Expanding the Surgical Robotics Community: an Intuitive Sim-to-Real Control Framework for Raven-II with a Budget-Friendly Gamepad Controller

Mai Bui, Natalie Chalfant, Cuiling Sun, Sean Fabrega, Haonan Peng, Kevin Huang, Yun-Hsuan Su

***Abstract*— This paper proposes a low-cost interface and refined digital twin for the Raven-II surgical robot. Previous simulations of the Raven-II, e.g. via the Asynchronous Multibody Framework (AMBF), presented salient drawbacks, including control inputs inconsistent with Raven-II software, and lack of stable, high-fidelity physical contact simulations. This work bridges both of these gaps, both (1) enabling robust, simulated contact mechanics for dynamic physical interactions with the Raven-II, and (2) developing a universal input format for both simulated and physical platforms. The method furthermore proposes a low cost, commodity game-controller interface for controlling both virtual and real realizations of Raven-II, thus greatly reducing the barrier to access for Raven-II research and collaboration. Overall, this work aims to eliminate the inconsistencies between simulated and real representations of the Raven-II. Such a development can expand the reach of surgical robotics research. Namely, providing end-to-end transparency between the simulated AMBF and physical Raven-II platforms enables a software testbed previously unavailable, e.g. for training real surgeons, for creating digital synthetic datasets, or for prototyping novel architectures like shared control strategies. Experiments validate this transparency by comparing joint trajectories between digital twin and physical testbed given identical inputs. This work may be extended and incorporated into recent efforts in developing modular or common software infrastructures for both simulation and control of real robotic devices, such as the Collaborative Robotics Toolkit (CRTK).**

***Index Terms*— Raven-II; AMBF; teleoperation; robot-assisted MIS; digital twin; human computer interfaces; CRTK.**

I. Introduction

A. Background

1) Surgical Robot Research Platforms: As medical robots take on more roles in the operating room and clinic, it is important to provide robust and accessible resources for research institutions to test and develop new tools and features. One of the most widely used systems is the da Vinci surgical robot, with system cost of 2 million dollars, restricted access to software and electronics, and limited distribution to select or certified medical facilities [1]. While the da Vinci Research Kit (dVRK) robot offers an alternative with open-source software and electronics [2], it remains highly selective regarding access and use. On the other hand, the Raven-II surgical robot was developed to provide common software and hardware environments for research innovations at a more affordable cost of around 250 thousand dollars, making it a more accessible option for research institutions and labs [3]. In other efforts to grow the surgical robotics research community, simulators such as the Asynchronous Multibody Framework (AMBF) [4] can bridge the gap by providing simluation capability to various robots in a variety of realistic physics-enabled environments. Such digital test beds can be used for rapid prototyping, experimentation and development of novel research features without requiring a physical device. These simulators greatly lower the hardware accessibility barrier and enable more ready involvemment in medical robotics research [5].

2) Robot Controller and Joysticks: In paired simulated and physical teleoperated robots, typically the same interfaces control both platforms [6]. However, many existing surgical robot operator interfaces, such as the daVinci surgical console, Microsurge, and Force Dimension exhibit costly custom platforms with integrated displays and highly articulated controllers [7], [8]. More affordable options such as the Phantom Omni can still be cost prohibitive at a price tag of about 1000 US dollars [9]. CRTK-based keyboard controllers like [10]–[12], grant manipulation capabilities to Raven-II with minimal additional expenses, requiring only a computer and simple software setup. However, they are not without limitations, sending only per joint or per Cartesian axis commands, making it cumbersome to maneuver the robot intuitively. In a continuous effort to promote collaboration and prevent redundant development, the proposed framework utilized the keyboard controllers as starter code.

3) Common Software Infrastructures: The Collaborative Robotics Toolkit (CRTK) is a unified API designed to transport robot commands and feedback, supporting teleoperation and coordinated control tasks. Its primary objective is to streamline research and education in cutting-edge human-robot collaborative fields, including semi-autonomous teleoperation and medical robotics. CRTK has been integrated into both the AMBF simulator and Raven-II and dVRK software, providing physics-enabled simulation support and compatibility with these platforms [13].

This material is based upon work supported by the National Science Foundation under Grant No. IIS-2101107. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

Mai Bui, Natalie Chalfant, Cuiling Sun, Sean Fabrega, and Yun-Hsuan Su are affiliated with Mount Holyoke College, Department of Computer Science, 50 College St, South Hadley, MA 01075 {bui23m, chalf22n, sun25c, fabre22s, msu}@mtholyoke.edu

Haonan Peng is affiliated with the University of Washington, Department of Electrical and Computer Engineering, 1410 NE Campus Pkwy, Seattle, WA 98195. penghn@uw.edu

Kevin Huang is with Smith College, Picker Engineering Program, 100 Green Street, Northampton, MA 01063. kevinhuang@smith.edu

B. Motivation

This research ultimately aims to create a robust simulation framework of the Raven-II robotic platform in AMBF that: (a) enables intuitive manipulation of the robot end-effector with little controller budget and usage barrier; (b) minimizes configuration and performance discrepancies between the simulated and physical platforms; and (c) is easy to translate a simulated teleoperation sequence to the physical Raven-II hardware for testing. The authors believe the proposed framework would greatly broaden the Raven-II surgical robotics research community, create potential surgical robotics curriculum design, and open the door to numerous outreach opportunities for under-resourced or K-12 institutions.

