MEAM 520

More Inverse Kinematics and Project 2: PUMA Light Painting

Katherine J. Kuchenbecker, Ph.D.
General Robotics, Automation, Sensing, and Perception Lab (GRASP)
MEAM Department, SEAS, University of Pennsylvania

GRASP LABORATORY

Lecture 23: November 21, 2013

MEAM 520

https://piazza.com/class/hf935b0sz1m5r3?cid=307

Google

piazza | MEAM 520 | Q & A | Course Page | Manage Class | Katherine J. Kuchenbecker

hw2 hw4 hw6 hw7 hw8 final_exam lecture10 lecture21 lecture27 project1 project2 midterm_exam other office_hours textbook matlab puma talks

Note History:

note

stop following 46 views

Actions

Choosing Your Team for PUMA Light Painting

Project 2 in MEAM 520 will be PUMA Light Painting. Each team of three students will write MATLAB code to make our PUMA 260 robot draw something interesting in the air with a colored light, as we take a long-exposure photograph.

The first step to this project is choosing your team. You must work with two other people in MEAM 520. Because we have 94 students there will be one team of four, but the vast majority of the teams will be teams of three. *The team of four has already been formed, so all further teams will be teams of three.*

Please figure out with whom you want to work. When you have chosen your team, send an email to meam520@seas.upenn.edu. The subject should be "Project 2 Team Selection". List the three full names in the body of the email. Only one person needs to submit the team. We will respond with a team number and post it on Piazza.

Looking for a teammate? Consider using the "Search for Teammates!" tool here on Piazza: @5

As another alternative, we are happy to assign you to a random partner or two. To ask us to do this, send an email to meam520@seas.upenn.edu with the subject "Project 2 Partner Request" with your full name (or two full names if in a pair) in the body of the email.

If possible, please do this by midnight (11:59 p.m.) on Thursday, November 21.

Any questions? Post a follow up. Thanks!

project2

edit · good note | 0

1 day ago by Katherine J. Kuchenbecker

followup discussions *for lingering questions and comments*

Start a new followup discussion

Compose a new followup discussion

Average Response Time: 12 min

Special Mentions: Katherine J. Kuchenbecker answered Thetadot/omega and Inductance in 13 min. 2 days ago

Online Now | This Week: 2 | 107

MEAM 520

https://piazza.com/class/hf935b0sz1m5r3?cid=308

piazza | MEAM 520 | Q & A | Course Page | Manage Class | Katherine J. Kuchenbecker

hw2 hw4 hw6 hw7 hw8 final_exam lecture10 lecture21 lecture27 project1 project2 midterm_exam other office_hours textbook matlab puma talks

Note History:

note

stop following 46 views

Actions

Teams for Project 2

Here is a list of team numbers and team members for Project 2. I will keep updating this as more teams are formed, following the guidelines explained in @307.

Team # - Members
201 - Jonathan Cousins, Rafi Pelles, Brandon Jennings
202 - Alex McCraw, Vivienne Clayton, Jay Davey
203 - Sarah Martezian, Kristin Marra, Kate Wessels, Carlie Badder
204 - Kristopher Li, Matthew Lisle, Mark Gallagher
205 - Wyatt Shapiro, Val Cohen, Bahram Banisadr
206 - Joe Hill, Dan LaMorte, Michael Latimer
207 - Akshitha Sriraman, Rahul Ajay Nafde, Prathik Prakash
208 - Te-Chuan Chou, Yifan Zhao, Jing Tang
209 - Shyamsundar Ramanathan, Vamsi Sonti, Shweta Krishnan
210 - Michael Rosenman, Yike Chen, Mariah Clark
211 - Jordan Landis, Nick Pesta, Iason Stamatiadis
212 - Jianqiao Li, Siyao Hu, Yu Fu
213 - Clara Midgley, Spyridon Karachalios, Adam Baitch
214 - Michael Lautman, Nate Baker, Nick Parrotta
215 - Antonino Mazzurco, Mitch Graves, Emily Plumb
#pin

project2

Please sit with your Project 2 team.

If you don't yet have a team, please sit with people you might want to work with for Project 2.

edit · good note | 0

6 hours ago by Katherine J. Kuchenbecker

followup discussions *for lingering questions and comments*

Average Response Time: 12 min

Special Mentions: Katherine J. Kuchenbecker answered Thetadot/omega and Inductance in 13 min. 2 days ago

Online Now | This Week: 2 | 107

Tentative MEAM 520 Calendar

Thursday 11/21 – Finish Inverse Kinematics,
Introduce Project 2

Tuesday 11/26 – Haptic Rendering
Introduce Homework 9
Tuesday 11/26 – Project 2 Part 1 (IK) due
Thursday 11/28 – Thanksgiving, no class

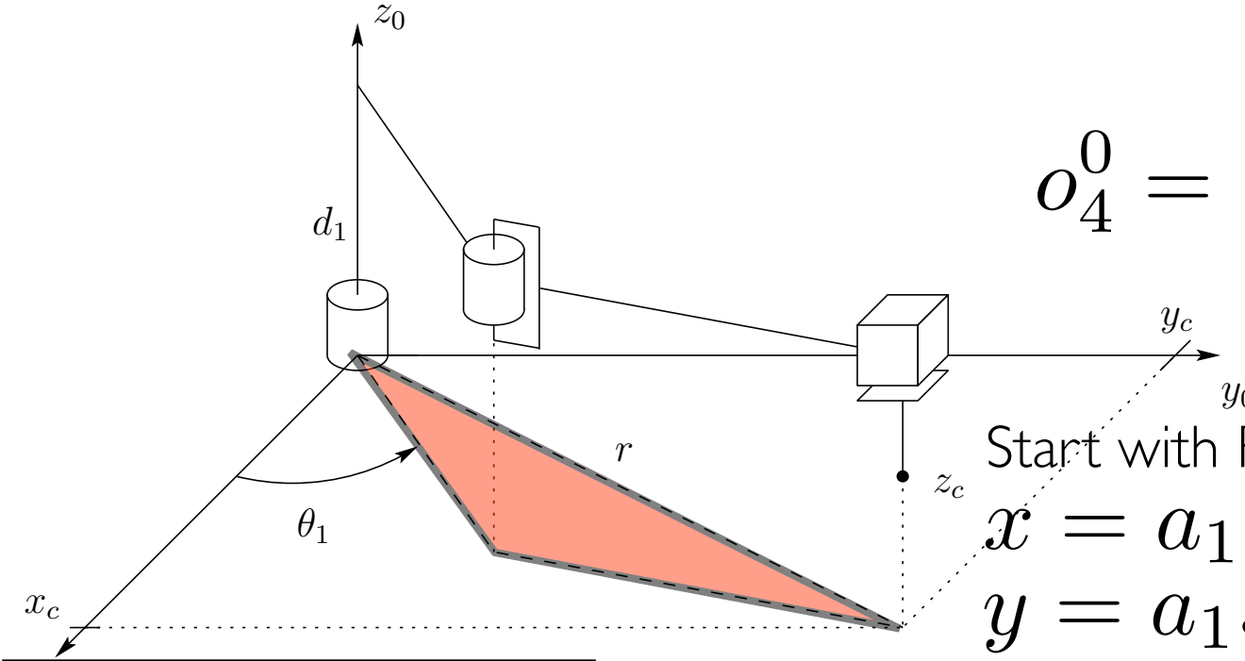
Tuesday 12/3 – Teleoperation
Thursday 12/5 – More Haptics and Teleoperation
Thursday 12/5 – Project 2 Part 2 (Sim Painting) due

Tuesday 12/10 – Mobile Robots
Tuesday 12/10 – Homework 9 (Haptics) due

Ongoing – Record Light Paintings on PUMA

Wednesday 12/18 – Final Exam noon to 2 p.m.

Inverse Position Kinematics Example: SCARA Robot



$$o_4^0 = \begin{bmatrix} o_x \\ o_y \\ o_z \end{bmatrix} \Rightarrow \begin{aligned} \theta_1 &= ? \\ \theta_2 &= ? \\ d_3 &= ? \end{aligned}$$

Example 3.10 SCARA Manipulator

As another example, we consider the SCARA manipulator whose forward kinematics is defined by T_4^0 from (3.30). The inverse kinematics solution is then given as the set of solutions of the equation

$$T_4^1 = \begin{bmatrix} R & o \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{124}c_4 + s_{124}s_4 & s_{124}c_4 - c_{124}s_4 & 0 & a_1c_1 + a_2c_{12} \\ s_{124}c_4 - c_{124}s_4 & -c_{124}c_4 - s_{124}s_4 & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & -1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.81)$$

We first note that the SCARA has only four degrees-of-freedom, not every possible H from ... allows a solution of (3.81). In fact we can easily see that there is no solution to (3.81) unless R is ...

$$R = \begin{bmatrix} \cdot & \cdot & s_{\alpha} \\ \cdot & \cdot & -c_{\alpha} \\ s_{\alpha} & -c_{\alpha} & 0 \end{bmatrix} \quad (3.82)$$

and if this is the case, the sum $\theta_1 + \theta_2 - \theta_4$ is determined by

$$\theta_1 + \theta_2 - \theta_4 = \alpha = \text{atan2}(r_{11}, r_{12}) \quad (3.83)$$

Projecting the manipulator configuration onto the x_0y_0 plane immediately yields the situation of Figure 3.12. We see from ...

$$\theta_2 = \text{atan2}\left(c_2, \pm\sqrt{1-c_2^2}\right) \quad (3.84)$$

where

$$c_2 = \frac{o_x^2 + o_y^2 - a_1^2 - a_2^2}{2a_1a_2} \quad (3.85)$$

$$\theta_1 = \text{atan2}(o_x, o_y) - \text{atan2}(a_1 + a_2c_2, a_2s_2) \quad (3.86)$$

We may then determine θ_4 from (3.83) as

$$\theta_4 = \theta_1 + \theta_2 - \alpha = \theta_1 + \theta_2 - \text{atan2}(r_{11}, r_{12}) \quad (3.87)$$

Finally d_3 is given as

$$d_3 = o_z + d_4 \quad (3.88)$$

Start with Forward Position Kinematics

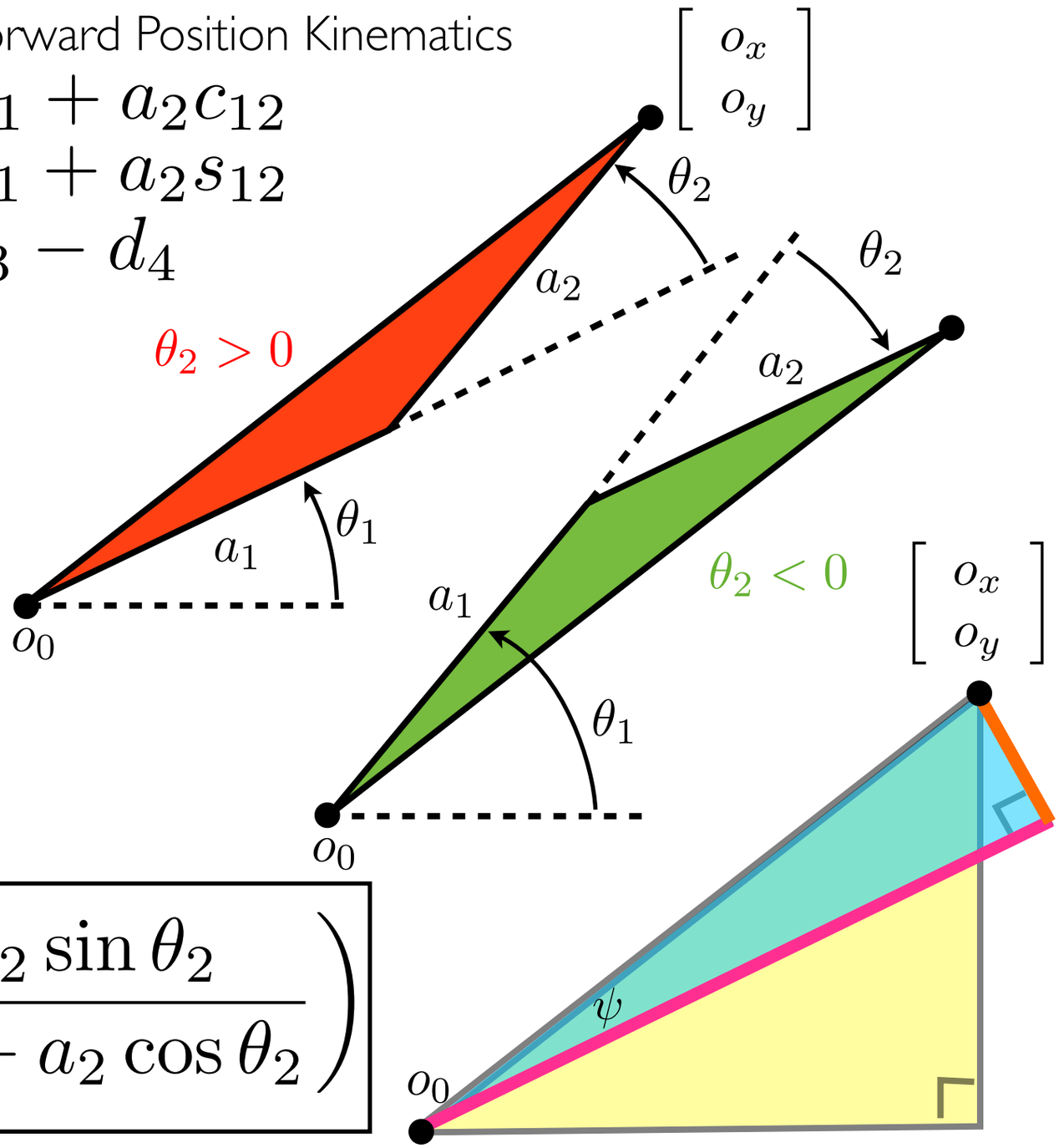
$$x = a_1c_1 + a_2c_{12}$$
$$y = a_1s_1 + a_2s_{12}$$
$$z = -d_3 - d_4$$

$$d_3 = -o_z - d_4$$

$$\cos\theta_2 = \frac{o_x^2 + o_y^2 - a_1^2 - a_2^2}{2a_1a_2}$$

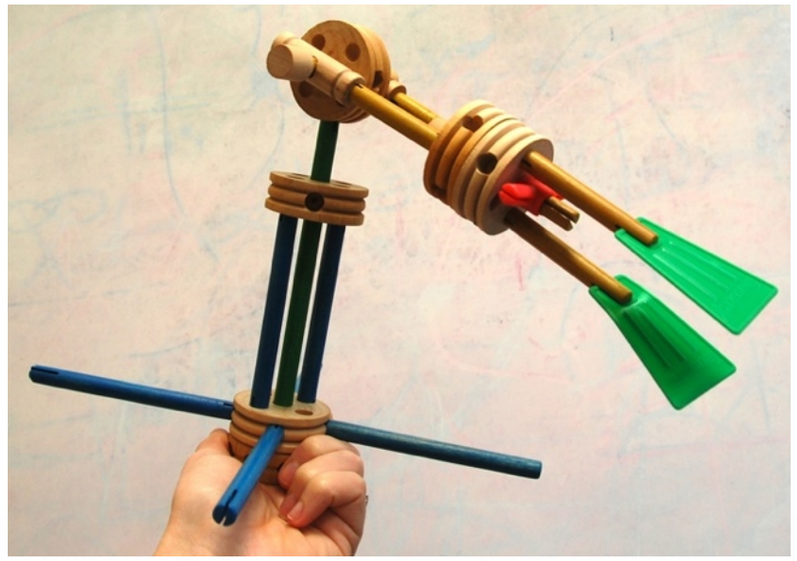
$$\theta_2 = \text{atan2}\left(\frac{\pm\sqrt{1-\cos^2\theta_2}}{\cos\theta_2}\right)$$

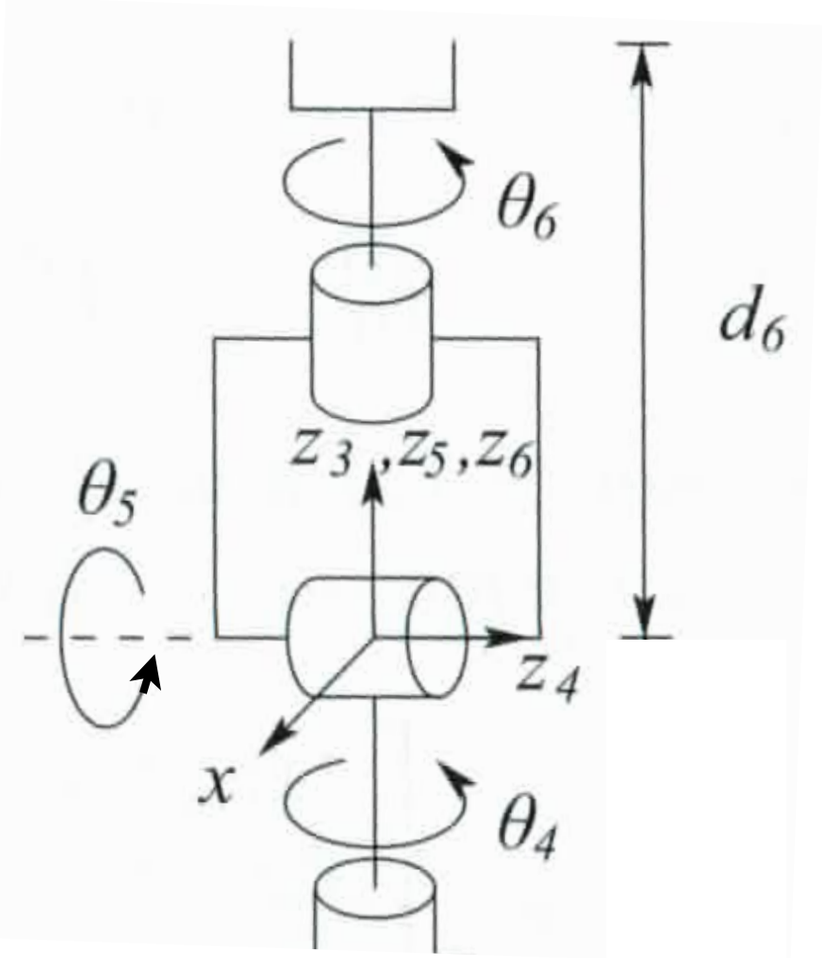
$$\theta_1 = \text{atan2}\left(\frac{o_y}{o_x}\right) - \text{atan2}\left(\frac{a_2\sin\theta_2}{a_1 + a_2\cos\theta_2}\right)$$



Inverse Orientation Kinematics

Given a rotation matrix **R** and the robot arm's joint angles, find the wrist joint angles that put the end-effector in the desired orientation.





