ME/CS 133a Final Project

VISUALIZING SERIAL CHAIN ROBOTS USING DENAVIT-HARTENBERG PARAMETERS IN A MATLAB GUI

December 8, 2017

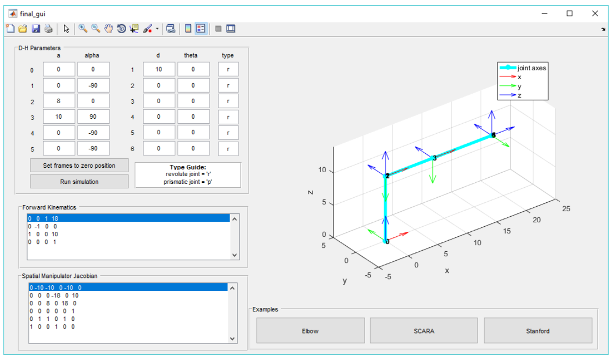
Maegan Tucker
California Institute of Technology
mtucker@caltech.edu

1 Introduction

Denavit-Hartenberg parameters are commonly used to define robotic mechanisms. The parameters follow a strict set of rules that specify the location and orientation of each link frame for a rigid robotic device. Each frame corresponds to one of the joint axes. Once the Denavit-Hartenberg (D-H) parameters are defined, the transformation between each frame can be calculated using the D-H parameters. The forward kinematics of the mechanism can then be calculated using all of the frame transformations. The Guided User Interface (GUI) outlined in this report visualizes a robot as defined by its D-H parameters and calculates the forward kinematics and spatial manipulator Jacobian of the robot given the final joint variables.

2 Project Summary

The main outcome of this project is a kinematics and visualization software. The inputs of the software are the Denavit-Hartenberg parameters as defined by the user. Then, using these Denavit-Hartenberg parameters, the software creates a visual representation of the specified robot and calculates the forward kinematics and spatial manipulator Jacobian. A figure of the initial GUI screen can be seen in Figure 1.



D-H Parameters

	a	alpha		d	theta	type
0	0	0	1	10	0	r
1	0	-90	2	0	0	r
2	8	0	3	0	0	r
3	10	90	4	0	0	r
4	0	-90	5	0	0	r
5	0	-90	6	0	0	r

Type Guide:
revolute joint = 'r'
prismatic joint = 'p'

Forward Kinematics

```
0  0  1 18
0 -1  0  0
1  0  0 10
0  0  0  1
```

Spatial Manipulator Jacobian

```
0 -10 -10   0 -10   0
0   0   0 -18   0  10
0   0   8   0  18   0
0   0   0   0   0   1
0   1   1   0   1   0
1   0   0   1   0   0
```

Figure 1: Initial GUI Output

The motivation for this project was to create a software that made it easier to visualize the relationship between Denavit-Hartenberg parameters and the associated robotic mechanism. The software can also be used to verify the validity of Denavit-Hartenberg parameters derived for a specific robot with known geometry. The software also makes it faster to obtain the forward kinematics and spatial manipulator Jacobian for different final configurations of robotic mechanisms.

The scope of this project covers Denavit-Hartenberg parameters, the modified transformation matrix, forward kinematics, and the spatial manipulator Jacobian. These principles are implemented into software using MATLAB and a guided user interface.

The software was created using MATLAB. The GUI was first outlined using the MATLAB GUIDE feature. The GUI was then edited in the callback functions to obtain the D-H parameters and the type of each joint from the user. These D-H parameters were used to calculate the total transformation of the mechanism from the base frame to the tool frame. The transformation between each frame was also calculated in order to plot the location and orientation of each link frame.The transformation matrix for the entire mechanism and the spatial manipulator Jacobian were calculated and displayed on the GUI. Additionally, three buttons were created that change the D-H parameters to illustrate three different example robotic mechanisms.

3 Project Details

3.1 Visualization of robotic mechanism

The visualization of the robotic mechanism was obtained from the transformation matrix of each frame individually. The calculation of all of the transformation matrices occurs in the subfunction "modtransform". The function uses the modified Denavit-Hartenberg frame transformation equation as shown below in Equation 1.

$$g_{i/i+1} = \begin{bmatrix} cos\theta_{i+1} & -sin\theta_{i+1} & 0 & a_i \\ sin\theta_{i+1}cos\alpha_i & cos\theta_{i+1}cos\alpha_i & -sin\alpha_i & -d_i sin\alpha_i \\ sin\theta_{i+1}sin\alpha_i & cos\theta_{i+1}sin\alpha_i & cos\alpha_i & d_i cos\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

The values for α and θ are both entered in degrees for ease of use with the GUI. For a robot with N Degrees of Freedom, the modtransform function would output N+1 transformation matrices. The first N transformation matrices are for each of the joint frames from the stationary frame. The last transformation matrix is the total transformation matrix from the stationary frame to the tool frame. A transformation matrix is defined in general as the following.

$$g_{AB} = \begin{bmatrix} R_{AB} & \overline{d_{AB}} \\ \overline{0}^T & 1 \end{bmatrix} \tag{2}$$

Thus, for the i^{th} transformation matrix of the first N transformation matrices, the transformation matrix gives us the location and orientation of the i^{th} frame. The location of the frame is given by $\overline{d_{AB}}$ and the rotation of the frame coordinates are given by the columns of R_{AB} where each column is a vector for each frame coordinate.

