Developing Collateral Control Systems for robotic surgical training

By: Bryan Birthwright and Joao Kawase
Mentor: Dr. Adnan Munawar Ph.D

CIIS Project
May 5, 2020

Problem Background

- As of 2019, over 2,800 hospitals have *da Vinci* systems installed (1)
- In 2017, Intuitive estimates that over 875,000 *da Vinci* procedures were performed (1)
- “researcher reckons that at most, one out of five residents at top-tier institutions are succeeding at robotic surgery” (2)
- Due to scarcity of *da Vinci* devices, hands-on time with these machines is difficult to obtain (3)
- Over 170 surgeons worldwide operating Da Vinci machines world wide, it is crucial that thorough and effective training techniques are widely available to everyone (3)

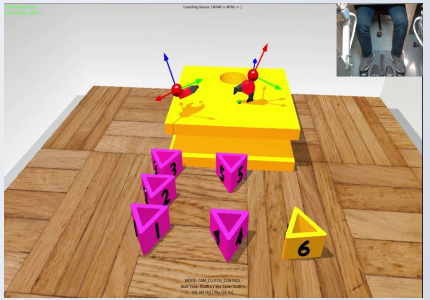
(1) Perez, R., & Schwaitzberg, S. (2019). Robotic surgery: finding value in 2019 and beyond. *Annals Of Laparoscopic And Endoscopic Surgery, 4*. doi:10.21037/ales.2019.05.02

(2) https://www.wired.com/story/med-students-are-getting-terrible-training-in-robotic-surgery/

(3) https://futurism.com/the-byte/surgeons-barely-trained-operating-robots

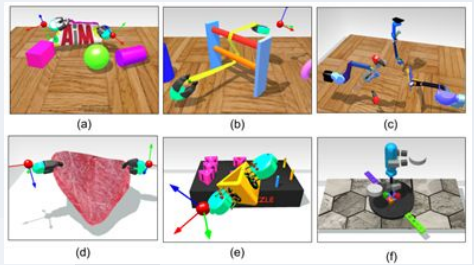
Significance

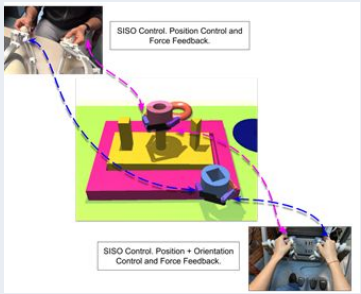
- Creating an new and effective training system could allow for decreased training times and acquisition of proficiency by surgical residents, allowing them to hone their skills and gain more efficient hours of training under simulations
- Virtual training environment allows for accessibility of system so programs and students around the world can utilize it



Project Approach

- Implement collateral control capabilities in AMBF
- Create suite of training puzzles/exercises for AMBF
- Code data collection script to record key performance metrics from users in the virtual environment





Dual Control Implementation

- Adding code to allow simulator to recognize multiple dvrk consoles
- Root linking consoles so dvrk consoles both control the same virtual end effectors
- Adjusting controller and haptic gains to generate different dual control schemes: **Symmetric Input - Symmetric Output (SISO), Symmetric Input - Asymmetric Output (SIAO), Asymmetric Input - Symmetric Output (AISO), Asymmetric Input - Asymmetric Output (AIAO)**