$$T_6^3 = A_4 A_5 A_6 = \begin{bmatrix} R_6^3 & o_6^3 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & c_4s_5 & c_4s_5d_6 \\ s_4c_5c_6 + c_4s_6 & -s_4c_5s_6 + c_4c_6 & s_4s_5 & s_4s_5d_6 \\ -s_5c_6 & s_5s_6 & c_5 & c_5d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

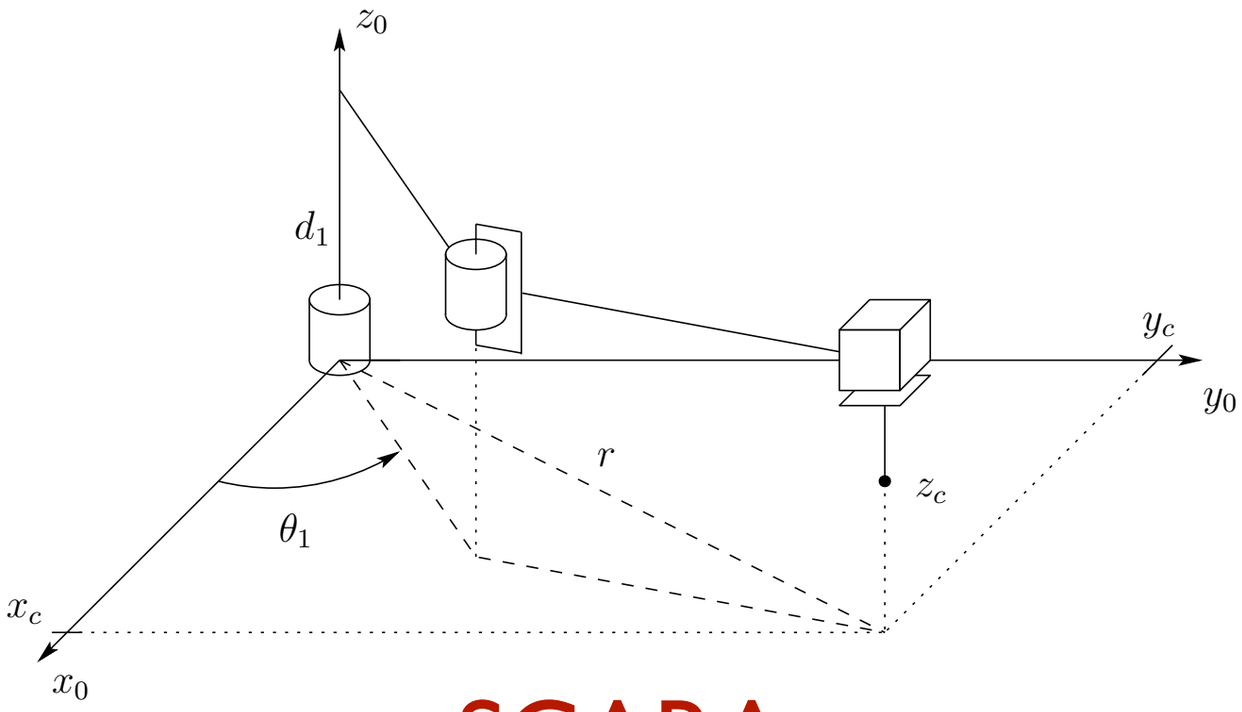
Equivalent to Euler angles for this choice of frames, where

$$\theta_4 = \phi \qquad \theta_5 = \theta \qquad \theta_6 = \psi$$

$$\mathbf{R} = \mathbf{R}_{z,\phi}\,\mathbf{R}_{y,\theta}\,\mathbf{R}_{z,\psi} = \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}$$

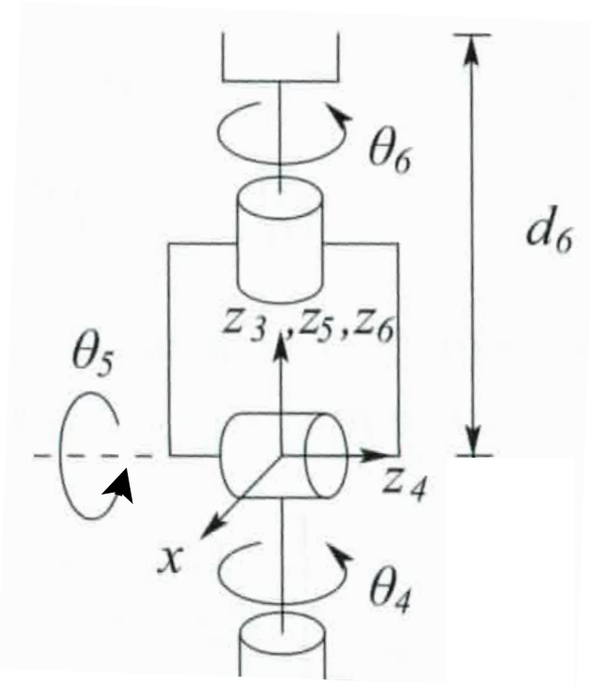
Two solutions for θ because s_θ can be either negative or positive.

SHV gives steps for inverse Euler.



SCARA

+



Spherical Wrist

Let's give the SCARA a spherical wrist and implement inverse orientation kinematics!

Talk to a partner about how to do this....
What steps are required?

Editor – /Users/kuchenbe/Documents/teaching/meam 520/lectures/23 project2/robotics23ik_code/scara_robot_with_wrist_fk.m

EDITOR | PUBLISH | VIEW

scara_robot_with_wrist_fk.m | scara_robot_with_wrist_circle_kuchenbe_v1.m | scara_robot_with_wrist_circle_kuchenbe_v

```
1  function [points_to_plot, x06, y06, z06] = scara_robot_with_wrist_fk(a1, a2, d6, theta1, theta2, d
2
3      % Calculate the six A matrices based on DH parameters.
4  -   A1 = dh_kuchenbe(a1,    0,   0, theta1);
5  -   A2 = dh_kuchenbe(a2,   pi,   0, theta2);
6  -   A3 = dh_kuchenbe(0,     0,  d3,      0);
7  -   A4 = dh_kuchenbe(0, -pi/2,   0, theta4);
8  -   A5 = dh_kuchenbe(0,  pi/2,   0, theta5);
9  -   A6 = dh_kuchenbe(0,     0,  d6, theta6);
10
11     % Calculate the position of the origin for each frame.
12 -   o = [0 0 0 1]';
13 -   o0 = o;
14 -   o1 = A1*o0;
15 -   o2 = A1*A2*o0;
16 -   o3 = A1*A2*A3*o0;
17 -   o4 = A1*A2*A3*A4*o0;
18 -   o5 = A1*A2*A3*A4*A5*o0;
19 -   o6 = A1*A2*A3*A4*A5*A6*o0;
20
21     % Put the points together.
22 -   points_to_plot = [[0 0 -2 1]' o0 o1 o2 o3 o4 o5 o6];
23
24     % Save the full transformation in T06.
25 -   T06 = A1*A2*A3*A4*A5*A6;
26
27     % Length of coordinate frame vectors, in meters.
28 -   vlen = 0.3;
29
30     % Calculate the coordinates of the x, y, and z unit vectors of frame 6 in
31     % frame 0. Each of these vectors starts at the origin of frame 6 and ends
32     % at the distance vlen along the designated axis.  We calculate the
33     % location of the end by multiplying T06 into a scaled unit vector in the
34     % correct direction.
35 -   x06 = [o6 (T06 * [vlen 0 0 1]')];
36 -   y06 = [o6 (T06 * [0 vlen 0 1]')];
37 -   z06 = [o6 (T06 * [0 0 vlen 1]')];
```

Link	a_i	α_i	d_i	θ_i
4	0	-90	0	θ_4^*
5	0	90	0	θ_5^*
6	0	0	d_6	θ_6^*

scara robot with wrist fk | Ln 1 Col 1

Editor – /Users/kuchenbe/Documents/teaching/meam 520/lectures/23 project2/robotics23ik_code/scara_robot_with_wr...

EDITOR | PUBLISH | VIEW

scara_robot_with_wrist_circle_kuchenbe_v1.m | scara_robot_with_wrist_circle_kuchenbe_v2.m | scara_robot_with_wr

```
1   %% scara_robot_circle_with_wrist_kuchenbe_v1.m
2   %
3   % This Matlab demonstrates inverse orientation kinematics with a SCARA
4   % robot with a spherical wrist.  This was shown in MEAM 520 lecture on
5   % November 21, 2013.
6
7   %% SETUP
8
9   % Clear all variables from the workspace.
10  clear all
11
12  % Clear the console, so you can more easily find any errors that may occur.
13  clc
14
15  % Define our time vector.
16  tStart = 0;    % The time at which the simulation starts, in seconds.
17  tStep = 0.04; % The simulation's time step, in seconds.
18  tEnd = 2*pi;     % The time at which the simulation ends, in seconds.
19  t = (tStart:tStep:tEnd)';  % The time vector (a column vector).
20
21  % Set whether to animate the robot's movement and how much to slow it down.
22  pause on;  % Set this to off if you don't want to watch the animation.
23  GraphingTimeDelay = 0.001; % The length of time that Matlab should pause between positions when
24
25
26  %% ROBOT PARAMETERS
27
28  % This problem is about the first three joints (RRP) of a SCARA
29  % manipulator.  This robot's forward kinematics are worked out on pages 91
30  % to 93 of the SHV textbook, though we are ignoring the fourth joint (the
31  % wrist).
32
33  % Define robot link lengths.
34  a1 = 1.0; % Distance between joints 1 and 2, in meters.
35  a2 = 0.7; % Distance between joints 2 and 3, in meters.
36  d6 = a1/4; % Offset from wrist center to end-effector, in meters.
37
38
```

script | Ln 3 Col 63

Editor – /Users/kuchenbe/Documents/teaching/meam 520/lectures/23 project2/robotics23ik_code/scara_robot_with_wr...

EDITOR | PUBLISH | VIEW

scara_robot_with_wrist_circle_kuchenbe_v1.m | scara_robot_with_wrist_circle_kuchenbe_v2.m | scara_robot_with_wr

```
38
39      %% DEFINE CIRCULAR MOTION
40
41      % We want the SCARA to draw a vertical circle parallel to the x-z plane.
42
43      % Define the radius of the circle.
44 -    radius = .4; % meters
45
46      % Define the y-value for the plane that contains the circle.
47 -    y_offset = -1; % meters
48
49      % Define the x and z coordinates for the center of the circle.
50 -    x_center = -.5; % meters
51 -    z_center = -1.2; % meters
52
53      % Set the desired x, y, and z positions over time given the cir
54 -    ox_history = x_center + radius * sin(t);
55 -    oy_history = y_offset * ones(size(t));
56 -    oz_history = z_center + radius * cos(t);
57
58
59      %% SIMULATION
60
61      % Notify the user that we're starting the animation.
62 -    disp('Starting the animation.')
63
64      % Show a message to explain how to cancel the simulation and graphing.
65 -    disp('Click in this window and press control-c to stop the code.')
66
67      % Initialize a matrix to hold the position of the robot's tip over time.
68      % The first row is the x-coordinate, second is y, and third is z, all in
69      % the base frame.  You are welcome to make this 4 rows if you want to use
70      % the homogeneous representation for points.  It has the same number of
71      % columns as t has rows, one tip position for every time step in the
72      % simulation.  We keep track of this history so we can trace out the
73      % trajectory of where the robot's tip has been.
74 -    tip_history = zeros(3,length(t));
75
```

Creating a circle the same way as before.

script | Ln 15 Col 26

Doing the same position IK as before.

Editor – /Users/kuchenbe/Documents/teaching/meam 520/lectures/23 project2/robotics23ik_code/scara_robot_with_wr...

EDITOR | PUBLISH | VIEW

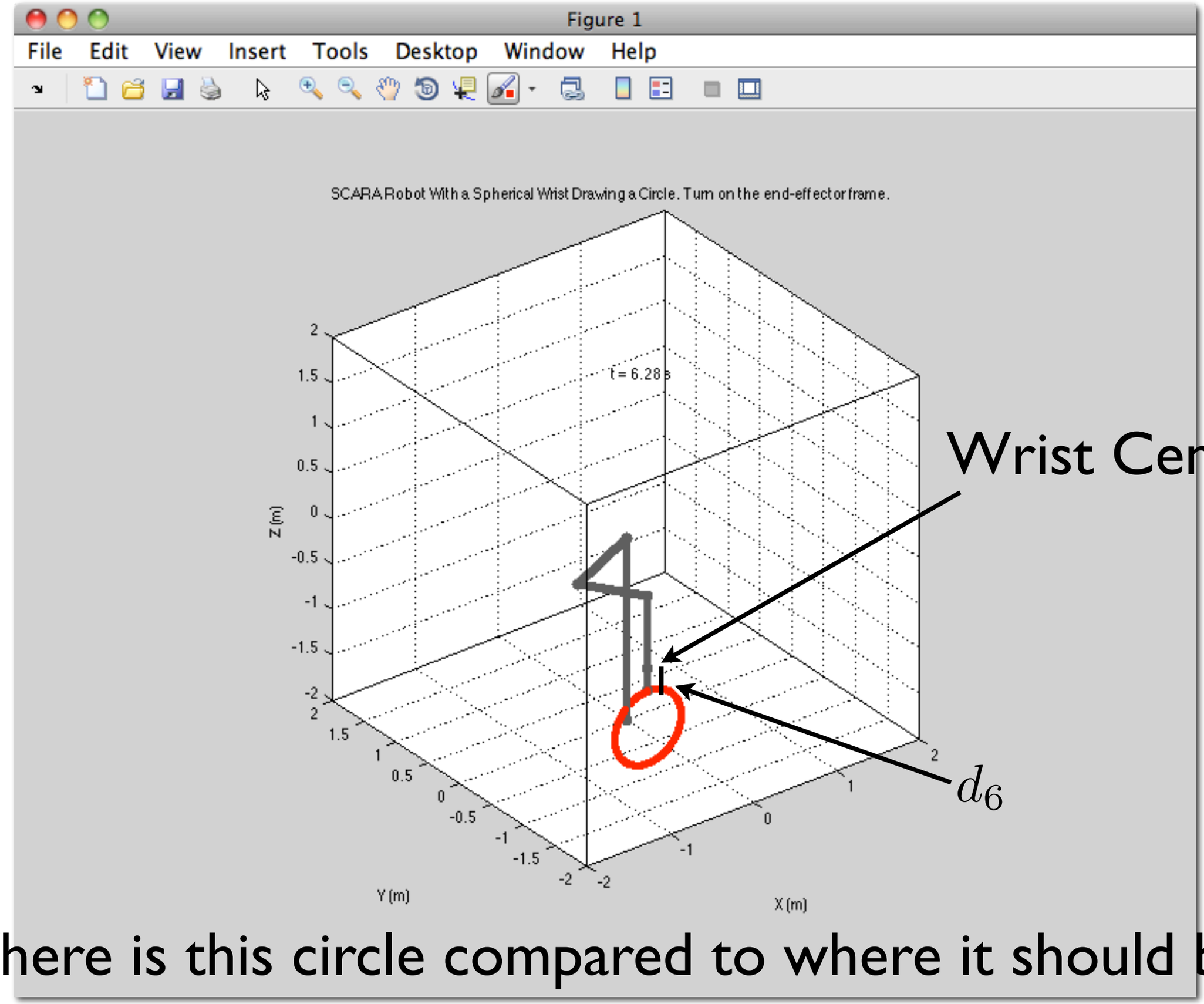
scara_robot_with_wrist_circle_kuchenbe_v1.m | scara_robot_with_wrist_circle_kuchenbe_v2.m | scara_robot_with_wr

```
76      % Step through the time vector to animate the robot.
77 -    for i = 1:length(t)
78
79          % Pull the current values of ox, oy, and oz from their histories.
80 -        ox = ox_history(i);
81 -        oy = oy_history(i);
82 -        oz = oz_history(i);
83
84          % Calculate theta1, theta2, and d3 given the robot's parameters (a1 and
            % a2) and the current desired position for its tip ([ox oy oz]').

            % Calculate the cosine of theta2 using law of cosines.
            c2 = (ox.^2 + oy.^2 - a1^2 - a2^2)/(2*a1*a2);

            % Calculate the positive and negative solutions for theta2.
            theta2_pos = atan2(sqrt(1-c2^2),c2); % Corrected solution from our derivation.
            theta2_neg = atan2(-sqrt(1-c2^2),c2); % Corrected solution from our derivation.

            % Arbitrarily choose the positive solution.
            theta2 = theta2_pos;

            % Uncomment this line to choose the negative solution instead.
            %theta2 = theta2_neg;

            % Calculate theta1 using a pair of inverse tangents.  Note that MATLAB
            % atan2 takes the numerator and then the denominator.
            theta1 = atan2(oy,ox) - atan2(a2*sin(theta2),a1+a2*cos(theta2));

            % Calculate d3.
            d3 = -oz; % Corrected solution from our derivation.

            % Use provided .p function to calculate the points of the robot that we
            % should plot to show in the animation.
            [points_to_plot, x06, y06, z06] = scara_robot_with_wrist_fk(a1, a2, d6, ...
                theta1, theta2, d3, 0, 0, 0);

            % Grab the final plotted point for
            tip_history(:,i) = points_to_plot(
```

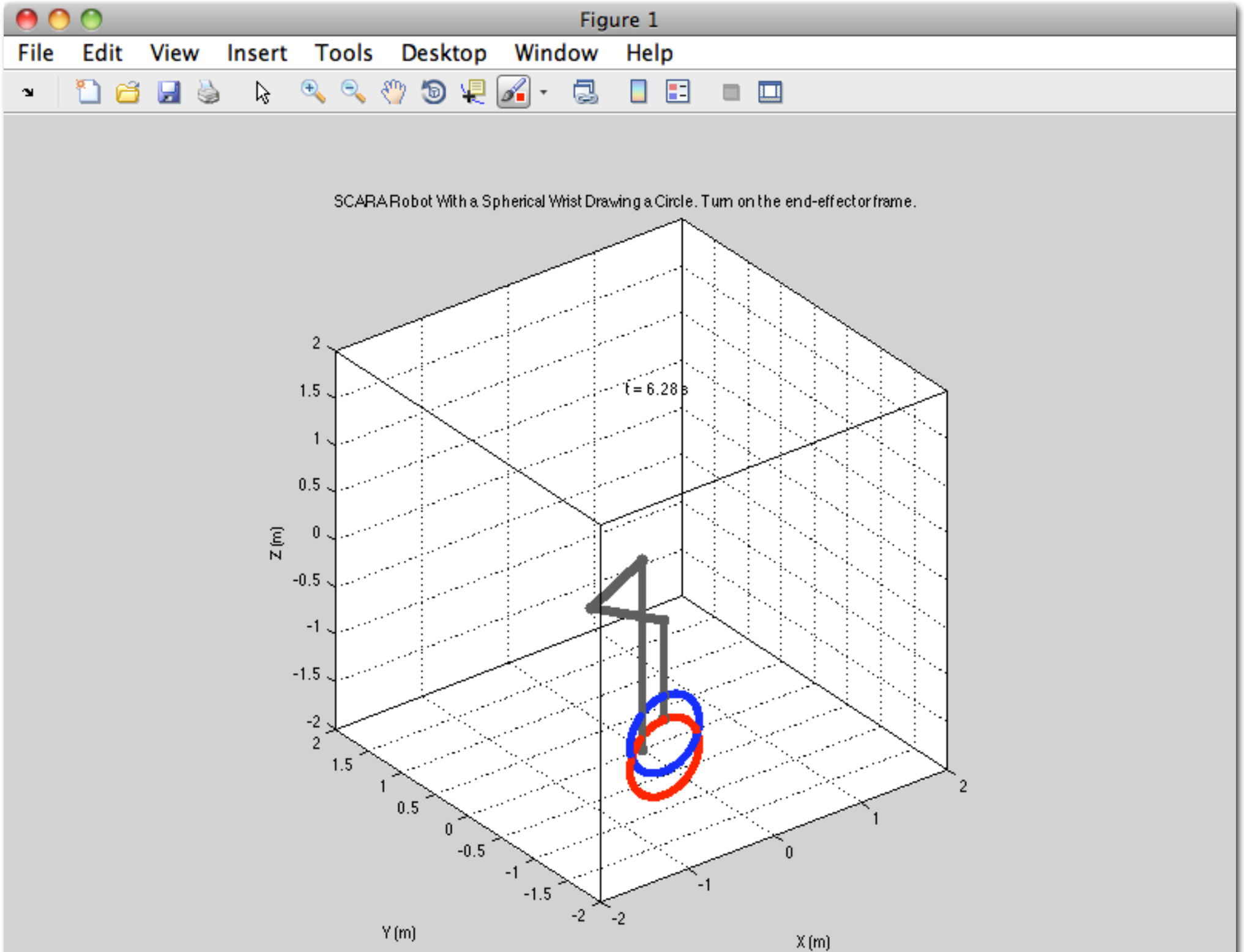
Setting wrist angles to zero for now.

script | Ln 15 Col 26

Run v1 code.