3.2 Forward Kinematics

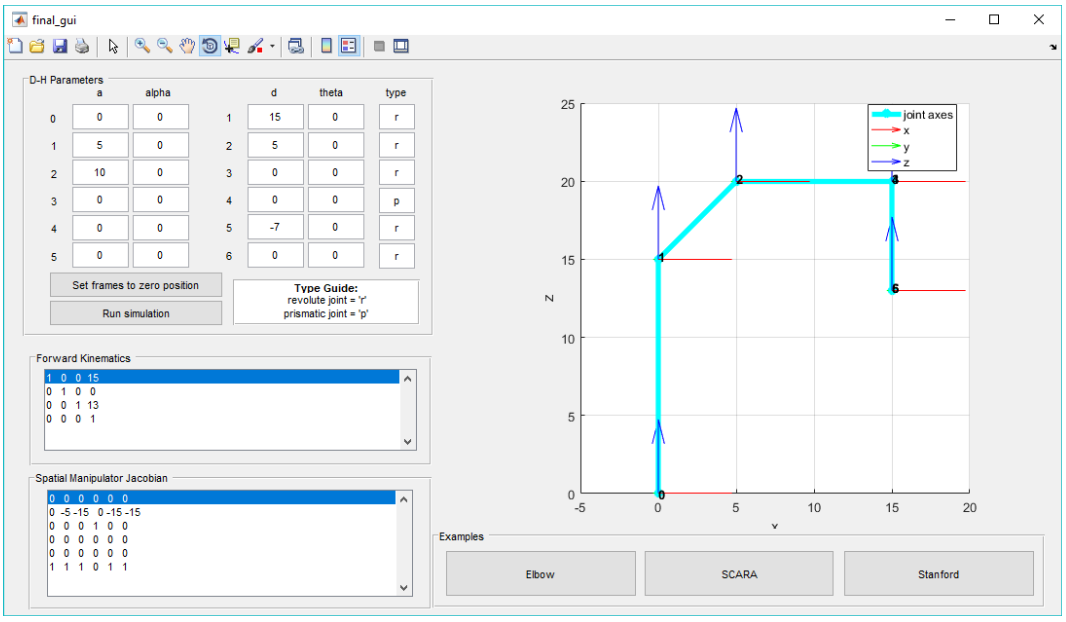
The forward kinematics of the entire mechanism was calculated using the transformation matrix of each frame as shown in the following equation.

$$g_{st}(\theta) = g_{l_0 l_1}(\theta_1) g_{l_1 l_2}(\theta_2) \dots g_{l_5 l_6}(\theta_6) g_{l_6 l_t} \tag{3}$$

For the purposes of this software, the tool frame was assumed to be the same as the last frame.This multiplication results in a final matrix as shown below.

$$g_{st} = \begin{bmatrix} R_{st} & \overline{d_{st}} \\ \overline{0}^T & 1 \end{bmatrix} \tag{4}$$

Where R_{st} is the rotation matrix of the tool frame coordinates from the stationary frame coordinates and $\overline{d_{st}}$ is the displacement of the tool frame from the stationary frame. This matrix was output to the GUI in the ”Forward Kinematics” panel. The user can vary the inputs for the joint variables and analyze how each change in joint variable effects the forward kinematics of the entire mechanism. The forward kinematics can be verified in the following software screenshot.



D-H Parameters

	a	alpha		d	theta	type
0	0	0	1	15	0	r
1	5	0	2	5	0	r
2	10	0	3	0	0	r
3	0	0	4	0	0	p
4	0	0	5	-7	0	r
5	0	0	6	0	0	r

Type Guide:
revolute joint = 'r'
prismatic joint = 'p'

Forward Kinematics

```
1  0  0 15
0  1  0  0
0  0  1 13
0  0  0  1
```

Spatial Manipulator Jacobian

```
0  0  0  0  0  0
0 -5 -15  0 -15 -15
0  0  0  1  0  0
0  0  0  0  0  0
0  0  0  0  0  0
1  1  1  0  1  1
```

Figure 2: Verification of Forward Kinematics of SCARA manipulator at $\theta = 0$

The forward kinematics indicates that the tool frame has zero rotation and a displacement of 15 units in the x-direction, 0 in the y-direction, and 13 in the z-direction.This results can be visually verified from the visual representation of the manipulator.