C. Contributions

Figure 1 shows an overview of the proposed sim-to-real control framework for Raven-II. To the best of the authors' knowledge, we are the first to simultaneously achieve:

- Design of two gamepad controller mapping schemes for Raven-II: two-arm partial mapping ($\mathcal{M}_2$) and one-arm full mapping ($\mathcal{M}_1$).
- Refactorization of the Raven-II kinematics to achieve a specialized Cartesian control with wrist locking ($\mathcal{C}_\mathcal{W}$).
- Motion interpolation to achieve smooth trajectories with a maximum velocity threshold ($\mathcal{C}_\mathcal{V}$) when the rate of raw controller setpoints is unreliable or infeasible.
- Easy motion replication and simultaneous dual platform operation between the AMBF-simulated ($\mathcal{R}_\mathcal{S}$) and physical ($\mathcal{R}_\mathcal{P}$) Raven-II through recorded gamepad controller inputs.
- Evaluation of trajectory consistency on two experimental datasets: Peg Transfer ($\mathcal{D}_\mathcal{I}$) and Wide Range ($\mathcal{D}_\mathcal{II}$).
- Open source [14] for the AMBF and Raven-II research community. (A demo video included.)

II. Methods

A. Dual Platform Controller: Software Architecture

To facilitate the intuitive sim-to-real transfer, the controller is able to relay ROS commands to either Raven proxies ($\mathcal{R}_\mathcal{S}$ or $\mathcal{R}_\mathcal{P}$) with a change of a platform flag upon startup. As shown in Table I, there are five operation modes, which demonstrate varying levels of interactiveness. These range from hard-coded motion ($\mathcal{O}_1$, $\mathcal{O}_5$), trajectory replay from recorded CSV files ($\mathcal{O}_3$, $\mathcal{O}_4$), to live teleoperation via a gamepad controller ($\mathcal{O}_2$).

1) Runtime Optimization: One essential component of real-time control is to ensure quick responsiveness of the robot to user inputs. To this end, the controller is designed to conduct three main tasks in parallel through multi threading: one to collect operation mode choices from the keyboard, one to process gamepad controller inputs using the Python Inputs Library, and one controller main thread to publish ROS topics to the robot proxy ($\mathcal{R}_\mathcal{S}$ or $\mathcal{R}_\mathcal{P}$) at a constant rate. This allows the the keyboard and gamepad control inputs to reach the controller at a constant update rate of 500 Hz with Ubuntu 20.04 LTS running on an AMD Ryzen 5800x.

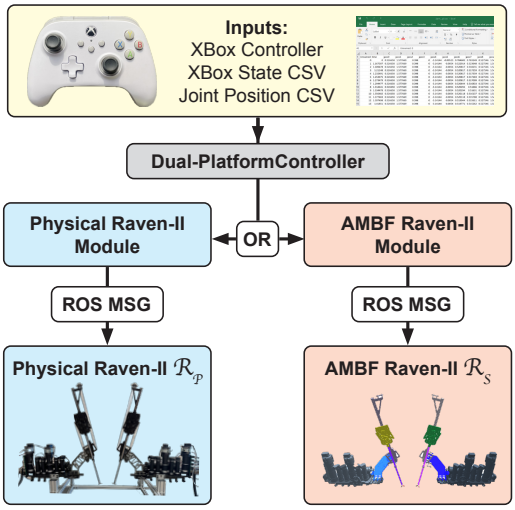

Fig. 1. The Sim-to-Real Control Framework for Raven-II.

2) The CSV Recorder (mode: $\mathcal{O}_2$, $\mathcal{O}_3$): The CSV recorder is a versatile tool that captures the time-stamped statuses of Raven-II and the gamepad controller and saving in CSV format. It can be initiated and stopped at any time during $\mathcal{O}_2$ or at the beginning of $\mathcal{O}_3$. When recording on the physical robot ($\mathcal{R}_\mathcal{P}$), the CSV contains all data from the ravenstate ROS topic. For AMBF ($\mathcal{R}_\mathcal{S}$), the CSV structure remains identical. However, due to the innate differences in dynamics calculation between the two platforms, only the time, joint position, joint velocity, and end effector Cartesian pose columns are populated – all others are designated NaN as placeholders. The time column reflects the elapse time since the start of recording. During recording, each command is added as a new row in the CSV at approximately every 70ms until recording is stopped, unless the velocity cap $\mathcal{C}_\mathcal{V}$ is reached and the interpolator extends the time required to complete the command (more in Section II-B.4).

Idx	Operation Mode	Description
$\mathcal{O}_1$	Homing Mode	Returns the robot to its specified home position
$\mathcal{O}_2$	Manual Control	Cartesian control of the end effector using a gamepad controller
$\mathcal{O}_3$	File Controller Inputs	Cartesian control of the end effector using a CSV of recorded XBOX controller states
$\mathcal{O}_4$	File JPos	Joint level control using a CSV of recorded joint positions
$\mathcal{O}_5$	Sine Dance	Moves all joints according to the sine function (currently only enabled for the AMBF Raven-II)

