Lab 7 Manual

Robotics & Automation
Dylan Losey, Virginia Tech

1 Introduction

Welcome to the seventh and final robotics lab! The GTAs are here to facilitate your learning. Please be kind, professional, and respectful to the GTAs.

Objective. This lab has two objectives:

- Creating and interacting with virtual objects
- Using trajectory generation for smooth point-to-point motions

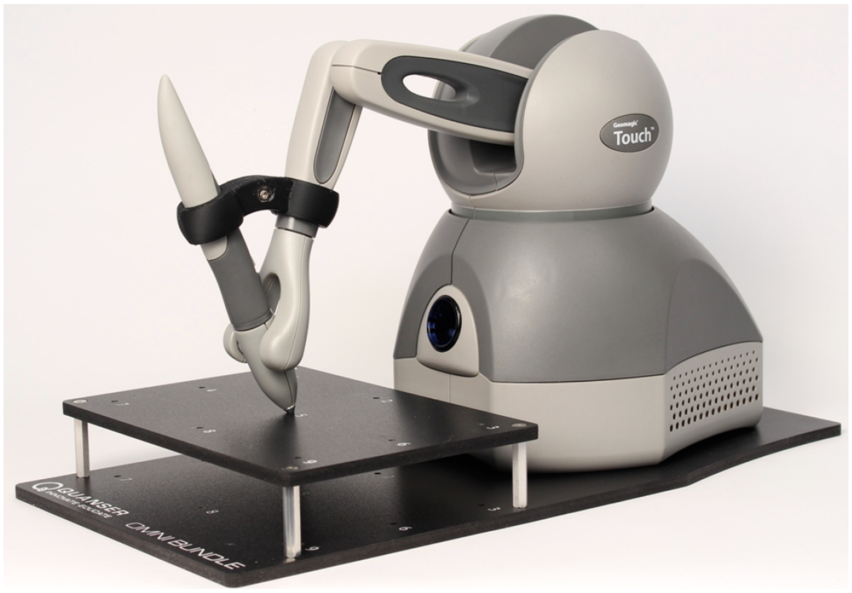


Figure 1: Geomagic Touch at your workstation.

2 Calibrating the Robot

Multiple groups use each workstation. Between sessions the robot needs to be re-calibrated. You must complete these steps at the start of *every* lab. On the plus side, you practiced these same steps during Labs 1–6, so you know what to do!

Action 1

Perform the following checks:

- The computer licences are paired with specific Geomagic Touch robots. Confirm that your computer has the same number as your Geomagic Touch workstation.
- Make sure that the ethernet cable coming out of the Geomagic Touch is connected to the back of the computer, and the robot's charging cable is plugged in.

- Check that the computer is connected to the wifi "quanser_UVS" with password "UVS_wifi".

Hint. The computers are slow. Patience is a virtue. But if your connection to the robot is continually lagging throughout this lab, we recommend that you restart the computer.

Action 2

Now we are ready to initialize our robot arm. Open the Geomagic Touch Diagnostic Tool application shown in Figure 2. Then complete the following action items:

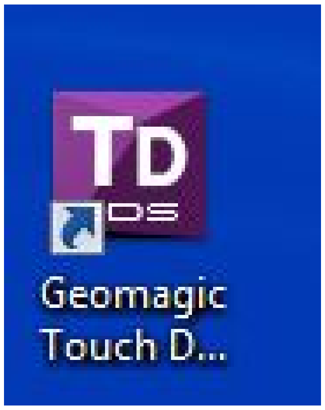
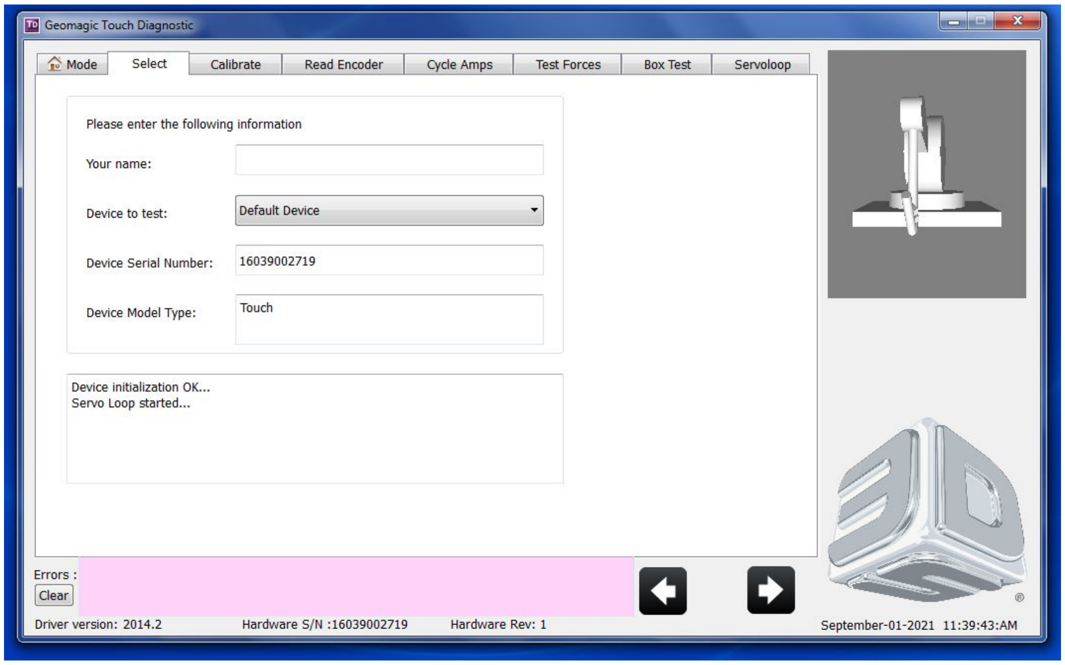


Figure 2: Application to check your connection to the Geomagic Touch.



Please enter the following information

Your name:

Device to test: Default Device

Device Serial Number: 16039002719

Device Model Type: Touch

```
Device initialization OK...
Servo Loop started...
```

Driver version: 2014.2 Hardware S/N :16039002719 Hardware Rev: 1 September-01-2021 11:39:43:AM

Figure 3: Geomagic Touch Diagnostic Tool. There is a 3D rendering of your robot arm in the top right. This should move in sync with your robot.

- When you open the application, you will see a tab at the top called **Select**. Click on this tab to see a small 3D model of your robot (shown in Figure 3). When you move the actual robot this 3D model should move as well. *If the 3D model is not moving when you move the robot ask a GTA for help.*

- Once you've ensured that the robot is connected, click on the **Calibrate** tab. Follow all the instructions on the screen to calibrate the robot (i.e., plug the pen into the holder for several seconds until the calibrate button turns green).
- Make sure to **close** the Diagnostic Tool after calibration. Otherwise it will conflict with Simulink in later steps.

Action 3

Let's open and run the initial Simulink model for this lab.

- In MATLAB, open the Control Simulink model:
  `..\Desktop\07 Trajectory_Planning\Trajectory_Planning_2015b_Start.mdl`
- You should see a Simulink diagram similar to Figure 4. This is the model that your team created during Lab 6.

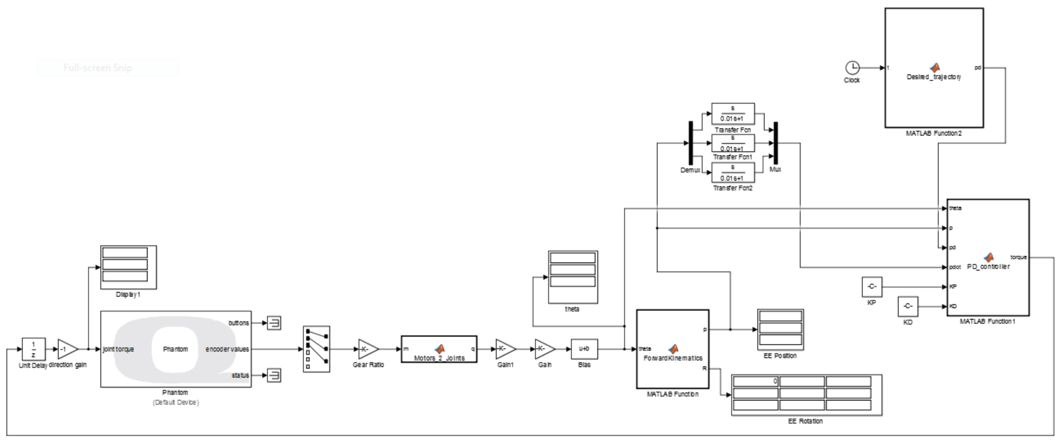


Figure 4: Initial Simulink diagram. Your team created the PD controller in Lab 6. This controller works in task space, i.e., the the control torque is based on the error between the desired (x, y, z) position and the actual (x, y, z) position. For initial calibration we have set p_d to **position 2** of the baseboard.