Where is this circle compared to where it should be?



Circle is shifted straight down by d6.

Editor – /Users/kuchenbe/Documents/teaching/meam 520/lectures/23 project2/robotics23ik_code/scara_robot_with_wr...

EDITOR | PUBLISH | VIEW

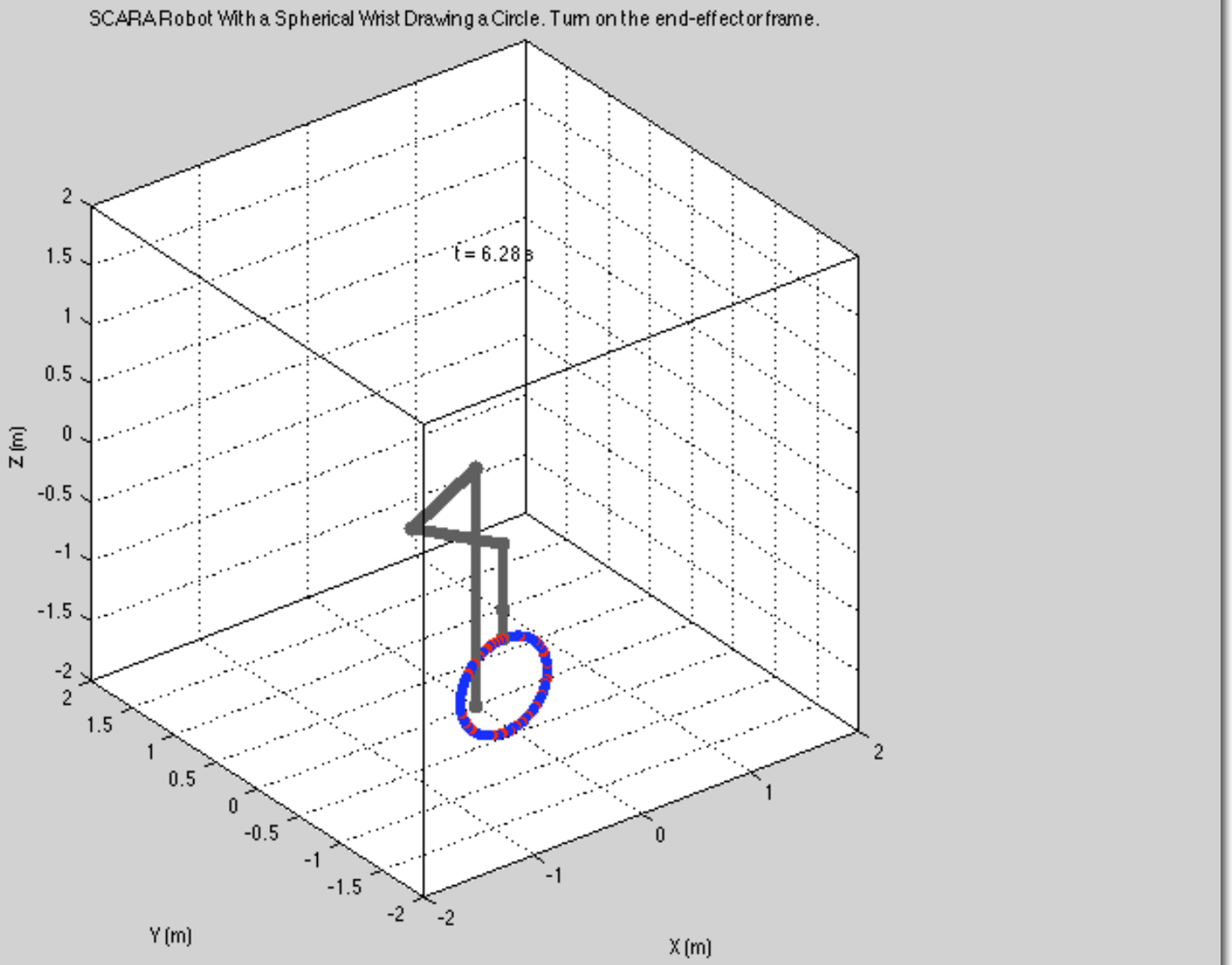
scara_robot_with_wrist_circle_kuchenbe_v1.m | scara_robot_with_wrist_circle_kuchenbe_v2.m | scara_robot_with_wr

```
88 -      c2 = (ox.^2 + oy.^2 - a1^2 - a2^2)/(2*a1*a2);
89
90        % Calculate the positive and negative solutions for theta2.
91 -      theta2_pos = atan2(sqrt(1-c2^2),c2); % Corrected solution from our derivation.
92 -      theta2_neg = atan2(-sqrt(1
93
94        % Arbitrarily choose the p
95 -      theta2 = theta2_pos;
96
97        % Uncomment this line to c
98        %theta2 = theta2_neg;
99
100       % Calculate theta1 using a
101       % atan2 takes the numerato
102 -     theta1 = atan2(oy,ox) - at
103
104       % Calculate d3.
105 -     d3 = -oz - d6; % Corrected
106
107       % Use provided .p function
108       % should plot to show in t
109 -     [points_to_plot, x06, y06,
110           theta1, theta2, d3, 0,
111
112       % Grab the final plotted p
113 -     tip_history(:,i) = points_
114
115       % Check if this is the fir
116 -     if (i == 1)
117           % Open figure 1.
118 -         figure(1);
119
120           % The first time, plot
121           % This is a 3D plot wi
122           % neighboring points,
123 -         hrobot = plot3(points_
124               '.-','linewidth',5
125
```

3 usages of "d6" found

Figure 1

File Edit View Insert Tools Desktop Window Help



Inverse Orientation Kinematics Questions

1. When the SCARA draws a circle with all three of its wrist joints at zero, how does the orientation of the end-effector frame change over time? Sketch and/or explain.

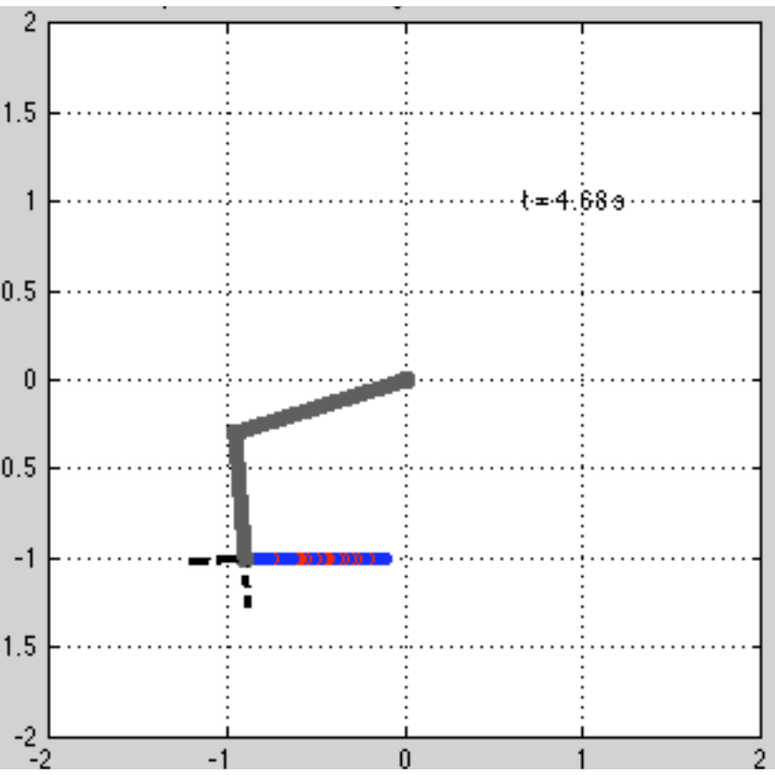
2. What shape will the SCARA draw if we set the fifth joint at plus or minus ninety degrees?

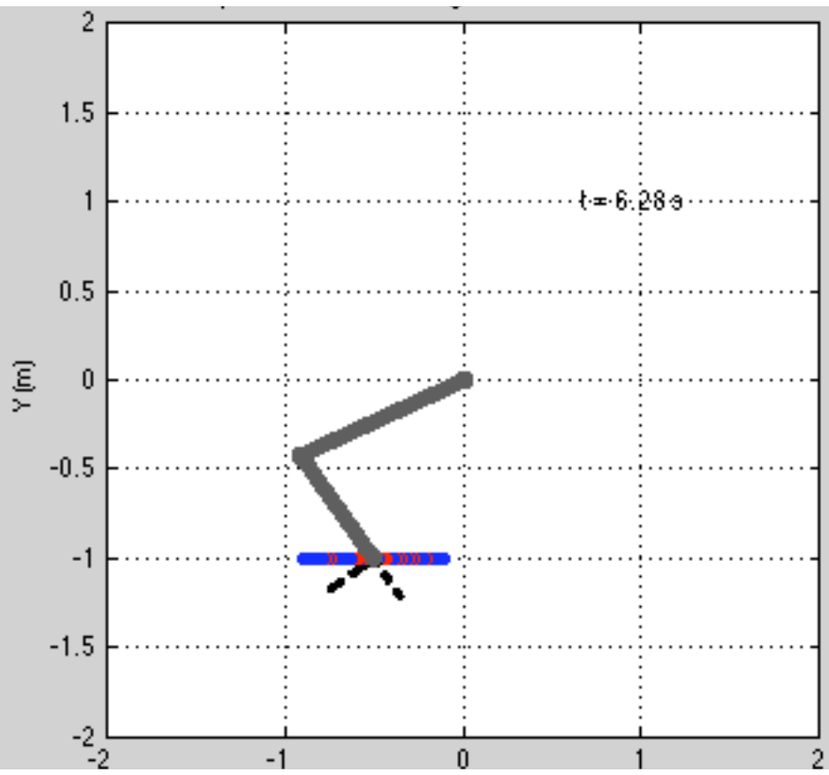
3. We want the end-effector's z-axis to point out toward the camera with its x-axis horizontal to the right. What rotation matrix **R** should we give to the inverse orientation kinematics?

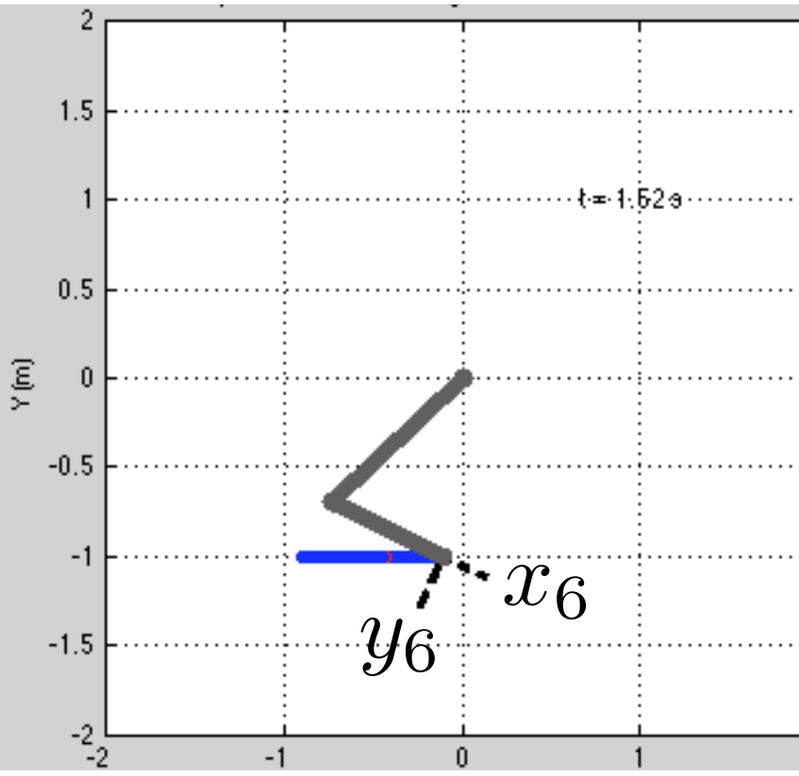
1. When the SCARA draws a circle with all three of its wrist joints at zero, how does the orientation of the end-effector frame change over time?

Talk with your partners.

