GRAK: A VISUAL ROBOTICS TOOLBOX USING MATLAB GRAPHICAL USER INTERFACE

Sibel Kaplan Yıldırım[1], Hüseyin Canbolat[2]

[1]Computer Engineering Dept., Mersin University, Ciftlikkoy, Mersin, Turkey
[2]Electronics and Communication Eng. Dept. Yildirim Beyazit Univ., Ankara, Turkey
[1]sibel_izmir@yahoo.com, [2]huseyin.canbolat@gmail.com; hcanbolat@ybu.edu.tr

ABSTRACT

The aim of this work is to execute the commands of the Robotics Toolbox.Version 7 via a graphical user screen out of the MATLAB command editor. The Robotics toolbox has the following menu items: Homogenous Transformations, Kinematics, Path Generation, Dynamics and Robot Object. The designed visual toolbox creates a window for each command asking the required and optional parameters. The visualization is achieved via the MATLAB Graphical User Interface (GUI). The developed interface supplies a field for each argument of the command through which the parameter values can be entered. Therefore, the user does not need to memorize the required parameters of the command. Thus, the commands can be executed faster and more effective. An application of GRAK for robotics or mechatronics education is given. The intended future work for the study is to develop a stand alone robotics software, which allows the user to design and analyze any kind of robotic mechanism in a user friendly environment.

Keywords: Graphical User Interface, Visual Robotics Toolbox, MATLAB, Animation and Simulation, Programming Environment

1. INTRODUCTION

Several software packages are developed for simulating and programming robot manipulators [1-3]. Robotic systems are extensively used especially in automotive industry. In such cases, the data communication is essential for the robot manipulators in order to coordinate the tasks to be performed [2]. Estimation of the performance of the system can be effectively done in computer simulations in the design of coordinated robotic units. However, the researchers write their own codes for themselves and then set up a physical prototype to get the actual performance. Generally, the robot manipulators used in these simulations and experiments have similar characteristics and most of the time, the simulations are done in the MATLAB environment. These simulations are time consuming due to the definition of the simulation code again and again each time. A general purpose simulation package will help researchers to reduce the time.

There are several Robotics Toolbox attempts in the literature [4, 5]. In [4], a software package called Robotica for Mathematica (RfM) is presented. Although RfM is still available, new releases are no longer supported by the team. *The Robotics Toolbox*, designed in [5] is a MATLAB based working environment executed via the command window. The toolbox is used in robotics courses at most of the educational institutes [6, 7]. The toolbox was presented in [5], as a software package that allows a MATLAB user to create and manipulate fundamental data types used in robotics such as homogeneous transformations, quaternions and trajectories. The toolbox has inverse and forward kinematics, Jacobians for n-link serial robot manipulators. The forward and inverse dynamics functions can be employed in the toolbox, however the required parameter values, such as the dynamic terms in the inertia and other matrices should be estimated and entered by the user to the toolbox. Toolbox includes some of the well-known serial robot manipulators, such as PUMA560 manipulator. It is also possible to define a robot in the toolbox. But this definition requires a lot of work to be completely used in the toolbox. Especially, the parameters required by dynamics should be carefully computed and fed to the software. We found this process very tedious for most of the users. On the other hand the toolbox is available free of charge at

several sites including the Mathworks FTP site and http://www.petercorke.com/Home.html .

The original Robotics Toolbox was created in the MATLAB environment, which is used extensively among the engineering community [6, 8]. However, the user should know and supply all the arguments required by a command. Although there are help files to remind how the command is formed, in case of an error, the user may consume time for the correct format of the command. In this work, the aim is to provide a visual interface to the user. To the best of our knowledge there is no such visual interface for robotics. A similar visual interface was developed for Microwave studies [9]. In this paper, the Visual Robotics Toolbox (GRAK- Görsel Robotik Araç Kutusu) is presented.

MATLAB Graphical User Interface (GUI) is a platform that allows the end user to easily utilize the graphical applications of MATLAB interactively. MATLAB GUI applications are necessary, because most of the software packages are graphic based and the graphical applications are easy to use.