TABLE I

Dual-Platform Controller Modes

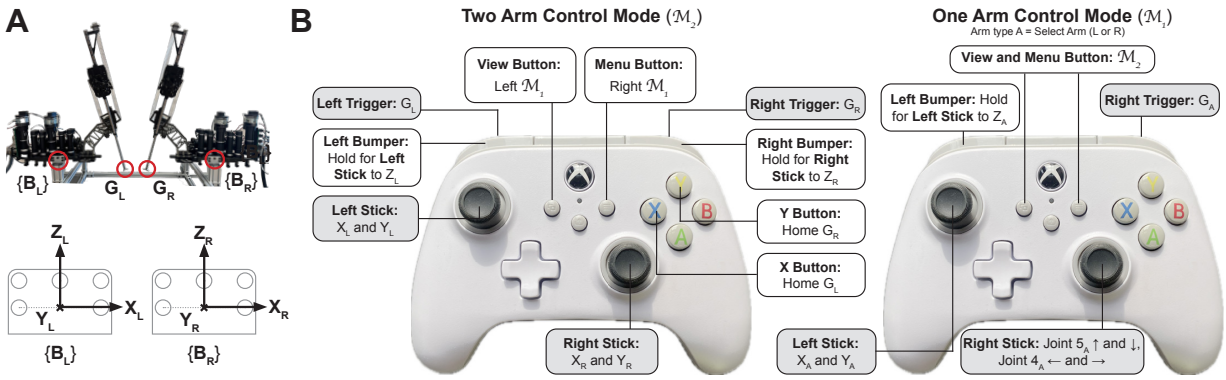

Fig. 2. **A**: Raven-II base frame $\{B\}$ and grasper G of each arm. **B**: The proposed gamepad control mapping schemes: the one arm full mapping scheme $\mathcal{M}_1$ and the two arm partial mapping scheme $\mathcal{M}_2$. The continuous control inputs are tinted in gray, whereas binary ones are not.

3) Trajectory Replay (mode: $\mathcal{O}_3$, $\mathcal{O}_4$): Both $\mathcal{R}_\mathcal{P}$ and $\mathcal{R}_\mathcal{S}$ can replicate CSV pre-recorded trajectories by following either the measured joint positions ($\mathcal{O}_4$) or recorded gamepad controller commands ($\mathcal{O}_3$). Therefore, sim-to-real cross platform trajectory replication can be achieved if one runs the CSV recorder in $\mathcal{O}_2$ mode on $\mathcal{R}_S$, and then conducts $\mathcal{O}_3$ with the recorded file on $\mathcal{R}_P$.

B. The Gamepad Controller: Teleoperation

1) Controller Mapping Schemes ($\mathcal{M}_1$ and $\mathcal{M}_2$): While a gamepad controller is affordable and ubiquitous, utilizing one to control a high degree of freedom manipulator such as Raven-II presents challenges. A gamepad controller typically has two sticks and two triggers, totaling 6 continuous control DOFs, whereas just the end effector configuration of each of the Raven-II arms are described by 6DOFs - describing both end effectors requires 12 DOFs.

In order to address this, modifiers that combine control inputs can enable mapping of gamepad inputs to higher dimensional devices. To work within the aforementioned limitations, two controller mapping schemes - $\mathcal{M}_2$ and $\mathcal{M}_1$ - are designed and depicted in Fig.2.

2) Handling Controller Stick Drift: The precision of gamepad controller joysticks can vary, and the joystick positions rarely return to exactly (0,0), or true center. This causes a subtle, yet consistent drifting motion when the controller is untouched. To accommodate for the undesirable drift during teleoperation, a deadzone is incorporated into the controller to ignore any perceived joystick readings with magnitude, $\sqrt{J_h^2 + J_v^2}$, less than a heuristically determined threshold of 0.15, where J_h, J_v are the raw joystick readings in the horizontal and vertical axes ranging from -1 to 1.

3) Wrist Locking Kinematics ($\mathcal{C}_\mathcal{W}$): shown in Fig.3, were developed to resolve insufficient number of independent continuous control inputs of the gamepad controller.

- In $\mathcal{M}_2$: for each arm, the position of the joint 5 frame is controlled by 3 gamepad DOFs. The revolute DOFs of by joints 4, 5, and 6 are locked at their home positions. The gripper, (G_L, G_R), is controlled separately.
- In $\mathcal{M}_1$: More complex manipulations are enabled with fine-grained direct control of joints 4,5, and 7 implemented – only the wrist joint 6 remained locked.

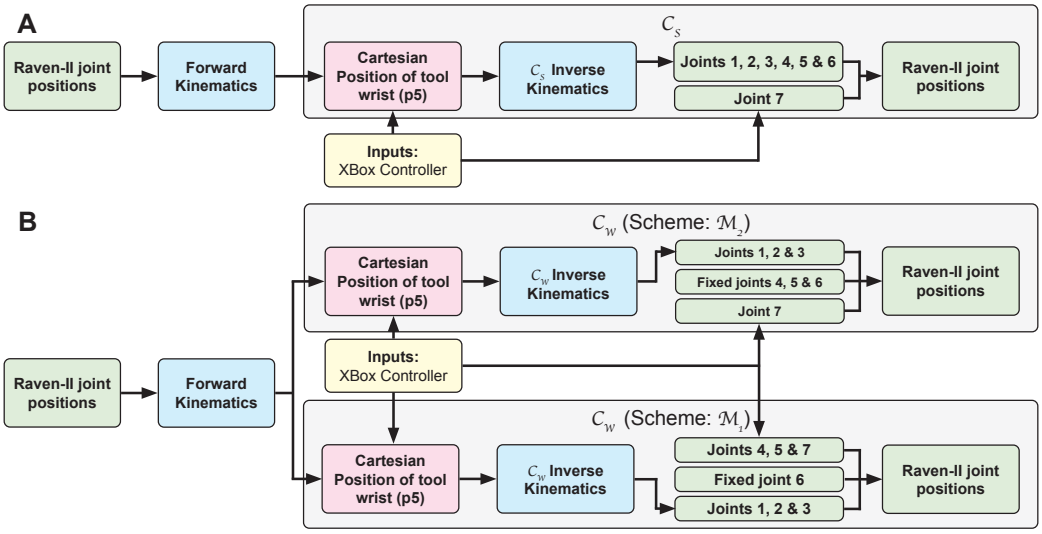

Fig. 3. **A**: Standard Kinematics ($\mathcal{C}_\mathcal{S}$) **B**: Wrist Locking Kinematics ($\mathcal{C}_\mathcal{W}$) under the two arm mapping scheme ($\mathcal{M}_2$) with wrist locking for joints 5, 6, & 7; and the one arm mapping scheme ($\mathcal{M}_1$) with direct joint level control of joints 5 & 6.

