Assignment 8:
Rendering with Dynamics and Graphics, Vibration Feedback and Teleoperation

PDF file due on Canvas by 11:59 pm PDT on Monday, June 8, 2020
(no late assignments accepted due to solution posting in advance of Quiz 3)

Optional Readings

- B. Hannaford, Design framework for teleoperators with kinesthetic feedback. IEEE Transactions on Robotics and Automation, 5(4):426-434, 1989. This is a seminal paper in the field of bilateral teleoperation.

- K. Hashtrudi-Zaad and S. E. Salcudean. Analysis of Control Architectures for Teleoperation Systems with Impedance/Admittance Master and Slave Manipulators. International Journal of Robotics Research, 20(6):419-445, 2001. Much of the material in Lecture 16 is drawn from this paper.

You can download papers from Canvas > Files > Papers.

What to submit for this assignment: Clean up your **code** so it includes only what is necessary for simulation of problems 1 and 2 below, and submit it to Canvas in a PDF file along with comments from part 3 below. Also, you must submit short (< 10 second) **video demos**. The video for the wall and the video for the mass-spring-damper should each be submitted to the same Canvas assignment as the PDF. The assignment will allow for multiple submissions (one after another, not simultaneously). Please use the re-submit assignment option to submit multiple times.

For the videos, please follow the structure seen in the example videos named A8P1_Example and A8P2_Example such that we can see the movement of your handle and the corresponding response in the graphical environment.

1. Haptic Rendering with Graphics

Here you will create a visualization of the one-degree-of-freedom virtual environment using graphics via the Processing language (http://processing.org). ***Make sure to download the latest version 3.5.3! The example project will not work with versions older than 3.0.*** To get started, use the sample Processing code posted with this assignment on course website. Download the Arduino starter code (**Assignment8-starter.ino**) and add code where prompted within the #ifdef statements.

Use the starter code for Processing in the Assignment 8 folder. See https://processing.org/reference/ for functions to draw shapes (e.g. an ellipse) and use built-in calculation functions like map().

Note that a higher serial communication rate (e.g. changing the rate in your Hapkit code and Processing code from 9600 to 38400) may improve performance. If you will also use the Serial Monitor for debugging, the rate must also be changed in the Serial Monitor window.

We will start with a virtual wall. This is the same as in Assignment 7, Problem 3A, with one difference: When the "haptic avatar" (a ball, a cube, whatever object representation you like) penetrates into the wall, do not *visually* show it penetrating into the wall. As you may recall from our discussions in lecture, this visual "trick" causes users to think that the wall is harder than it actually is.

Submit a video on Canvas showing the graphics and your handle in the same shot. Show that when you move your handle, the ball in the graphic program moves accordingly. Be sure to show that the handle stops at the wall, and that the ball does not visually penetrate the wall. See the instructions before Problem 1 for what to submit.

Also submit Arduino and Processing code.

2. Haptic Rendering of Dynamics

Create a mass-spring-damper simulation with the following effects: The user can make and break contact with the mass via the Hapkit handle, and applying forces to the mass should cause the system to oscillate. To make the simulations consistent throughout the class, please make the mass-spring-damper system at equilibrium at x = 0.5 cm (so that the system will apply zero force if it begins at rest and the handle is vertical), and the Hapkit position should always be to the left of the mass. (In other words, the Hapkit position should not "pop through" to the other side of the mass.)

In the starter code the mechanical properties of the system have been defined for you as follows:
Mass: m = 2 kg
Damping: b = 1 Ns/m
Spring stiffness: k = 300 N/m
Stiffness of interaction between user and mass: k_{user} = 1000 N/m
Equilibrium position of mass: 0.5 cm

Your graphics should show the mass, a line representing the spring-damper connection with a stationary wall, the wall, and your haptic avatar. You *do not* need to make a nice-looking spring/damper that stretches – we just want you to make a rough visualization (like a line that lengthens).

Submit your video, Arduino code, and Processing code.

3. Vibration

In this question you will compare two methods of implementing vibration. First, you will use the Hapkit motor and feel the vibration through the Hapkit handle. Second, you will connect the small vibration/coin motor from your kit to the alligator clips and feel the vibration directly from the motor.