3.3 Spatial Manipulator Jacobian

The spatial manipulator Jacobian illustrates the twist for each joint axis in the mechanism. The i^{th} column of the Jacobian corresponds to the i^{th} joint axis twist. For a revolute joint, the joint twist was calculated as follows.

$$\xi_i = \begin{bmatrix} -\omega_i \times q_i \\ \omega_i \end{bmatrix} \tag{5}$$

Where ω_i is the direction of the i^{th} joint axis and q_i is the location of the i^{th} joint frame from the stationary frame. Both ω_i and q_i are in reference to the stationary frame.

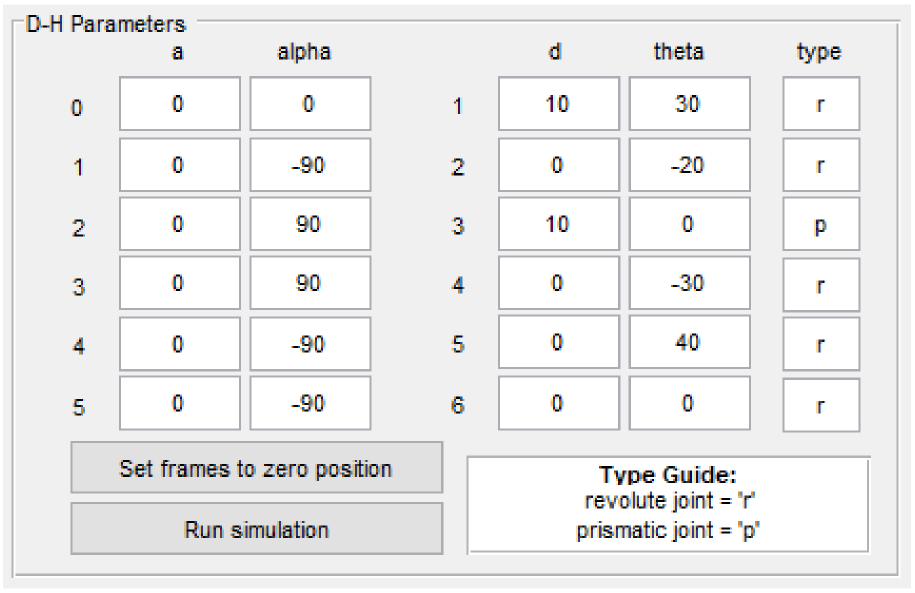
For a prismatic joint, since the pitch of the joint is infinite, the joint twist is calculated as follows.

$$\xi_i = \begin{bmatrix} \omega_i \\ \hline 0 \end{bmatrix} \tag{6}$$

The Jacobian for the entire mechanism is output to the GUI in the "Spatial Manipulator Jacobian" panel. This output also allows users to investigate how different joint parameters influence the Jacobian.

3.4 User Interface

The primary purpose of the software is the ability to obtain user input. The layout of the user input section of the software can be seen in the following screenshot.



	a	alpha		d	theta	type
0	0	0	1	10	30	r
1	0	-90	2	0	-20	r
2	0	90	3	10	0	p
3	0	90	4	0	-30	r
4	0	-90	5	0	40	r
5	0	-90	6	0	0	r

Type Guide:
revolute joint = 'r'
prismatic joint = 'p'

Figure 3: Software User Interface

The user has the ability to enter the Denavit-Hartenberg parameters for a desired robotic mechanism using the text entry boxes. The user also can enter the type of joint for each frame. The possible options are 'r' for revolute joints, and 'p' for prismatic joints. These joints can also be combined to form other kinds of joints. For example, a cylindrical joint could be formed by combining a revolute and prismatic joint.

To originally create the visualization of the robot manipulator in the desired orientation, the theta values for specific links may need to be adjusted. For example, the theta value for the 5th link frame of the Elbow Manipulator, shown in Figure 5, needs to be -90 degrees in order to achieve the desired tool frame orientation. Thus, the user first inputs the Denavit-Hartenberg parameters with the theta values that achieve the desired frame orientations. Then, the user selects the "Set frames to zero position" button to identify the theta values as offset angles to the initialized positions. The user interface will then reinitialize the theta values to be zero so that any further changes to the theta values will reflect the desired changes in joint variables. However all calculations will be for the addition of the offset angles and the change in joint variables. For prismatic joints, the joint variable will be added to the constant d for a given link. Once the user has initialized the zero position for all the joint frames and entered the desired joint variables, the user then clicks on the "Run simulation" button to rerun the calculations with the new joint variables. There is no need to re-initialize the zero position for a robotic mechanism once it has been initialized.

