Robot Simulator in MATLAB

Lodes, A.

***Abstract*—The joint configuration of any robot can be described by the Denavit-Hartenberg parameters. These parameters are enough to obtain a working visualization of the robot described. Presented is a MATLAB program which models any robot given a set of corresponding DH parameters. This simulation allows the user to visualize the joints and movements of the modeled robot.**

***Index Terms*—Robot Simulator, DH representation, MATLAB**

I. Introduction

THE Denavit-Hartenberg representation of forward kinematic equations of robots has become the standard technique for modeling robots and their motions. The technique summarizes the relationship between two joints in concise set of four parameters. Any robot can be modeled using the DH representation.

A computer code has been created in MATLAB to implement the modeling of any robot with only the DH parameters as input. The purpose of the simulator is to create an accurate visual representation of any type of robot and its motions. The simulator also allows for the independent manipulation of each joint of the modeled robot.

Presented in this study are the details of this simulator as well as background on the DH representation and some analysis on how effectively the simulator models some example robots.

II. BACKGROUND

In 1955, Denavit and Hartenberg published a paper [1] explaining a kinematic notation that was eventually adapted as a method to represent robots. The method defines robots as a sequence of joints, each with a degree of freedom. Each joint has its own reference frame complete with a z and x axis, the intersection of which defines the joint's origin. Each joint is defined as either prismatic, when the motion is a linear translation along the joint's z axis, or revolute, when the motion is a rotation about the joint's z axis. Each joint is defined iteratively in terms of the transformation necessary from the previous joint. The first joint is defined as a transformation from a reference origin and axis. This technique is detailed in Figure 1 and Figure 2.

The transformation from joint *n* and joint *n+1* has four steps as follows. First, rotate an angle of θ_{n+1} about the z_n axis. This aligns x_n with x_{n+1}. Second, translate along the z_n axis a distance d_{n+1} to make the x_n and x_{n+1} axis collinear. Third, translate along the x_n axis a distance α_{n+1}. This makes the two origins in the same location. Finally, rotate the z_n axis about the x_{n+1} axis an angle of α_{n+1}. This process aligns both the origins and reference frames of joint *n* and joint *n+1* and is depicted in Figure 2.

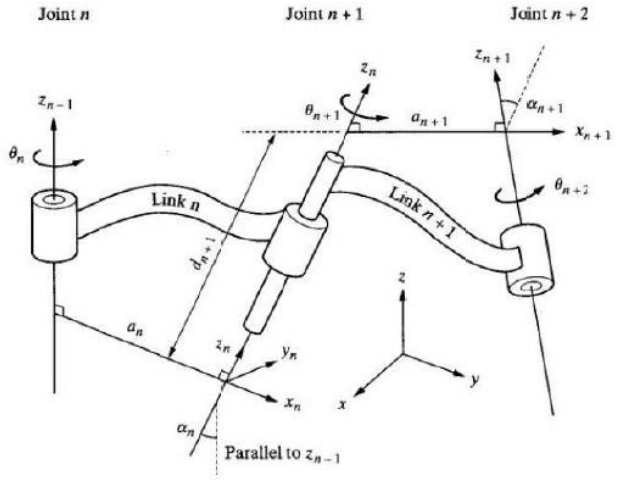

Fig. 1. Denavit-Hartenberg represenation [2].

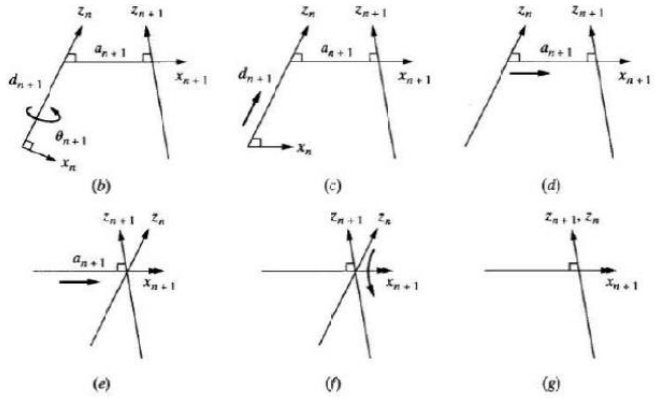

Fig 2. Transformation from joint *n* and joint *n+1* in DH representation [2].

