# Developing Collateral Control Systems for Robotic Surgical Training

Spring 2020
601.456 Computer-Integrated Surgery II
Bryan Birthwright and Joao Kawase
May 9, 2020

# Technical Summary

## Problem Background:

Ever since 2000, when the FDA cleared the Da Vinci surgical robot for clinical use, its popularity and prominence have increased at an ever-accelerating pace. With over 3800 hospitals all around the world having at least one Da Vinci system installed as of 2019 and +1million procedures being conducted with them each year, these systems are prevalent and undoubtedly part of the future of medicine.[1] This widespread adoption, however, does not come without its own concerns surrounding its use and operators. With its increased use, more and more attention has been brought to the training and certification practices which prepare residents and surgeons for using such machines for real procedures, and what has been discovered is concerning. According to an article by Simon, "a researcher reckons that at most, one out of five residents at top-tier institutions are succeeding at robotic surgery."[2] This is a very worrisome statement, and the researcher Simon is referring to is not alone. In another article by Tangermann, "[researcher] found a troubling trend: doctors were "barely trained" in how to operate robots in the operating room". This can be accredited to several factors surrounding the availability and training practices with the robotic systems. The first and most prominent issue is the fact that because of the highly limited number of the systems accessible to surgeons, let alone residents, hands-on experience is difficult to come by with these machines, which is a valuable aspect of gaining proficiency in their use. Another issue is that current virtual training programs are not specifically catered to training (although that is their primary purpose) in regards to how no attempts are made to increase the efficiency of the learning process. Many of the current virtual training environments simply give you a task to accomplish and let you practice completing said task but do little to nothing in an attempt to help the user become accustomed to the controls and the Da Vinci console itself.[3] Therefore, it is clear that a better system for training future and current surgeons in using these surgical systems is of vital importance if they are to remain at the

[1] Perez, R., & Schwaitzberg, S. (2019). Robotic surgery: finding value in 2019 and beyond. *Annals Of Laparoscopic And Endoscopic Surgery, 4*. doi:10.21037/ales.2019.05.02

[2] Simon, Matt. "Med Students Are Getting Terrible Training in Robotic Surgery." *Wired*, Conde Nast, 15 Mar. 2018, www.wired.com/story/med-students-are-getting-terrible-training-in-robotic-surgery/.

[3] "Human Surgeons Are 'Barely Trained' on Operating Room Robots." *Futurism*, The Byte, 17 Mar. 2019, futurism.com/the-byte/surgeons-barely-trained-operating-robots.

forefront of modern medicine and to help extend the limits of what physicians are capable of doing with these incredible tools.

## Approach:

In order to most effectively solve the current issues with robotic surgical training two central aspects need to be addressed, accessibility to a robust training platform which can translate directly to the using a full da Vinci machine itself and having an advanced training scheme which facilitates and accelerates the learning process. This project attempts to address these by developing new collateral control schemes inside a sophisticated virtual training platform which can be accessed and utilized with minimal effort. In doing so, we attempt to improve the efficiency of the entire training process within an accessible program which could be used by a large variety of different training curricula.

The virtual training platform used was the AMBF (Asynchronous Multibody Framework) simulator developed by our mentor in this project Dr. Adnan Munawar Ph.D. Utilizing this rigid body simulator, specifically designed to simulate surgical tools and machines as well as compatibility with the dVRK (da Vinci research kit), we were able to adapt the code in order to allow for multiple control consoles (in this case 2 dVRK's) to be used at the same time to control the same end-effectors (virtual tools). By developing this system along with multiple different shared control schemes which will be explained in the next section, we believe a more efficient training scheme for users can be made so they can become more comfortable using the control console quickly and then be able to demonstrate greater proficiency when tasked with completing a puzzle on their own. Along with these shared control schemes, a small suite of 5 puzzles were created for the virtual environment to provide a more experimental opportunity for testing these systems as well as adapting a data acquisition script to gather data on proficiency metrics within the virtual environment. The purpose of developing these 3 components is to eventually conduct a user