4 Software Outputs

Since the software is intended to be very user dependent, three examples of outputs of the software are illustrated in this section. These examples can also be investigated in the software using the three "Examples" buttons. All of these examples were based off of robotic mechanisms discussed in "A Mathematical Introduction to Robotic Manipulation" [1].

4.1 Example 1: Elbow Manipulator

The first example of a robotic mechanism is the Elbow Manipulator. A diagram of this device, can be seen in Figure 4.

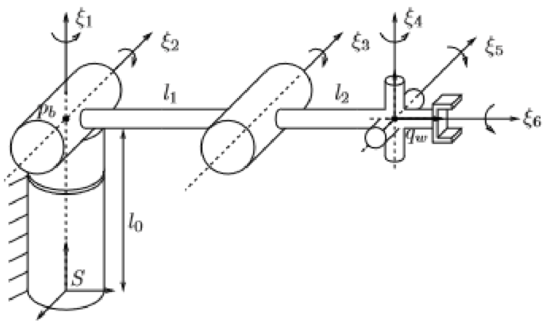
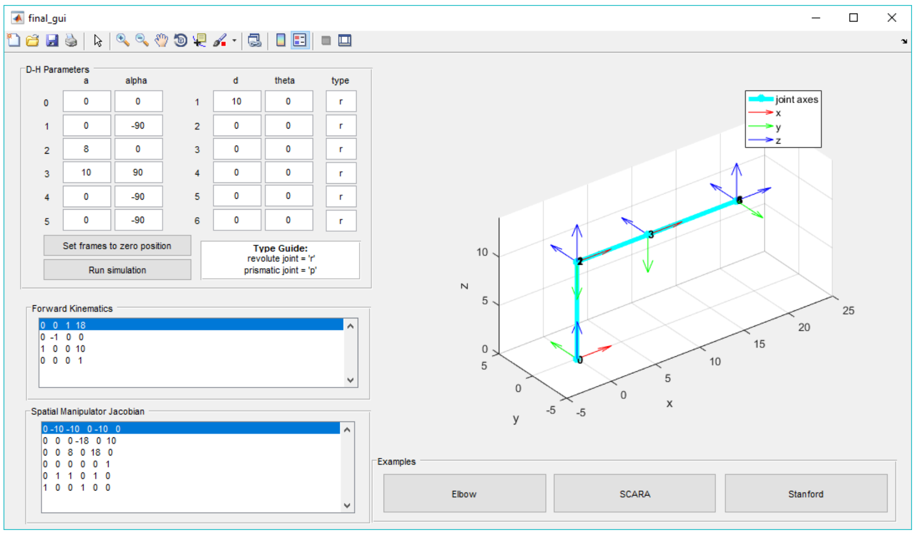


Figure 4: Elbow Manipulator at $\theta = 0$ [1]

This same mechanism can be illustrated using the software by inputing the Denavit-Hartenberg parameters of the elbow manipulator shown in Figure 4. The screenshot of the output of the software can be seen in Figure 5



D-H Parameters

	a	alpha		d	theta	type
0	0	0	1	10	0	r
1	0	-90	2	0	0	r
2	8	0	3	0	0	r
3	10	90	4	0	0	r
4	0	-90	5	0	0	r
5	0	-90	6	0	0	r

Set frames to zero position

Run simulation

Type Guide:
revolute joint = 'r'
prismatic joint = 'p'

Forward Kinematics

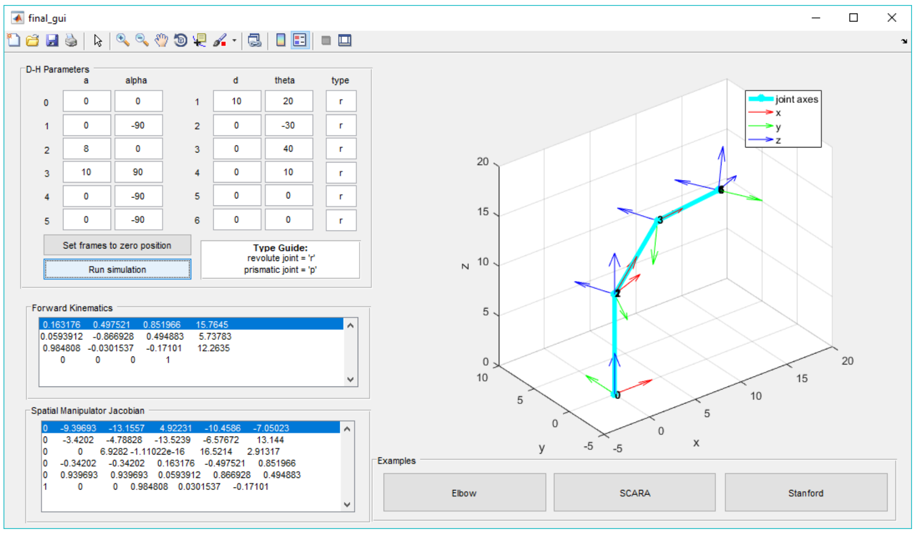
```
0  0  1  18
0 -1  0   0
1  0  0  10
0  0  0   1
```