The transformation between reference frame *i - 1* and reference frame *i* can be easily calculated by following these above steps and is shown in Equation 1.

$$T_{\mathrm{i}}^{i-1} = \begin{bmatrix} c\theta_{\mathrm{i}} & -s\theta_{\mathrm{i}}c\alpha_{\mathrm{i}} & s\theta_{\mathrm{i}}s\alpha_{\mathrm{i}} & a_{\mathrm{i}}c\theta_{\mathrm{i}} \\ s\theta_{\mathrm{i}} & c\theta_{\mathrm{i}}c\alpha_{\mathrm{i}} & -c\theta_{\mathrm{i}}s\alpha_{\mathrm{i}} & a_{\mathrm{i}}s\theta_{\mathrm{i}} \\ 0 & s\alpha_{\mathrm{i}} & c\alpha_{\mathrm{i}} & d_{\mathrm{i}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

A. Lodes is with the Department of Computer and Electrical Engineering, University of Missouri-Columbia, Columbia, MO 65211 USA (e-mail: alvb4@mail.missouri.edu).

Following this technique iteratively, each joint can be described by the previous until all of the joints of the robot have been described. The resultant parameters θ, d, a, α are the critical parameters to define one joint in terms of the previous. The values of these parameters for each joint are often represented in a table known as a DH table. Every robot can be described by its DH table. For a more exhaustive review of the Denavit-Hartenberg Representation see references [1], [2].

This simulator uses the DH parameters in the table to model the robot and its motions. This is described in detail in the following sections.

III. APPROACH AND IMPLEMENTATION

The aim of the simulation was to model any robot with only the DH parameters. Careful consideration was given to the modeling of the robot links. The DH parameters of a robot do not specify exact link dimensions, base or end dimensions, or joint limitations. Without extra information given, some assumptions had to be made regarding the link shape and size, default base and end affecter shapes were chosen, and values were assigned to joint limitations. These assumptions have considerable impact on the final visualization of a robot and its movements and will be described in thorough detail below.

Other considerations were given to the workings of the simulator program itself. The graphical user interface(GUI) was carefully designed to allow a number of options for loading DH parameters. The figure displaying the robot as well as the GUI controls for animating the robot were chosen to be both aesthetically pleasing and functional. These program design decisions are detailed in this section. Also presented are some practical decisions regarding the implementation in MATLAB of the animation of the robot joints.

A. Link Visualization

As described in Section II, the DH parameters specify the relationship between the origins of each joint. The physical link between each origin pair is not described in the DH parameters.

There is only one constraint on each link; the link must be physically continuous between the two joints it connects. A link could be straight, curved, or angled to achieve this constraint. Given no other information, it is best for simulation to try and visualize links in a way that accurately reflects joint motion.

An easy assumption would be to draw a line between the two origins. This is a simple interpretation, but in practice is too often misrepresentative of the actual physical robot being modeled. This assumption can lead to some difficulties in visualizing the movement of the robot modeled.

The end-user often focuses on the moving links, and not what those link movements represent in terms of joint movements. This is not desirable, as in fact it does detract from properly visualizing the robot for the end user.

It is because of this undesirable consequence that this simulation has chosen to make a different assumption about link structure. Each link in this simulator is modeled as a simple shape along the z axis of the origin which is has dimensions related to the *d* and *a* parameters from the DH parameters for that joint transformation. By definition, the *n+1* origin is by distance *d* along the z axis and distance *a* along the x axis away from the *n* origin. The approach taken for this simulator is to make the link *d* in length along the z axis from the origin and *2*a* or *a* wide along the x axis, centered at the *n* origin (depending on if the joint is revolute or prismatic). This approach visualizes each link along its full range of motion as can be seen in Section IV, Robot Simulations and Results.

For prismatic joint *n*, a rectangular box is used in this simulator as the link between joints *n* and *n+1*. This box is *d* in height along the z axis from the origin and is *a* wide and *a* long centered at the origin.

For revolute joint *n*, a cylinder is used in this simulator as the link between joints *n* and *n+1*. This cylinder is *d* in height along the z axis from the origin and has a radius of *a* from the origin.

In order to visualize joint animation, each link has a minimum height and width. Only the link is drawn with these dimensions, the DH table for the robot is not affected. The default minimum height and width value of the simulator is 10, and can be changed within the source code as necessary.

Additionally each joint is colored in order of appearance, red, blue, yellow, orange, purple, gold, army green, cyan, magenta, grey. After the tenth joint, the colors repeat beginning again at red.

Also of note is the difficulty in scaling the link sizes to show all of the joints within proportion for any robot. This program sidesteps this limitation by allowing the user to zoom in, zoom out, and rotate the figure through the menu bar on the top of the GUI.