Figure 1

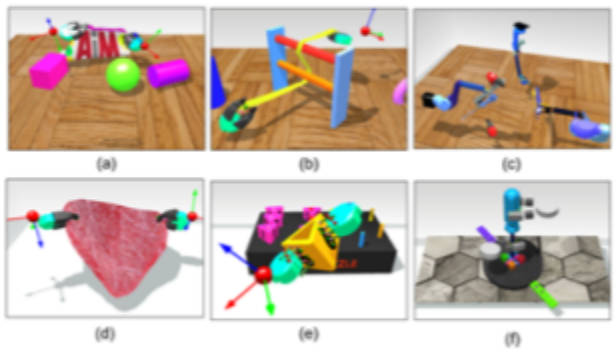



Figure 1 shows virtual end-effectors manipulating a variety of objects in the simulated AMBF environment

study in order to gather real data about the effectiveness of the developed systems so improved training schemes can be developed in the future.

## Results:

A breakdown of all the developed components for this project can be found in the table below and a link to the Wiki page, project repo and AMBF simulator can be found in the *Technical Appendix* section of this report.

| File/**Directory** | Methods changed/added | Purpose |
|---|---|---|
| *Bridge.cpp* | `DVRK_Bridge::DVRK_Bridge(const std::string &arm_name, int bridge_frequency): _freq(bridge_frequency){}`<br><br>`void DVRK_Bridge::get_arms_from_rostopics( std::vector<std::string> &arm_names){}` | These methods were changed so that the communication between ROS and the dVrks would recognize the second console named MTM2(L/R). |
| *CdVRKDevices.cpp* | `cDvrkDevice::cDvrkDevice(unsigned int a_deviceNumber){}` | This method was changed slightly in order to include a more generic method of left and right arm assignment for the dVrk. Now any dVrk following the naming convention of "MTM[Number][L/R]" will be correctly assigned. |
| *input_devices.yaml* | `Input device[], MTM1R, MTM1L, MTM2R, MTM2L` | New device names were added to the list along with their descriptions which link MTM#R devices together and MTM#L devices together. It is in these description sections where different control schemes are made and their specific configurations are explained further down this section. |

| | | |
|---|---|---|
| *data_collect.py* | Includes a function to create preliminary sets of pose data (one trainee set and one expert set) in the form [tx, ty, tz, roll, pitch, yaw] and uses that data to calculate comparison metrics on the total path length, orientation, and average motion smoothness(time integral of jerk squared) | The main application of these functions is to assist the collection of data in the user study to be performed in the future. Due to technical difficulties, the script is not currently integrated with ROS and cannot receive pose information from a dVRK. |
| ***beamPuzzle*** | Created in blender, converted to meshes and a .yaml description file for the AMBF simulator to interpret | This puzzle tasks users to build a system of beams using corner poles with rings to connect them (image found below) |
| ***boxPuzzle*** | Created in blender, converted to meshes and a .yaml description file for the AMBF simulator to interpret | This puzzle tasks users with filling a 5 sided black box with tetris-like shapes (image found below) |
| ***fishPuzzle*** | Created in blender, converted to meshes and a .yaml description file for the AMBF simulator to interpret | This puzzle is reminiscent of carnival games where the user must use a simple fishing rod to place and remove a set of 8 fish from a stationary platform (image found below) |
| ***seesawPuzzle*** | Created in blender, converted to meshes and a .yaml description file for the AMBF simulator to interpret | This puzzle tasks users with constructing a seesaw and balancing a collection of objects on top of it (image found below) |
| ***towerPuzzle*** | Created in blender, converted to meshes and a .yaml description file for the AMBF simulator to interpret | This puzzle tasks users with constructing a small tower using Jenga-like blocks (image found below) |

