ME 449 Assignment 1

From Mech
Jump to navigationJump to search

In Chapter 4, we will study the forward kinematics problem for open-chain robots: finding the configuration of the robot's end-effector $T_{sb}(\theta) \in SE(3)$ given the vector of joint positions θ. The forward kinematics problem is easy to solve using a formula called the "product of exponentials," which uses the matrix exponential of this chapter. In this project, CoppeliaSim will solve the forward kinematics for you.

The goal of this project is to test your understanding of the matrix log for rotations, to give you a little practice using the MR library of functions, and to familiarize you with CoppeliaSim.

You will submit **a single zip file** that includes **a single pdf file and one video (make sure we can play it)** to Canvas. The file name should be FamilyName_GivenName_asst1.zip (for me, it would be Lynch_Kevin_asst1.zip). You will have to concatenate all your responses and screenshots into a single pdf file. The video should be only a few MB; don't submit an overly long, hi-res video.

All assignments will be graded based on correctness, how clearly you organize your homework (the grader should easily find all of your solutions), and how well you follow the instructions. You will lose points if you don't follow the instructions or if the assignment is difficult for the grader to grade.

Contents

Part 1: CoppeliaSim Simulation and Exploration

In part 1, you will explore some of the functionality and built-in models for CoppeliaSim. Go to this page (http://hades.mech.northwestern.edu/index.php/Getting_Started_with_the_CoppeliaSim_Simulator) for a refresher on getting started with CoppeliaSim.

A CoppeliaSim scene may include objects (https://www.coppeliarobotics.com/helpFiles/en/objects.htm) (like shapes (https://www.coppeliarobotics.com/helpFiles/en/shapes.htm), joints (https://www.coppeliarobotics.com/helpFiles/en/joints.htm), or sensors) and one or more models (https://www.coppeliarobotics.com/helpFiles/en/models.htm). A model consists of a number of objects connected to each other.

Models can have sensing and motion capabilities as defined by their attached child scripts (https://www.coppeliarobotics.com/helpFiles/en/childScripts.htm). A child script can be threaded (which creates a new computation thread; this is generally discouraged) or non-threaded. A non-threaded script defines the "actuation" and "sensing" functions for an object or model, and these functions are invoked by the main script at each simulation step. More information can be found at Coppelia's webpage on main and child scripts (https://www.coppeliarobotics.com/helpFiles/en/simulationScripts.htm), the CoppeliaSim User Manual (https://www.coppeliarobotics.com/helpFiles/index.html), and the CoppeliaSim forums (https://forum.coppeliarobotics.com/).

Default models can be found in the model browser (left pane, or open the pane by clicking the robot icon in the left toolbar).

A scene also includes a main script (https://www.coppeliarobotics.com/helpFiles/en/mainScript.htm), which handles the simulation loop. At each simulation step, the main script calls (a) "actuation" functions that simulate the motion of the system and (b) "sensing" functions that simulate the sensors. Generally this main script should not be edited.

Part 1A: Building a world for a mobile robot

In this part, you will explore the CoppeliaSim model browser and learn how to create your own scene using the default models. Anything in the model browser can be dragged and dropped onto the checkerboard floor of the scene. It can then be moved or reoriented using the **Object/item shift** and the **Object/item rotate** buttons in the top toolbar (the cubes with the arrows around them). Also try using the controls at the top of the window to zoom the camera in and out, pan the camera, etc.

Use File>New scene to create a new blank scene. Add to the scene the Pioneer_p3dx, a mobile 2-wheeled differential drive robot with 16 sonar sensors, to the scene. (Find it under "robots/mobile" in the model browser). Clicking the **Start/resume simulation** button (the "play" button) will cause the robot to move forward at a set velocity. To see where this behavior is defined, stop the simulation and open the child script associated with the Pioneer robot. (Make sure the simulation is stopped, not just paused.) You can open the child script by clicking on the folded paper icon in the left toolbar and double-clicking "Child script (Pioneer _p3dx)" or by double-clicking the folded paper icon next to "Pioneer_p3dx" in the Scene hierarchy pane. You can see the variable v0 that defines the forward speed, modify it, and play the simulation again to see the robot move at a different speed.

