Soft and Biohybrid Robotics – Graded Assignment #3

Due Date: See Moodle page/Syllabus for the due date and time.

Required Materials: You will need a computer running either Mac OS (Intel CPU only), Linux, or Windows. Three files need to be submitted in the end on Moodle: a one page write up with up to 500 words as pdf, a video less than 50MB in mp4, and a code archive in zip format. The video needs to be brief (less than 180 seconds) and self-explanatory.

Naming of the files:

1) Report: "<Your Name>_report.pdf"
2) Video: "<Your Name>_video.mp4"
3) Code archive: "<Your Name>_code.zip"

Task 1 Modeling and Controlling Soft Finger with SOFA (20 points)

Objectives: Evaluate how well you have understood the modeling and control lectures (Lectures 8-11). This assignment should lead to a more comprehensive understanding of the key steps required to model a soft robot using a finite element solution. Please read notes and final remarks at the end before getting started with this assignment.

Task 1a: Set up using SOFA binaries and run the default caduceus simulation (3 points)

Your first task is to download and set up SOFA v23.12 on your computer following the instructions on this page:
https://www.sofa-framework.org/download/
Remember to check the instructions on the GitHub releases page for your OS and follow them.

Note that you are not required to build SOFA from source. Please use the provided binaries.

Now that you have installed the SOFA software, start it by running the ***runSofa*** executable. You should now see the default simulation of the caduceus. Click ***Animate*** to start the simulation and apply forces on the snake using ***Shift + Click and drag.***

To view the code used to run this scene *(XML, not Python)*, open the file in a text editor (the path should be:
<SOFA-installation-directory>/share/sofa/examples/Demos/caduceus.scn)

Make a screen recording of your running environment and your interaction with this scene (not more than 15 seconds of recording) and add it to your submission video. No code submission is needed for this task.

Task 1b: Set up the SOFA plugins necessary for your tasks (5 points)

Set up and load the following SOFA plugins:

- *SofaPython3*
- *STLIB*
- *SoftRobots*
- *SoftRobots.Inverse*

Note: All plugins come with the SOFA binary installation. All that needs to be done is to add the plugins to SOFA itself through their dynamic library files.
On the SOFA GUI, go to "Edit > Plugin Manager", and then click "Add", navigate to the above path add the lib

Optional: Feel free to load additional plugins as well. List of plugins is here: https://www.sofa-framework.org/applications/plugins/

Now execute the *Finger* example in the *SoftRobots.Inverse* plugin.
(path to the example: *<SOFA-installationdirectory>/plugins/SoftRobots.Inverse /docs/sofapython3/component/constraint/CableActuator/Finger.pyscn*)

You can interact with the finger by moving the yellow target sphere around. Try to create an interesting deformation. A short screen recording of your interactions in your environment should suffice (not more than 15 seconds). Add it to your submission video. No code submission is needed for this task.

Task 1c: Implement a simple finger position controller (6 points)

Adapt the above scene from Task 1b such that you can control the position of the finger. Implement a controller to track a sinusoidal trajectory of the fingertip position.

This task requires making the scene interactive. Here is an SofaPython3 example that should help you in that task: *https://github.com/sofa-framework/SofaPython3/blob/master/examples/example-scriptcontroller.py*

Make a screen recording showing how you can control the finger's position and also how you can track a sinusoidal trajectory (not more than 30 seconds) and add it to your submission video. Report on the tracking error metrics. Properly comment your SOFA scene code for this task and add it to your submission. Report your findings in your write-up.

Task 1d: Simulate a tendon-driven 3-finger gripper (6 points)

Modify the above scene from Task 1c by duplicating the finger to create a 3-fingered gripper.

Create a spherical object of radius 4 cm. Your task is to grasp this sphere using your 3-finger gripper (*clue: you will need to add collision properties),* assuming you have perfect knowledge of the position of the sphere, and assuming that you can arbitrarily set the pose of the gripper.

Make a screen recording actuating it to grasp three of the objects in the (less than 1 min) and add it to your submission video. Properly comment your SOFA scene code for this task and add it to your submission. Report your findings in your write-up.

Task 2: Training a Lunar Lander to Safely Land with RL (30 points)

Objectives: Evaluate how well you have understood the lecture on machine learning for soft robotics and get you familiar with coding in Jupyter Notebook. In this assignment, you will complete the code to use RL to safely land the lunar lander in Gym Environment. Please read the notes and final remarks at the end before getting started with this assignment.

Please download SBRC024_Assignment3_Task2.ipynb and rename it to <Your Name>_Assignment3_Task2.ipynb

Below is a summary of the subtasks, please refer to the Jupyter Notebook file for the detailed task description.

Task 2a: Set up Gym Environment and run one episode with visualization (4 points)

Complete the code to run one single episode with randomly sampled action, and you will need to set the render flag correctly to visualize the episode. Make a screen recording showing the episode visualization and add it to your submission video. Report on the Total Reward of this episode.

Task 2b: Complete the code for Q Network architecture (8 points)

Complete the code to make a simple 3-layer network with hidden layer input and output size 64 as the Q Network. Hint: the architecture should be three fully connected layers with ReLU in between. You don't need to report anything for this subtask.

Task 2c: Complete the code for Q Network learn function (8 points)

Complete the code to calculate

1. Target Q value from Bellman equation
2. MSE loss term

You don't need to report anything for this subtask.

Task 2d: Train the agent to safely land the Lunar Lander (10 points)

Run the code to train the agent for 1000 episodes. You need to achieve a final average score of at least 200 with a safe landing animation to get full points for this task. You

can play with the hyperparameter to train for more or fewer episodes, however, if your reward is not improving, go back and carefully check your code in Task 2b and 2c.

Report on your final average score of training with the reward history plot. Make a screen recording showing the episode visualization and add it to your submission video. Please also report on the total reward of this visualized episode.

Save your Jupyter Notebook with ALL cell outputs and add the file to the code archive.

Important notes, applicable to all tasks:

- SOFA binaries are currently ***not*** supported on arm64 macs, but x86 macs should work fine, and so should x86 Linux and Windows machines. For arm64 macs, you can either use it through Rosetta or build from source.
- For Windows users: download and install Python3.8 NOT through the Microsoft store but directly from python.org. Make sure you select the option to add it to the PATH env variable**. Now s**et the env variable "SOFA_ROOT=<sofa-installation-dir>" and add to (if it already exists) or set "PYTHONPATH=<sofa-installation-dir>\plugins\SofaPython3\lib\python3\site-packages"
- For Mac users: when you open SOFA for the first time, you might be required to approve each library in SOFA manually *(in System Settings → Privacy and Security)*. One workaround is to run this command to approve all libraries at once: “xattr -dr com.apple.quarantine <SOFA-installation-dir>”
- Please combine all screen recordings into a single video and add titles to distinguish between tasks.
- Please make sure to save ALL cell outputs in Jupyter Notebook. Do NOT clear cell outputs. Submission without cell outputs will be penalized up to half the points of the task.
- The problem has to be solved individually; unique submissions are required. Copying code from a classmate is obviously not allowed.
