Homework 9:
Haptic Rendering with the Phantom

MEAM 520, University of Pennsylvania
Katherine J. Kuchenbecker, Ph.D.

December 5, 2013

This assignment is due on **Tuesday, December 10, by midnight (11:59:59 p.m.)** Your code should be submitted via email according to the instructions at the end of this document. To help you handle the intensity of the end of the semester, **late submissions will be accepted with no penalty until midnight on Saturday, December 14**. Submissions after Saturday will be penalized by 10% for each partial or full day late, up to 20%. After Monday, December 16, no further assignments may be submitted.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you write down must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. If you get stuck, post a question on Piazza or go to office hours!

Pair Programming

Because it requires the use of a real robot, you must do this assignment with a partner; individual submissions are not allowed. Working with a partner will help keep you safe, keep the Phantom safe, and ensure everyone has a chance to work with the robot in the time available. You may work with any other student in the class. If you're looking for a partner, consider using the "Search for Teammates!" tool on Piazza.

You should work closely with your partner throughout this assignment, following the paradigm of pair programming. You will turn in one set of MATLAB scripts for which you are both jointly responsible, and you will both receive the same grade. Please follow these pair programming guidelines, which were adapted from "All I really need to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000:

- Start with a good attitude, setting aside any skepticism and expecting to jell with your partner.
- Don't start writing code alone. Arrange a meeting with your partner as soon as you can.
- Use just one computer, and sit side by side; a desktop computer with a large monitor is better for this than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (using the mouse and keyboard or recording design ideas) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every thirty minutes, *even if one partner is much more experienced than the other.* You may want to set a timer to help you remember to switch.
- If you notice a bug in the code your partner is typing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Recognize that pair programming usually takes more effort than programming alone, but it produces better code, deeper learning, and a more positive experience for the participants.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.

Task Description

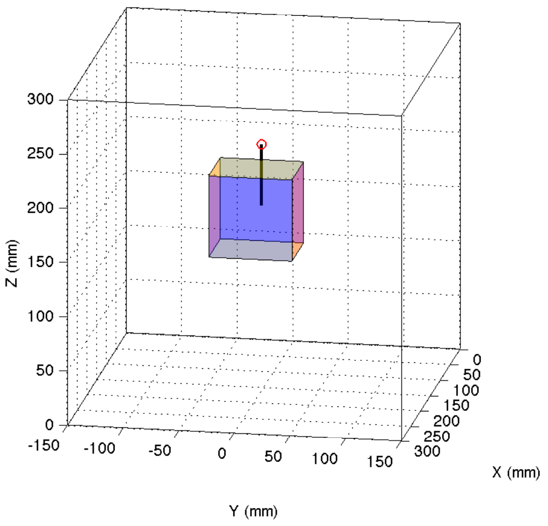
Like Homework 8, this assignment centers on the SensAble Phantom Premium 1.0, an impedance-type haptic interface with three actuated rotational joints. Designed to be lightweight, stiff, smooth, and easily backdrivable, this type of robotic device enables a human user to interact with a virtual environment or control the movement of a remote robot through the movement of their fingertip while simultaneously feeling force feedback.

As described below, there are three parts to this assignment. The first part aims to familiarize you with the functionality of the Phantom and its simulator. The second and third parts ask you to modify provided starter code to achieve a desired behavior of the Phantom. Everything you need to know to accomplish this assignment was covered in lecture. To learn more about haptic rendering, we encourage you to look over "The PHANToM Haptic Interface: A Device for Probing Virtual Objects" by Thomas H. Massie and J. Kenneth Salisbury, which was handed out in class and is also posted on Piazza under Lecture Resources.

Begin by downloading the **starter code** from the assignment resources section on Piazza; it is provided as a zip file named `hw09_starter.zip`. You will first develop your code for this assignment using the simulated version of the Phantom that is included in the starter code. We are providing you with p-coded versions of all the functions you need. The code starts the Phantom by calling `phantomStart(hardware)`, where `hardware` is a Boolean value. Setting `hardware` to `false` starts the simulated Phantom so you can work on your code on any computer. Instead of getting encoder readings from the real Phantom, the system simulates the presence of a human user by reading a pre-recorded trajectory from the included `encsHistory.mat` file. You choose the duration of the test by setting the value of `nCycles` at the top of the starter code.

Once your team has written a good draft of the code for each section, you will need to test it on the real Phantom and make any necessary refinements. Detailed instructions for testing on the Phantom are included at the end of this document.