A. Hapkit motor vibration
    a. Download and run **Assignment8-hapkitVibe.ino**. Feel free to edit the frequency and amplitude as desired.
B. Coin motor vibration
    a. Disconnect the alligator clips from the Hapkit motor and connect them to the free leads of the coin motor (the red and blue wires). Note that the cut ends of the alligator clip should still be connected to your Hapkit Board, the only thing we are changing is which motor we are actuating.
    b. Download and run **Assignment8-vibeMotor.ino**. Feel free to edit the duty cycle but do not go too high or you may burn out the motor (recommended limit is about 0.5 duty cycle).
    c. For this experiment, do not attempt to mount the coin motor onto the Hapkit or your body, just hold it with your fingertips.
C. <u>What to submit:</u> Comment on the differences in what you feel in parts A and B. These comments should be in the same PDF file as your code from Problems 1 and 2.

4. OPTIONAL: Teleoperation

In these interesting times we live in, wouldn't it be nice to interact via touch with another person that we cannot be with in person due to social distancing?

In this OPTIONAL part of the assignment, you will set up, with a partner, remote teleoperation over the internet using your Hapkits. This will require having **python** and the **pyserial package** installed to act as a link between the serial and TCP/IP socket connection. One person must be the "server" and one person must be the "client". The server role requires enabling port forwarding on your internet router to allow an inbound connection from the other client-side Hapkit. Note that either Hapkit can be the master or the follower using this protocol.

If you are able to set this up and test it with a partner before the Interactive Session on Tuesday, June 9, 2020, the teaching team would like to "shake hands" with you to say goodbye at the end of the quarter. Let us know by Piazza post if you get it working. Also, if you want to give it a try but are uncomfortable with "command line-fu" or setting up port forwarding, please post to Piazza so that your peers and the teaching team can assist!

A. Installing Python and Miniconda

If you already have a working python installation on your computer you do not need to follow these steps. For more experienced users, if you use another package manager such as pip or a different virtual environment, you can use that instead – you don't need to use Miniconda.

Windows 10:
Follow the instructions here for installation of Miniconda (not the silent mode)
https://docs.conda.io/projects/conda/en/latest/user-guide/install/windows.html
* Note that you should install the Python 3 version

MacOS:
Follow the instructions here for installation of Miniconda (not the silent mode)
https://docs.conda.io/projects/conda/en/latest/user-guide/install/macos.html
* Note that you should download and install the bash version of the Python 3 installation.

You should now have a fully functional python development environment that will serve you well in your future engineering endeavors! If the `conda list` command in the instructions linked above did not work, please post to Piazza to get help.

B. Create a new development environment and install pyserial

From Anaconda prompt (Windows) or terminal (macOS), create an environment:
```
conda create --name myenv
```

Activate it by typing
```
conda activate myenv
```

Now install pyserial by typing:
```
conda install pyserial
```

C. Download python script and Arduino sketch

From the Assignment 8 folder on Canvas, download the corresponding python file for your role. If

you are the client, download "hapkit_client.py" and if you are the server, download "hapkit_server.py".

Both people must download the "Assignment8_tele" Arduino sketch for the Hapkits.

D. (For server role) Enable port forwarding on your router and check your public IP address

As the server, you will have to allow the client side to make an inbound connection to your computer. However, if you are connected to the internet via a router, the client can only see the router and you will need to tell the router to forward the data to your computer. While different routers have different interfaces for changing their internal settings, the conceptual steps are similar.

First, you will need to know your router's, and your computer's, local I.P address. In Windows, do this by typing `ipconfig` into the command line and checking the default gateway (router's) and IPv4 (your computer's) IP addresses. For MacOS, type `netstat -rn |grep default` to get the router IP address and `ifconfig |grep inet` and look for the your computer's IP address, `inet` (not `inet6`) entry that is not `127.0.0.1` . Note them down.

For most conventional routers, once you know your router's IP address, you can type it into your browser and open up a configuration page. Log in using the admin username and password (if nobody in your household knows this information, it is probably left at the defaults of that brand/model and a quick Google search should turn it up).