Dual Control Implementation

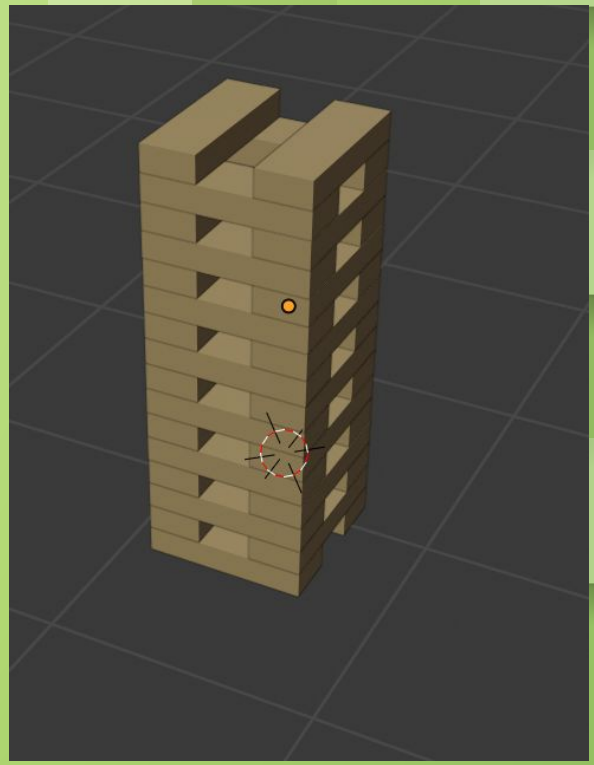
```
MTM1L:
    hardware name: MTM1L
    workspace scaling: 5
    simulated multibody: "../multi-bodies/grippers/pr2 gripper small red.yaml"
    haptic gain: {
        linear: 0.03,
        angular: 1 }
    controller gain: {
       linear: {P: 20.0, D: 2.0},
       angular: {P: 5.0, D: 1.0}}
    location: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

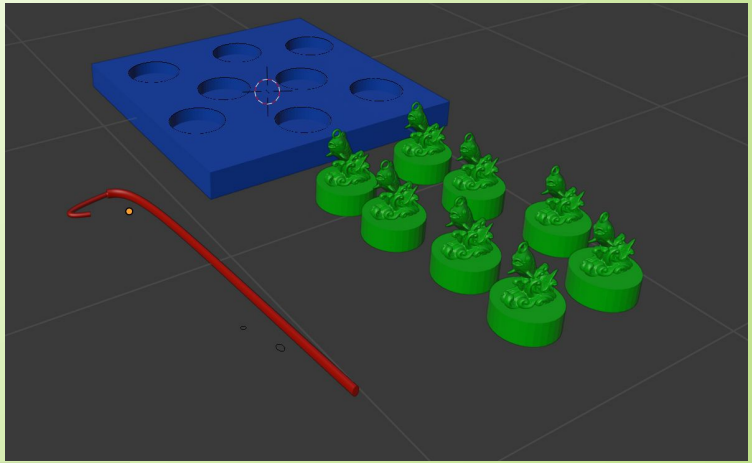
- SIAO: (MTM2L) haptic gain: {L: 0.0, A: 0.0}
- AISO: (MTM1L) controller gain: {L:{P: 0.0, D: 0.0}}
  (MTM2L) controller gain: {A:{P: 0.0, D: 0.0}}
- AIAO: (MTM1L) haptic gain: {A:0.0}, controller gain: {L:{P: 0.0, D: 0.0}}
  (MTM2) haptic gain: {L:0.0}, controller gain: {A:{P: 0.0, D: 0.0}}

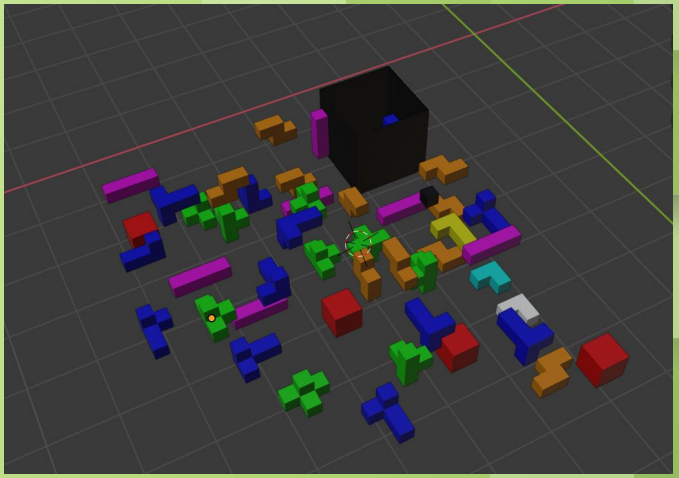
Training Puzzles

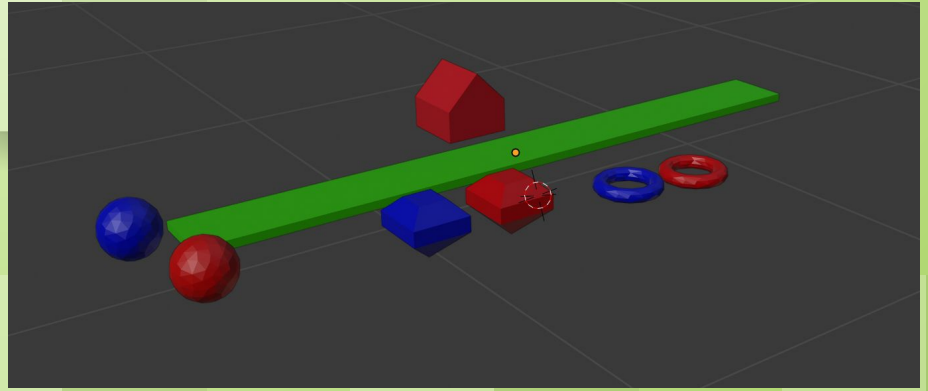
- Simple and familiar
- Easy to understand
- Requires using all methods of manipulation

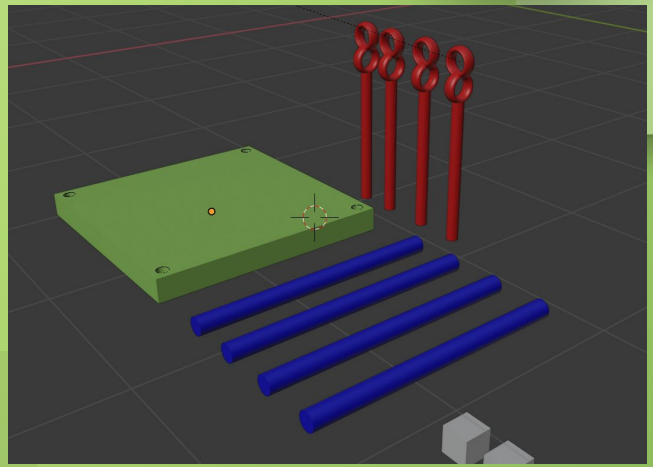


Training Puzzles

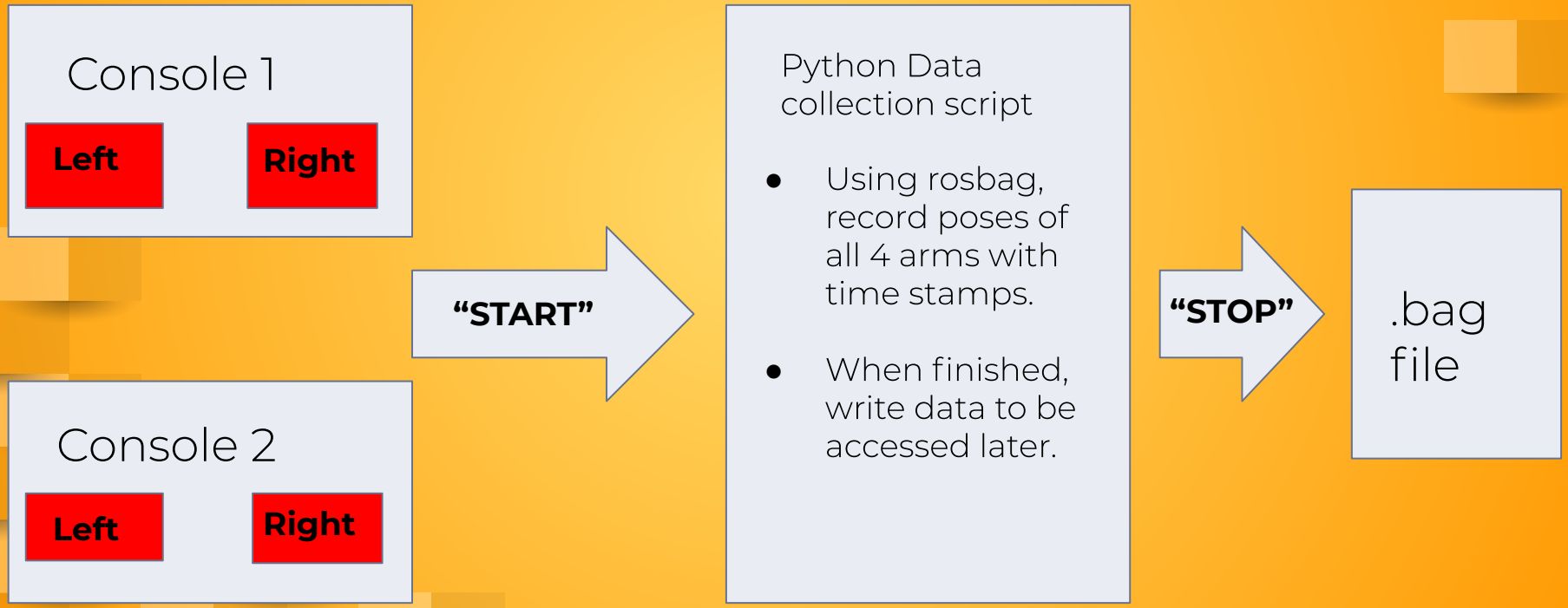








Data Collection for User Study



Data Collection for User Study

3 Main metrics:

- Total Path Length
- Controller orientation
- Path smoothness

Plan for testing:

Create simple ROS publishers to publish poses in place of dVRK consoles.

```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import Float64


def talker():
    pub = rospy.Publisher('chatter1', Float64, queue_size=15)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        data = [tx, ty, tz, qw, qx, qy, qz]
        rospy.loginfo(data)
        pub.publish(data)
        rate.sleep()
```

Future Work

Before completion of final report:

- Finish testing data collection script that can be used alongside our collateral control system in a user study.
- Complete documentation

After this semester:

- Get hands on experience with dvrks to fine tune dual control systems
- Conduct user study for collateral control effectiveness in training

Lessons Learned:

- Learn to use tools (Blender) sooner in order to build proficiency and be able to develop project more fully
- Maintain regular meetings to help advance project beyond initial expectations

Current State

Deliverables:

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

Work distribution:

- **Dual Control Implementation**: Joao Kawase and Bryan Birthwright
- **Training Puzzles**: Joao Kawase
- **Data Collection Script**: Bryan Birthwright

CREDITS

Special Thanks to Dr. Munawar for offering this project and for helping us with everything we needed

Thank You

Any questions?