1. Haptic Box Demo

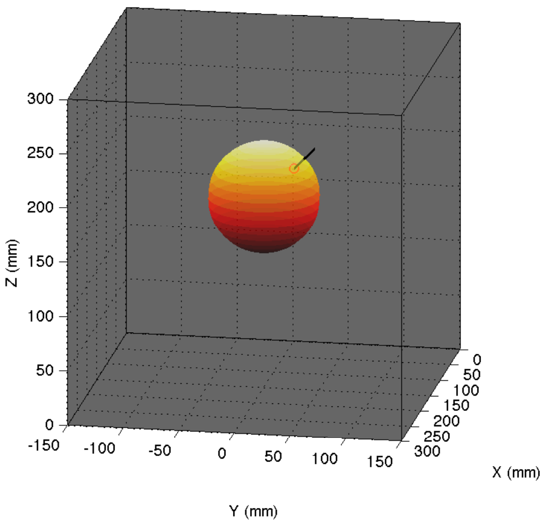


After you download the starter code from Piazza, run `haptic_box_demo.m` and look at how it is written. This demonstration creates a virtual haptic box for the user to feel, as seen in the illustration at right. The user is trapped inside the virtual box and feels a virtual spring force each time they contact a wall. The position of the Phantom tip is shown as a red circle, the box is shown in transparent colors, and a scaled version of the force vector is shown as a thick black line. The location and size of the box are controlled through variables such as `boxWidthX`; all dimensions are in millimeters.

The provided software simulates the presence of a human user by default (because you probably don't have a Phantom connected to your computer). Look at how the forces `Fx`, `Fy`, and `Fz` are calculated from the positions `hx`, `hy`, and `hz`, and watch how the force vector changes as the simulated user moves around. This is the type of mapping you will need to create in this assignment.

Play around with this simulation to make sure you understand it. Try changing the location of the box and the stiffness of the walls. Notice that the code uses variables rather than hard-coded values (magic numbers) so it's easier to modify, and it includes comments to explain what's being done at each step of the calculation; your code should do the same. Once you understand how the haptic box demo works, you should move on to the next part. There is no deliverable for this part of the assignment.

2. Haptic Ball



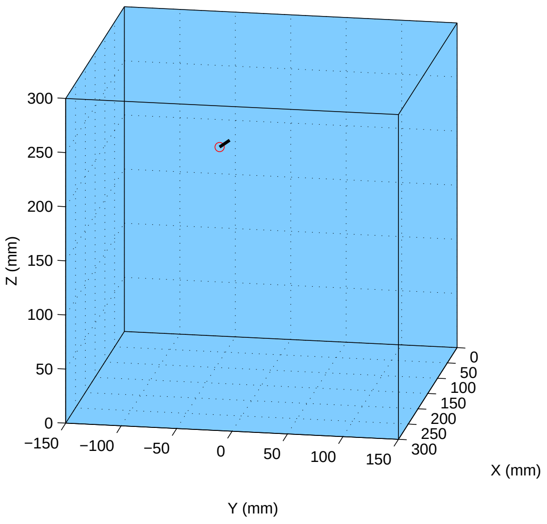
Complete the haptic ball scene that has been started for you in `haptic_ball_starter.m`. Change the filename to include your team members' PennKeys (`_pennkey1_pennkey2`) instead of `_starter`, and enter the names of your team members at the top of the file in the `studentNames` variable.

The graphics for this scene are shown in the image at right. Again, the position of the Phantom tip is shown as a red circle, the ball is shown as a transparent warm-colored sphere, and a scaled version of the force vector is shown as a thick black line. The background is gray like the outer space around the sun.

Modify the code between the two lines of stars to push the user out of the ball whenever they come inside, using a virtual spring mapping. The force should push them straight out of the ball, and the magnitude of the force should be proportional to their penetration depth. The stiffness of your spring should be `k` (already defined for you in the code using units of newtons per millimeter). The ball's surface should not include any friction or other haptic effects. Please comment your code.

Use the graphics to debug your calculations in simulation (with hardware set to false) until both of your team members are convinced the calculations are correct. Only then should you try your code on the real Phantom, following the instructions at the end of this document.

3. Haptic Damping



Complete the haptic damping scene that has been started for you in `haptic_damping_starter.m`. Change the filename to include your team members' PennKeys, and enter the names of your team members at the top of the file in the `studentNames` variable.

The graphics for this scene are shown in the image above. Again, the position of the Phantom tip is shown as a red circle, and a scaled version of the force vector is shown as a thick black line. The background is blue like water.

Modify the code between the two lines of stars to output a viscous damping force that always acts to slow the user down. This force should act in the opposite direction to the user's velocity vector, and its magnitude should scale with the magnitude of their velocity vector. The viscosity of your damper should be `b` (already defined for you in the code using units of newtons per millimeter per second). Please comment your code.