Top views, looking down over the robot





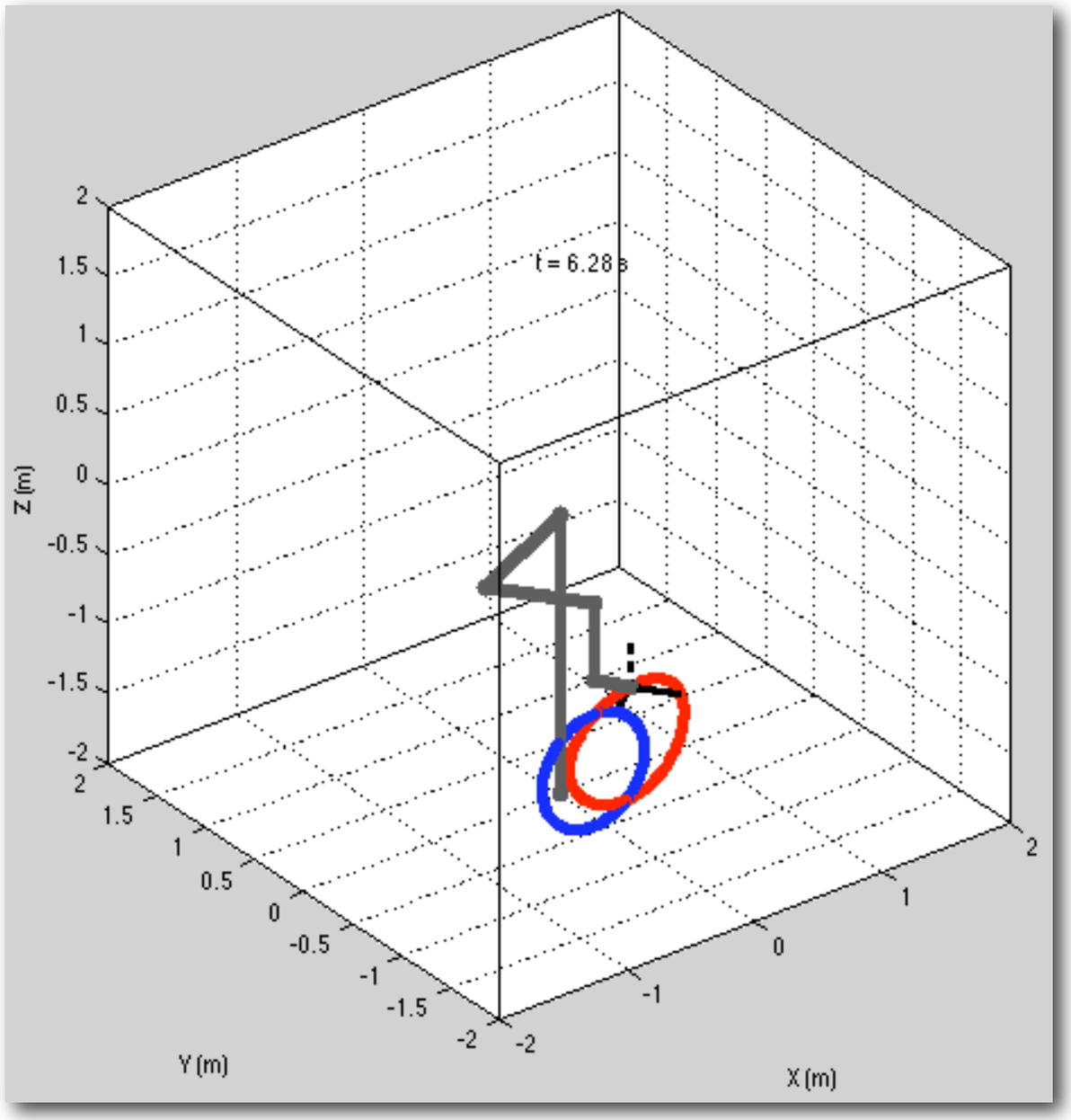


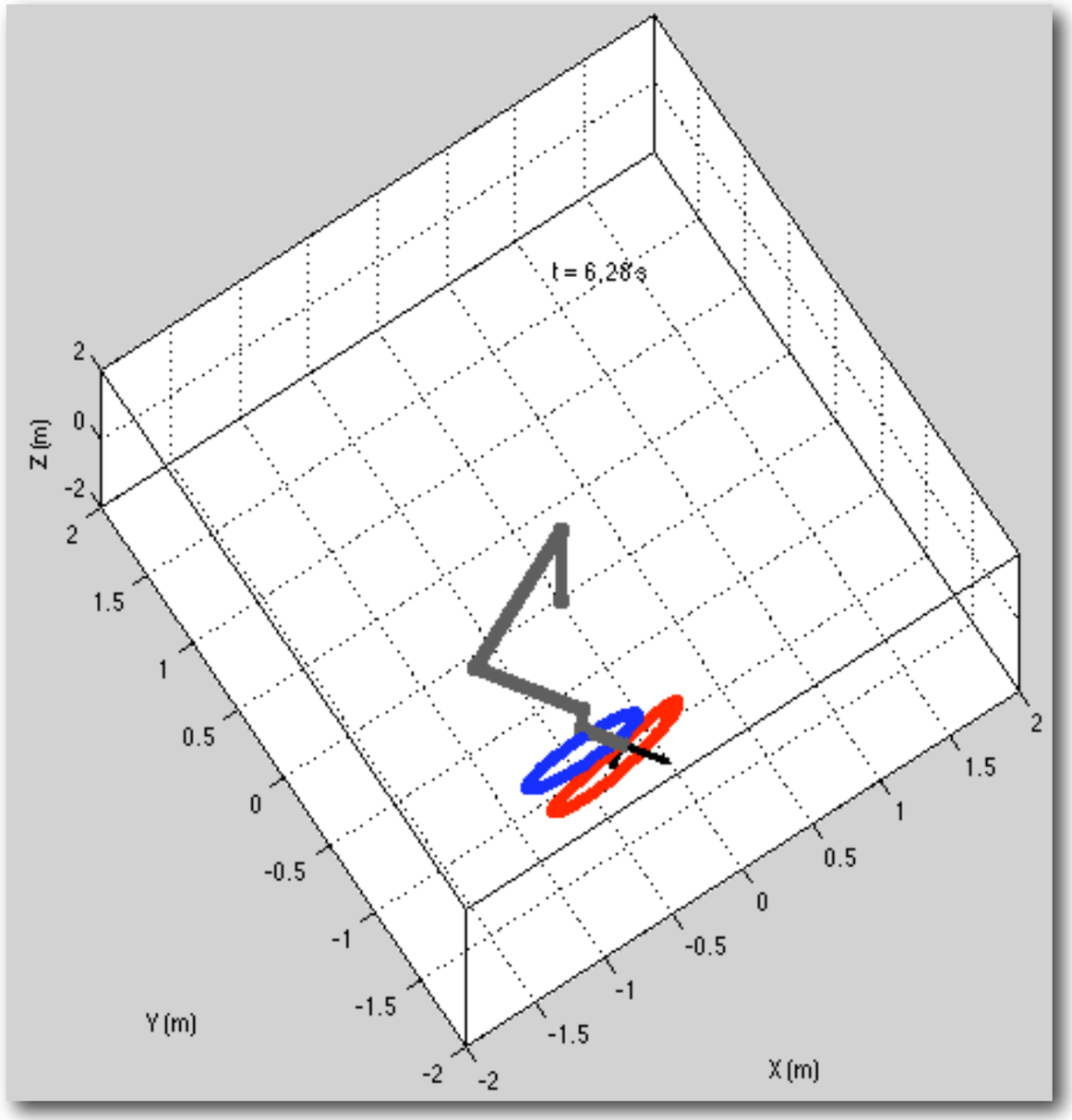
z6 always points down (same as z3)

x6 always points along forearm (same as x3)

2. What shape will the SCARA draw if we set the fifth joint at plus or minus ninety degrees?

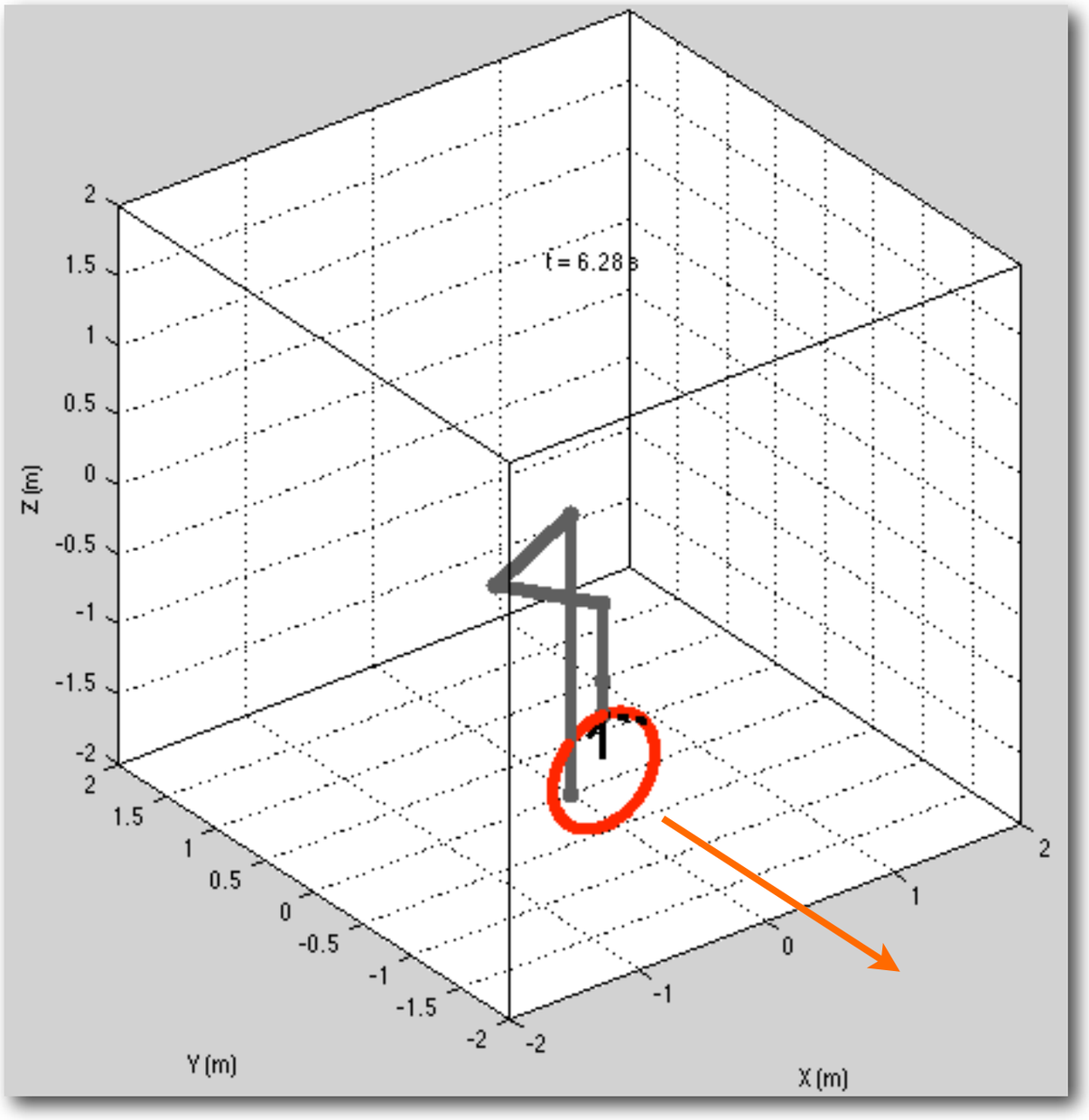
$$\theta_5 = 90^\circ$$

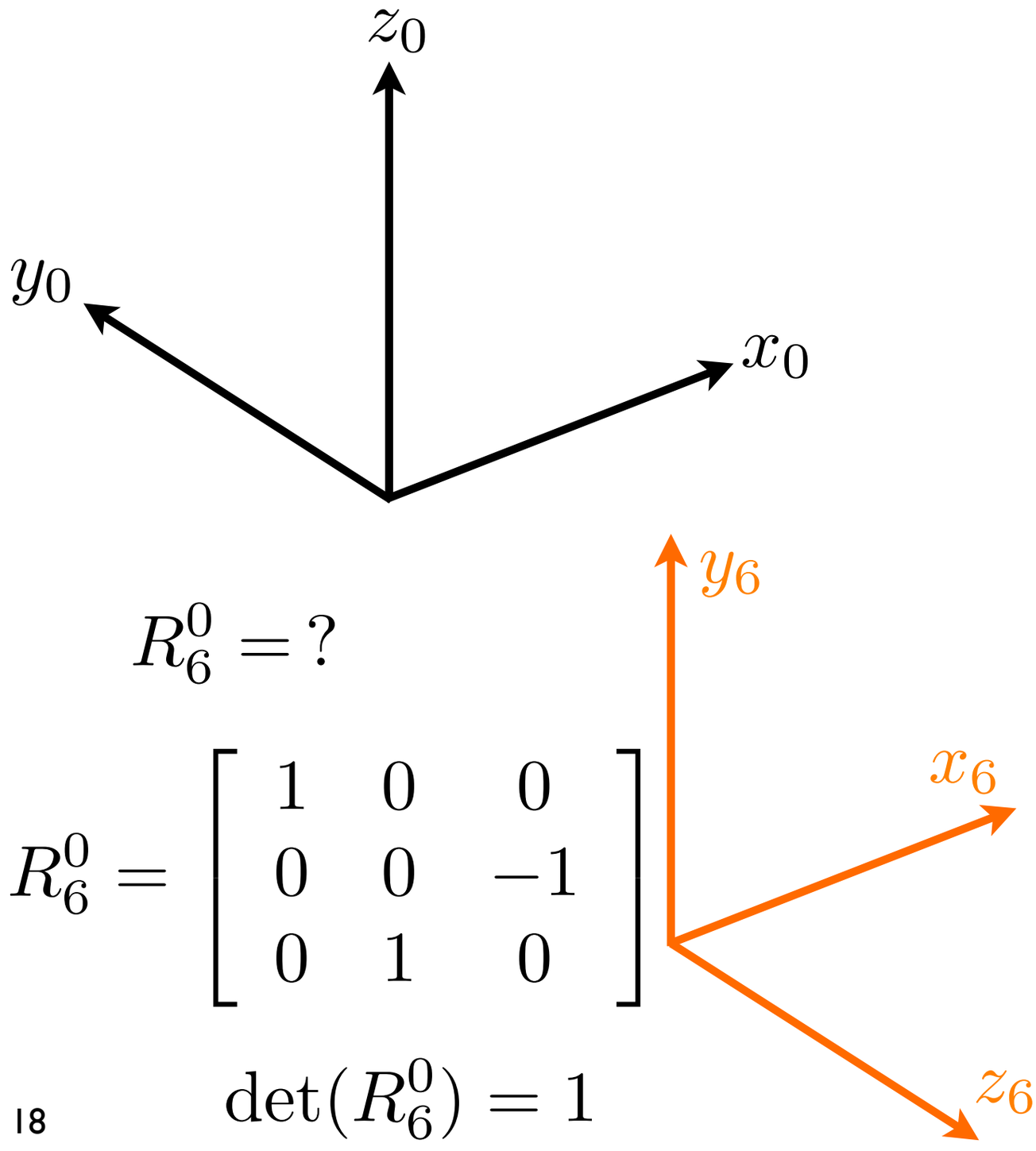




Shifted up and distorted!

3. We want the end-effector's z-axis to point out toward the camera with its x-axis horizontal to the right. What rotation matrix **R** should we give to the inverse orientation kinematics?





$$R_6^0 = ?$$

$$R_6^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\det(R_6^0) = 1$$

Editor – /Users/kuchenbe/Documents/teaching/meam 520/lectures/23 project2/robotics23ik_code/scara_robot_with_wr...

EDITOR | PUBLISH | VIEW

scara_robot_with_wrist_circle_kuchenbe_v1.m | scara_robot_with_wrist_circle_kuchenbe_v2.m | scara_robot_with_wr

```
27
28      % This problem is about the first three joints (RRP) of a SCARA
29      % manipulator.  This robot's forward kinematics are worked out on pages 91
30      % to 93 of the SHV textbook, though we are ignoring the fourth joint (the
31      % wrist).
32
33      % Define robot link lengths.
34 -    a1 = 1.0; % Distance between joints 1 and 2, in meters.
35 -    a2 = 0.7; % Distance between joints 2 and 3, in meters.
36 -    d6 = a1/4; % Offset from wrist center to end-effector, in meters.
37
38
39      %% DEFINE CIRCULAR MOTION
40
41      % We want the SCARA to draw a vertical circle parallel to the x-z plane.
42
43      % Define the radius of the circle.
44 -    radius = .4; % meters
45
46      % Define the y-value for the plane that contains the circle.
47 -    y_offset = -1; % meters
48
49      % Define the x and z coordinates for the center of the circle.
50 -    x_center = -.5; % meters
51 -    z_center = -1.2; % meters
52
53      % Set the desired x, y, and z positions over time given the circle parameters.
54 -    ox_history = x_center + radius * sin(t);
55 -    oy_history = y_offset * ones(size(t));
56 -    oz_history = z_center + radius * cos(t);
57
58      % Set the end-effector orientation that we want.
59 -    R = [1   0   0; ...
60           0   0  -1; ...
61           0   1   0]
62
63 -    det(R)
64
```

What next?

Kinematic Decoupling

Created R and checked determinant

script | Ln 63 Col 7

given $\mathbf{H} = \begin{bmatrix} \mathbf{R} & o \\ 0 & 1 \end{bmatrix}$ and a certain manipulator with n joints

find $q_1, \ldots, q_n$ such that $\mathbf{T}_n^0(q_1, \ldots, q_n) = \mathbf{H}$ ugly!

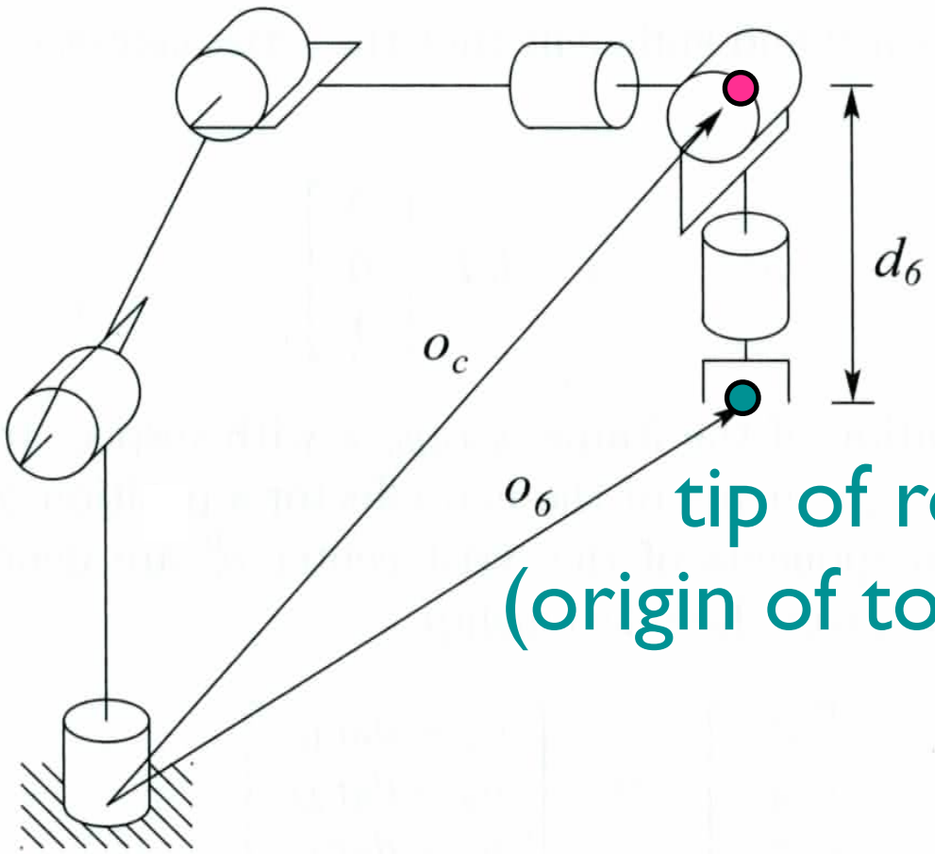
The inverse kinematics problem may or may not have a solution.

Even if one exists, the solution may or may not be unique.

Trick: Exploit the kinematic structure of the manipulator.

If the robot has a spherical wrist, use **Kinematic Decoupling**.

wrist center



do position first!

tip of robot (origin of tool frame)

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix}$$

position

$$R = R_3^0 R_6^3$$

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R$$

orientation

Editor – /Users/kuchenbe/Documents/teaching/meam 520/lectures/23 project2/robotics23ik_code/scara_robot_with_wr...