4) Motion Interpolation with Joint Velocity Cap ($\mathcal{C}_\mathcal{V}$): To ensure teleoperation safety and motion consistency between $\mathcal{R}_\mathcal{S}$ and $\mathcal{R}_\mathcal{P}$, linear interpolation was applied to maintain all commands within experimentally determined joint velocity limits $\mathcal{C}_\mathcal{V}$. Inspired by the concept of "interpolate" in CRTK [13], large motion commands that exceed $\mathcal{C}_\mathcal{V}$ are decomposed into smaller ones, while extending the time required to complete the commanded distance (Fig.5). In this study, $\mathcal{C}_\mathcal{V}$ is set as 0.087 (rad/s) for joints 1,2; 0.262 (rad/s) for joints 4-7; and 0.02 (m/s) for joint 3.

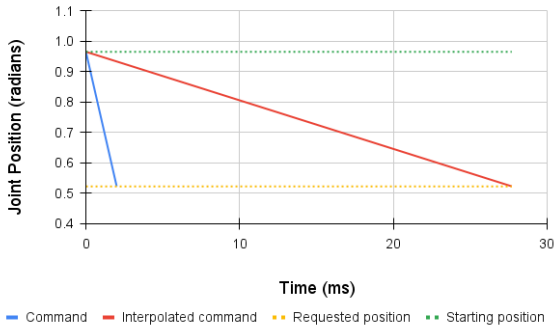

Fig. 5. Demonstration of joint 1 transitioning from 0.96 to 0.52 radians in approximately 2 ms without caps and in 27.7 ms with velocity caps.

C. Closing the Performance Gap: $\mathcal{R}_\mathcal{S}$ v.s. $\mathcal{R}_\mathcal{P}$

1) Collision Mesh Simplification for $\mathcal{R}_\mathcal{S}$: Previously, the simulated Raven-II could only operate in open space. Whenever a collision with an external object occurred, the AMBF physics engine update frequency would drop from the target 200 Hz to around 40 Hz (Fig.4-F). This was caused by the default collision mesh being directly calculated from the original Raven-II 3D model (Fig.4-A), which contains 204,746 vertices. To reduce collision calculation overhead, each robot link is coarsely approximated by a minimum bounding cube or cylinder. Meanwhile, the grasper collisions were simplified in blender using a combination of manual modifications and the decimate modifier. This resulted in a salient reduction of the simplified mesh to 384 vertices (Fig.4-B), and enabled interactions between $\mathcal{R}_\mathcal{S}$ with both soft or rigid objects (Fig.4-D and E) while maintaining a realtime dynamic loop frequency of 150-200 Hz (Fig.4-F).

Parameter (m)	$\mathcal{R}_\mathcal{P}$	$\mathcal{R}_\mathcal{S}$
DH value: d_4	-0.47	0.0
Home pos: Joint 1	$\pi/6$	$7\pi/36$
Home pos: Joint 3	0.4	-0.07
Limits: Joint 3	$0.23 \leq x \leq 0.56$	$-0.23 \leq x \leq 0.1$

TABLE II

A SUMMARY OF THE PARAMETER DISCREPANCIES ($\mathcal{R}_\mathcal{P}$ AND $\mathcal{R}_\mathcal{S}$)

2) Kinematic Parameter Discrepancies: In this study, robot kinematics and Denavit-Hartenberg parameters were initially adopted from Su's C++ controller [11]; the home joint positions and joint limits were obtained from the Raven-II code [15]. However, even though $\mathcal{R}_\mathcal{S}$ shares a structurally identical and to-scale appearance as $\mathcal{R}_\mathcal{P}$, it is not actuated in a cable-driven fashion – joint level coordinate frames differed between the two kinematic models. Table II summarizes additional modifications necessary in order to remove the performance discrepancies between $\mathcal{R}_\mathcal{S}$ and $\mathcal{R}_\mathcal{P}$.

3) Graphical Disparity Analysis Tool: A comprehensive data analysis tool that visualizes the robot trajectory CSV files, highlighting performance disparities between $\mathcal{R}_\mathcal{S}$ and $\mathcal{R}_\mathcal{P}$. By providing insights into joint errors and various correlating factors, developers can effectively fine-tune parameters (including setup configurations and command rates) to minimize the performance gap based on their specific computing environments. Therefore, this tool not only facilitates the authors' research in the development phase, but also serves as a valuable resource for future users seeking to refine their codes. Currently under development, the authors aim to finalize it, making it a valuable addition to the proposed research toolkit in the near future.