When correctly implemented, viscous damping should feel like you are moving your hand through molasses or honey. There should not be any jittering in the force. This means you will need to low-pass filter your velocity vector to reduce quantization noise. Be careful to implement the velocity filter correctly; examine the graph produced at the end of the simulation to confirm that the filter has the effect you want.

Use the graphics to debug your calculations in simulation (with hardware set to false) until both of your team members are convinced the calculations are correct. Only then should you try your code on the real Phantom, following the instructions at the end of this document.

Testing Your Code on the Phantom

The actual Phantom robot is located in Towne B2 and is attached to the same computer that runs the PUMA robot. All members of this class have been given card-swipe access to Towne B2 for this homework assignment. The Phantom and its computer are the only equipment you'll be using. You should not touch anything else in the room, nor should you attempt to use the PUMA robot.

- You must reserve the Phantom computer in order to test your code. You may reserve only one sixty-minute slot (or two thirty-minute slots) at a time. Respect the next team by finishing on time. The link to the Phantom reservation system will be posted on Piazza.
- Use your Penn ID card to swipe into Towne B2. You may need to swipe your card several times to get the door to unlock.
- Log in to the computer using the "puma student" account. The password is "meam520" without the quotation marks. This is a Linux computer running Ubuntu 12.04. Please do not reboot the computer or modify any of its settings. If you run into computer issues, post a note on Piazza.
- Once logged in, double-click the "Homework 9" folder on the Desktop. Create a folder for your team, naming it with your PennKeys (i.e., `pennkey1_pennkey2`). Do not look at, modify, or run any other code that may be on this computer.
- Copy only your team's main script files (`haptic_box_demo.m`, `haptic_ball_pennkey1_pennkey2.m`, and `haptic_damping_pennkey1_pennkey2.m`) over to the folder you just created; do not copy the rest of the Phantom simulator files. The easiest methods for transferring your files are to use a USB key or

to email them to yourself as attachments. You can access the Internet using the Firefox web browser link on the desktop. Open a file browser by clicking "Places" in the upper-left menu bar.

- Once you have your files transferred to this computer, double-click the large MATLAB icon on the Desktop. It will ask, "Do you want to run "MATLAB", or display its contents?" Choose **Run in Terminal**. MATLAB should open. Change the working directory to your folder.
- Before you run any of your code, you **MUST** check the calibration of the Phantom. With the Phantom's emergency stop pressed down, call `phantomStart(true)` from the command line. Then hold the robot in its zero configuration and call `phantomJointAngles` from the command line. If the joints do not all read very close to zero radians at this pose, the robot is not correctly zeroed, so you must run `phantomZero` from the command line before proceeding. Call `phantomStop` when done.
- You must have the Phantom emergency stop down (no forces) the first time you run any code, even the demo code. Watch the graphical output to see if your code is doing what it should. The black line shows a scaled force vector.
- You must firmly grip the Phantom gimbal every time you run any code. If the thimble is too big for your finger, wrap a rubber band around it to make it tighter, but also hold on. Do not let the robot move freely until you are confident your code is doing what it should. The robot can be damaged if it slams into the joint limits.
- Someone **MUST** have their hand on the emergency stop whenever the Phantom is running. Push down the emergency stop as soon as anything unusual happens.
- Start by running `haptic_box_demo.m` to get a feel for how it works. (Make sure you run it first with the emergency stop down. Then make sure to hold firmly onto the thimble and have a hand over the emergency stop the entire time it is running.)
- Then run each of the scripts that you edited for this assignment. Run them first with `hardware` set to `false`, and make sure they are doing what you intend. Then run the code with `hardware` set to `true` but with the emergency stop down, so you can see how the forces change with your motion. Once you're confident the force vector is correct, run it again with the emergency stop up. Update your code to improve it if needed.
- Please immediately post on Piazza if you notice any problems with the Phantom!

Submitting Your Code

Follow these instructions to submit your code:

1. Start an email to `meam520@seas.upenn.edu`
2. Make the subject *Homework 9: Your Name and Your Teammate's Name*, replacing *Your Name* and *Your Teammate's Name* with the appropriate full names.
3. Attach your two correctly named MATLAB files to the email as individual attachments; please do not zip them together or include any other attachments. The files should be the following:
    - `haptic_ball_pennkey1_pennkey2.m`
    - `haptic_damping_pennkey1_pennkey2.m`
4. Optionally include any comments you have about this assignment.
5. Send the email.

You are welcome to resubmit your code if you want to make corrections. To avoid confusion, please state in the new email that it is a resubmission, and include all of your MATLAB files, even if you have not updated all of them.