The robot moves forward by setting the speed of each wheel to be the same, as long as the 16 sonar sensors do not detect an obstacle. The speeds of the wheels may be changed differentially if one or more sonar sensors detect an obstacle, which causes the robot to turn. (See if you can make sense of the algorithm by looking at the code.) This simple algorithm for obstacle avoidance is from the work of Valentino Braitenberg, and a robot implementing it is often called a "Braitenberg vehicle." (Braitenberg also proposed other simple reactive control algorithms for mobile robots.)

Add at least five 80 cm high walls to the scene (find them in infrastructure/walls/80cm high walls in the model browser). Arrange and orient the walls (click on the icons at the top that show a cube being translated or rotated) such that when the simulation is started, the Pioneer_p3dx interacts with at least three of the walls. To interact with a wall means to drive towards it and avoid bumping into it.

Your task:

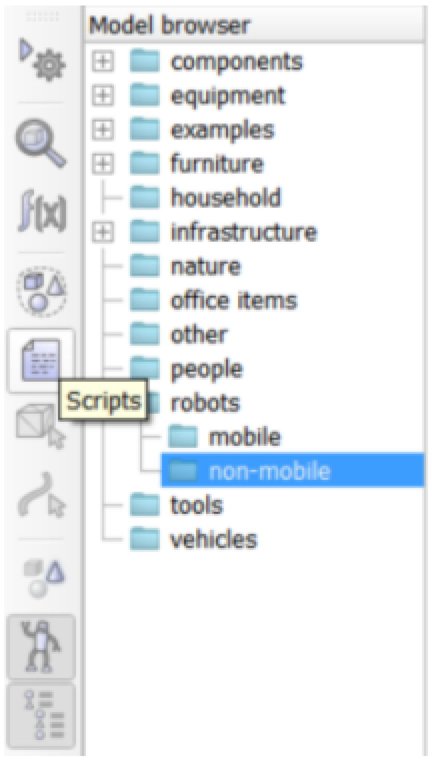
- Take a screen recording video of the robot moving through your world and interacting with at least three walls. In the video, use the zoom, pan, and other buttons to move the camera view while the robot is moving. You can change the robot's default speed in its child script to make it easier to capture your video with the moving camera. (If you don't know how to take a screen recording, you can check out CoppeliaSim Introduction (http://hades.mech.northwestern.edu/index.php/CoppeliaSim_Introduction).)

Part 1B: Exploring the UR5 Scene 1

Open Scene 1 for interactive manipulation of the Universal Robots UR5 robot, a popular 6R robot. (You can go to the CoppeliaSim Introduction (http://hades.mech.northwestern.edu/index.php/CoppeliaSim_Introduction) to download Scene 1 and the other scenes used in this course). In this scene, there is a model of a UR5 robot, which consists of a collection of joints (https://www.coppeliarobotics.com/helpFiles/en/joints.htm) connecting shapes (links), from the base out to the end-effector.

When you run this scene you will see a window with two tabs: "Enter Config and SE(3) Value" and "Joint Angle Sliders". Go to the "Joint Angle Sliders" tab, move the sliders corresponding to the six joints, and watch how the robot moves.

Make sure your scene 1 simulation is stopped so you can open up a script. Click the "Scripts" button on the toolbar on the left side of the screen (shown below) to see the scripts being run by the scene.



Double-click the "Child script (UI_Script)" to open it. You will see a script written in the Lua programming language. Early in the file, you might notice that some functions look similar to functions written in the Modern Robotics Library. For example, *so3andp2se3(R,p)* in this script resembles *RpToTrans(R,p)* in the Modern Robotics Library.