MATLAB GUI has three basic components: GUI window, GUI objects and GUI function Callback. GUI window contains all the used objects. It is the medium that the objects are placed. GUI objects are elements with a specific function, such as buttons, sliders, and axes. `Callback` is the most important component of GUI. `Callback` dictates an object what to do. In an m-function type graphical application all the objects should be given a function using `Callback` [9].

The codes in visual environment are object oriented. This makes the visual toolbox functional and easy to use. Once the visual toolbox is prepared, the user needs not to know MATLAB and MATLAB GUI commands. Everything can be done by use of menu items. The user should only enter the parameter values and press a button to get the result.

In this work, the Visual Robotic Toolbox (GRAK) is presented by examples. First the details of the GRAK design are presented and some sample applications follows. Then the limitations of GRAK and the future work to improve the software will be discussed in conclusion.

2. VISUAL ROBOTIC TOOLBOX (GRAK)

A main screen is used for the visual robotic toolbox (GRAK). The commands of the robotics toolbox are used through the main screen. The details for the main screen will be introduced in the design section and some examples will be presented.

Robotics Toolbox is a useful tool in the computation of kinematics, dynamics, path generation subjects of robotics area. Peter I. Corke created the first version of Robotics Toolbox in 1996 [5]. In this work, the Release6 (2001) is used. The Robotics Toolbox is a free software and available at the Mathworks company, which owns the MATLAB software, web site http://www.mathworks.com/. The Toolbox can manipulate serial robot manipulators and the commands can be modified by the user. This allows the user to create his own command using his favorite algorithm. Indeed, the modification can be done for almost all MATLAB commands. However, the company provides support only its official versions. On the other hand, the available Robotics Toolbox versions come "as is" and there is no support for it. However, one can easily access the latest versions at Peter Corke's official web site www.petercorke.com .

All commands of the Robotics Toolbox are carried in to GRAK. Along those commands, the `drivebot` command, which animates the robot in a simple 3D plot is used in some visualized command windows.

2.1 Design

The commands of Robotics Toolbox are listed as menu items in GRAK main window. The MATLAB Graphical User Interface (GUI) was utilized in developing the GRAK. Using the objects of GUI, a small window for each command and the associated `*.m` files are prepared. These are called through the menu items at the main GRAK window. Also a help menu is prepared for the usage of GRAK. A view of the main window is given in Figure 1.

Figure 1. A snapshot of the GRAK main window

Although GRAK uses MATLAB files and commands, the user does not need to use of any MATLAB commands. All operations are carried

through the GRAK main window and menu items. Additionally, a user is not required to know any MATLAB software to use the Visual Robotics Toolbox (GRAK) via the visual menu items and the associated command windows, which asks every parameter needed to execute the command. The user should only enter the appropriate values in the related place on the command window and press the execution button to get the result.

The GRAK has all the required *.m and command window files for each command supplied by the Robotics Tollbox. The following are the main menu items used in the designed toolbox:

- *Homogeneous transformations*
- *Trajectory generation*
- *Kinematics*
- *Dynamic*
- *Robot Object*
- *Help*

Also, the commands associated with these menus are listed under the tab of each menu.

2.2 Sample Applications

The usage of GRAK is illustrated in the following examples.

2.2.1. Rpy2tr (Roll/Pitch/Yaw) Command

Roll, Pitch and Yaw angles are defined as Φ radians around z-axis, θ radians around y-axis, and Ψ radians around x-axis, respectively, as shown in Figure 2.