EDITOR | PUBLISH | VIEW

scara_robot_with_wrist_circle_kuchenbe_v1.m | scara_robot_with_wrist_circle_kuchenbe_v2.m | scara_robot_with_wr

```
82      % Step through the time vector to animate the robot.
83 -    for i = 1:length(t)
84
85          % Pull the current values of ox, oy, and oz from
86 -        ox = ox_history(i);
87 -        oy = oy_history(i);
88 -        oz = oz_history(i);
89
90          % Calculate where the wrist center position need
91 -        oc = [ox oy oz]' - d6*R(:,3);
92
93          % Pull out the components of wrist center and store in ox, oy, oz.
94 -        ox = oc(1);
95 -        oy = oc(2);
96 -        oz = oc(3);
97
98          % Calculate th                    s parameters (a1 and
99          % a2) and the                     [ox oy oz]').
100
101         % Calculate th                    es.
102 -       c2 = (ox.^2 +
103
104         % Calculate the positive and negative solutions for theta2.
105 -       theta2_pos = atan2(sqrt(1-c2^2),c2); % Corrected solution from our derivation.
106 -       theta2_neg = atan2(-sqrt(1-c2^2),c2); % Corrected solution from our derivation.
107
108         % Arbitrarily choose the positive solution.
109 -       theta2 = theta2_pos;
110
111         % Uncomment this line to cho
112         %theta2 = theta2_neg;
113
114         % Calculate theta1 using a                    AB
115         % atan2 takes the numerator
116 -       theta1 = atan2(oy,ox) - atan
117
118         % Calculate d3.
119 -       d3 = -oz; % Corrected solution from our derivation.
```

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix}$$

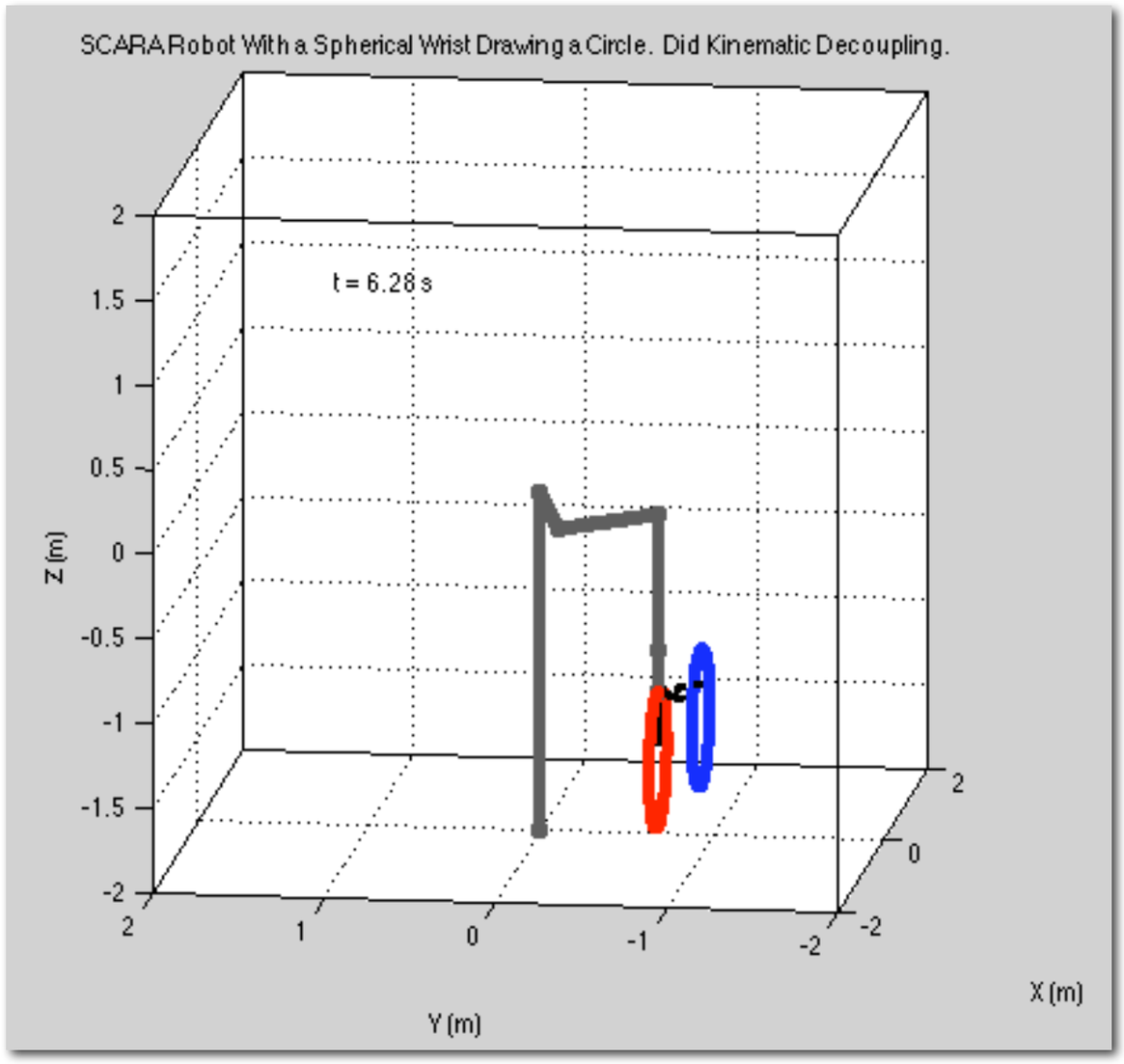
position

Put wrist center into variables that held tip position

Same inverse position kinematics, but remove d6 offset

script | Ln 119 Col 37

Run v2 code.



Is this correct?

Yes.

How can you tell?

The wrist is where it needs to be to put the tip on the blue circle, pointing in negative y0 direction.

Next step?

Calculate wrist joint angles.

What questions do you have ?

Inverse Kinematics

given $\mathbf{H} = \begin{bmatrix} \mathbf{R} & o \\ 0 & 1 \end{bmatrix}$ and a certain manipulator with n joints

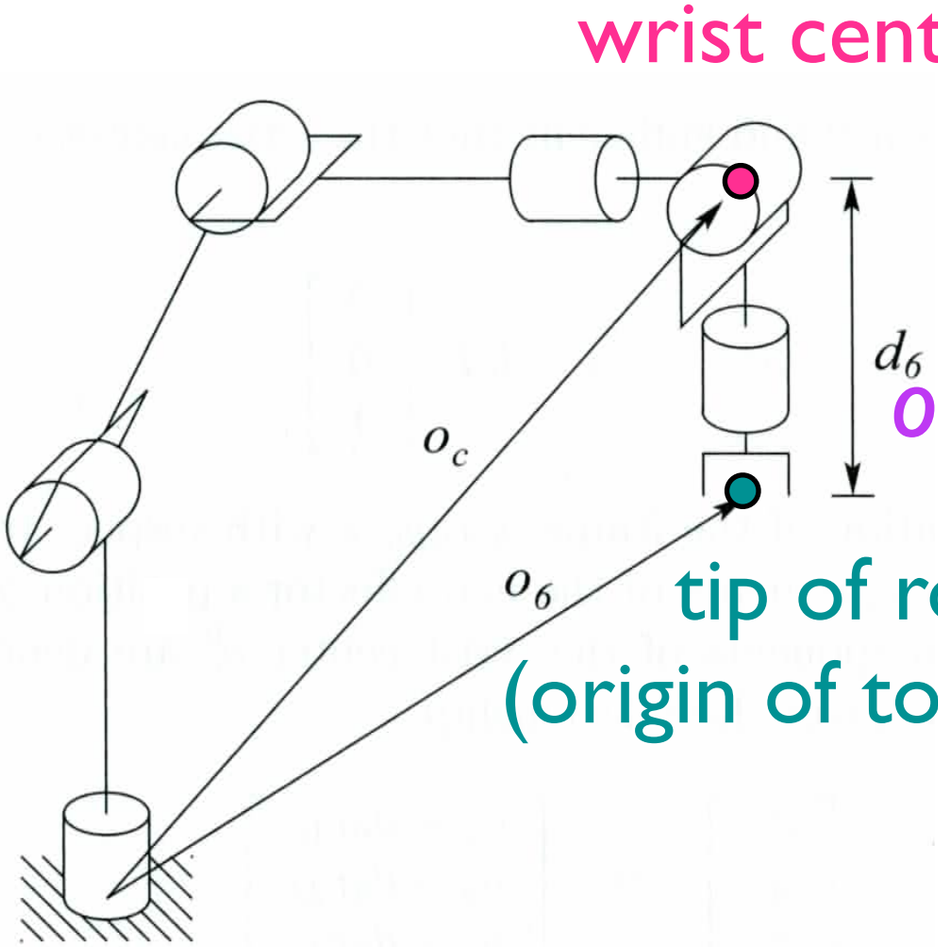
find $q_1, \ldots, q_n$ such that $\mathbf{T}_n^0(q_1, \ldots, q_n) = \mathbf{H}$ ugly!

The inverse kinematics problem may or may not have a solution.

Even if one exists, the solution may or may not be unique.

Trick: Exploit the kinematic structure of the manipulator.

If the robot has a spherical wrist, use **Kinematic Decoupling**.



wrist center

do orientation second!

tip of robot
(origin of tool frame)

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} o_x - d_6 r_{13} \\ o_y - d_6 r_{23} \\ o_z - d_6 r_{33} \end{bmatrix}$$

position

$$R = R_3^0 R_6^3$$

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R$$

orientation

Inverse of the orientation the arm causes up to third frame

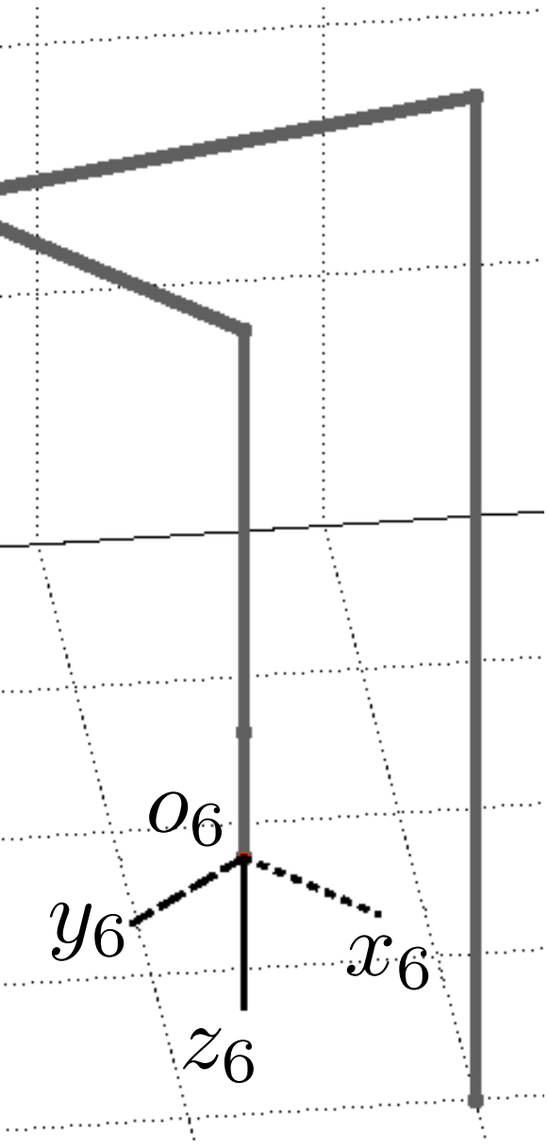
known or unknown?

$$R_6^3 = \left(R_3^0\right)^T R$$

What the wrist needs to do, unknown

What we want to accomplish, known

4. How can we calculate R03 from the available information? We have scara_robot_with_wrist_fk.m**, which returns points to plot along the robot, plus coordinates of x06, y06, and z06 line segments.**



- **Call** scara_robot_with_wrist_fk.m **with chosen values for theta1, theta2, and d3, and zero for theta4, theta5, and theta6. This causes frame 6 to be parallel to frame 3.**
- Calculate x, y, and z unit vectors for frame 6 in frame 0 by subtracting start of each line segment from the end and then normalizing.
- Assemble unit vectors into the matrix R03.

Editor - /Users/kuchenbe/Documents/teaching/meam 520/lectures/23 project2/robotics23ik_code/scara_robot_with_wr...

EDITOR | PUBLISH | VIEW

cle_kuchenbe_v1.m | scara_robot_with_wrist_circle_kuchenbe_v2.m | scara_robot_with_wrist_circle_kuchenbe_v3.m

```
109 -       oy = oc(2);
110 -       oz = oc(3);
111
112         % Calculate theta1, theta2, and d3 given the robot's parameters (a1 and
113         % a2) and the current desired position for its tip ([ox oy oz]').
114
115         % Calculate the cosine of theta2 using law of cosines.
116 -       c2 = (ox.^2 + oy.^2 - a1^2 - a2^2)/(2*a1*a2);
117
118         % Calculate the positive and negative solutions for theta2.
119 -       theta2_pos = atan2(sqrt(1-c2^2),c2); % Corrected solution from our derivation.
120 -       theta2_neg = atan2(-sqrt(1-c2^2),c2); % Corrected solution from our derivation.
121
122         % Arbitrarily choose the positive solution.
123 -       theta2 = theta2_pos;
124
125         % Uncomment this line to choose the negative solution instead.
126         %theta2 = theta2_neg;
127
128         % Calculate theta1 using a pair of inverse tangents.  Note that MATLAB
129         % atan2 takes the numerator and then the denominator.
130 -       theta1 = atan2(oy,ox) - atan2(a2*sin(theta2),a1+a2*cos(theta2));
131
132         % Calculate d3.
133 -       d3 = -oz; % Corrected solution from our derivation.
134
135         % Use FK to get orientation of frame 3 relative to frame 0 by keeping
136         % wrist angles all zero.  Must normalize vectors and assemble into R03.
137 -       [~, x06, y06, z06] = scara_robot_with_wrist_fk(a1, a2, d6, theta1, theta2, ...
138             d3, 0, 0, 0);
139 -       ux = (x06(1:3,2) - x06(1:3,1)); ux = ux / norm(ux);
140 -       uy = (y06(1:3,2) - y06(1:3,1)); uy = uy / norm(uy);
141 -       uz = (z06(1:3,2) - z06(1:3,1)); uz = uz / norm(uz);
142 -       R03 = [ux uy uz];
143
144         % Calculate R36.
145 -       R36 = R03'*R;
146
```

script | Ln 145 Col 18

Calculating R03

Now we have R36. What next?

Do inverse Euler angles following the book's method.

To find a solution for this problem we break it down into two cases. First, suppose that not both of r_{13}, r_{23} are zero. Then from Equation (2.26) we deduce that $s_\theta \neq 0$, and hence that not both of r_{31}, r_{32} are zero. If not both r_{13} and r_{23} are zero, then $r_{33} \neq \pm 1$, and we have $c_\theta = r_{33}$, $s_\theta = \pm\sqrt{1 - r_{33}^2}$ so

$$\theta = \text{Atan2}\left(r_{33}, \sqrt{1 - r_{33}^2}\right) \tag{2.28}$$

or

$$\theta = \text{Atan2}\left(r_{33}, -\sqrt{1 - r_{33}^2}\right) \tag{2.29}$$

where the function Atan2 is the **two-argument arctangent function** defined in Appendix A.

If we choose the value for θ given by Equation (2.28), then $s_\theta > 0$, and

$$\phi = \text{Atan2}(r_{13}, r_{23}) \tag{2.30}$$
$$\psi = \text{Atan2}(-r_{31}, r_{32}) \tag{2.31}$$

If we choose the value for θ given by Equation (2.29), then $s_\theta < 0$, and

$$\phi = \text{Atan2}(-r_{13}, -r_{23}) \tag{2.32}$$
$$\psi = \text{Atan2}(r_{31}, -r_{32}) \tag{2.33}$$

Thus, there are two solutions depending on the sign chosen for θ.

If $r_{13} = r_{23} = 0$, then the fact that R is orthogonal implies that $r_{33} = \pm 1$, and that $r_{31} = r_{32} = 0$. Thus, R has the form

$$R = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & \pm 1 \end{bmatrix}. \tag{2.34}$$

If $r_{33} = 1$, then $c_\theta = 1$ and $s_\theta = 0$, so that $\theta = 0$. In this case, Equation (2.26) becomes

$$\begin{bmatrix} c_\phi c_\psi - s_\phi s_\psi & -c_\phi s_\psi - s_\phi c_\psi & 0 \\ s_\phi c_\psi + c_\phi s_\psi & -s_\phi s_\psi + c_\phi c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c_{\phi+\psi} & -s_{\phi+\psi} & 0 \\ s_{\phi+\psi} & c_{\phi+\psi} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Thus, the sum $\phi + \psi$ can be determined as

$$\phi + \psi = \text{Atan2}(r_{11}, r_{21}) = \text{Atan2}(r_{11}, -r_{12}) \tag{2.35}$$

Since only the sum $\phi + \psi$ can be determined in this case, there are infinitely many solutions. In this case, we may take $\phi = 0$ by convention. If $r_{33} = -1$, then $c_\theta = -1$ and $s_\theta = 0$, so that $\theta = \pi$. In this case Equation (2.26) becomes

$$\begin{bmatrix} -c_{\phi-\psi} & -s_{\phi-\psi} & 0 \\ s_{\phi-\psi} & c_{\phi-\psi} & 0 \\ 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{2.36}$$

The solution is thus

$$\phi - \psi = \text{Atan2}(-r_{11}, -r_{12}) \tag{2.37}$$

As before there are infinitely many solutions.

Special case for cos (theta) = +1, –1

Remember that SHV lists atan2 arguments as atan2(denominator, numerator), which is opposite MATLAB.

Rotation Matrix to Euler Angles?

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Use atan2 to determine ϕ for both θ options

$$= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}$$

Use atan2 to determine ψ for both θ options

Two solutions for θ because sign of s_θ is not known.

$$\phi = ? \quad \theta = ? \quad \psi = ?$$

Editor - /Users/kuchenbe/Documents/teaching/meam 520/lectures/23 project2/robotics23ik_code/scara_robot_with_wr...