Spatial Manipulator Jacobian

```
0 -10 -10   0 -10   0
0   0   0 -18   0  10
0   0   8   0  18   0
0   0   0   0   0   1
0   1   1   0   1   0
1   0   0   1   0   0
```

Figure 5: Elbow Manipulator at $\theta = 0$

The joint variables can be adjusted for each frame to explore the forward kinematics and spatial manipulator Jacobian. A screenshot of the output of the software for a specific variation of the joint variables can be seen in Figure 6



D-H Parameters

	a	alpha		d	theta	type
0	0	0	1	10	20	r
1	0	-90	2	0	-30	r
2	8	0	3	0	40	r
3	10	90	4	0	10	r
4	0	-90	5	0	0	r
5	0	-90	6	0	0	r

Set frames to zero position

Run simulation

Type Guide:
revolute joint = 'r'
prismatic joint = 'p'

Forward Kinematics

```
 0.163176    0.497521    0.851966    15.7645
0.0593912   -0.866928    0.494883    5.73783
 0.984808  -0.0301537   -0.17101     12.2635
    0          0           0           1
```

Spatial Manipulator Jacobian

```
0   -9.39693   -13.1557    4.92231   -10.4586   -7.05023
0   -3.4202    -4.78828   -13.5239   -6.57672    13.144
0       0       6.9282 -1.11022e-16  16.5214    2.91317
0   -0.34202   -0.34202    0.163176  -0.497521   0.851966
0    0.939693   0.939693   0.0593912  0.866928   0.494883
1       0          0       0.984808   0.0301537  -0.17101
```

Figure 6: Elbow Manipulator for arbitrary joint variables.

4.2 Example 2: SCARA Manipulator

Another example of a robotic mechanism is the SCARA Manipulator. Figure 5 shows a diagram of this mechanism (MLS,p.88) [1].

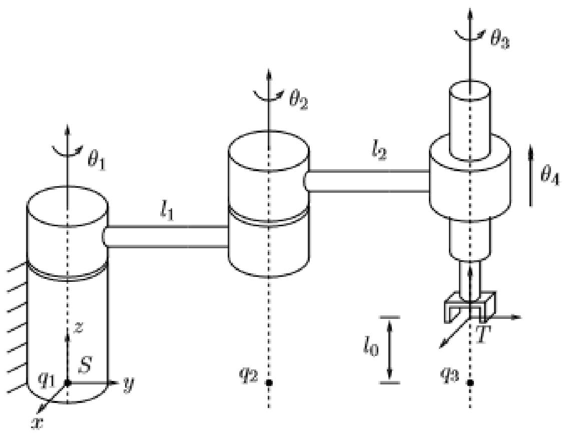
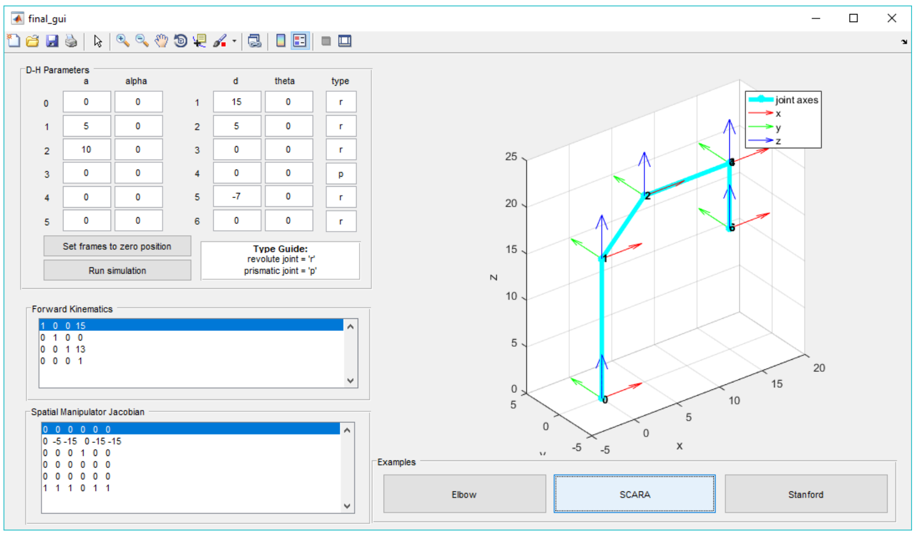


Figure 7: SCARA Manipulator at $\theta = 0$

The output of the software for the SCARA manipulator D-H parameters can be seen in Figure 8.