Once dual control functionality was implemented within the AMBF simulator the issue of creating different variations of dual control schemes became the main concern and remains the backbone of the adaptability of such a system. In this project 4 different configurations were developed based on the simple idea of how the input and output of each of the consoles is to

behave in a binary fashion. This being that either the users contribute to input or not and whether they feel output or not. From this you get the possible configurations of Symmetric Input-Symmetric Output (SISO), Asymmetric Input-Symmetric Output (AISO), Symmetric Input-Asymmetric Output (SIAO), and finally Asymmetric Input-Asymmetric Output (AIAO). To achieve these different control schemes all that has to be done is change the controller (input) gains and haptic (output) gains in the *input_devices.yaml* file listed above.

Figure 2

```
MTM1L:
    hardware name: MTM1L
    workspace scaling: 5
    simulated multibody: "../multi-bodies/grippers/pr2 gripper small red.yaml"
    haptic gain: {
        linear: 0.03,
        angular: 1 }
    controller gain: {
       linear: {P: 20.0, D: 2.0},
       angular: {P: 5.0, D: 1.0}}
    location: {
      position: {x: -0.5, y: 0.0, z: 0},
      orientation: {r: 0, p: 0.0, y: 0}}
    button mapping: {
      a1: 1,
      a2: 6,
      next mode: 3,
      prev mode: 4}
    pair cameras: [camera2] # The cameras paired with this IID-SDE pair
    # visible: True
    # visible size: 0.005
```

```
MTM2L:
    hardware name: MTM2L
    workspace scaling: 5
    #simulated multibody: "../multi-bodies/grippers/pr2 gripper small red.yaml"
    root link: "/ambf/env/BODY r_gripper_palm_link"
    haptic gain: {
        linear: 0.03,
        angular: 1 }
    controller gain: {
       linear: {P: 20.0, D: 2.0},
       angular: {P: 5.0, D: 1.0}}
    location: {
      position: {x: -0.5, y: 0.0, z: 0},
      orientation: {r: 0, p: 0.0, y: 0}}
    button mapping: {
      a1: 1,
      a2: 6,
      next mode: 3,
      prev mode: 4}
    pair cameras: [camera2] # The cameras paired with this IID-SDE pair
    # visible: True
    # visible size: 0.005
```

Figure 2 shows the description of the left manipulators for dVrk's 1 and 2 in the *input_devices.yaml*

Figure 2 shows an example of SISO dual control configuration and the right manipulators would have the same configuration between themselves. In order to achieve the other control schemes only minor changes need to be made to these descriptions regarding the haptic gain and controller gain values. Below are images of what the different configurations would be:

Figure 3

```
MTM1L:
    hardware name: MTM1L
    workspace scaling: 5
    simulated multibody: "../multi-bodies/grippers/pr2 gripper small red.yaml"
    haptic gain: {
        linear: 0.03,
        angular: 1 }
    controller gain: {
       linear: {P: 20.0, D: 2.0},
       angular: {P: 5.0, D: 1.0}}
MTM1L:
    hardware name: MTM1L
    workspace scaling: 5
    simulated multibody: "../multi-bodies/grippers/pr2 gripper small red.yaml"
    haptic gain: {
        linear: 0.03,
        angular: 1 }
    controller gain: {
       linear: {P: 0.0, D: 0.0},
       angular: {P: 5.0, D: 1.0}}
MTM1L:
    hardware name: MTM1L
    workspace scaling: 5
    simulated multibody: "../multi-bodies/grippers/pr2 gripper small red.yaml"
    haptic gain: {
        linear: 0.03,
        angular: 0 }
    controller gain: {
       linear: {P: 0.0, D: 0.0},
       angular: {P: 5.0, D: 1.0}}
```

```
MTM2L:
    hardware name: MTM2L
    workspace scaling: 5
    #simulated multibody: "../multi-bodies/grippers/pr2 gripper small red.ya
    root link: "/ambf/env/BODY r_gripper_palm_link"
    haptic gain: {
        linear: 0.0,
        angular: 0 }
    controller gain: {
       linear: {P: 20.0, D: 2.0},
       angular: {P: 5.0, D: 1.0}}
MTM2L:
    hardware name: MTM2L
    workspace scaling: 5
    #simulated multibody: "../multi-bodies/grippers/pr2 gripper small red
    root link: "/ambf/env/BODY r_gripper_palm_link"
    haptic gain: {
        linear: 0.03,
        angular: 1 }
    controller gain: {
       linear: {P: 20.0, D: 2.0},
       angular: {P: 0.0, D: 0.0}}
MTM2L:
    hardware name: MTM2L
    workspace scaling: 5
    #simulated multibody: "../multi-bodies/grippers/pr2 gripper small red.y
    root link: "/ambf/env/BODY r_gripper_palm_link"
    haptic gain: {
        linear: 0.0,
        angular: 1 }
    controller gain: {
       linear: {P: 20.0, D: 20.0},
       angular: {P: 0.0, D: 0.0}}
```