EDITOR | PUBLISH | VIEW

cle_kuchenbe_v1.m | scara_robot_with_wrist_circle_kuchenbe_v2.m | scara_robot_with_wrist_circle_kuchenbe_v3.m

```
144         % Calculate R36.
145 -       R36 = R03'*R;
146
147         % Do inverse Euler angle kinematics - phi around Z, theta around new Y,
148         % and psi around new Z.  Calculate two solutions for theta.
149 -       costheta = R36(3,3);
150 -       sintheta_pos =
151 -       sintheta_neg =
152 -
153 -
154
155
156 -
157
158         % Check sine of theta to determine the way to calculate the other two angles.
159 -       if (sin(theta) > 0)
160
161             % Calculate phi.
162 -           phi =
163
164             % Calculate psi.
165 -           psi =
166
167 -       else
168
169             % Calculate phi.
170 -           phi =
171
172             % Calculate psi.
173 -           psi =
174
175 -       end
176
177         % Convert from Euler angles to joint angles.
178 -       theta4 = phi;
179 -       theta5 = theta;
180 -       theta6 = psi;
181
```

Code Hidden

Run v3 code
(will not be posted
because it's too similar
to project 2)

script | Ln 149 Col 25

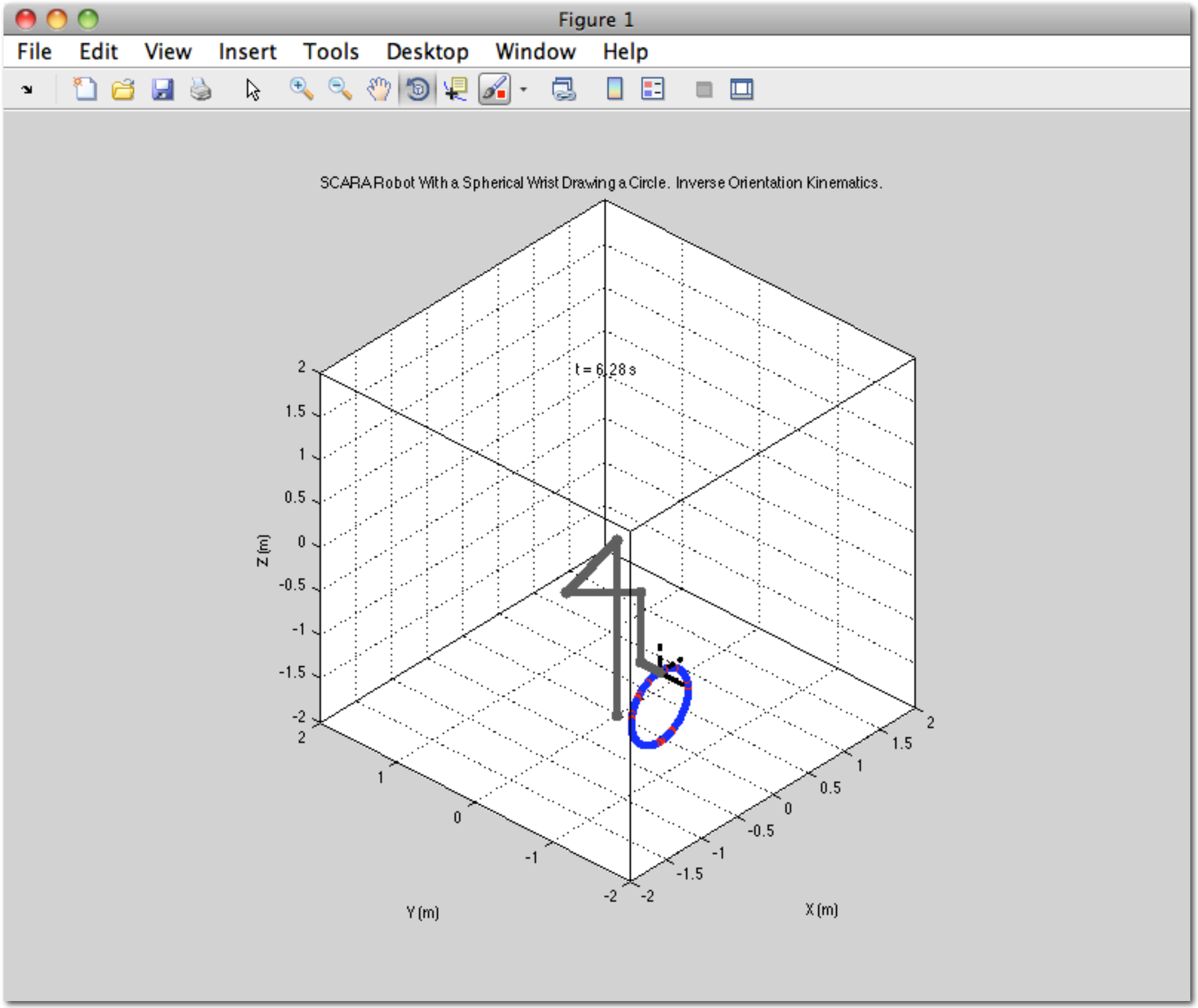
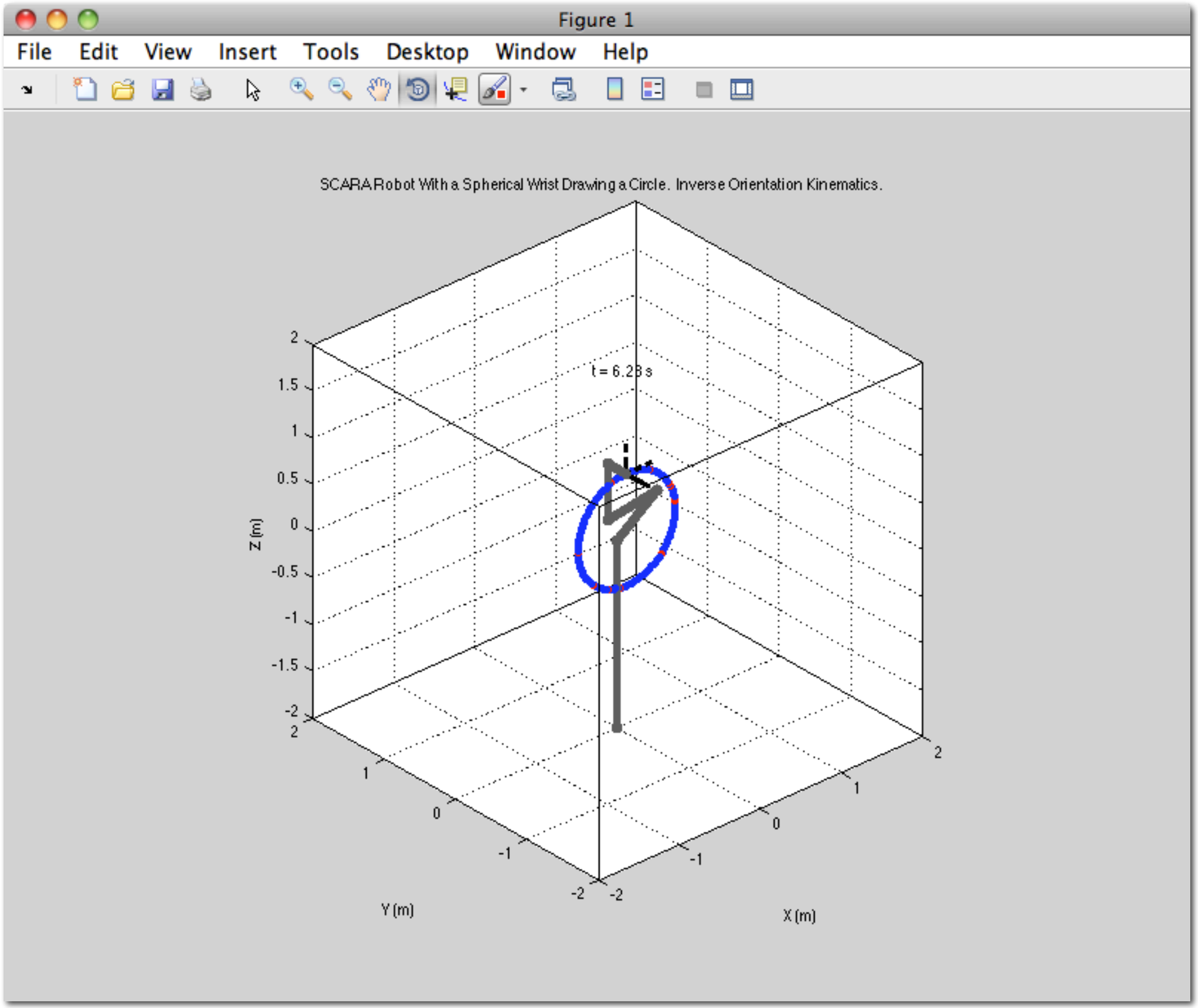


Figure 1
File Edit View Insert Tools Desktop Window Help
SCARA Robot With a Spherical Wrist Drawing a Circle. Inverse Orientation Kinematics.
t = 6.28 s
Z (m)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
Y (m)
2
1
0
-1
-2
X (m)
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Will this code work for other circles? Yes.



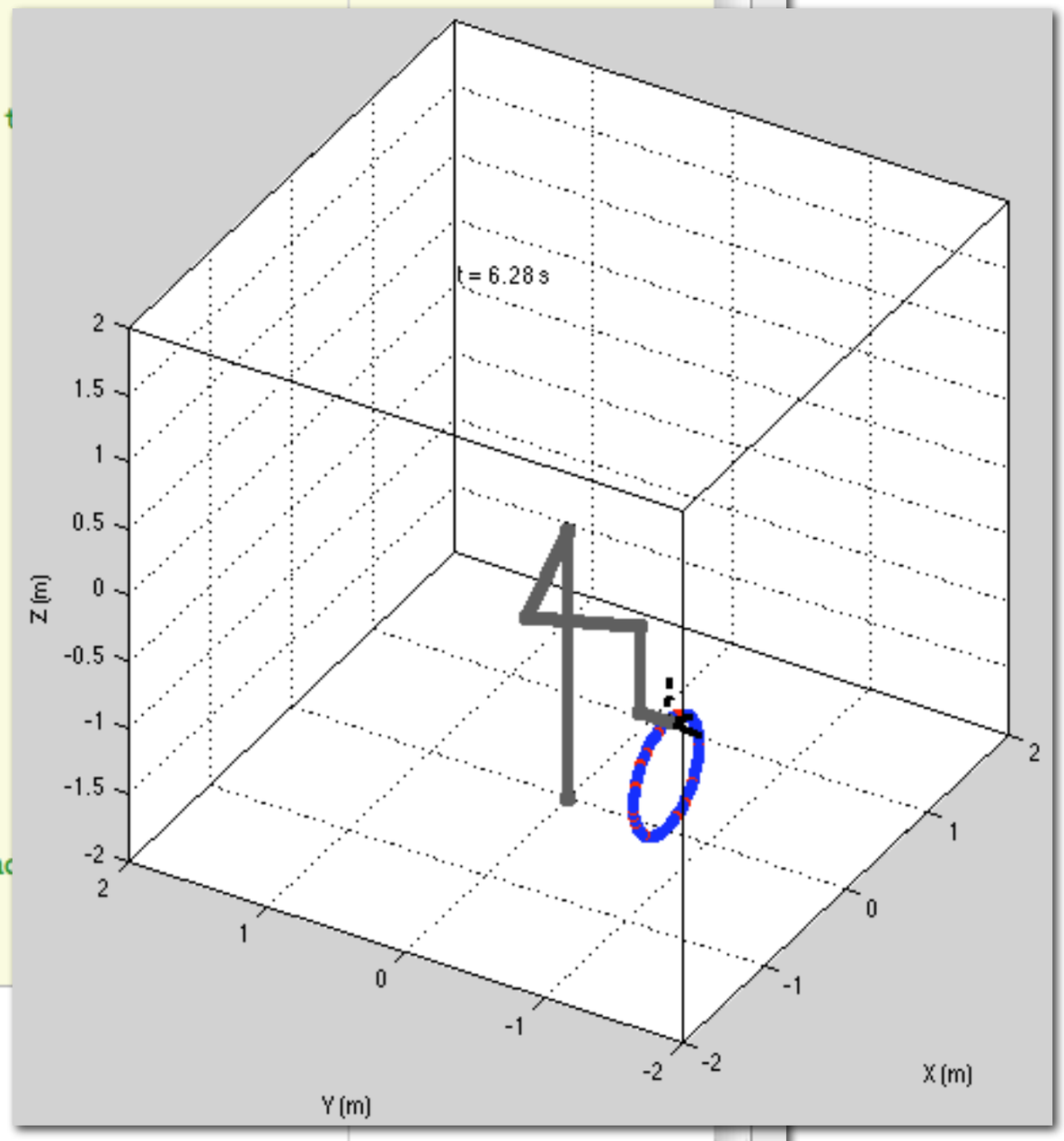
Will this code work for other rotation matrices? Yes.

Editor – /Users/kuchenbe/Documents/teaching/meam 520/lectures/23 project2/robotics23ik_code/scara_robot_with_wr...

EDITOR | PUBLISH | VIEW

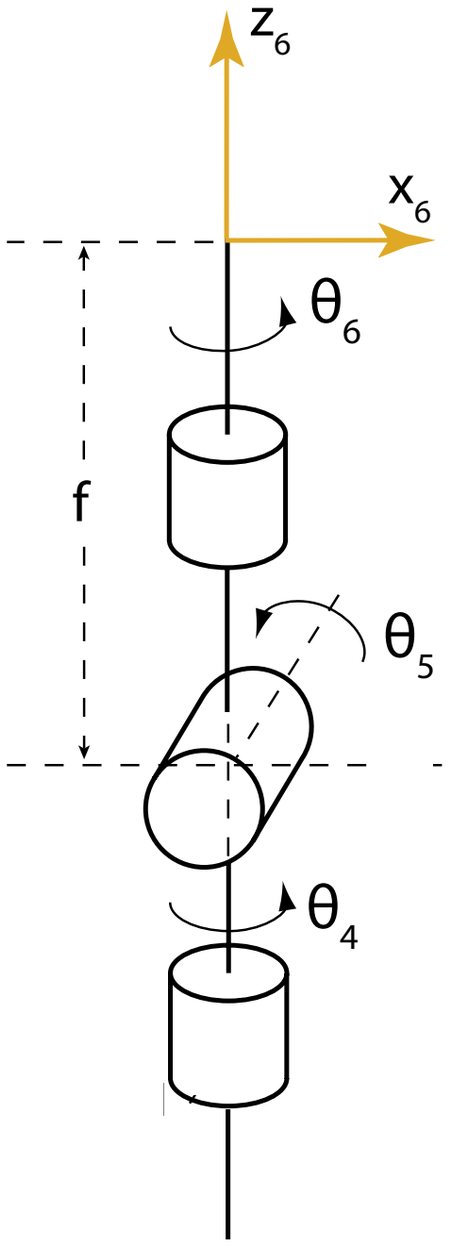
cle_kuchenbe_v1.m | scara_robot_with_wrist_circle_kuchenbe_v2.m | scara_robot_with_wrist_circle_kuchenbe_v3.m

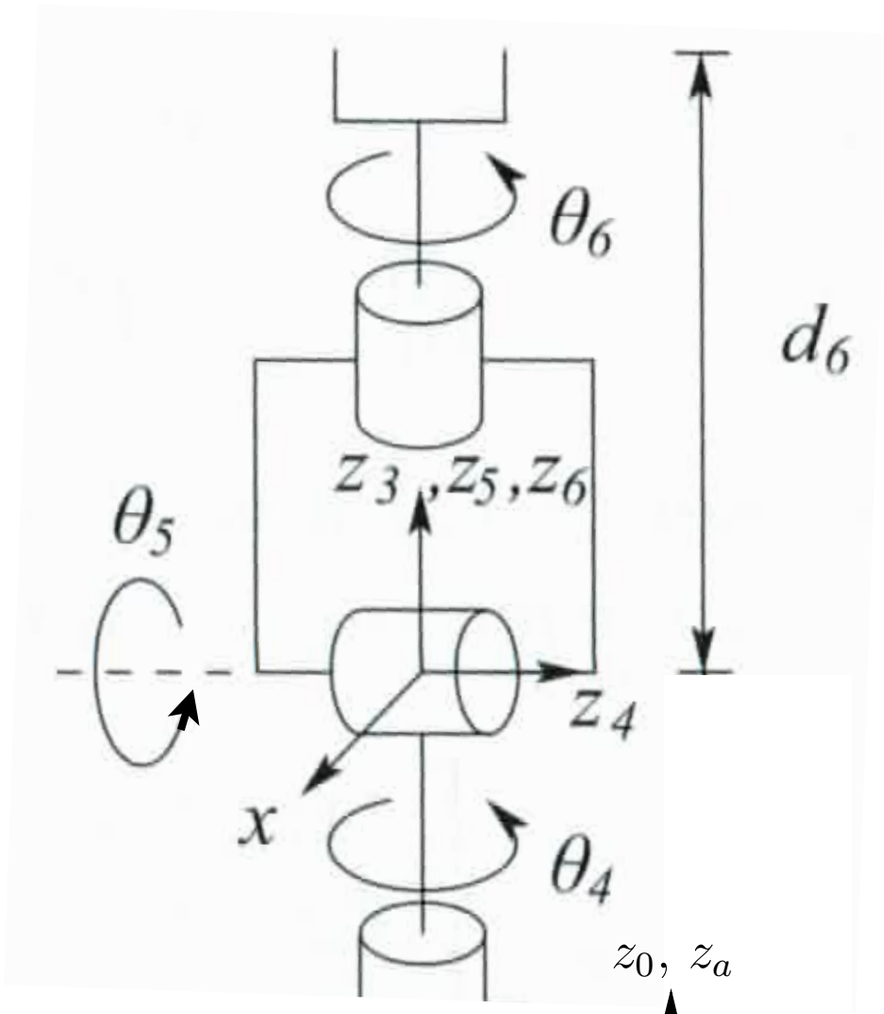
```
48
49      % Define the x and z coordinates for the center of the circle.
50 -    x_center = -.5; % meters
51 -    z_center = -1.2; % meters
52
53      % Set the desired x, y, and z positions over time given t
54 -    ox_history = x_center + radius * sin(t);
55 -    oy_history = y_offset * ones(size(t));
56 -    oz_history = z_center + radius * cos(t);
57
58      % Set the end-effector orientation that we want.
59 -    R = [1  0  0; ...
60           0  0 -1; ...
61           0  1  0];
62
63      % Add a rotation around the z-axis of frame zero.
64 -    th = 45;
65 -    Rzth = [cosd(th) -sind(th) 0; ...
66              sind(th)  cosd(th) 0; ...
67                     0         0 1];
68
69      % Add a rotation about the x axis of frame zero.
70 -    al = 60;
71 -    Rxal = [1 0 0; ...
72              0 cosd(al) -sind(al); ...
73              0 sind(al)  cosd(al)];
74
75      % Pre-multiply R by Rxal and Rzth to get rotations around
76 -    R = Rxal*Rzth*R
77
78
79      %% SIMULATION
80
81      % Notify the user that we're starting the animation.
82 -    disp('Starting the animation.')
83
84      % Show a message to explain how to cancel the simulation and graphing.
85 -    disp('Click in this window and press control-c to stop the code.')
```



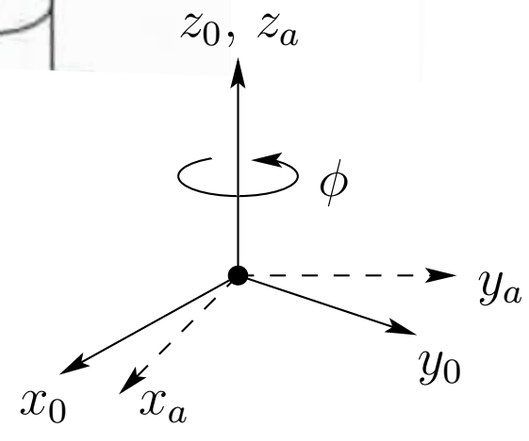
script | Ln 70 Col 8

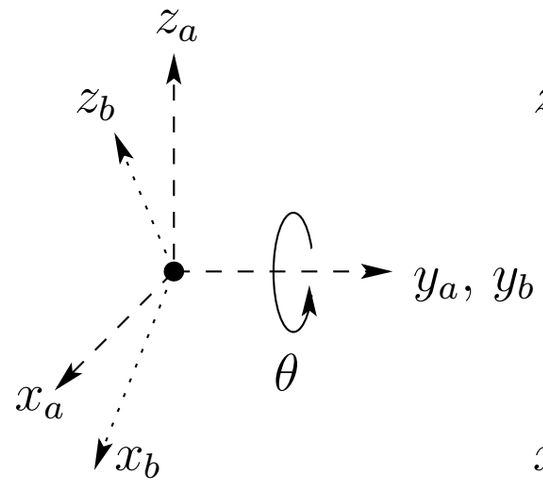
5. How does our model of the PUMA's spherical wrist (below left) differ from SHV's spherical wrist (below right) when compared to ZYZ Euler angles (bottom)?