Scroll down to line 242. From here to the end of the file, this code examines the type of "call" to the child script by the main script and performs the appropriate functions. For example, if the sim_call_type is sim.syscb_init, the simulation has started and the user interface should be generated. (There is a lot of XML code here defining the user interface.) If the sim_call_type is sim.syscb_actuation, then the joint angles entered by the user in the UI are applied to the UR5 model, the UI is updated, and the transformation matrix for the end-effector is calculated. If the sim_call_type is sim.syscb_sensing, nothing happens, and if it is sim.syscb_cleanup, the UI is destroyed as the simulation ends.

The XML code for the UI starts on line 251. This XML code controls the appearance of the Custom UI, including the layout and content of text boxes, sliders, tab titles, and font size. The XML syntax can be found at Custom UI Plugin XML Syntax (https://www.coppeliarobotics.com/helpFiles/en/customUIPluginXMLSyntax.htm). For this specific scene, the different text boxes and sliders are arranged in different groups. Each group has a layout which determines how the objects inside the group will be displayed and label text which determines what text will be displayed and how it will be displayed. If you'd like, you can consult the XML syntax for the attributes each element can have.

Your task: Modify the UI_Script to change an aspect of the scene, and you will use this modified scene for Part 2, below. For example, you can choose to change the Custom UI layout, change the titles and words of the Custom UI, or change font sizes in the Custom UI. Some example changes include:

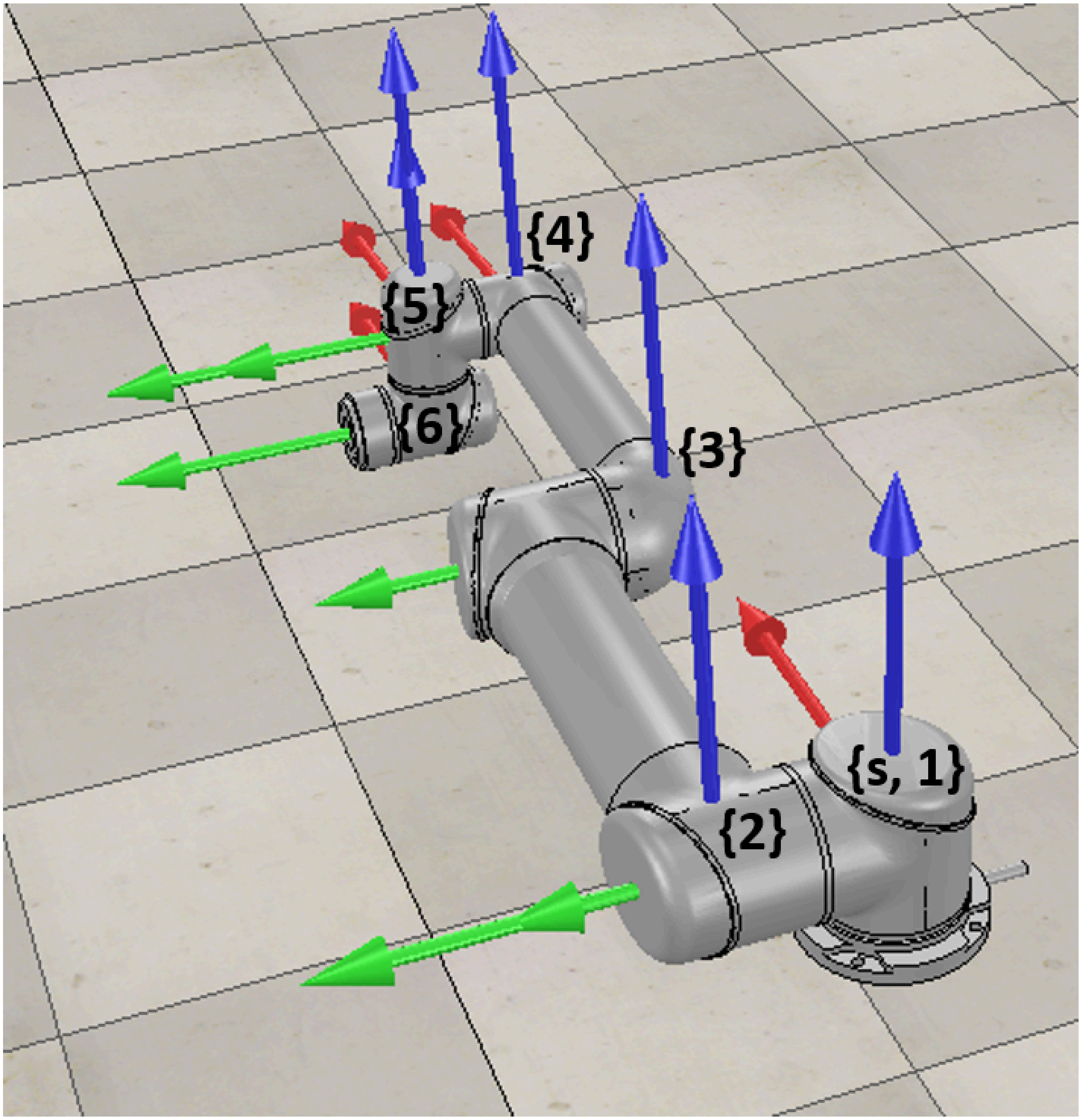
- Changing the layout: consult the XML syntax and change <group layout="vbox"> to another type (hbox,form, grid, none). Using line 257, changing <group layout="vbox"> to <group layout="hbox"> changes the orientation of the items in the group containing "Configuration Entry", "Current configuration", and "Messages" on the "Enter Config and SE(3) Value" tab of the Custom UI from vertically arranged to horizontal.

