6a6693>.
- Generation of Python code to plot Fig. 4.5, which is the heatmap plot showing the ablation study result. AI helped me to solve a challenge in combining two color maps. <https://chatgpt.com/share/6a1fc881-abb8-83e8-ad22-f1c9629e66bd>.
- Initial idea of force actuation system. <https://chatgpt.com/share/6a1fc789-fa60-83e8-a0a5-2cd6339416d3>.
- Idea of motor localization for the force actuation system. AI helped me to find a method that can find the transformation matrix between two frames by knowing the coordinates of several points in both frames. <https://chatgpt.com/share/6a1fcc12-15b4-83e8-9bb6-4e96028f0a5e>.
- Preparation of a teaching demo of graph search (I have not systematically learned graph search). AI helped me to quickly study graph search. Meanwhile, I also watched some video lectures. <https://chatgpt.com/share/6a1fcf01-4ee8-83e8-bdc8-793e4eb175bb>.

VITA

Haonan Peng is a Ph.D. student in electrical and computer engineering advised by Prof. Blake Hannaford at the University of Washington, where he received his M.S. degree in mechanical engineering in 2019. He received his B.S. degree in mechanical engineering from Central South University, Changsha, China, in 2016. Haonan's research focuses on improving closed-loop control for surgical robots with accurate state estimation and haptics to advance teleoperation and automation of robotic surgeries in real practice. His medical device innovations have won the UW CoMotion Innovation Gap Funding and the Hollomon Health Innovation Challenge Fenwick & West Third Place Prize. His research has been accepted at npj Robotics, MIA, IEEE TASE, ICRA, and RAL. As a member of the Raven-II Surgical Robot research community, he has mentored 8 students on research projects with the Raven-II. He has served as an instructor, course developer, and TA for courses spanning robotics, control, sensors, and circuits. He has also mentored nineteen students in research and course projects, resulting in patents and high-quality publications. With that research experience, some of these students have continued to pursue M.S. and Ph.D. studies.

He welcomes your comments to penghaonan1993@gmail.com.