- Build the simulation by going to the QUARC menu and clicking on **Build**. When the MATLAB Command Window shows that the model has been downloaded to the target, run the simulation by opening the QUARC menu and clicking on **Start**.
- Place the robot's end-effector on **position 2** of the baseboard. The positions on the **display** for each revolute joint should read:
$$\theta_1 = 0.0 \quad \theta_2 = -0.3 \quad \theta_3 = 0.26 \tag{1}$$
  If your positions do not match, modify the **bias** block until the robot is correctly calibrated. You should also confirm that the forward kinematics function is correct. At **position 2**, the end-effector position p_{sb} should be close to:
$$p_x = 0.13 \quad p_y = 0 \quad p_z = -0.09 \tag{2}$$
  If the joint position is correct but the end-effector position is wrong, first double check the forward kinematics block, and then ask a GTA for assistance.
- Adjust the controller parameters K_p and K_d as needed during calibration.

3 Virtual Objects

In the first part of this lab we will use the robot to render virtual objects. When you push the pen into these objects, it will *feel* as though you have bumped into a wall or run into the side of an obstacle. This is an example of *haptics*.

Action 4

Let's get started by rendering a virtual wall.

- Modify your controller to resist external forces in the y axis:

$$f_y = \begin{cases} -1000 \cdot p_y \text{ if } p_y < 0 \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

  where $f = [f_x, f_y, f_z]$ is the force in each axis, and p_y is the position of the robot's end-effector in the y-axis. Set $f_x = f_z = 0$.
- Review the diagram below (Figure 5) to remind yourself of the x, y, and z axes for the robot. Determine where p_y is positive and where it is negative.

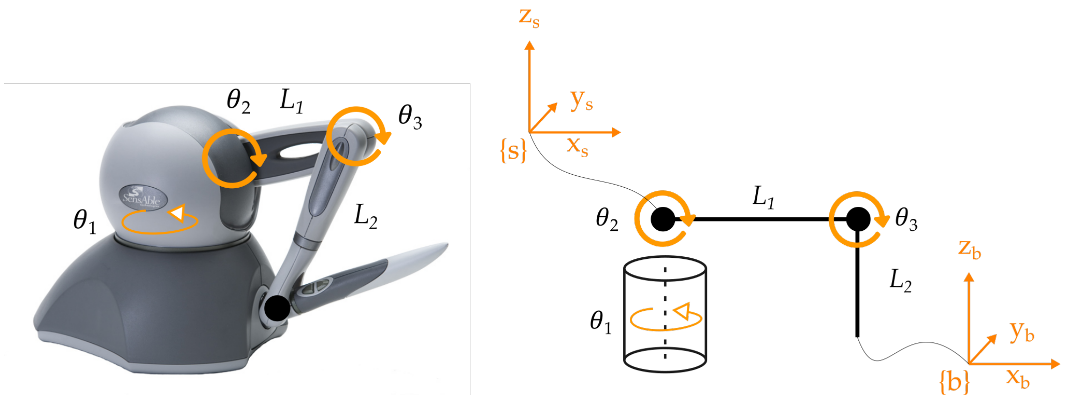


Figure 5: Joints of the Geomagic Touch. On right the robot is shown in its home position.

- **Important:** *Before running the model, make sure to move the robot to the right side so that* $p_y > 0$. *If you fail to do this the robot will start "inside" the wall and apply unsafe torques.*
- Connect the controller output to the **Phantom** block and run the model.
- Try moving the pen around the workspace. When you are on the right side, you should be able to move freely. But once you move the pen to the left, you will run into a wall. Take turns moving along this wall in the x-z plane.

Question 1

Using the process outlined above, create a new virtual object (e.g., sphere, cube, window) of your choice. Show your working virtual object to a GTA. **Important:** *Remember to always start the robot's end-effector outside of the virtual object.*

4 Trajectory Generation

We often need the robot to follow a *smooth* trajectory from start to goal. In lecture we are discussing motion planning algorithms. Here you will focus on interpolating along those motion plans for smooth trajectories.