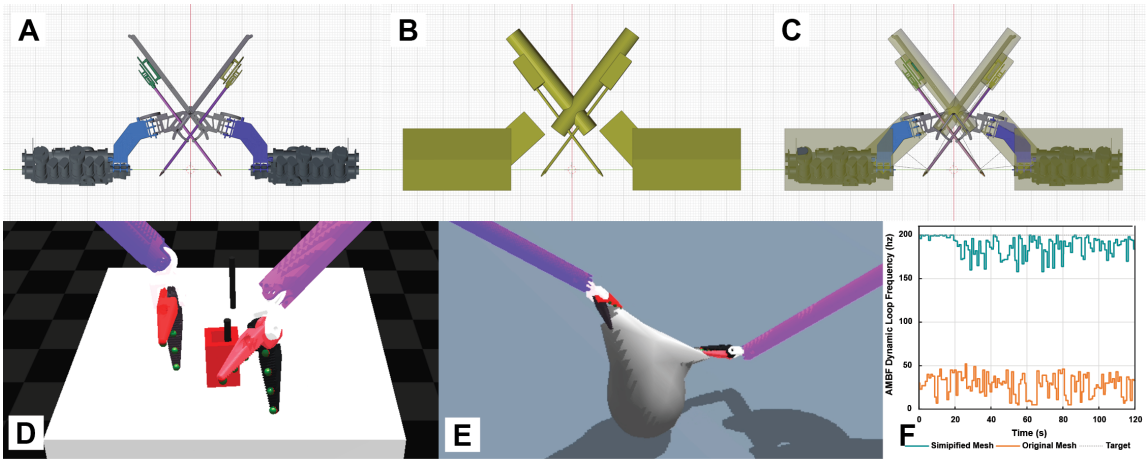

Fig. 4. Collision mesh simplification for AMBF simulated Raven-II $\mathcal{R}_\mathcal{S}$. **A**: Original mesh with 204,746 vertices, **B**: Simplified mesh with 384 vertices that demonstrates a closer collision approximation around the ROI (tool tip), **C**: Simplified collision mesh overlayed on $\mathcal{R}_\mathcal{S}$. **D,E**: $\mathcal{R}_\mathcal{S}$ successfully interacts with objects after collision mesh simplification. **F**: A comparison of the AMBF Dynamic Loop Frequency before and after collision mesh simplification in a soft body interaction experiment conducted on virtualized Ubuntu 20.04 LTS running in Parallels on Apple M1 max, with a target frequency of 200Hz.

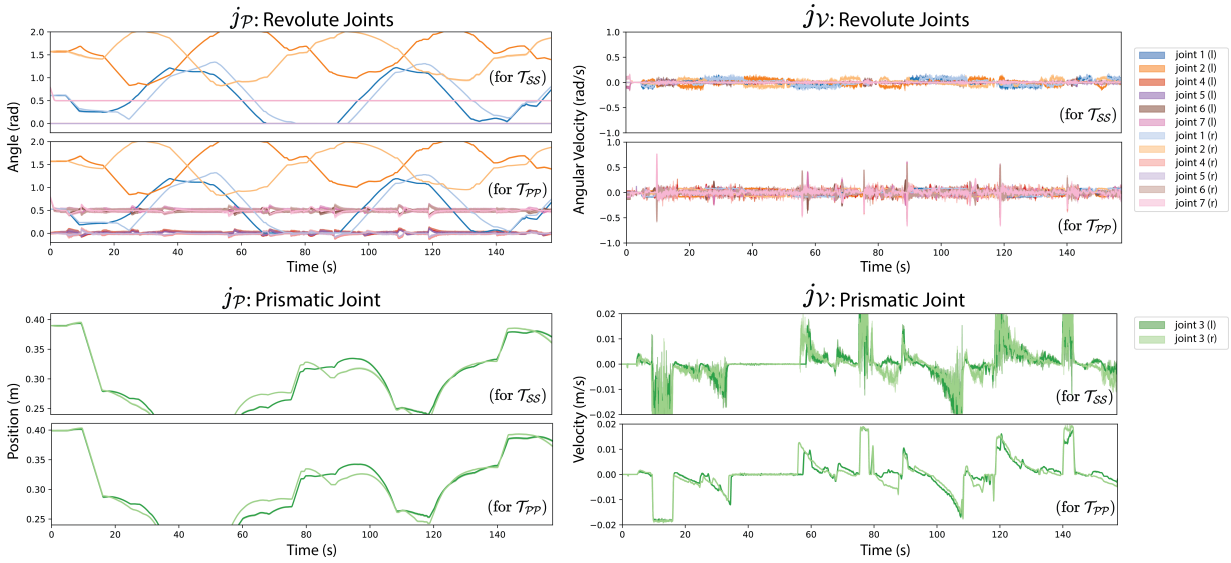

Fig. 6. The time stamped $j_{\mathcal{P}}$ (left) and $j_{\mathcal{J}}$ (right) trajectories for $\mathcal{T}_{\mathcal{SS}}$ and $\mathcal{T}_{\mathcal{PP}}$ in task $\mathcal{D}_{II}$. The revolute (top) and prismatic (bottom) joints are separately displayed and color coded. The area between the maximum and minimum joint values across the three trials are filled.

III. Experiments and Results

A. Experimental Design

The gamepad control framework and cross platform consistency were assessed through trajectory replication from two tasks: peg transfer $\mathcal{D}_{\mathcal{I}}$ and wide range motion $\mathcal{D}_{\mathcal{II}}$.

1) Original Trajectories: All original trajectories were commanded using $\mathcal{O}_2$. The $\mathcal{D}_{\mathcal{I}}$ task involved a single peg transfer from one post to another, and original trajectories were recorded using three different gamepad mapping schemes: (1) $\mathcal{M}_2$, (2) $\mathcal{M}_1$ of the left arm, and (3) $\mathcal{M}_1$ of the right arm. The $\mathcal{D}_{\mathcal{II}}$ wide range motion task used gamepad mapping $\mathcal{M}_2$, and both arms were moved along a rectangular path on an XY plane.

2) Playback Trajectories: The four aforementioned trajectories were recorded using the method described in II-A.2. For each original trajectory, $\mathcal{O}_3$ is used three times to execute the recorded trajectory on both $\mathcal{R}_{\mathcal{P}}$ and $\mathcal{R}_{\mathcal{S}}$. This results in a total of 24 playback trajectories.