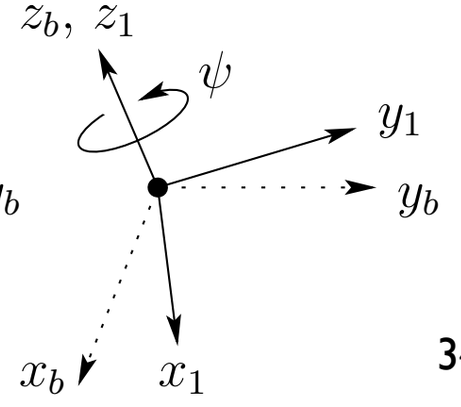




Work with your team to figure this question out for Project 2.







What questions do you have ?

Project 2

PUMA Light Painting

MEAM 520 Light Paintings – Fall 2012

https://plus.google.com/u/0/photos/108741623268221783763/albums/582013534809

Google

Google+ Search for your photos +Katherine Share

Photos Highlights Albums More Hangouts

MEAM 520 Light Paintings - Fall 2012

41 photos · Shared privately · Oct 27 - Dec 18 · Select

Highlights All photos Share Tag people Organize Add photos

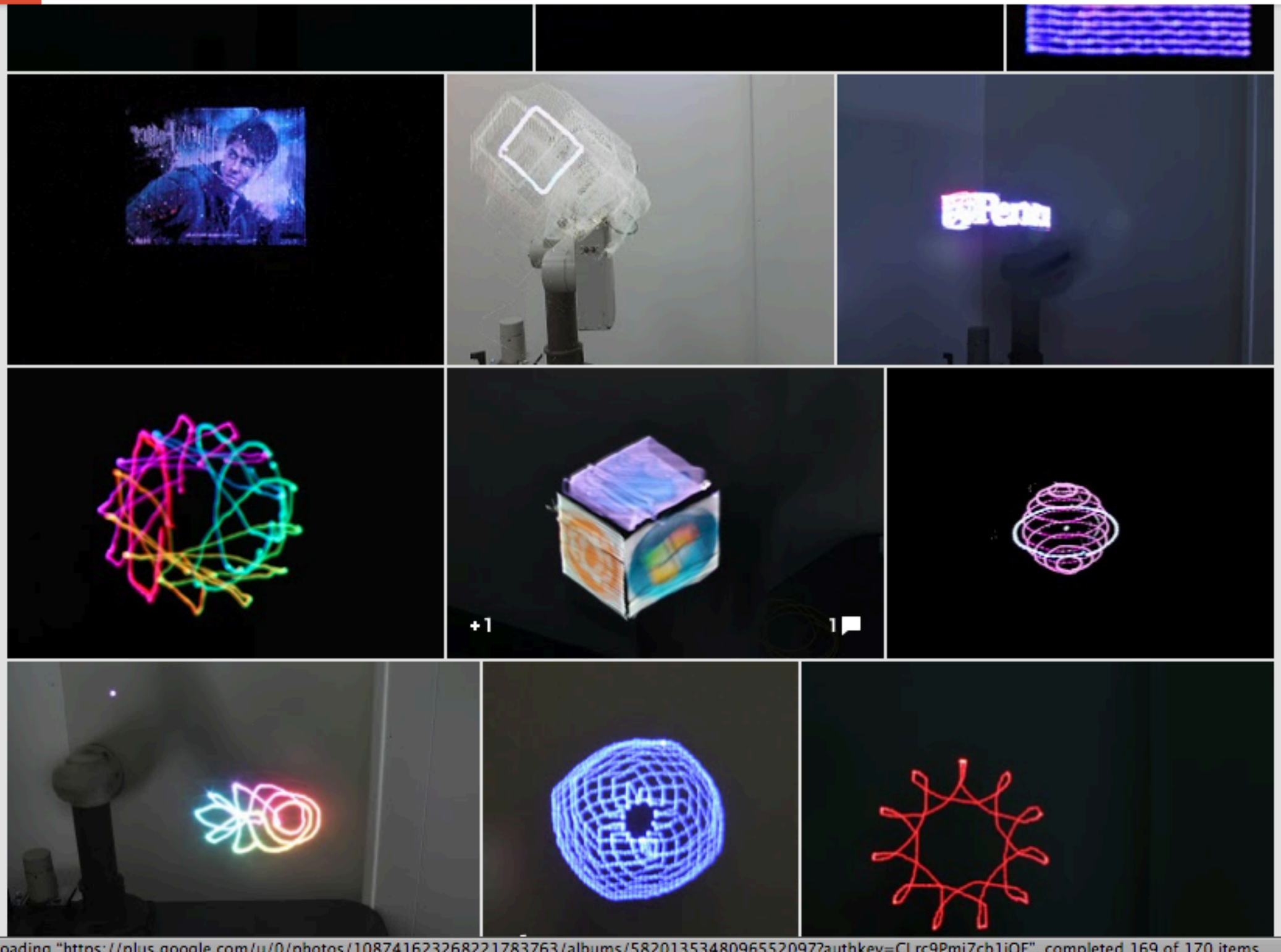
Loading "https://plus.google.com/u/0/photos/108741623268221783763/albums/5820135348096552097?authkey=CLrc9Pmj7cb1iQE", completed 169 of 170 items

MEAM 520 Light Paintings - Fall 2012

https://plus.google.com/u/0/photos/108741623268221783763/albums/582013534809

Google

Highlights Albums More



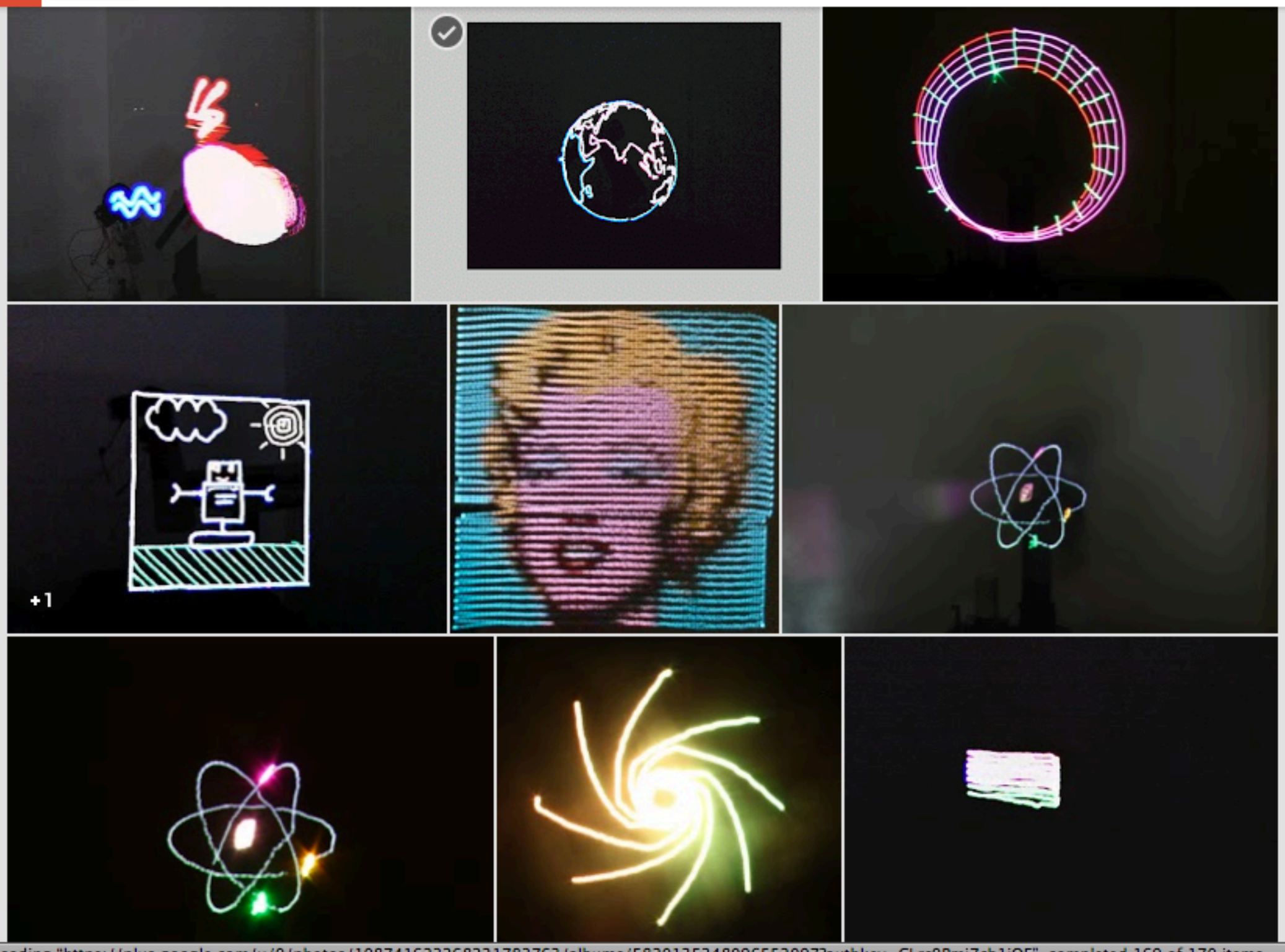
Loading "https://plus.google.com/u/0/photos/108741623268221783763/albums/5820135348096552097?authkey=CLrc9Pmj7cb1iQE", completed 169 of 170 items

MEAM 520 Light Paintings - Fall 2012

https://plus.google.com/u/0/photos/108741623268221783763/albums/582013534805

Google

Highlights Albums More



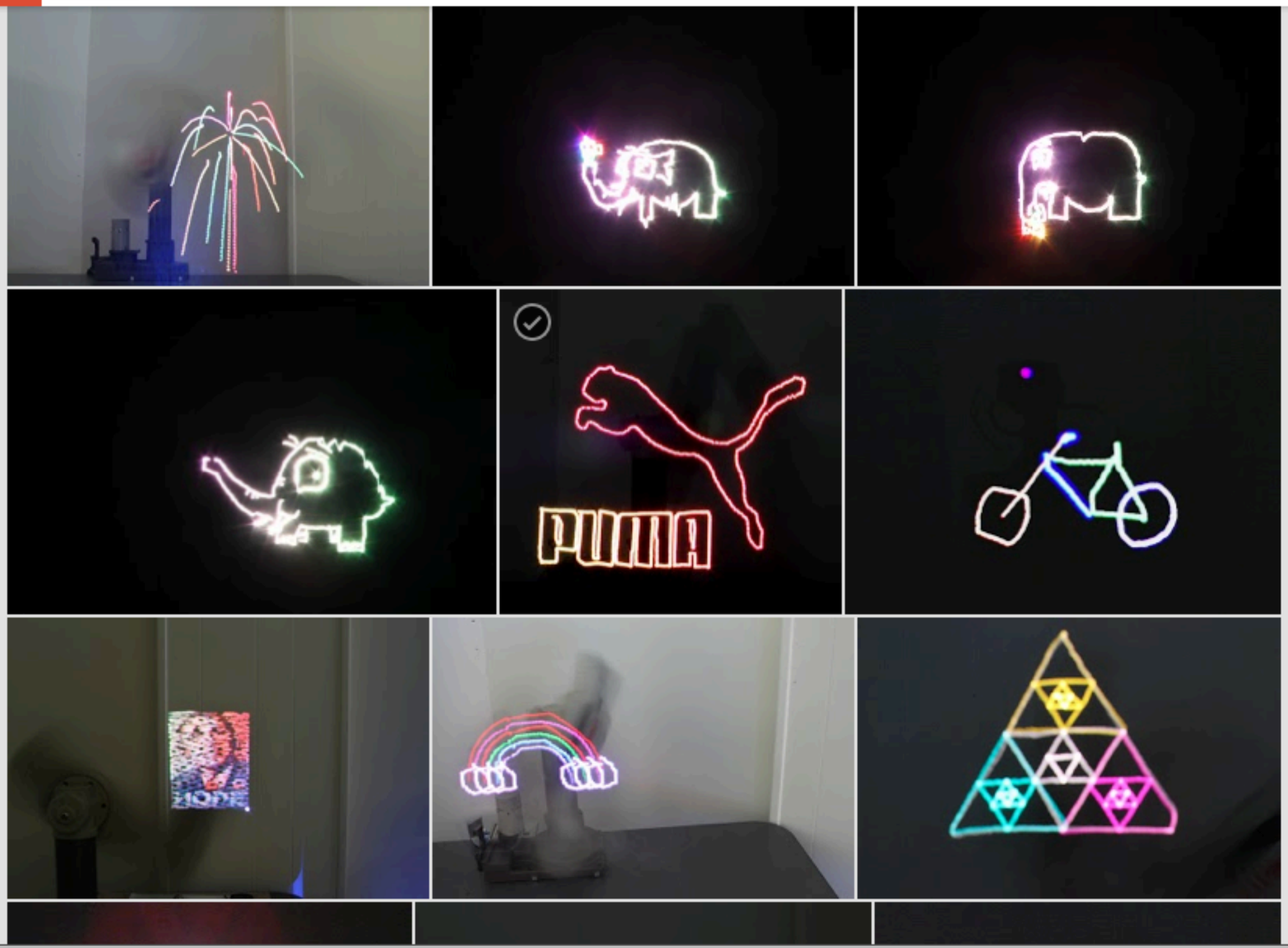
Loading "https://plus.google.com/u/0/photos/108741623268221783763/albums/5820135348096552097?authkey=CLrc9Pmj7cb1iQE", completed 169 of 170 items

MEAM 520 Light Paintings – Fall 2012

https://plus.google.com/u/0/photos/108741623268221783763/albums/582013534809

Google

Highlights Albums More



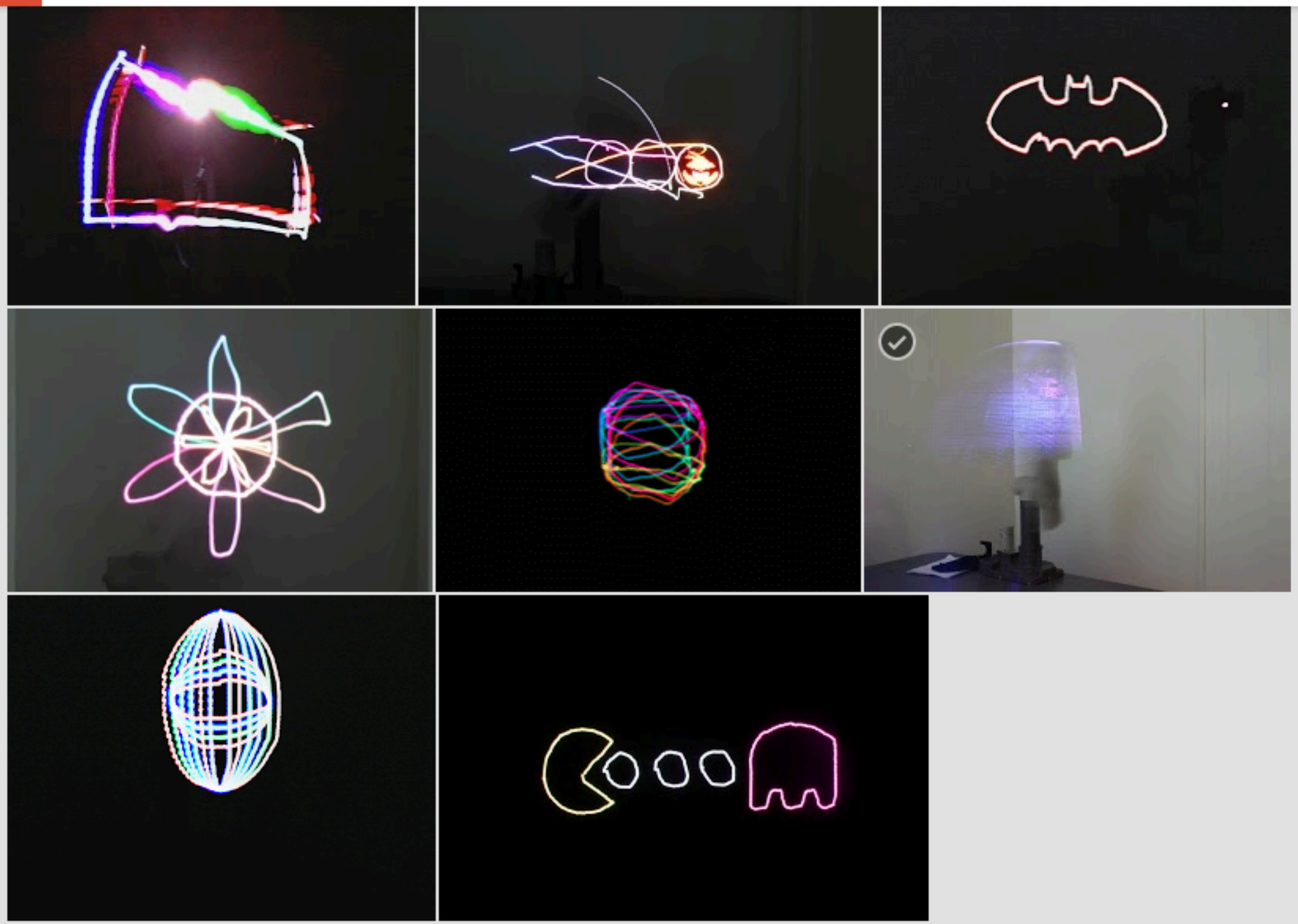
Loading "https://plus.google.com/u/0/photos/108741623268221783763/albums/5820135348096552097?authkey=CLrc9Pmj7cb1iQE", completed 169 of 170 items

MEAM 520 Light Paintings - Fall 2012

https://plus.google.com/u/0/photos/108741623268221783763/albums/582013534805

Google

Highlights Albums More



Loading "https://plus.google.com/u/0/photos/108741623268221783763/albums/5820135348096552097?authkey=CLrc9Pmj7cb1iQE", completed 169 of 170 items

Drawing precise, arbitrary shapes with a robot requires you to solve the robot's full inverse kinematics (position and orientation), so that is the first component of this project.