Figure 3 shows the 3 configuration schemes for collateral control in order from top to bottom: SIAO, AISO, AIAO

As can be seen, very minor changes can result in entirely different control schemes, opening up a whole range of possible configurations to explore in the future. We decided to start with these simple ones since they embody very generic schemes that can be manipulated and fine-tuned as more experimentation and data acquisition is conducted while using them.

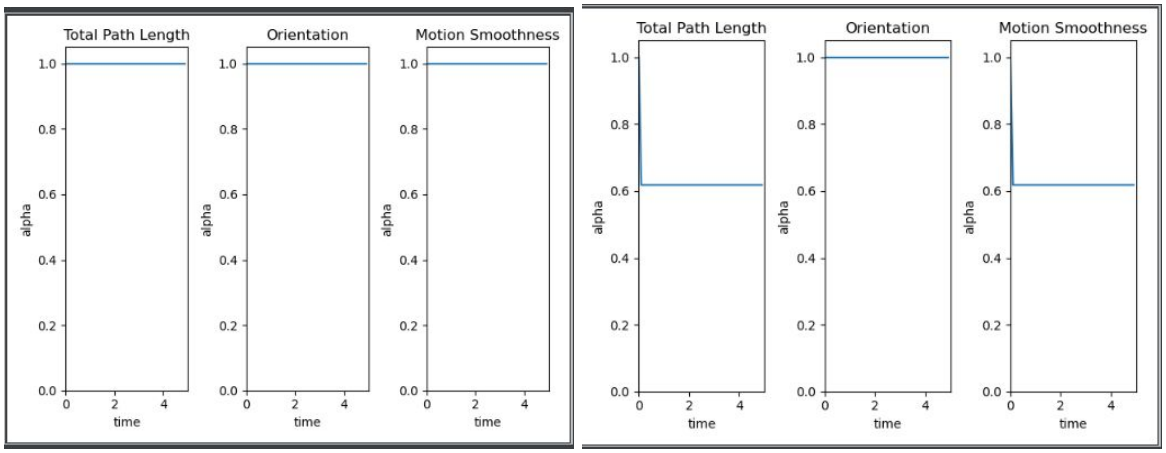
In order to make data collection during the user study more manageable, we created a python script intended to take data for us and also calculate comparison metrics on that data to compare the trainee's performance to that of the expert while the two are completing a puzzle. Our original intention was to implement the data collection script so that it can collect pose information from the 4 dVRK arms and calculate three main metrics: total cartesian path length(Γ), controller orientation(Θ), and average motion smoothness(Ψ). This would be accomplished through rosbag, a ROS package that can be used to record and store/read timestamped data from specified rostopics. Once these values are calculated we can calculate an alpha value that is representative of how closely the trainee is following the expert's movements as a measure of