B. Joint Limitations

The limits of each joint type were arbitrarily chosen. Revolute joints have limits of -180º to 180º. Prismatic joints have a maximum value of 150 which can be modified within the source code as necessary. Some error checking is done within the program to ensure the user cannot enter a Theta or alpha angle in the DH parameter beyond the joint limits. No error checking is done for the DH parameters of the prismatic joint *d* value, however this could be added easily if necessary.

Joint limits are checked when controlling the robot animations. The slider bars are only able to slide within the joint limits, and the edit fields check the joint limits before executing code. This is to minimize the number of errors generated by the program if an improper input is given.

No consideration is given to the individual links or joints overlapping or interfering with each other. The model allows for the robot joints to act independently. This allows for movements in the model that would not be possible when for a physical robot. This effect is intentional, as it does allow the user to visualize when overlapping is possible. Because of the link visualization implementation described above, the model

may show two links overlapping in situations where the physical robot's links may not overlap. Without additional link information or joint limitations, this is unavoidable.

C. Base and End Affecter

The shapes of the base and end affecter were chosen to help with aesthetics and the visualization of robot movements. The base is a cylinder tall enough to go from the minimum z coordinate of the world, to the world reference point. The base cylinder is of the default link radius. The end affecter is a small gripper like tool, and was added to better visualize the movement of the last joint.

Neither the base nor the end affecter is end-user editable. However minor changes in the source code are all that is necessary to change their size or appearance.

D. GUI Implementation

The simulator program opens with a welcome message and several buttons to choose how to load a DH table. Figure 3 shows this GUI. The five buttons on the right each load a different pre-made robot. The values for the DH tables of these robots have been saved in *.mat* files generated by the .m files. To change these premade values, the source code for the *.m* files can be modified and run to reflect the new desired DH parameters.

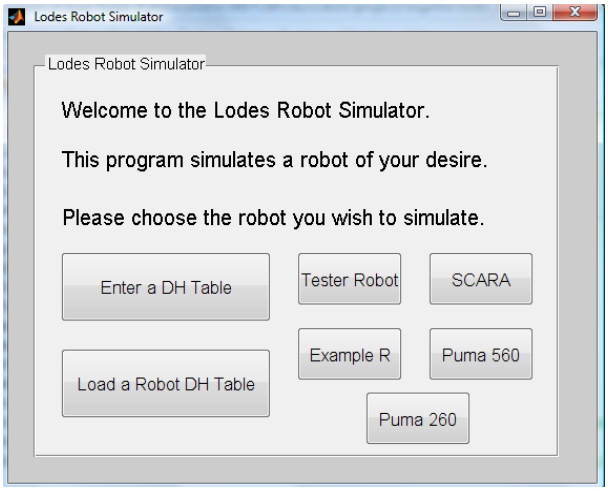

Fig. 3. Welcome GUI of the MATLAB program.

The buttons on the left allow the user to work with a custom robot. *Load a Robot DH Table* loads a premade DH table named *Robot.mat*. This input file has certain constraints and needs to follow the structure of the previously made example robot *.m* files.

Enter a DH Table allows the user to enter a DH Table with the GUI shown in Figure 4. First the program asks the user the number of joints the robot will have in another GUI, Figure 5. The size of the table in the GUI in Figure 4 is adjusted to reflect the number of joints entered by the user. The table itself has five columns. The first column is a drop box to have the user choose whether the joint is either prismatic or revolute. The other four columns are the DH parameters themselves. This table was implemented using the MATLAB function *uitable*. When finished entering a table, the user must click the *Continue* button to move on to the robot model. This user entered DH table is saved in the file *Robot.mat* so the user may load it again when restarting the program.

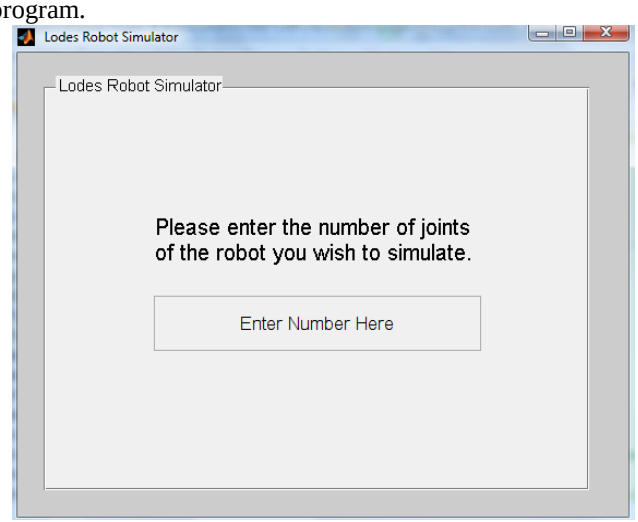

Fig. 4. GUI which determines the number of joints of a user loaded robot.