Editor - /Users/kuchenbe/Documents/teaching/meam 520/projects/2 puma paint/ik/team200_plot_puma.m

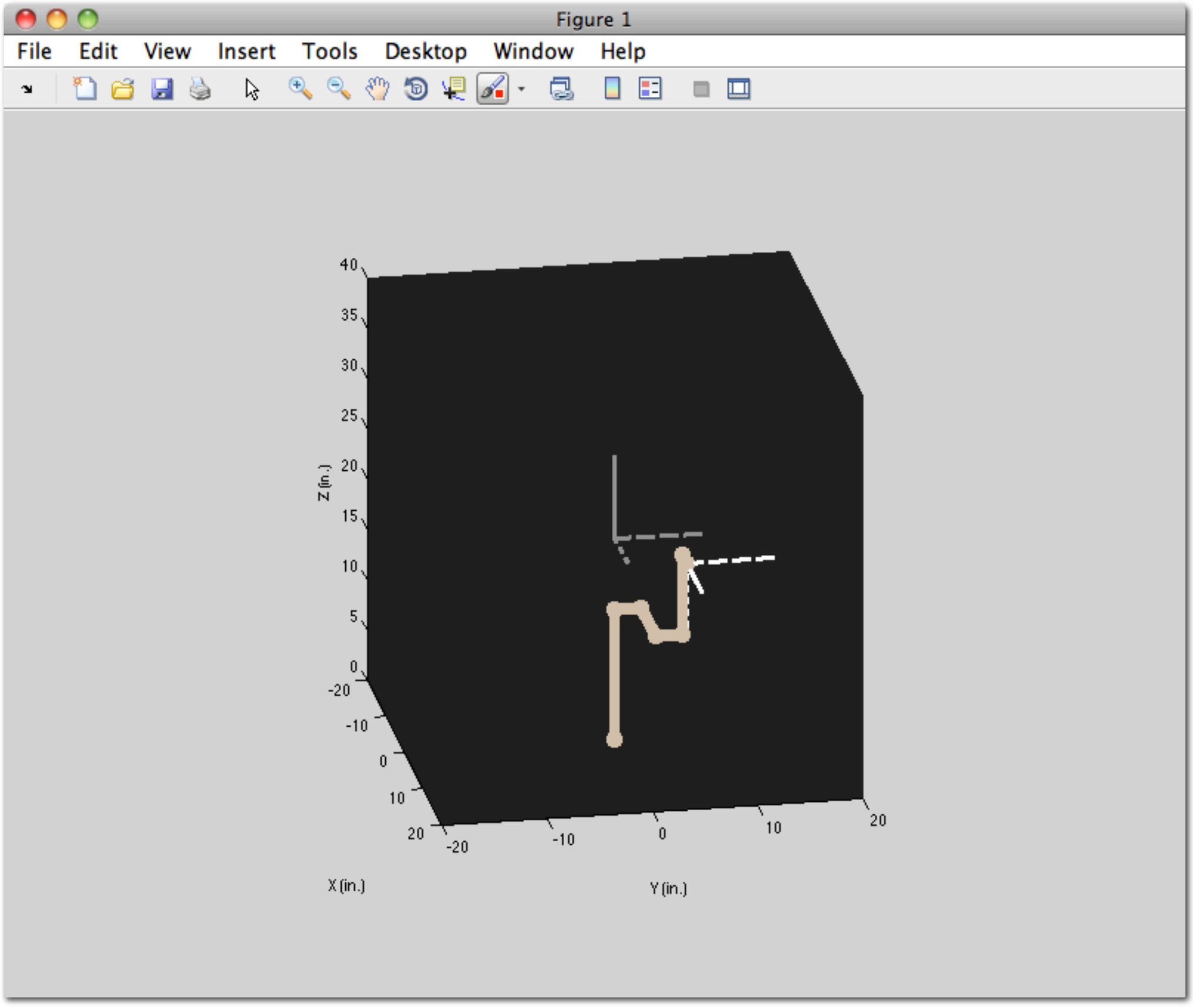
EDITOR | PUBLISH | VIEW

team200_plot_puma.m | team200_puma_ik.m | team200_testing.m

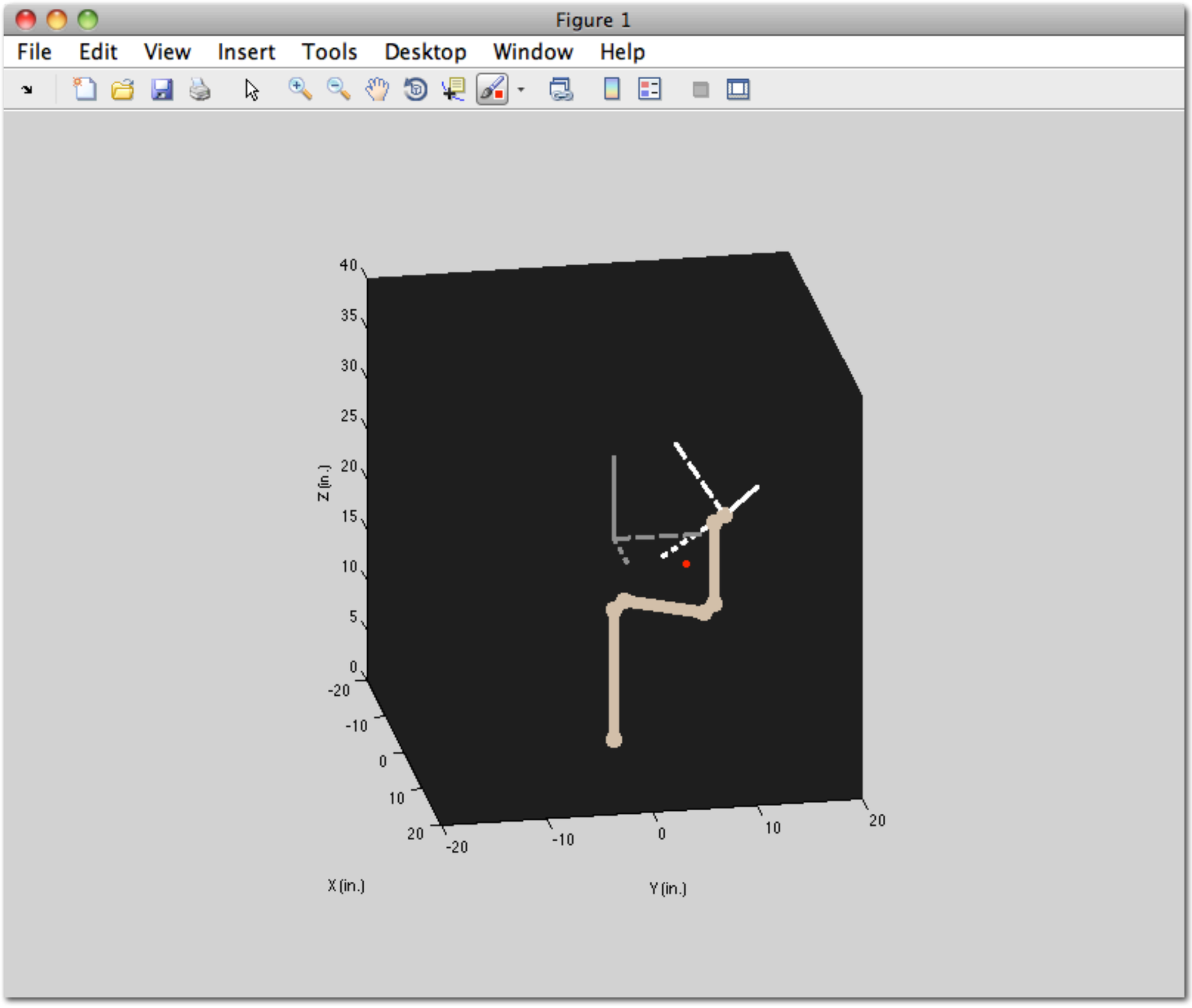
```
1  function hout = team200_plot_puma(x, y, z, phi, theta, psi,theta1, theta2, theta3, theta4, theta5
2  %
3  % This Matlab function is part of the starter code for the inverse
4  % kinematics part of Project 2 in MEAM 520 at the University of Pennsylvania.
5  % It plots many things, as explained below.
6  %
7  % A gray coordinate frame at the position and orientation specified by the
8  % first six inputs.  x, y, and z specify the desired position of the origin
9  % of the sixth frame with respect to frame zero.  phi, theta, and psi are
10 % ZYZ Euler angles that specify the desired orientation of the sixth frame
11 % with respect to frame zero.
12 %
13 % The PUMA robot in the configuration specified by the six joint
14 % angle inputs, which are the seventh through twelfth arguments, named
15 % theta1 through theta 6.
16 %
17 % A point of light in the specified RGB color at the end-effector tip.
18 % Each color value should range from 0 to 1.
19 %
20 % This function returns a vector of handles to the items it has plotted so
21 % that this function can be used to update a plot rather than replace it.
22 %
23 % The final input argument, which is optional, is a vector of handles from
24 % a previous call to this function.  When provided, this function updates
25 % the plot rather than replacing it.
26
27
28 %% ROBOT PARAMETERS
29
30 % This problem is about the PUMA 260 robot, a 6-DOF manipulator.
31
32 % Define the robot's measurements.  These correspond are constant.
33 a = 13.0; % inches
34 b =  2.5; % inches
35 c =  8.0; % inches
36 d =  2.5; % inches
37 e =  8.0; % inches
38 f =  2.5; % inches
39
```

team200_plot_puma | Ln 8 | Col 35

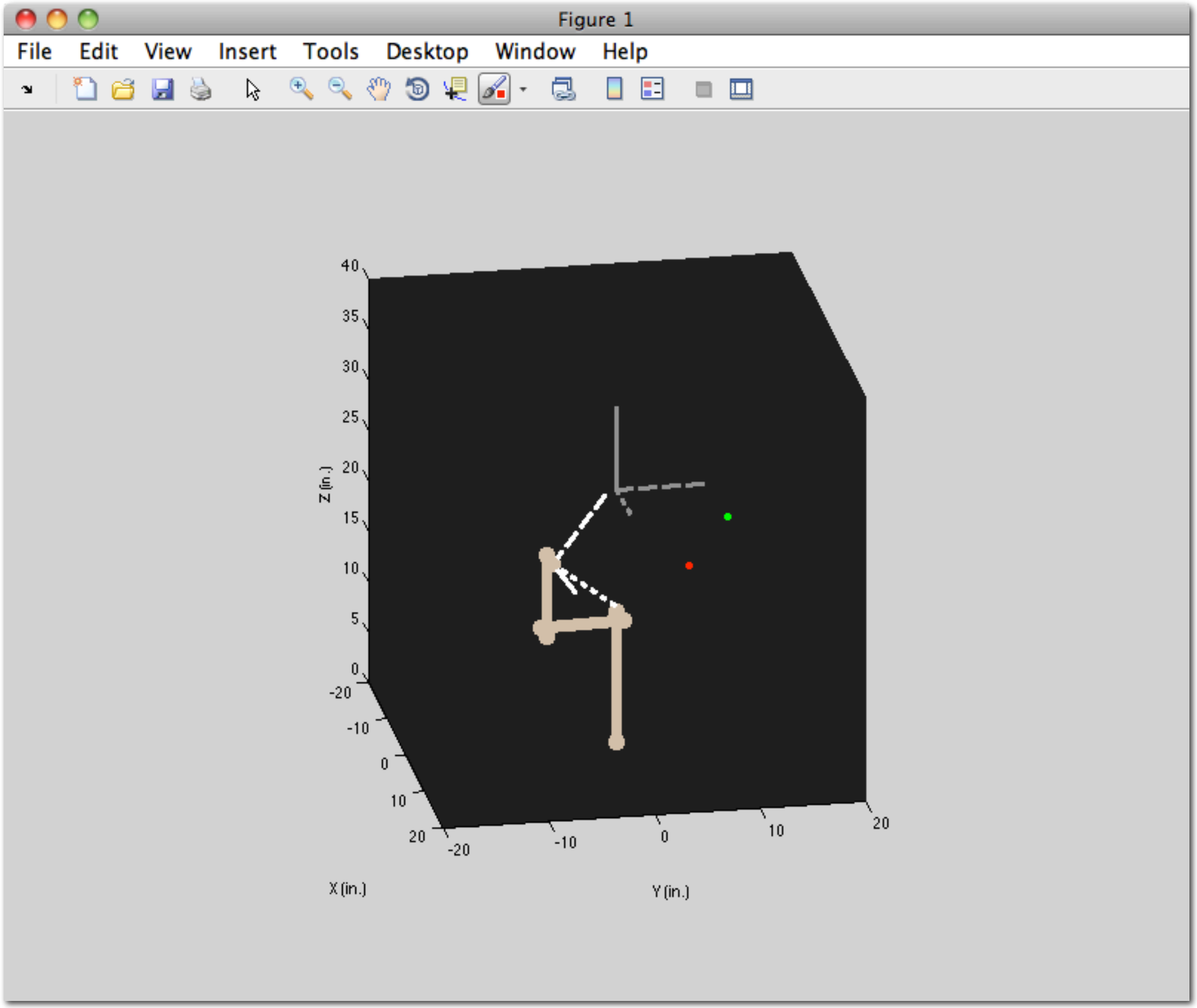
```
>> h = team200_plot_puma(0,0,20,0,0,0,0,0,0,0,0,-pi/2,0,1,0,0);
```



```
>> team200_plot_puma(0,0,20,0,0,0,1,0,0,pi/2,-pi/2,.9,0,1,0,h);
```



```
>> team200_plot_puma(0,0,25,0,0,0,-1.5,0,0,pi/2,-pi/2,.9,0,0,1,h);
```



Editor - /Users/kuchenbe/Documents/teaching/meam 520/projects/2 puma paint/ik/team200_puma_ik.m

EDITOR | PUBLISH | VIEW

team200_plot_puma.m | team200_puma_ik.m | team200_testing.m

```
1  function thetas = team200_puma_ik(x, y, z, phi, theta, psi)
2  %% team200_puma_ik.m
3  %
4  % Calculates the full inverse kinematics for the PUMA 260.
5  %
6  % This Matlab file provides the starter code for the PUMA 260 inverse
7  % kinematics function of project 2 in MEAM 520 at the University of
8  % Pennsylvania.  The original was written by Professor Katherine J.
9  % Kuchenbecker. Students will work in teams modify this code to create
10 % their own script. Post questions on the class's Piazza forum.
11 %
12 % The first three input arguments (x, y, z) are the desired coordinates of
13 % the PUMA's end-effector tip in inches, specified in the base frame.  The
14 % origin of the base frame is where the first joint axis (waist) intersects
15 % the table. The z0 axis points up, and the x0 axis points out away from
16 % the robot, perpendicular to the front edge of the table.  These arguments
17 % are mandatory.
18 %
19 %     x: x-coordinate of the origin of frame 6 in frame 0, in inches
20 %     y: y-coordinate of the origin of frame 6 in frame 0, in inches
21 %     z: z-coordinate of the origin of frame 6 in frame 0, in inches
22 %
23 % The fourth through sixth input arguments (phi, theta, psi) represent the
24 % desired orientation of the PUMA's end-effector in the base frame using
25 % ZYZ Euler angles in radians.  These arguments are mandatory.
26 %
27 %     phi: first ZYZ Euler angle to represent orientation of frame 6 in frame 0, in radians
28 %     theta: second ZYZ Euler angle to represent orientation of frame 6 in frame 0, in radians
29 %     psi: third ZYZ Euler angle to represent orientation of frame 6 in frame 0, in radians
30 %
31 % The output (thetas) is a matrix that contains the joint angles needed to
32 % place the PUMA's end-effector at the desired position and in the desired
33 % orientation. The first row is theta1, the second row is theta2, etc., so
34 % it has six rows.  The number of columns is the number of inverse
35 % kinematics solutions that were found; each column should contains a set
36 % of joint angles that place the robot's end-effector in the desired pose.
37 % These joint angles are specified in radians according to the
38 % order, zeroing, and sign conventions described in the documentation.  If
39 % this function cannot find a solution to the inverse kinematics problem,
40 % it will pass back NaN (not a number) for all of the thetas.
```

team200_puma_ik | Ln 2 Col 21

Editor - /Users/kuchenbe/Documents/teaching/meam 520/projects/2 puma paint/ik/team200_puma_ik.m

EDITOR | PUBLISH | VIEW

team200_plot_puma.m | team200_puma_ik.m | team200_testing.m

```
53          % This the correct way to call this function, so we don't need to do
54          % anything special.
55 -      elseif (nargin > 6)
56 -          error('Too many input arguments.  You need six.')
57 -      end
58
59
60          %% CALCULATE INVERSE KINEMATICS SOLUTION(S)
61
62          % For now, just set each theta to NaN (not a number).  This is what you
63          % should output if there is no solution to the inverse kinematics problem
64          % for the position and orientation that were passed in.  For example, this
65          % would be the correct output if the desired position for the end-effector
66          % was outside the robot's reachable workspace.  We use this sentinel value
67          % of NaN to be sure that the code calling this function can tell that
68          % something is wrong and shut down the PUMA.
69 -        th1 = NaN;
70 -        th2 = NaN;
71 -        th3 = NaN;
72 -        th4 = NaN;
73 -        th5 = NaN;
74 -        th6 = NaN;
75
76          % You should update this section of the code with your IK solution.
77          % Please comment your code to explain what you are doing at each step.
78          % Feel free to create additional functions as needed - please name them all
79          % to start with team2XX_, where 2XX is your team number.  For example, it
80          % probably makes sense to handle inverse position kinematics and inverse
81          % orientation kinematics separately.
82
83
84          %% FORMAT OUTPUT
85
86          % Put all of the thetas into a column vector to return.
87 -        thetas = [th1 th2 th3 th4 th5 th6]';
88
89          % By the very end, each column of thetas should hold a set of joint angles
90          % in radians that will put the PUMA's end-effector in the desired
91          % configuration.  If the desired configuration is not reachable, set all of
92          % the joint angles to NaN.
```

team200_puma_ik | Ln 22 Col 2

Editor - /Users/kuchenbe/Documents/teaching/meam 520/projects/2 puma paint/ik/team200_testing.m

EDITOR | PUBLISH | VIEW

team200_plot_puma.m | team200_puma_ik.m | team200_testing.m

```
%% team200_testing.m
%
% This Matlab script is part of the starter code for the inverse
% kinematics part of Project 2 in MEAM 520 at the University of Pennsylvania.
% It tests the team's inverse kinematics code for various configurations.

%% SETUP

% Clear all variables from the workspace.
clear all

% Clear the console, so you can more easily find any errors that may occur.
clc

% Set whether to animate the robot's movement and how much to slow it down.
pause on;  % Set this to off if you don't want to watch the animation.
GraphingTimeDelay = 0.001; % The length of time that Matlab should pause between positions when g


%% CHOOSE INPUT PARAMETERS

testType = 0;

switch(testType)
    case 0

        % Draw a vertical circle parallel to the x-z plane.

        % Define the radius of the circle.
        radius = 4; % inches

        % Define the y-value for the plane that contains the circle.
        y_offset = 7; % inches

        % Define the x and z coordinates for the center of the circle.
        x_center = 4; % inches
        z_center = 15; % inches

        % Create a time vector.
```

script | Ln 2 | Col 2

Final starter code for this part of Project 2 will be posted later today or tomorrow.

You can start working on solving the PUMA's inverse position kinematics and inverse orientation kinematics now.

We strongly encourage you to draw sketches and think about the problem before starting to program.

Work with your teammates.

Your first draft of team2XX_puma_ik.m is due by midnight on Tuesday, November 26.

Try to get as much working as you can. At a minimum, you must have some reasonable calculations.

We will also ask you what you think you will draw and where in space you will draw it.