D-H Parameters

	a	alpha		d	theta	type
0	0	0	1	15	0	r
1	5	0	2	5	0	r
2	10	0	3	0	0	r
3	0	0	4	0	0	p
4	0	0	5	-7	0	r
5	0	0	6	0	0	r

Set frames to zero position

Run simulation

Type Guide:
revolute joint = 'r'
prismatic joint = 'p'

Forward Kinematics

```
1  0  0  15
0  1  0  0
0  0  1  13
0  0  0  1
```

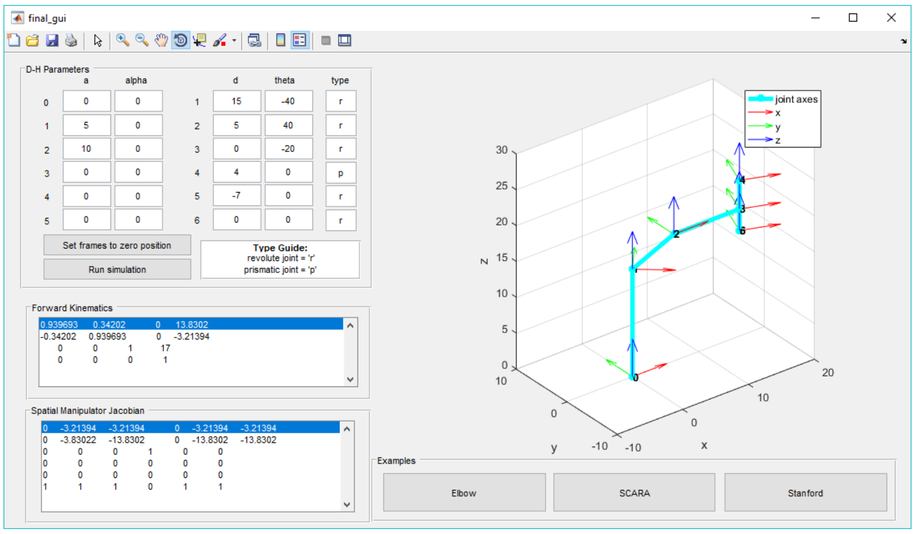
Spatial Manipulator Jacobian

```
0  0  0  0  0  0
0  -5 -15  0 -15 -15
0  0  0  1  0  0
0  0  0  0  0  0
0  0  0  0  0  0
1  1  1  0  1  1
```

Figure 8: SCARA Manipulator at $\theta = 0$

The constant parameters for the scara manipulator in the software, correponsding to the variables in Figure 7, are l_1=5, l_2=10, l_0=13. In the digram of the SCARA manipulator, the robot clearly has a horizontal connection between joint axes 1 and 2 with a vertical offset between frames. However, the software has no way of knowing the geometry of the links connecting joint frames, and thus the software connects the frames with a straight line. This is one drawback of using visualization software. However, the kinematics of the mechanism will still be the same regardless of the connecting link geometry.

A variation of the SCARA manipulator can be seen in Figure 9 for arbitrary joint variables.



D-H Parameters

	a	alpha		d	theta	type
0	0	0	1	15	-40	r
1	5	0	2	5	40	r
2	10	0	3	0	-20	r
3	0	0	4	4	0	p
4	0	0	5	-7	0	r
5	0	0	6	0	0	r

Set frames to zero position

Run simulation

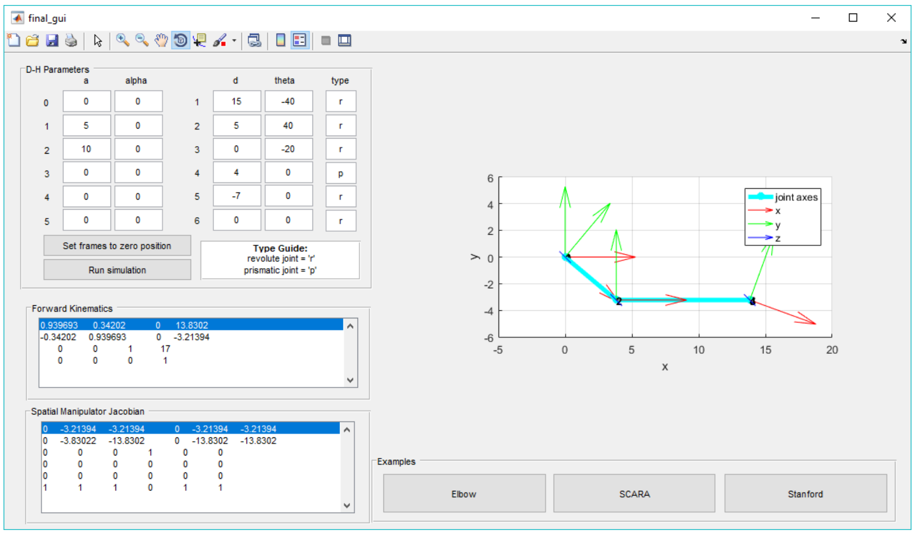
Type Guide:
revolute joint = 'r'
prismatic joint = 'p'