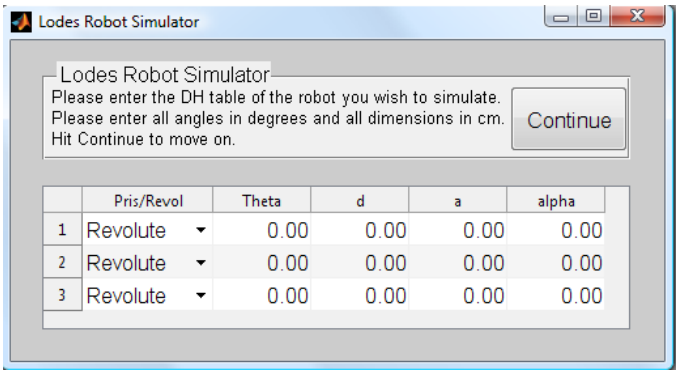

Fig. 5. GUI which allows the user to input the parameters of the DH table.

After creating or loading the DH parameters of a robot, the program opens up two windows. One is a non editable table of the DH parameters of the robot modeled. The second window, an example of which is seen in Figure 6, is the 3D model of the robot along with a GUI for controlling the animation of the joints of the robot. The slider controls and edit fields allow for the user to manipulate the joints of the robot. This controls initiate animation to the desired new position.

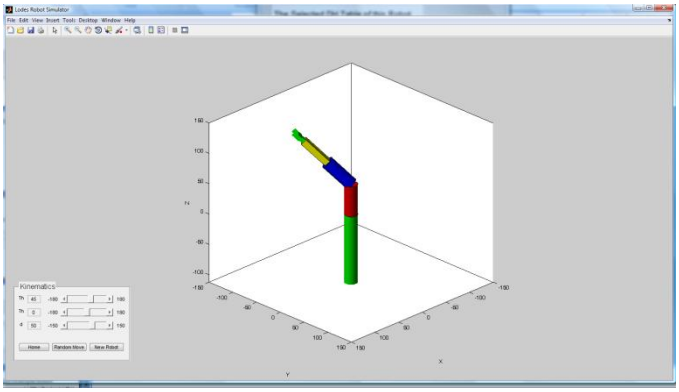

Figure 6 The robot model GUI with kinematic controls

The *Home* button returns the robot to its original position described by the original DH table. The *Random Move* button creates a random motion in each of the joints of the robots. The *New Robot* button sends the user back to the welcome screen shown in Figure 3 to choose a new Robot to model. A zoomed in picture of the kinematic panel of this GUI is shown in Figure 7.

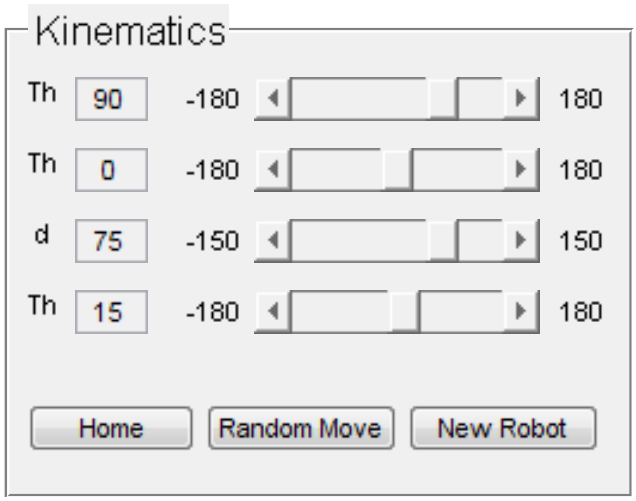

Fig. 7. Example kinematics control panel with buttons.

E. Animation

Animation of the robot is implemented by creating a linear 10 step progression between the current position and the desired new position. The robot is redrawn entirely for each new position, deleting the previously drawn robot. The number of steps as well as the delay between steps (0.5 sec) can be changed as necessary in the source code.

IV. ROBOT SIMULATIONS AND RESULTS

The robot simulator has been tested with five different robot configurations. The five robots consist of the widely known SCARA Robot, the Puma 560, the Puma 260, one other robot used in reference [2] known here as Example R , and the robot used to originally construct the simulator, known as Tester Robot.

Each robot model has been put through the full functionality of the program. Ultimately the simulator program is only as good as how effective it is at visualizing robots, so with that guideline each robot will be compared to its model. The reader is referred to the program itself as well as its source code to better familiarize himself with the accuracy of the models.