$$\alpha_{\Gamma}(T) = 1 - \left| \frac{\Gamma_{m_2}(t) - \Gamma_{m_1}(t)}{\Gamma_{m_2}(t) + \Gamma_{m_1}(t)} \right|$$

$$\alpha_{\Theta}(t) = 1 - \left| \frac{\Theta_{m_2}(t) - \Theta_{m_1}(t)}{\Theta_{m_2}(t) + \Theta_{m_1}(t)} \right|$$

$$\alpha_{\Psi}(t) = 1 - \left| \frac{\Psi_{m_2}(t) - \Psi_{m_1}(t)}{\Psi_{m_2}(t) + \Psi_{m_1}(t)} \right|$$

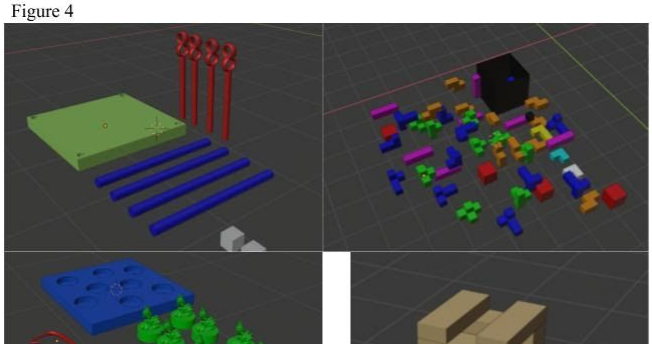
trainee expertise. Alpha values closer to 1 signify a well performing trainee as opposed to alpha values closer to 0. Unfortunately, we were unable to integrate our script with ROS to actually collect data from dVRK arms. In lieu of this, we have included a helper function within our script that generates 2 sets of mock pose data (a 3d translation along with RPY angles) for a number of very simple test scenarios (no motion in both sets, translational motion in one or both sets, rotational motion in both sets, etc.). For the time being we have used this mock data to help test our script and produce some simple mock data.




For example, in the left plot, the alpha values are shown for a scenario where the trainee and expert are moving toward the same point at equal speeds, while the right plot shows the same motion with some fluctuations in the trainee's y-coordinate, which was constant earlier.

For the puzzles, when developing them we decided to follow a small set of guidelines in order to make them as intuitive as possible. Firstly, we wanted to keep them simple and familiar to users. Overly complex tasks and challenging puzzles don't help the user learn how to better use the robotic system and we believe would only serve to hide and mask the data we were really looking for. The purpose of these puzzles isn't to test a users ability to solve a puzzle but instead their proficiency in using the robot to perform a task, therefore by keeping the puzzles simple all users will start off on the same footing so that proficiency in using the robot is the main factor in their performance. Secondly, we wanted to make the puzzles easy to understand. The users shouldn't be overwhelmed by what they see in front of them and by making the tasks straightforward users will already have an intuitive sense of how to complete it,

Figure 4



minimizing the difficulty of translating intended motion to robotic manipulation. Finally, we wanted to make puzzles that required a variety of manipulations to complete. Simply picking up and placing an object down is too simple and provides minimal data to analyze, in regards to position, orientation and smoothness, but by adding variation to the puzzles we can collect a much wider range of information on their performances. Therefore, we tried making puzzles that would incorporate using many different aspects of control such as introducing line of sight obstacles to promote clever camera manipulation, inserting objects into one another to encourage finesse and smoothness and also physics aspects to encourage using boths hands together for stability. Using these principles the 5 puzzles mentioned above were created and Figure 4 are images showing them in a virtual environment.

