CMPSCI 403: Introduction to Robotics: Perception, Mechanics, Dynamics, and Control

HW 04

Joshua Holden

October 3, 2021

1 Robot Kinematic Simulation

In order to solve and display the joint rotations and positions for the given configurations, we use the $eul2rotm$ equivalent $eulerToOrientation$ and the SE(3) representation $SE3$ as found on Github(https://github.com/jbholden271/CS403-2021F):

```
% joint rotations in radians
q_base = [0,0,0,0,0,0];
q_1a = [0, pi/2, 0, pi/6, pi/2, 0];
q_1b = [0, 2*pi/3, 0, pi/3, pi/2, 0];
q_2a = [0, pi/2, pi/2, pi/6, pi/2, 0];
q_2b = [0, pi/3, pi/4, pi/3, pi/2, 0];
q_test = [0.8, 0.4, -pi/2-0.4, -0.9, pi/2+0.3, 0.7];
q_test2 = [0.8,0.4,-pi/2-0.4,0,0,0];
% "inject" rotations into the kinematic chain
q = q_test;
% define each frame
frame_0 = SE3([0,0,0],eulerToOrientation([q(1),0,0]));
frame_1 = SE3([0,0,0.15],eulerToOrientation([0,q(2),0]));
frame_2 = SE3([0.3,0,0],eulerToOrientation([0,q(3),0]));
frame_3 = SE3([0.15,0,0],eulerToOrientation([0,0,q(4)]));
frame_4 = SE3([0.1,0,0],eulerToOrientation([0,q(5),0]));
frame_5 = SE3([0.07,0,0],eulerToOrientation([0,0,q(6)]));
frame_6 = SE3([0.05,0,0],eulerToOrientation([0,0,0]));
% calculate joint posiitons and rotations
frame_1r = mult(frame_0, frame_1);
frame_2r = mult(frame_1r, frame_2);
frame_3r = mult(frame_2r, frame_3);
frame_4r = mult(frame_3r, frame_4);
frame_5r = mult(frame_4r, frame_5);
frame_6r = mult(frame_5r, frame_6);
% draw figure
figure
hold on
grid on
% draw lines
drawLine3D(SE3.getPos(frame_0),SE3.getPos(frame_1r));
drawLine3D(SE3.getPos(frame_1r),SE3.getPos(frame_2r));
drawLine3D(SE3.getPos(frame_2r),SE3.getPos(frame_3r));
drawLine3D(SE3.getPos(frame_3r),SE3.getPos(frame_4r));
drawLine3D(SE3.getPos(frame_4r),SE3.getPos(frame_5r));
drawLine3D(SE3.getPos(frame_5r),SE3.getPos(frame_6r));
% draw frames
```

```
drawCoordinate3DScale(SE3.getRot(frame_0),SE3.getPos(frame_0),0.5);
drawCoordinate3DScale(SE3.getRot(frame_1r),SE3.getPos(frame_1r),0.03);
drawCoordinate3DScale(SE3.getRot(frame_2r),SE3.getPos(frame_2r),0.03);
drawCoordinate3DScale(SE3.getRot(frame_3r),SE3.getPos(frame_3r),0.03);
drawCoordinate3DScale(SE3.getRot(frame_4r),SE3.getPos(frame_4r),0.03);
drawCoordinate3DScale(SE3.getRot(frame_5r),SE3.getPos(frame_5r),0.03);
drawCoordinate3DScale(SE3.getRot(frame_6r),SE3.getPos(frame_6r),0.1);
% label axis, set view
xlabel('$x$','interpreter','latex','fontsize',20)
ylabel('$y$','interpreter','latex','fontsize',20)
zlabel('$z$','interpreter','latex','fontsize',20)
axis equal
view(25,30)
% print SE3 representation of end-effector
disp(frame_6r.Matrix)
```

This homework is contained within the "kinSim.m" file.