Forward Kinematics

```
0.939693   0.34202     0    13.8302
-0.34202   0.939693    0    -3.21394
    0          0       1      17
    0          0       0       1
```

Spatial Manipulator Jacobian

```
0   -3.21394   -3.21394      0   -3.21394   -3.21394
0   -3.83022   -13.8302      0   -13.8302   -13.8302
0       0          0         1       0          0
0       0          0         0       0          0
0       0          0         0       0          0
1       1          1         0       1          1
```



D-H Parameters

	a	alpha		d	theta	type
0	0	0	1	15	-40	r
1	5	0	2	5	40	r
2	10	0	3	0	-20	r
3	0	0	4	4	0	p
4	0	0	5	-7	0	r
5	0	0	6	0	0	r

Set frames to zero position

Run simulation

Type Guide:
revolute joint = 'r'
prismatic joint = 'p'

Forward Kinematics

```
0.939693   0.34202     0    13.8302
-0.34202   0.939693    0    -3.21394
    0          0       1      17
    0          0       0       1
```

Spatial Manipulator Jacobian

```
0   -3.21394   -3.21394      0   -3.21394   -3.21394
0   -3.83022   -13.8302      0   -13.8302   -13.8302
0       0          0         1       0          0
0       0          0         0       0          0
0       0          0         0       0          0
1       1          1         0       1          1
```

Figure 9: SCARA Manipulator for arbitrary joint variables

The SCARA manipulator demonstrates the use of a prismatic joint in a robotic mechanism and how that influences the Jacobian.

4.3 Example 3: Stanford Manipulator

A third example of a robotic mechanism is the Stanford Manipulator, shown in Figure 8 (MLS, p.119) [1].

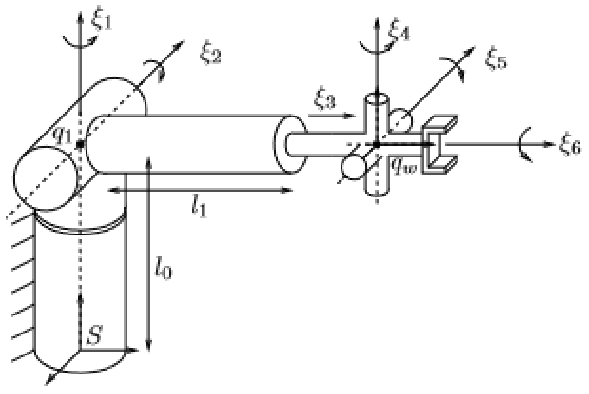
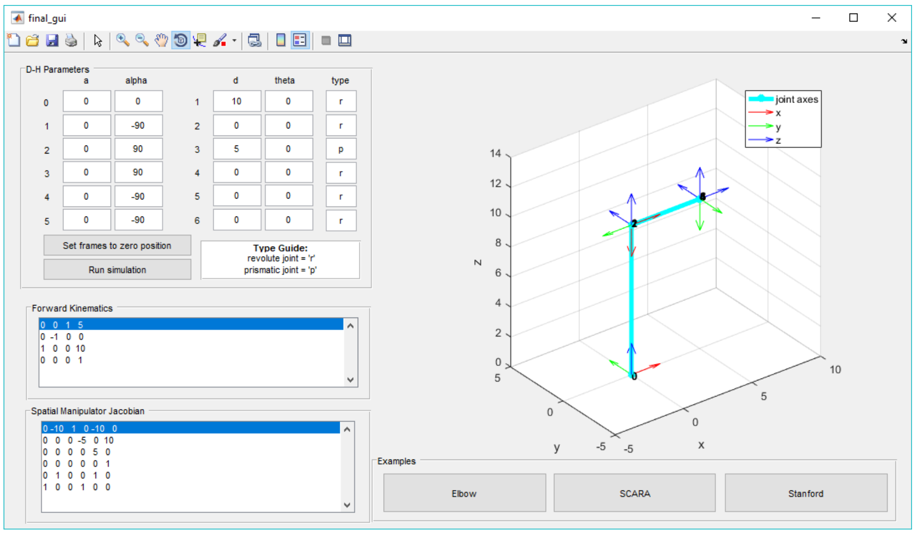


Figure 10: Stanford Manipulator at $\theta = 0$

The output of the software for the Standford Manipulator and its corresponding D-H parameters can be seen in Figure 11.