B. Data Analysis

For simplicity of notation, $\mathcal{T}_{\mathcal{SS}}$, $\mathcal{T}_{\mathcal{PP}}$, $\mathcal{T}_{\mathcal{PS}}$ are respectively used to denote consistency analysis among the $\mathcal{R}_{\mathcal{S}}$ trials, the $\mathcal{R}_{\mathcal{P}}$ trials, and between the $\mathcal{R}_{\mathcal{P}}$ and $\mathcal{R}_{\mathcal{S}}$ trials. The velocity and position of joint i are represented as $j_{\mathcal{V}i}$ and $j_{\mathcal{P}i}$.

1) Single Platform Repeatability ($\mathcal{D}_{II}$): Figure 6 reveals $j_{\mathcal{P}}$ and $j_{\mathcal{V}}$ trajectories for $\mathcal{T}_{\mathcal{SS}}$ and $\mathcal{T}_{\mathcal{PP}}$ on task $\mathcal{D}_{II}$. Overall, $j_{\mathcal{P}}$ presents more consistent results than $j_{\mathcal{V}}$; $\mathcal{R}_{\mathcal{S}}$ presents more consistent results than $\mathcal{R}_{\mathcal{P}}$. Additional findings include:

- $j_{\mathcal{P}1-3}$ show similar consistency between $\mathcal{T}_{\mathcal{SS}}$ and $\mathcal{T}_{\mathcal{PP}}$, but $j_{\mathcal{P}4-7}$ are significantly less consistent for $\mathcal{T}_{\mathcal{PP}}$.
- Between $\mathcal{T}_{\mathcal{SS}}$ and $\mathcal{T}_{\mathcal{PP}}$, the former experienced more damping in the prismatic joint ($j_{\mathcal{V}3}$), and the latter presents more jitter in the revolute joints ($j_{\mathcal{V}1,2,4-7}$).

2) Cross Platform Consistency ($\mathcal{D}_I$ and $\mathcal{D}_{II}$): The average $j_{\mathcal{P}}$, $j_{\mathcal{V}}$ trajectory error bar charts for $\mathcal{T}_{\mathcal{SS}}$, $\mathcal{T}_{\mathcal{PP}}$, and $\mathcal{T}_{\mathcal{PS}}$ are shown in Fig.7. Results are encouraging, as positioning errors were under 2 mm and .04 rad for prismatic and revolute joint respectively. Similarly, velocities deviated by less than 2 mm/s and 0.11 rad/s. Additional insights include:

- Within-platform consistency shows better prismatic joint tracking on the physical platform, $\mathcal{T}_{\mathcal{PP}}$, and better revolute joint tracking on the simulated, $\mathcal{T}_{\mathcal{SS}}$.
- Cross platform error for revolute joints track $\mathcal{T}_{\mathcal{PP}}$.
- Cross platform error magnitude for prismatic joints track $\mathcal{T}_{\mathcal{SS}}$, yet position-velocity error ratios with $\mathcal{T}_{\mathcal{PP}}$.

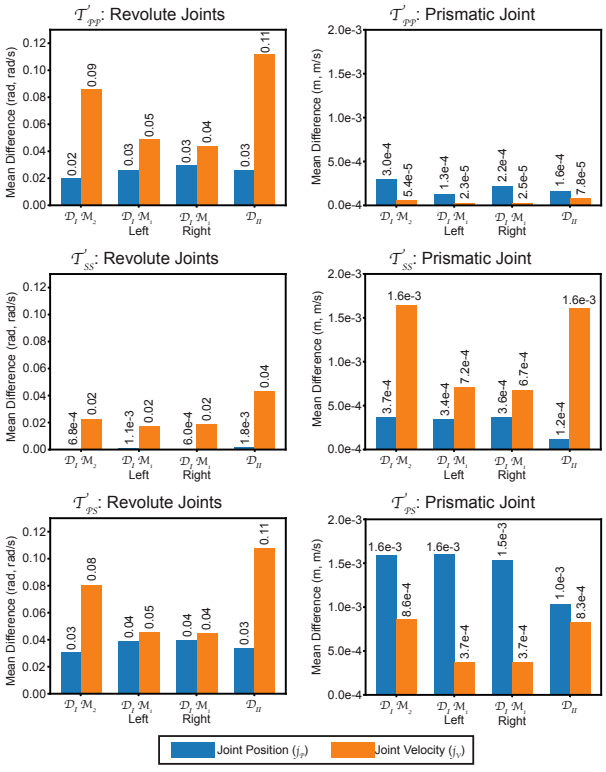

Fig. 7. Inconsistency bar charts averaging across joints and time.