- Changing the words: Find the text you want to change in the XML code. The text will be surrounded by quotation marks and have <big> and </big> on its sides. For example, line 258 "label text="<big> Configuration Entry:</big>" controls the text "Configuration Entry" on the "Enter Config and SE(3) Value" tab of the Custom UI. Change the words "Configuration Entry" to change the text displayed in that specific section.

- Changing the font: Using line 258 label text="<big>Configuration Entry:</big>"
  - The font can be changed to small using: label text="<small>Configuration Entry:</small>"
  - The font can be changed to a specific size using: label text="<font size=20>Configuration Entry: </font>", where 20 is the desired font size.

Part 2: Joint Angle Calculations

The 6R UR5 robot is shown below at its home configuration. Eight frames are defined: the fixed frame {s} at the base, frames {1} through {6} attached to links 1 through 6, and the end-effector frame {b} which is fixed relative to link 6. (The frame {b} is not shown in the image.) The red arrow is the x-axis, the green arrow is the y-axis, and the blue arrow is the z-axis. Frames {s} and {1}-{6} are aligned when the robot is at its home configuration, i.e., each rotation matrix R_{ij} (where i, j could be s or any number 1 through 6) is the identity matrix.



The rotation axes for joint i, defined in frame $\{i\}$, are
$\hat{\omega}_1 = (0,0,1), \hat{\omega}_5 = (0,0,-1), \hat{\omega}_2 = \hat{\omega}_3 = \hat{\omega}_4 = \hat{\omega}_6 = (0,1,0)$.

For some set of joint angles θ, we have the following relations between the orientations of the joint frames:

- R_{13} = [[-0.7071, 0, -0.7071]; [0, 1, 0]; [0.7071, 0, -0.7071]]
- R_{s2} = [[-0.6964, 0.1736, 0.6964]; [-0.1228, -0.9848, 0.1228]; [0.7071, 0, 0.7071]]
- R_{25} = [[-0.7566, -0.1198, -0.6428]; [-0.1564, 0.9877, 0]; [0.6348, 0.1005, -0.7661]]
- R_{12} = [[0.7071, 0, -0.7071]; [0, 1, 0]; [0.7071, 0, 0.7071]]
- R_{34} = [[0.6428, 0, -0.7660]; [0, 1, 0]; [0.7660, 0, 0.6428]]
- R_{s6} = [[0.9418, 0.3249, -0.0859]; [0.3249, -0.9456, -0.0151]; [-0.0861, -0.0136, -0.9962]]

- R_{6b} = [[-1, 0, 0]; [0, 0, 1]; [0, 1, 0]]

where {b} is the end-effector frame not shown in the figure.