### Significance:

With over 170 surgeons worldwide operating Da Vinci machines world wide, it is crucial that thorough and effective training techniques are widely available to everyone.[4] Our project aims to help bring surgical robotic training a step beyond current systems of virtual training by introducing new collateral control schemes to help increase the efficiency of training and hands-on time with these machines. By creating innovative training systems, we could allow for decreased training times and acquisition of proficiency by surgical residents, allowing them to hone their skills and gain more efficient hours of training under simulations. Also, developing such control systems and puzzles in an accessible, virtual environment allows for programs around the world to utilize, learn and adapt these systems helping improve and standardize the robotic surgical training programs. This would help improve overall care provided, increasing procedure effectiveness and streamlining surgical training processes.

## Management Summary

### Work Distribution:

Work for this project was distributed relatively evenly between the two group members. Below is a comprehensive breakdown of what each member of the team did in the project development and presentation materials:

[4] "Human Surgeons Are 'Barely Trained' on Operating Room Robots." *Futurism*, The Byte, 17 Mar. 2019, futurism.com/the-byte/surgeons-barely-trained-operating-robots.

Bryan Birthwright:

- Helped develop puzzle ideas and examples
- Developed and coded data acquisition script as well as testing with mock experimental data
- Helped develop dual control schemes
- Contributed to several sections of Project Proposal and Final Report

Joao Kawase:

- Develop puzzle ideas, created them in Blender, and converted them into the AMBF compatible format
- Implemented dual control capability into the AMBF simulator
- Developed dual control schemes for the AMBF simulator
- Contributed to several sections of Project Proposal and Final Report

## Goals and Accomplishments:

When we initially took on this project the intention was to develop these puzzles, dual control schemes and data acquisition script for the purpose of conducting a user study to test their applicable effectives and collect real user data. However, due to extenuating circumstances, access to dVrk consoles and users for the study were not feasible, preventing us from being able to test our implementations and puzzles as well as conducting our user study. Therefore, based on our initial deliverable goals as presented below, we were only able to satisfy our minimum deliverables. However, in doing so we have created and implemented all the tools necessary for conducting a full user study to test the effectiveness of the dual control systems, hence paving the way for future projects.

| Deliverables | Description |
|---|---|
| **Minimum** | Implementation of a dual console/shared control with dVRK system and AMBF simulator, 5-6 puzzles to be used in study, design user study and collect mock data |
| **Expected** | Dual console/shared control, puzzles, data acquisition script, and **conduct user study with actual subjects** |
| **Maximum** | All of the above along with writing a paper on results of user study |

## Future Work:

Next steps for this project primarily consist of using the tools we have created to conduct a full user study and collect real world data in order to properly analyze the effectiveness of these dual control schemes as training tools. Using insights from the study, modifications to existing

dual control schemes and entirely new control schemes can be made to further improve training effectiveness. One such possibility is an adaptive dual control system which constantly changes based on the real time measured performance of a user. However, before such work can be done our system must be rigorously tested on actual dVrks to ensure its stability, performance and accuracy. Our data collection script will also need to be integrated with ROS so that it can collect data from the dVRK arms being used in the user study.

### Lessons Learned:

Some important organizational and management lessons were learned during this project. First, learning and building proficiency with the development tools to be used early in the project life cycle is very important to the later implementation of the project. In this case, learning to use Blender early on (even though it wasn't a massive hurdle) would have dramatically helped with the sophistication of the puzzles developed and later one would prove useful to developing increasingly complex and interesting puzzles for the simulator. On the more managerial side, having more regular meetings as a group and with our mentor would have helped take this project way beyond our initial expectations despite the dramatic change in circumstances. Being limited in resources allowed for more time to be spent on developing more minute features to our puzzles and data acquisition scripts, but with the sudden change we were satisfied with being able to fulfill our initial deliverables.

## Technical Appendix

### Wiki Page:

**https://ciis.lcsr.jhu.edu/dokuwiki/doku.php?id=courses:456:2020:projects:456-2020-12:project-12**

**Deliverables Repo Link: https://github.com/jkawase1/CIIS-ambf**

**AMBF simulator Repo Link: https://github.com/WPI-AIM/ambf**