D-H Parameters

	a	alpha		d	theta	type
0	0	0	1	10	0	r
1	0	-90	2	0	0	r
2	0	90	3	5	0	p
3	0	90	4	0	0	r
4	0	-90	5	0	0	r
5	0	-90	6	0	0	r

Type Guide:
revolute joint = 'r'
prismatic joint = 'p'

Forward Kinematics

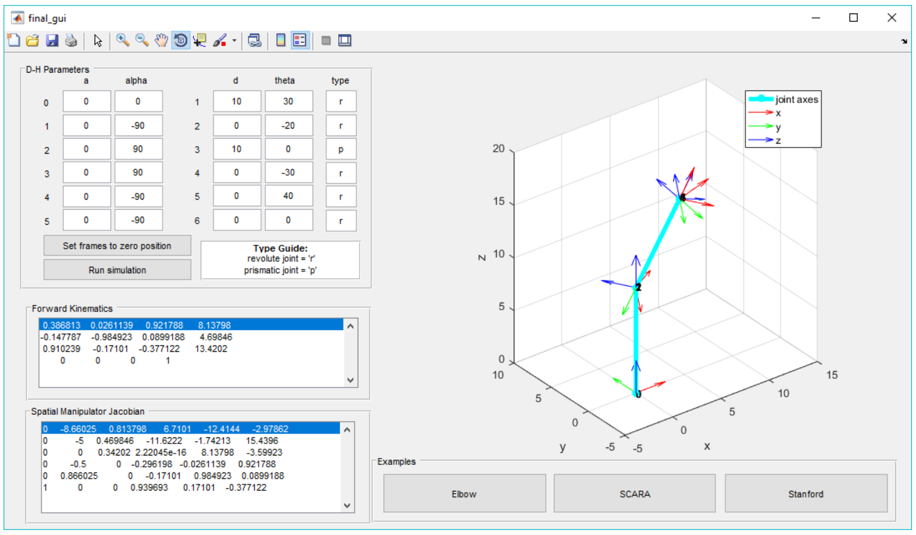
```
0  0  1  5
0 -1  0  0
1  0  0 10
0  0  0  1
```

Spatial Manipulator Jacobian

```
0 -10  1  0 -10  0
0  0  0 -5  0 10
0  0  0  0  5  0
0  0  0  0  0  1
0  1  0  0  1  0
1  0  0  1  0  0
```

Figure 11: GUI Stanford Manipulator at $\theta = 0$

The constant parameters for the Stanford manipulator in the software, corresponding to the variables in Figure 11, are l_0=10 and l_1=5.A variation of the Stanford manipulator can be seen in 12 for arbitrary joint variables.


final_gui

D-H Parameters

	a	alpha		d	theta	type
0	0	0	1	10	30	r
1	0	-90	2	0	-20	r
2	0	90	3	10	0	p
3	0	90	4	0	-30	r
4	0	-90	5	0	40	r
5	0	-90	6	0	0	r

Set frames to zero position

Run simulation

Type Guide:
revolute joint = 'r'
prismatic joint = 'p'

Forward Kinematics

```
 0.386813   0.0261139   0.921788    8.13798
-0.147787  -0.984923   0.0899188    4.69846
 0.910239   -0.17101   -0.377122   13.4202
     0          0          0          1
```

Spatial Manipulator Jacobian

```
0   -8.66025   0.813798     6.7101    -12.4144    -2.97862
0        -5    0.469846   -11.6222    -1.74213    15.4396
0         0     0.34202  2.22045e-16   8.13798    -3.59923
0      -0.5          0   -0.296198   -0.0261139   0.921788
0   0.866025          0   -0.17101    0.984923    0.0899188
1         0          0   0.939693    0.17101     -0.377122
```

Figure 12: GUI Stanford Manipulator at $\theta = 0$

In this specific example, $\theta_1 = 30°$, $\theta_2 = -20°$, $\theta_3 = 5$ (displacement in d), $\theta_4 = -30°$, $\theta_5 = 40°$. Each of these joint variables defines the change in joint angle (or displacement for prismatic joints) from the defined $\theta = 0$ position.

5 Conclusion

Overall, this software provides a graphical representation of robotic mechanisms and illustrates how Denavit-Hartenberg parameters influence the geometry of a robot mechanism as well as the forward kinematics and spatial manipulator Jacobian. It also demonstrates how changes to the mechanism's joint variables changes the location and orientation of the tool frame as well as the Jacobian. Future iterations of the software could include adding more options for types of joints as well as implementing inverse kinematics.

References

[1] R. M. Murray, Z. Li, and S. S. Sastry, "A mathematical introduction to robotic manipulation," 1994.