Question 2

Use linear interpolation to move the robot between end-effector positions p_{start} and p_{goal}:

$$p_d(t) = p_{start} + s(t)(p_{goal} - p_{start}) \tag{4}$$

where $p_d(t) = p_{start}$ when $s = 0$ and $p_d(t) = p_{goal}$ when $s = 1$. For smooth motion we choose the scalar $s(t)$ as a cubic polynomial:

$$\begin{aligned} s(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ \dot{s}(t) &= a_1 + 2a_2 t + 3a_3 t^2 \end{aligned} \tag{5}$$

This scalar must satisfy four constraints. At the start of the trajectory we need $s(0) = 0$ and $\dot{s}(0) = 0$, and at the end of the trajectory we need $s(T) = 1$ and $\dot{s}(T) = 0$, where T is the total time the desired trajectory takes ($T > 0$ is a variable you get to choose). These constraints ensure that the desired trajectory starts and ends at rest, and reaches from p_{start} to p_{goal}. Given these four constraints, solve for the four parameters a_0, a_1, a_2, and a_3.

Question 3

Now that you have a_0, a_1, a_2, and a_3, we can plug those values back into Equation 4 and Equation 5. Write the equation you will use for $p_d(t)$ and $\dot{p}_d(t)$.

Action 5

Update your Simulink model to implement the equations from Question 3. Your working model should smoothly move the robot between an initial and final end-effector position.

- Consider adding a **Matlab Function** block that evaluates Equation 4 and Equation 5. This function should input time, p_{start}, p_{goal}, and T, and output p_d and $\dot{p}_d$. Make sure to hold $p_d(t) = p_{goal}$ and $\dot{p}_d = 0$ when $t > T$.
- Connect your desired position to the PD controller developed in the previous lab.
- Use p_{start}, p_{goal}, and T of your choice. You can check the position display to find a reasonable p_{start} and p_{goal}. For time, leverage the **Clock** block.

Question 4

Run the model with different values for T, p_{start}, and p_{goal}. Describe the motions you see with the trajectory generator. When does the robot arm move the fastest? When does it move the slowest? How does changing T change how the robot follows the motion plan?

5 Trajectory Recording & Playback

In the final part of this lab you will decide the robot's trajectory. First, you will physically guide the robot through a motion of your choice. The robot will record the positions you visit, and then play back the motion you provided.

Action 6

First we need to record your desired trajectory.

- Send $\tau = 0$ joint torques to the **Phantom** block (or disconnect the torque command).
- Add a **To Workspace** block and connect it to the output of forward kinematics. Use this block to record the position p of the end-effector.
- Within the **To Workspace** block set the variable to *simout* and set saved format to *Structure With Time*.
- Run the model and guide the robot through a motion of your choice. To begin, we recommend a motion where the robot starts and ends in a similar position. Move the robot slowly to get a good trajectory.
- Return to the Matlab workspace after the model has stopped. You should now see a variable titled **simout**. *We recommend saving your workspace (i.e., saving the simout variable) in case you accidentally override it later in the lab.*

Action 7

Next we will use the recorded trajectory as the robot's desired trajectory.

- Explore the **simout** variable in Matlab and print the time (simout.time) and the recorded positions (simout.signals.values).

- Make sure that the positions changed; we have noticed a bug where occasionally only the last joint position is recorded.
- Create a trajectory ξ where the first column is time and the remaining columns are the desired position:

$$\xi = [\texttt{simout.time, simout.signals.values}] \tag{6}$$

- Return to the Simulink model and make a **Constant** block. Set the variable to ξ, and connect this block to a Matlab Function.
- You have now imported the recorded trajectory ξ into your Simulink model. Modify your the Matlab Function so that it inputs time and outputs the corresponding p_d from ξ. As a hint, try comparing the current time to the trajectory time: the desired position p_d should be the demonstrated position at the closest point in time.
- Reconnect your PD controller to the **Phantom** block.

Question 5

Run the system and adjust the controller as needed. The robot should replay the motion you originally demonstrated (you may need to hold the pen during the replay). Show your working trajectory replay to a GTA.

Question 6

Often we record motions that are faster (or slower) than we want the robot to replay them. Or we make mistakes during the motion; e.g., moving the robot too close to an obstacle. Within your team discuss what changes you could make to "fix" a demonstrated trajectory.