Your task:

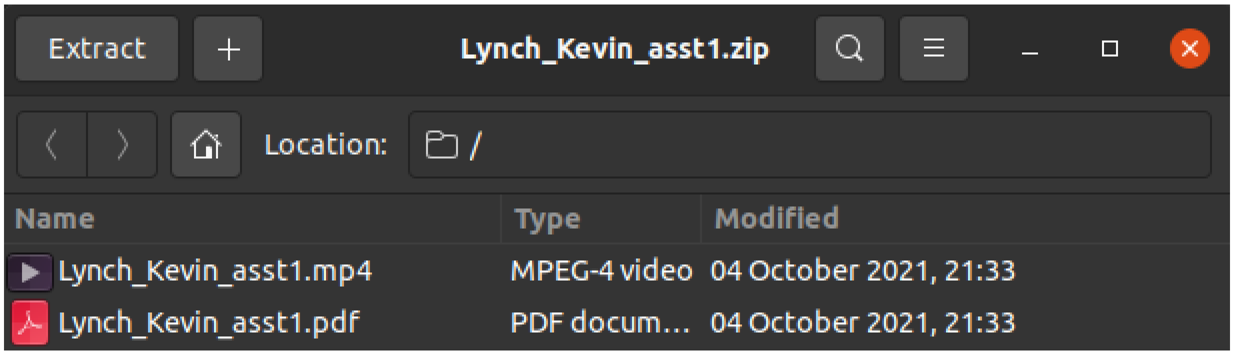
- Find the six-vector of joint angles $\boldsymbol{\theta}$ given the $\boldsymbol{R}_{ij}$ above. (You will likely want to calculate the rotation matrices $\boldsymbol{R}_{s,1}$ and $\boldsymbol{R}_{i,i+1}$ and use the MR code library, e.g., MatrixLog3.)
- Enter the joint angles you found into your modified Scene1_UR5 in CoppeliaSim to see the configuration of the robot.
- Calculate R_{sb} using the information given, and verify that your joint angle vector $\boldsymbol{\theta}$ is correct by entering the joint angles into the scene and comparing your R_{sb} to the rotation matrix portion of the T_{sb} calculated by the scene under the "Enter Config and SE(3) Value" tab.

What to Submit via Canvas

Turn in a single zip file. The file name should be FamilyName_GivenName_asst1.zip (for me, it would be Lynch_Kevin_asst1.zip). This file should have:

- Part 1A: A single video file showing a screen recording of your Pioneer_p3dx robot moving through your custom scene. NOTE: make sure your video is in a common format and reasonably sized (e.g., less than 20 MB).
- A single pdf file including:
  - Part 1B: A screenshot of your changed code in the child script. Explain what you changed about the scene/UI. (A small change suffices.)
  - Part 1B & 2: A screenshot of the scene, clearly showing the modified UI, the SE(3) calculation, and the robot at the correct configuration.
  - Part 2: The list of the six joint angles and R_{sb} you calculated and a brief explanation of the method (including your code, which likely calls the MR code) you used to calculate them.

The contents of your zip file should look like what you see below (with your name substituted):



Name	Type	Modified
Lynch_Kevin_asst1.mp4	MPEG-4 video	04 October 2021, 21:33
Lynch_Kevin_asst1.pdf	PDF docum...	04 October 2021, 21:33

If you do not know how to take a screenshot, you can use one of the following:

- *Mac*: use Preview>File>Take Screenshot, or Cmd-Shift-3 and look for the screenshot on your desktop.
- *Windows*: Use the PrtScn button (or Windows Key + PrtScn or Alt _ PrtScn, etc.). You can also search for the Snipping Tool.
- *Linux*: you can use Screenshot or PrtScrn.

Retrieved from "https://hades.mech.northwestern.edu//index.php?title=ME_449_Assignment_1&oldid=26892"

- This page was last edited on 11 October 2024, at 12:27.
- This page has been accessed 19,724 times.