A. Tester Robot

The Tester Robot was created alongside writing the source code of this program. It was intentionally made simple for the author to visualize the robot and its motions. This robot was created before animation was implemented, so at the time, manually changing the DH table was the only way to ensure correct operation of the joints. Because the Tester Robot was designed in this fashion it can only be compared to the ideal which the DH table defines. The DH table for this robot is shown in Table 1, and the model for this robot is shown in Figure 8.

Table 1. DH Table for Tester Robot.

	Pris/Revol	Theta	d	a	alpha
1	Revolute	45.00	50.00	0.00	45.00
2	Revolute	0.00	50.00	0.00	0.00
3	Prismatic	0.00	50.00	0.00	0.00

The green shape on the bottom is the base, the green shape on the top is the end affecter, as described in Section III. The other three shapes are the link of the robot described by the above table. Upon inspection, the sizes of all the joints seem modeled correctly when taking into account the minimum joint dimensions. Here, with all zero *a* parameters, each link is drawn with its minimum default value.

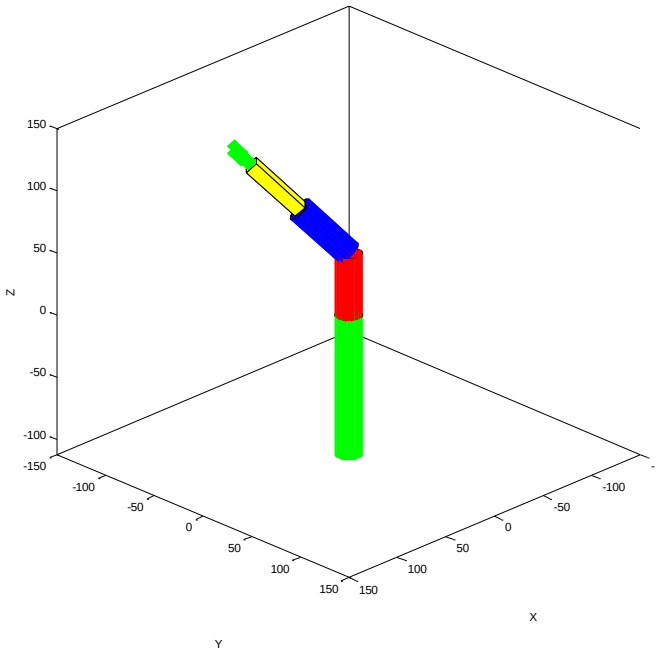

Fig. 8. Program model of Tester Robot

Movement of the joints of this robot, as in all of the robots model is easy to visualize when observing the effect of the motion on the later links. Here the rotation of link 1, shown red, between joint 1 and joint 2 changes the orientation of all of the other links. Such a rotation shows the blue and yellow links rotating about the axis of the red link. This is best observed when interacting with the simulator and animating the robot with the kinematics controls.

With this simple example, testing shows each joint to move properly. A particular observation to point out is the third joint, shown yellow, is a prismatic joint. When the *d* parameter of this joint is modified, this link is shown to extend/retract.

B. *SCARA Robot*

The SCARA Robot is an excellent robot for testing purposes as it is simple yet widely known. The DH parameters for the SCARA Robot are shown in Table 2. A representation of the SCARA Robot is shown in Figure 9 from reference [2]. From this clear definition, it should be possible to understand the accuracy of the program's model of the SCARA Robot, shown in Figure 10.

Table 2. DH Table for the SCARA Robot

	Pris/Revol	Theta	d	a	alpha
1	Revolute	90.00	0.00	50.00	0.00
2	Revolute	0.00	0.00	20.00	180.00
3	Prismatic	0.00	75.00	0.00	0.00
4	Revolute	15.00	10.00	0.00	0.00

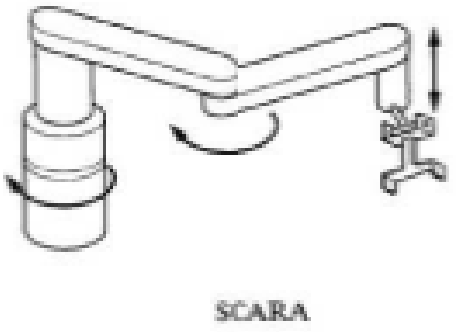

Fig. 9. The SCARA Robot [2].