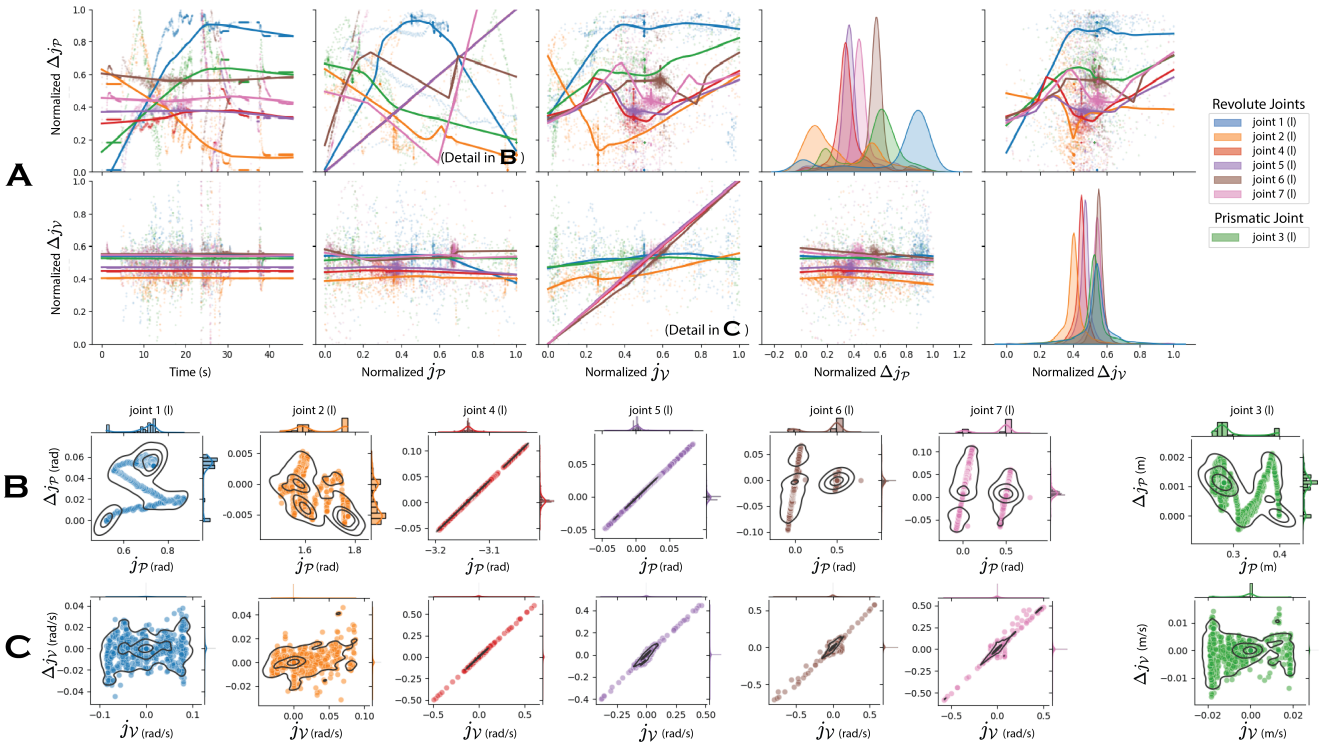

Fig. 8. **A** displays the normalized correlation and the optimal LOWESS regression curves of $j_{\mathcal{P}}$, $j_{\mathcal{V}}$ with a variety of potential impacting factors for $\mathcal{T}_{\mathcal{PS}}$. **B** (and **C**) further presents correlation graphs between $j_{\mathcal{P}}$ (and $j_{\mathcal{V}}$) and its cross platform difference $\Delta j_{\mathcal{P}}$ (and $\Delta j_{\mathcal{V}}$) with kernel density estimates (black curves), and marginal distributions. All variables (except time) in **A** are normalized to 0-1.

3) Correlating Factors that Impact Consistency ($\mathcal{D}_I$): Figure 8 aims to identify prominent correlating factors for inconsistencies in $j_{\mathcal{P}}$ and $j_{\mathcal{V}}$ with $\mathcal{T}_{\mathcal{PS}}$. From Fig.8-**A**:

- $j_{\mathcal{P}}$ and $\Delta j_{\mathcal{P}}$ display no correlation with $\Delta j_{\mathcal{V}}$.
- Time is not correlated with $\Delta j_{\mathcal{V}}$, but correlated with $\Delta j_{\mathcal{P}}$ although the correlation pattern is joint dependent.
- $\Delta j_{\mathcal{P}}$ does not affect $\Delta j_{\mathcal{V}}$, $\Delta j_{\mathcal{V}}$ positively affects $\Delta j_{\mathcal{P}}$.
- $j_{\mathcal{V}}$ has a subtle impact on $\Delta j_{\mathcal{P}}$. although correlations for j_1, j_3 don't appear linear.

Additional conclusions can be drawn from Fig.8-**B**,**C** that:

- Strong positive correlations are observed for $j_{\mathcal{P}4,5}$ and $j_{\mathcal{V}4-7}$ with their respective cross platform differences - $\Delta j_{\mathcal{P}4,5}$ and $\Delta j_{\mathcal{V}4-7}$.
- The correlation between $j_{\mathcal{V}}$ and $\Delta j_{\mathcal{V}}$ presents no offset, but it's not the case for $j_{\mathcal{P}}$ and $\Delta j_{\mathcal{P}}$.
- $j_{\mathcal{P}6,7}$ has segmented positive correlations with $\Delta j_{\mathcal{P}6,7}$.
- $\Delta j_{\mathcal{P}1,3}$ presents bimodal distributions.

IV. Conclusion

A. Summary

In this study, a low cost control framework was developed for the Raven-II surgical robot physical, $\mathcal{R}_{\mathcal{P}}$, and AMBF simulated, $\mathcal{R}_{\mathcal{S}}$, platforms. Extensive efforts were made to minimize the performance gap with trajectory following and replication. Figures 6-8 reveal an overall consistent motion across $\mathcal{T}_{\mathcal{PP}}$, $\mathcal{T}_{\mathcal{SS}}$ and $\mathcal{T}_{\mathcal{PS}}$. Specifically, mean differences for the revolute $j_{\mathcal{P}}$ emerged at 0.03, 0.0018 and 0.03 radians respectively for the three $\mathcal{T}$ categories in $\mathcal{D}_{\mathcal{II}}$. The mean deviations for primsatic joint $j_{\mathcal{P}3}$ were 0.16, 0.12 and 1 mm. For within-platform consistencies, the revolute and prismatic joints respectively showcased more consistent results in $\mathcal{T}_{\mathcal{SS}}$ and $\mathcal{T}_{\mathcal{PP}}$. Inconsistencies in $\mathcal{T}_{\mathcal{SS}}$ often resulted from intermittent high-intensity oscillations, whereas inconsistencies in $\mathcal{T}_{\mathcal{PP}}$ appear as persistent moderate jitters (Fig.6). Lastly, correlations between cross platform consistencies $\mathcal{T}_{\mathcal{PS}}$ and various impacting factors were discussed.