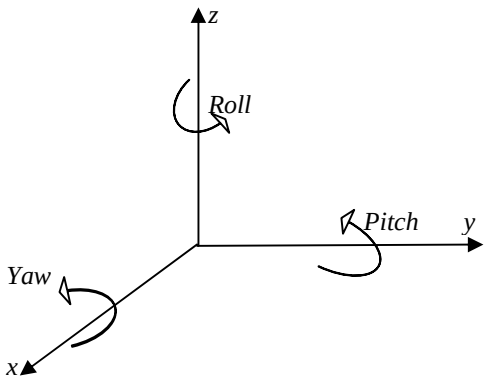

Figure 2. Roll / Pitch /Yaw Angles

The MATLAB notation of Roll-Pitch-Yaw angle computation is `rpy2tr` in the Robotics Toolbox. The command should have the following forms:

```
TR = rpy2tr([r p y])
```

or

```
TR = rpy2tr(r, p, y).
```

This command returns a homogenous transformation matrix determined by the given Roll/Pitch/Yaw (`r,p,y`) angles in the argument. A sample screen shot is seen in Figure 3. This command window can be opened through the *Homogenous Transformations* menu of the GRAK (Figure 1).

Figure 3. The screen shot of the GUI window for `rpy2tr` Command in GRAK

2.2.2. User Defined Robot

The Robotics Toolbox has some predefined robots, such as PUMA and 2-link planar robots. The toolbox is capable of creating a new robot defined by the user. The user should define the parameters for each link given in Table 1. The detailed definition of these parameters can be found in any robotics book such as [10]. The definition of a robot object is done through somewhat complicated commands in the Robotics Toolbox. The robot definition is made through a GUI window in GRAK. The user can assign as many links as he wishes for his user defined robot. The robot definition window of GRAK makes the basic definition of a new robot makes it easier. The user can employ the defined robot in the Robotics Toolbox and in GRAK, of course. To use his robot, the reader should save it under a certain name. Then the defined robot can be called by supplying the name of the robot.

However, the user defined robots has no dynamic parameters. We can find no clear definition of dynamic parameters for a user defined robot in the Robotics Toolbox. On the other hand, the details of predefined robot parameters demonstrate that the required dynamic analysis should be done manually, in order to give the exact values of the robot dynamic parameters. Due to this, the dynamic parameters for a robot object

definition are not included in the robot object window of GRAK. One of the challenging works for GRAK is that the complete definition of a user defined robot including the dynamic parameters besides the basic parameters for a Denavit-Hartenberg homogenous transformation. This study requires not only defining a parameter entry window, but also a complete module to get the dynamic equations using the given parameters and the geometry of each link. Such a module will be a very powerful dynamic tool for robot researchers.

Table 1: The link parameters of a robot

Parameter	Symbol	Short Definition
`alpha`	α	Joint twist angle
`A`	A	Link length
`theta`	θ	Joint angle
`D`	D	Distance between two joints
`sigma`	σ	Joint type (Revolute if zero, prismatic if nonzero)

The following figure shows the Robot Object window (Figure 4). To open this window, the Robot Object menu item should be selected in the main GRAK window (Figure 1).

Figure 4. The screenshot of Robot Object Constructor GUI window of GRAK

In this example, a 6-link robot is constructed and saved under the name `myrobot`. The user can add new links by clicking the "Add" button and delete a link using the "Delete" button. After the parameter values are entered the robot can be saved under the name typed in the "Robot Name" window by pressing the "Save Robot" button. Pressing the "Robot" button, starts another robot object definition, while the "New Values" button allows to change values of the parameters.

The simplified simulation of `myrobot` defined in Figure 4, is seen in Figure 5. The simulation window is opened, when the user presses the "Robot" button. This is a feature of the Robotics Toolbox, we have just employed the `drivebot` command to activate it. The simulation window is another challenging issue of future work. A better view of robots can be supplied in accordance with the dynamic parameter computation module described above. The geometry supplied to the dynamic module can be imported to the simulation window. The user can study the workspace and the physical limitations on the link using the simulation window.