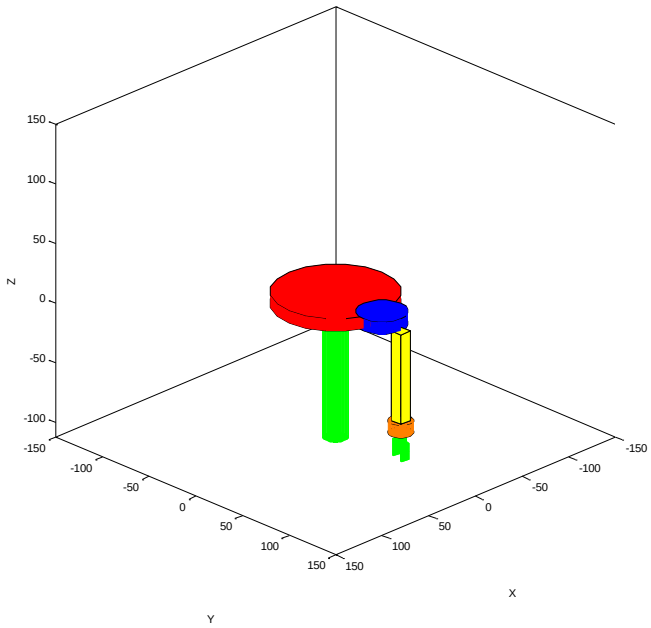

Figure 10. Program model of the SCARA Robot.

The comparison of Figure 9 and Figure 10 shows precisely the oddities of our assumptions for link visualization. The first link connecting joint 1 and joint 2, shown in red, is a revolute joint. It does not appear similar to the picture of the actual robot. The actual robot has something closer to just a rectangular box connecting joint 1 and joint 2. This box is aligned with the x axis of joint 2 or in DH terms is rotated θ_1 about the z axis of the base. The model of the robot occupies the entire range of motion of this actual robot's link.

A rotation in joint 1 is represented by moving the subsequent links along the circumference of the red circle. Similarly, a rotation in joint 2 is represented by moving the subsequent links along the blue circle. The prismatic joint, due to the assumptions in joint limitations is allowed to travel in the negative z direction, allowing for the link to be above the blue and red circles. With this type of transformation, the tool is still pointed down, as would be expected as the prismatic joint is not revolving or reflecting in any way, it is merely translating along an axis.

C. *Puma 560*

The Puma 560 is another good robot to model. The DH parameters for the Puma 560 are shown in Table 3. A representation of the Puma 560 is shown in Figure 11 from reference [2] and the program's model of the robot is shown in Figure 12.

Table 3. DH Table for the Puma 560.

	Pris/Revol	Theta	d	a	alpha
1	Revolute	90.00	0.00	0.00	-90.00
2	Revolute	0.00	15.00	43.00	0.00
3	Revolute	90.00	0.00	-2.00	90.00
4	Revolute	0.00	43.00	0.00	-90.00
5	Revolute	45.00	0.00	0.00	90.00
6	Revolute	0.00	6.00	0.00	0.00

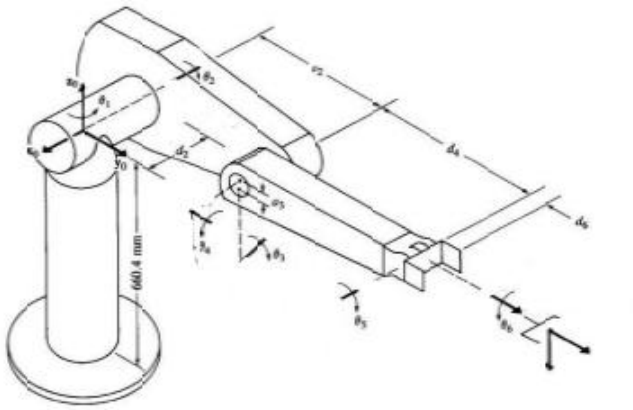

Fig. 11. The Puma 560 [2].

Comparing Figure 11 and Figure 12, the appearance of the robot again looks different. The first and second links are visualized a bit differently than in Figure 11.

Rotation in joint 1 causes the blue link and each subsequent link to rotate about the z axis. This is very easy to visualize in the model, but is more difficult in the drawing of the Puma 560 in Figure 11. The drawing's links do not seem to suggest a possible rotation about the base. On closer inspection of the figure however, the θ_1 is shown to convey this information. In the model, the red link suggests such a movement immediately.