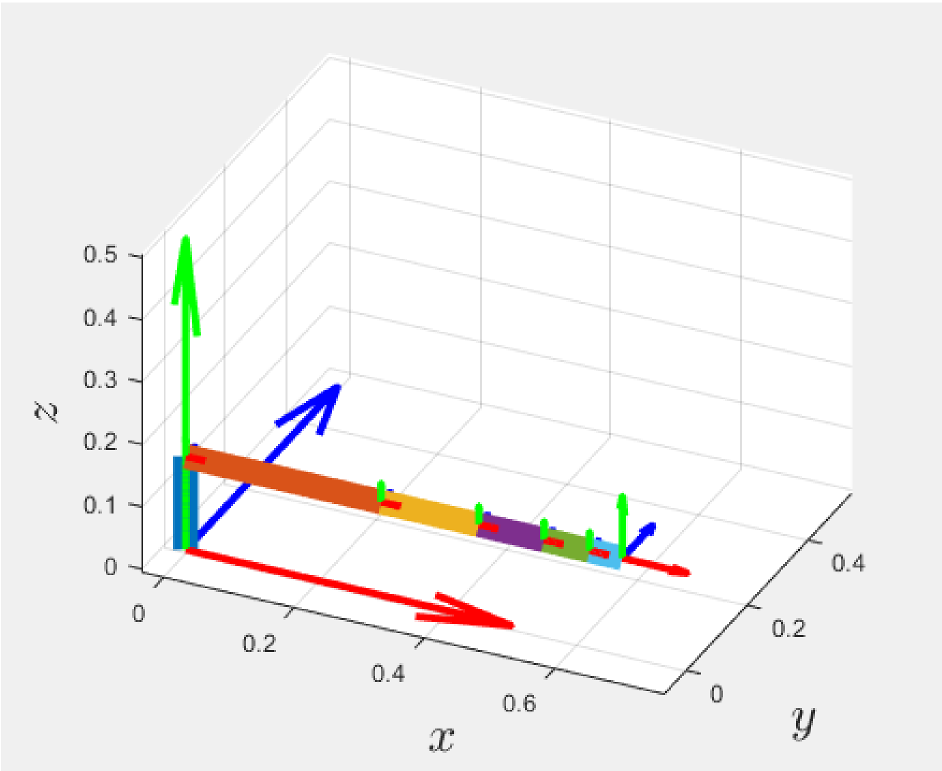

Figure 1: Base configuration

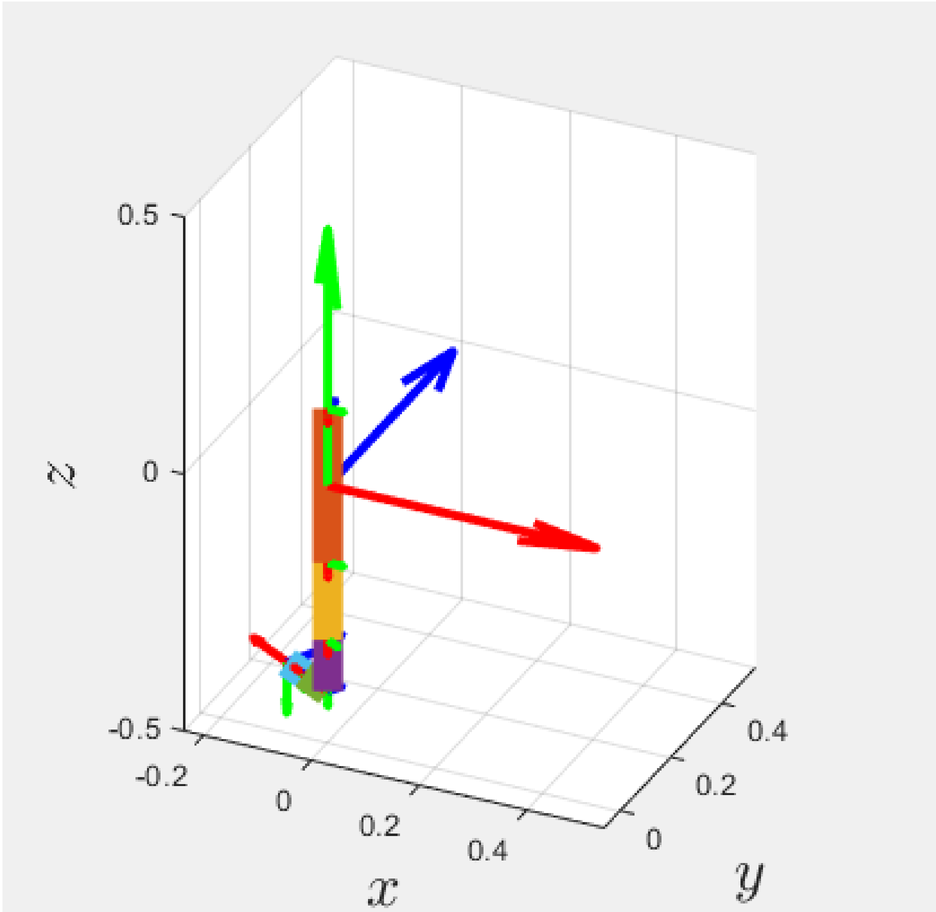

Figure 2: Question 1.a

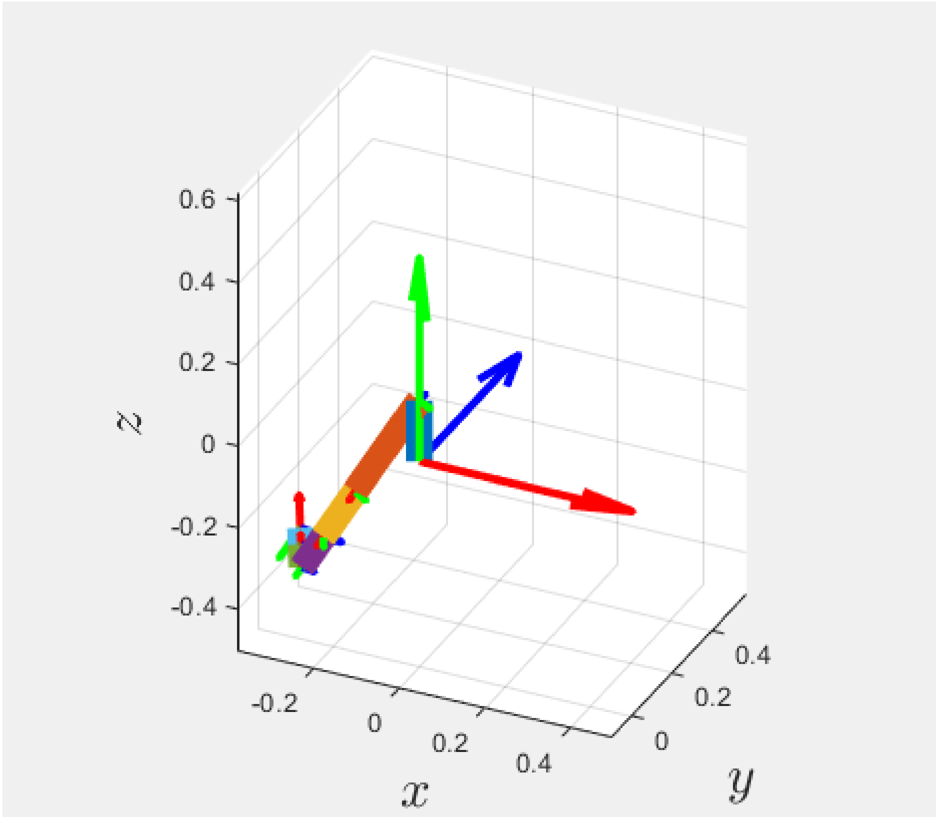

Figure 3: Question 1.b

2 End Effector in SE(3)

The end effector of the robot arm can be found in relation to the global frame by creating a kinematic chain of SE(3) representations of joints, and multiplying them together in sequence to receive ${}_0T_{ee}$.

For question 2.a)

```
-0.0000    0.0000   -1.0000   -0.2500
 0.5000    0.8660   -0.0000    0.0600
 0.8660   -0.5000   -0.0000   -0.0461
      0         0         0    1.0000
```

For question 2.b)

```
-0.4830    0.8365   -0.2588    0.0273
 0.8660    0.5000   -0.0000    0.1039
 0.1294   -0.2241   -0.9659   -0.3358
      0         0         0    1.0000
```

Notice that these are 4x4 SE(3) matrices, the SO(3) matrix is just the 3x3 matrix in the top left corner, with the first 3 rows of the rightmost column are the translation, in the typical SE(4) format.