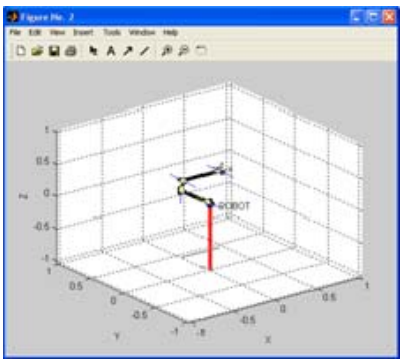

Figure 5. The Simplified Simulation of `myrobot`

2.2.3. Forward Kinematics

In the field of robotics, forward kinematics is defined as computing the position and orientation of the end-effector as a function of given joint variables. A joint variable is the angle θ for a revolute joint or the distance D for a prismatic joint. The definition of these variables is given in Table 2. A joint variable is symbolized with q_i and defined as follows:

$$q_i = \begin{cases} \theta_i & \textit{revolute} \\ d_i & \textit{prismatic} \end{cases}$$

In robotics toolbox, the command for forward kinematics is `fkine`, and its GRAK window is shown in Figure 6. The forward kinematics window can be opened through the *Kinematics* menu in the main GRAK window. Again, the user should choose his robot and enter the joint variables in the related areas of the window. Then he should press the "*Calculated*" button to get the resultant homogenous transformation matrix. He can repeat the calculations for another set of joint variables by pressing "*New Values*" button and then entering the new values.

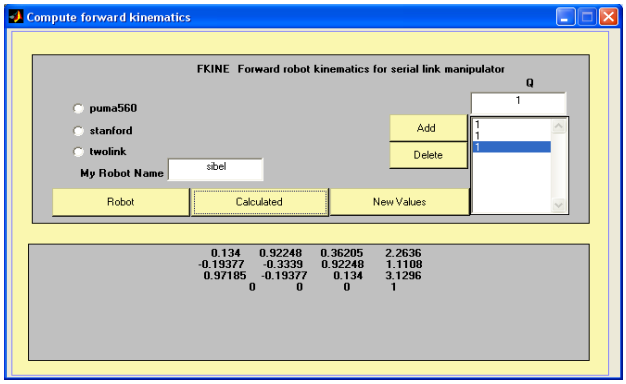

Figure 6. The GUI screenshot for forward kinematics command `fkine`

As it can be seen in Figure 6, the user can choose one of the predefined robots *(Puma560, Stanford, Twolink)* or a user defined robot by typing its name (`sibel` in Figure 6). After the required joint variables are entered and the "*Calculate*" button is pressed, the GRAK calls the corresponding Robotics Toolbox command to perform the computation and return the result to the user. In Figure 6, a three link robot is selected and the forward kinematics is computed. The orientation and position of the robot can be seen by pressing "Robot" button, as in the Robot Object Constructor window.

The MATLAB m-file for forward kinematics GRAK window is the following:

```
function
Calculated_Callback(hObject,
eventdata, handles)
   try
   l1=findobj(gcbf,'Tag','listbox1');
   str5=findobj(gcbf,'Tag','sonuc');
   str1 = get(l1, 'String');
   len = length(str1);
   sayac=len
   if len > 0
   for index = 1:len
   sonuc1=str1{sayac};
   sonuc1=str2num(sonuc1);
   q_d(1,index)=sonuc1;
   sayac=sayac-1 ;
   end
   end
   r1=findobj(gcbf,'Tag','puma560');
   r2=findobj(gcbf,'Tag','stanford');
   r3=findobj(gcbf,'Tag','twolink');
   r1=get(r1,'Value');
   r2=get(r2,'Value');
   r3=get(r3,'Value');
   if r1==1
   puma560;
   t=fkine(p560,q_d);
   elseif r2==1
   stanford;
   t=fkine(stanf,q_d);
   elseif r3==1
   twolink;
   t=fkine(t1,q_d);
   else
   robotname=findobj(gcbf,'Tag','Robo
   tName');
   robotname=get(robotname,'String');
   strname=strcat(robotname,'_tez');
   eval(strname);
   q_d=num2str(q_d)
   q_d=strcat('[',q_d,']');
   strson=strcat('fkine(',robotname,'
   ,',q_d, ')');
   t=eval(strson);
   end
   t=num2str(t);set(str5,'String',t);
   catch
   errordlg('Yanlış yada eksik değer
   girildi.','Hata
   Penceresi','modal');end
```