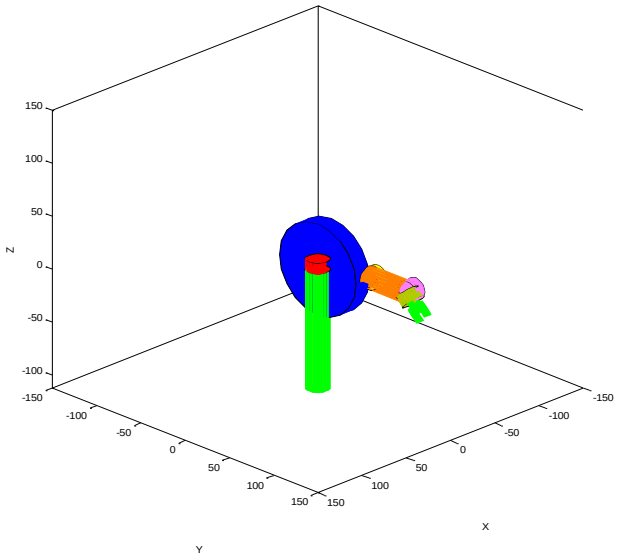

Fig. 12. Program model of the Puma 560.

An additional advantage to this particular robot's model is the yellow and pink revolute joints. For the drawing in Figure 11, these joints have to be described with the added axis and angle labels. In the model, these joints are shown as minimally sized links, yielding immediate information on their motion capabilities.

D. Puma 260

The DH parameters for the Puma 260 are shown in Table 4. An actual picture of the Puma 260 is shown in Figure 13 and the program's model of the robot is shown in Figure 14. The model is difficult to visualize in the default view, so Figure 14 shows the model rotated and zoomed in, a feature of the simulator program describe in Section III.

Table 4. DH Table for the Puma 260.

	Pris/Revol	Theta	d	a	alpha
1	Revolute	0.00	0.00	0.00	-90.00
2	Revolute	0.00	12.50	20.00	0.00
3	Revolute	0.00	0.00	-1.00	90.00
4	Revolute	0.00	20.00	0.00	-90.00
5	Revolute	0.00	0.00	0.00	90.00
6	Revolute	0.00	6.00	0.00	0.00

Comparing the two images, the 2nd link seems particularly different in appearance. This again is an artifact of the assumptions made about visualizing the links without link information. However, the link gives immediate information about the motion of its joint. Rotation about z axis of the blue joint will be reflected in the yellow and subsequent joints moving along the circumference of the blue cylinder.

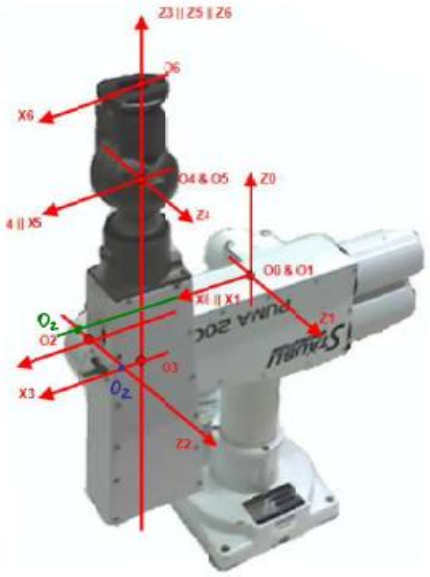

Fig. 13. The Puma 260.

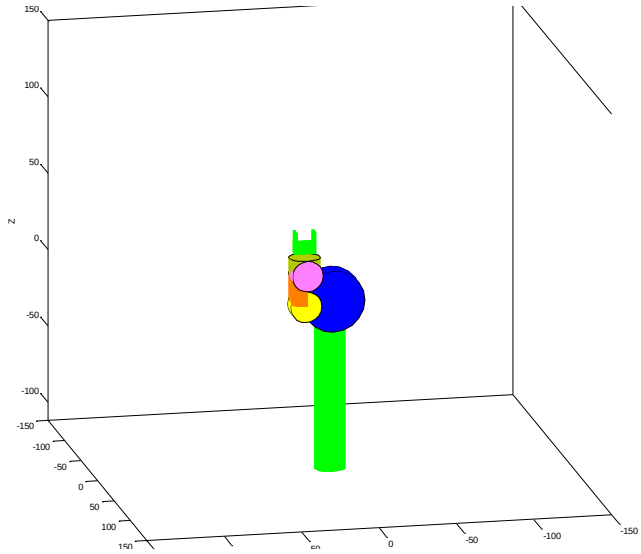

Fig. 14. Program model of the Puma 560. The model has been rotated and zoomed in for this image.

E. The Example R

The Example R Robot is shown in Figure 15 from reference [2]. The DH parameters for Example R are shown in Table 5 and the program model of the robot is shown in Figure 16.