Check how to do port forwarding for your router on the internet. You can see if you can the instructions for your router here: https://portforward.com/router.htm. For Xfinity routers, refer to this page: https://www.xfinity.com/support/articles/port-forwarding-xfinity-wireless-gateway. For AT&T, refer to this page: https://www.att.com/support/article/u-verse-high-speed-internet/KM1123072/. The port you are going to forward is: `65432`.
*__Note that you might be using a dynamic IP address assigned to you by your router, which can change. At the time that you Zoom call your partner, you should check to see if your local IP has changed and update the port forwarding configuration to redirect to the correct IP address.__

Lastly, you will need to check your public IP address. The easiest way to do that is to visit https://whatismyipaddress.com/ and check. Note that this can change day by day. **Check and share this IP address** with your partner right before you do your Zoom call.

E. Upload Arduino sketch and check serial port name

For stability, do not plug in your DC power supply for now.

Launch the Arduino app and upload the sketch you downloaded. Also note the name of your serial port which is given at the bottom right corner of the app. For Windows it should be "COMxx" and for MacOS it should be "/dev/…". This default sketch has a default calibration for the Hapkit, it should work for most Hapkits, but you can update the calibration if you wanted to.

F. Edit scripts to point to the correct serial port and IP address.

For the client
Open up your python script "haptkit_client.py" in a text editor and change the HOST string variable to the public IP address given to you by your partner who is acting as the server. Change the

ser_port string variable to that of your serial port name. (Both of these lines are indicated with "MODIFY THIS".) Save your changes.

For the server
Open up your python script "haptkit_client.py" in a text editor and change the ser_port string variable to that of your serial port name. (Both of these lines are indicated with "MODIFY THIS".) Save your changes.

You are now ready to begin remote teleoperation.

G. Schedule a zoom session between you and your partner for the teleoperation session

For the best experience, we recommend making sure your camera feed has a visual of your Hapkit so that you and your partner can see each other's Hapkits being remotely teleoperated.

H. Teleoperation

With the Hapkit USB cable connecting the Hapkit Board to your computer, DC power disconnected, and Hapkit paddle centered:

In the Anaconda prompt (Windows) or terminal (macOS), both partners navigate to the directory with the python script. To change directory, use `cd your_path_name`.

Activate your conda environment (see Part B).

The server-side partner launches their script by typing: `python hapkit_server.py`

The client-side partner then launches their script by typing: `python hapkit_client.py`

Both partners should receive a confirmation once the connection is successful.

Now, the client-side can activate their motor: While holding onto your Hapkit paddle with one hand, plug in the DC power with the other.

The server side should now be able to teleoperate the client-side Hapkit. At least at first, BE CAREFUL TO MOVE SLOWLY to not cause instability. The client-side partner can lightly grasp onto their Hapkit paddle to feel their partner's movements.

Return both Hapkits to the centered position and swap roles. Do this by having the client-side unplug their DC power supply and the server-side plugging theirs in. Remember to STABILIZE your Hapkit with one hand as you plug it in.

That was unilateral teleoperation. To do bilateral teleoperation, return both Hapkits to the center position and connect the DC power supplies for both Hapkits. Bilateral teleoperation is more likely to go unstable, and you should be CAREFUL AND MAKE SLOW MOVEMENTS. If at any time the system goes fully unstable or starts limit cycling at high frequency or amplitude, then both sides should grasp onto their Hapkit paddles to stabilize the system and/or remove power. Disconnecting an alligator clip may be easier than unplugging the power supply.

To stop the teleoperation, hit ctrl/cmd + c to terminate the terminal script.

Try lightly grasping your paddles and take turns to transmit your movements to each other's

Hapkits. The other person can try providing some resistance or pushing back on the other person's movements. Another fun experiment is to have one person's hand act as a physical barrier. As one user teleoperates the Hapkit to contact the barrier/hand, they should feel the contact.

I. Questions:

a. Play with the gains of the system, when does the system go unstable? Besides the gains, what other factor affects system stability?

b. In what way did unilateral teleoperation feel different from bilateral teleoperation? What factors do you think contributed to these differences?

c. How would you modify the `force` expression in the Arduino sketch to achieve position scaling on the teleoperation?