B. Future Directions

Currently, simultaneous control is achieved by running two separate instances of the controller, one for $\mathcal{R}_{\mathcal{P}}$ and one for $\mathcal{R}_{\mathcal{S}}$ both in $\mathcal{O}_2$. As isolated instances, there is no way to check and maintain synchronization, which is evident in the relatively larger $\Delta j_{\mathcal{V}}$ than $\Delta j_{\mathcal{P}}$. Further improvements can be made to enable synchronous control of both $\mathcal{R}_{\mathcal{P}}$ and $\mathcal{R}_{\mathcal{S}}$ using a single controller instance. Additionally, the authors will further explore the capabilities of AMBF soft body interactions and to what extent $\mathcal{R}_{\mathcal{S}}$ can be used to predict applied forces on $\mathcal{R}_{\mathcal{P}}$ when interacting with deformable objects. Even though $\mathcal{R}_{\mathcal{S}}$ shares a structurally identical and to-scale appearance as $\mathcal{R}_{\mathcal{P}}$, it is not actuated in a cable-driven fashion. This makes raw joint torque estimations in $\mathcal{R}_{\mathcal{S}}$ incomparable with $\mathcal{R}_{\mathcal{P}}$. The authors will look into active learning based sim-to-real approaches to minimize this gap. Lastly, to broaden the impact of the intuitive sim-to-real gamepad control framework for Raven-II, the authors will: (a) conduct human experiments and optimize the mapping schemes based on user feedback; (b) incorporate this framework in future outreach opportunities; and (c) contribute the developed source code to the AMBF GitHub page [4].

REFERENCES

[1] E. Singer, "The slow rise of the robot surgeon," Apr 2020. [Online]. Available: https://www.technologyreview.com/2010/03/24/122356/the-slow-rise-of-the-robot-surgeon/

[2] P. Kazanzides, Z. Chen, A. Deguet, G. S. Fischer, R. H. Taylor, and S. P. DiMaio, "An open-source research kit for the da vinci® surgical system," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 6434–6439.

[3] B. Hannaford, J. Rosen, D. W. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. N. Kosari, and L. White, "Raven-ii: An open platform for surgical robotics research," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 4, pp. 954–959, 2013.

[4] A. Munawar and G. S. Fischer, "An asynchronous multi-body simulation framework for real-time dynamics, haptics and learning with application to surgical robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6268–6275.

[5] K. Kunkler, "The role of medical simulation: an overview," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 2, no. 3, pp. 203–210, 2006.

[6] K. Huang, D. Subedi, R. Mitra, I. Yung, K. Boyd, E. Aldrich, and D. Chitrakar, "Telelocomotion—remotely operated legged robots," *Applied Sciences*, vol. 11, no. 1, p. 194, 2020.

[7] H. Ashrafian, O. Clancy, V. Grover, and A. Darzi, "The evolution of robotic surgery: surgical and anaesthetic aspects," *BJA: British Journal of Anaesthesia*, vol. 119, no. suppl_1, pp. i72–i84, 2017.

[8] A. Tobergte, P. Helmer, U. Hagn, P. Rouiller, S. Thielmann, S. Grange, A. Albu-Schäffer, F. Conti, and G. Hirzinger, "The sigma. 7 haptic interface for mirosurge: A new bi-manual surgical console," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3023–3030.

[9] A. Simorov, R. S. Otte, C. M. Kopietz, and D. Oleynikov, "Review of surgical robotics user interface: what is the best way to control robotic surgery?" *Surgical Endoscopy*, vol. 26, no. 8, pp. 2117–2125, Aug 2012. [Online]. Available: https://doi.org/10.1007/s00464-012-2182-y

[10] H. Peng, "Haonanpeng/raven2_crtk_python_controller: A python keyboard controller for raven-ii surgical robot, based on the crtk api," Jun 2018. [Online]. Available: https://github.com/HaonanPeng/raven2_CRTK_Python_controller

[11] Y.-H. Su, "Ambf_raven-ii_c++_controller: A c++ keyboard controller for ambf raven-ii, based on the crtk api." [Online]. Available: https://github.com/WPI-AIM/ambf/tree/ambf-2.0/ambf_controller/raven2

[12] S. Fabrega, "Ambf_raven-ii_python_controller: A python keyboard controller for ambf raven-ii, based on the crtk api." [Online]. Available: https://github.com/MHC-RobotSimulators-Research/ambf/tree/raven_ml/ambf_controller/raven2/scripts

[13] Y.-H. Su, A. Munawar, A. Deguet, A. Lewis, K. Lindgren, Y. Li, R. H. Taylor, G. S. Fischer, B. Hannaford, and P. Kazanzides, "Collaborative robotics toolkit (crtk): Open software framework for surgical robotics research," in *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, 2020, pp. 48–55.

[14] N. Chalfant and M. Bui, "Source code of the proposed sim-to-real gamepad controller framework for raven-ii," Jun 2024. [Online]. Available: https://github.com/MHC-RobotSimulators-Research/Raven2_standardized_controller

[15] "Raven-ii surgical robotics research platform open source software distribution." [Online]. Available: https://github.com/uw-biorobotics/raven2