Table 5. DH Table for the Example R Robot

	Pris/Revol	Theta	d	a	alpha
1	Revolute	0.00	50.00	0.00	-90.00
2	Prismatic	-90.00	25.00	10.00	90.00
3	Prismatic	90.00	25.00	0.00	90.00
4	Revolute	45.00	10.00	0.00	0.00

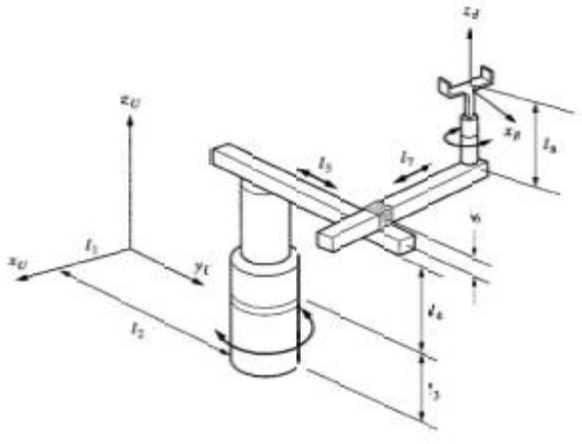

Fig. 15. The Example R Robot [2].

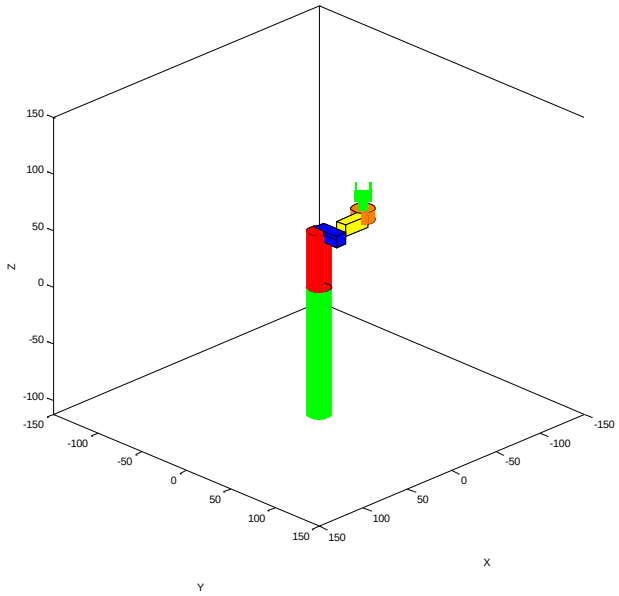

Fig. 16. Program model of the Example R Robot.

The Example R model seems to accurately reflect the appearance of the robot and depicts information about the movements of the joints. It is a good example of how links of prismatic joints are drawn.

V. CONCLUSION

The robot simulator program developed allows for the modeling of any robot given a set of DH parameters. Given the assumptions made in visualization of the links, the program was shown to adequately model several example robots. While the simulator may not always accurately visualize each link properly, the models tend to represent the links in ways that help the user understand the dynamic motion of the robot.

This program would be an excellent teaching device for the study of robots. The source code could be adapted easily to more accurately model any specific robot for advanced study.

Future recommendations for improvement of the simulator program include more robust error handling, the stability to handle multiple button presses when animating, and user definable range of motion limits to each joint. Also, the option of including more link information possibly in the form of a CAD file could be implemented for more accurate visualization. Additional work could be done to make the simulator backward compatible with previous versions of MATLAB. The program was written with MATLAB version 7.8.0.347 (R2009a), 64-bit edition. Some *uicontrol* functions used are not supported by older versions of MATLAB and can cause the program to have errors or not run.

ACKNOWLEDGMENT

A Lodes would like to thank Dr. G. Desouza and Y. Dong for their teaching efforts in the Introduction to Mechatronics and Robotic Vision class at the University of Missouri-Columbia. A Lodes would also like to thank Walla Walla Washington for the very informative 3D MATLAB Kinematic Model of the Puma 762.

REFERENCES

[1] Denavit, J., R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *ASME Journal of Applied Mechanics*, June 1955, 215-221.

[2] S. B. Niku, *Introduction to Robotics Analysis, Systems, Applications*, Upper Saddle River, NJ: Prentice Hall, 2001, pp 67-71, 92-94.

Adam Lodes (S'2006) received dual B.S. degrees in electrical engineering as well as a BS in mathematics at the University of Missouri-Columbia in 2007 where he is currently working toward a Ph.D. degree in electrical engineering.

He is currently a Graduate Research Assistant for the Center for Physical and Power Electronics at the University of Missouri-Columbia. His research interests include air plasma sources, spectroscopy, plasma diagnostics, and pulse power systems.