2.2.4. Cartesian Trajectory Generation

A trajectory is the set of points, which are traced by the a joint during its motion in a time interval. For Cartesian trajectory generation, the path of the end-effector is determined between the given initial (Point1) and final (Point2) points. The Robotics Toolbox command for Cartesian trajectory generation is `ctraj`. This command is visualized in GRAK as seen in Figure 7. To open this window, the user should go to the "Trajectory Generation" menu and select the related item under this menu. The user should enter the initial and final points of the trajectory along with start and finish time. Pressing "Calculated" button will generate the required trajectory and open the figure with subplots of first three joint trajectories (Figure 8). This visual item returns the graphics of the joint trajectories. In this first version of GRAK, `ctraj` is a bit limited. Indeed, it can be made more flexible by adding buttons in the window to select the subplots demonstrated in the figure.

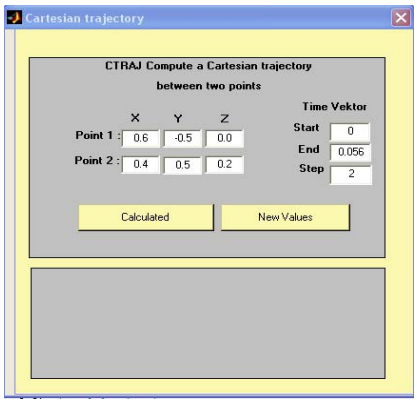

Figure 7. The screenshot for `ctraj` visual GUI

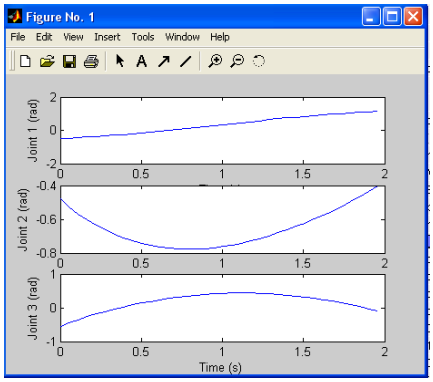

Figure 8. Joint trajectories computed by `ctraj`

3. A POSSIBLE APPLICATION

In the robotics (or mechatronics) education, the computations of several complex formulae, such as, the position of end-effector through forward kinematics and Jacobian calculations, should be used extensively. However, due to the computational burden and time consumption in performing these computations, the usage of computers is inevitable after the introductory level. Generally, people write their own codes for their applications or use the Robotics Toolbox for MATLAB. The Robotics Toolbox is a command base toolbox. Due to this, the student should unnecessarily learn the arguments of the commands. The difficulty is that the arguments should be known by the user, there is no direct help to the user. Mostly, the students learn only the compulsory arguments. They have no information for the optional arguments. For example, we could find no sample application of a new (or virtually created) robot design through the Robotics Toolbox, even though the toolbox has tools for design of one's own robot. But there are plenty of examples for other popular applications of predefined robots, such as PUMA, in the Toolbox. However, the design of a virtual robot is an excellent teaching tool for grasping the fundamental robotics principles. The robot design is also true for teaching mechatronics principles, since the robotics can be considered as a mature and broad application area for most of the mechatronics laws.

The use of Robotics Toolbox is limited for educational purposes and requires some preliminary study for use in a laboratory. The GRAK can be used easily for such an application, because all the commands are visualized with all required and optional arguments. The preliminary work for GRAK is much shorter than Robotics Toolbox itself. Thus the student can focus on understanding the basic principles instead of memorizing the parameters (arguments) of the commands.

4. RESULTS AND DISCUSSION

This work contains an extended application of MATLAB GUI to make the Robotics Toolbox a graphics based application. In this respect, our work along with [9] will provide an important start point for visualizing the MATLAB Toolboxes.

During our study, it was seen that the Robotics Toolbox are used only in the introductory level although there are some advanced tools in the Robotics Toolbox, such as dynamic analysis. However this feature requires the full derivation of the robot dynamics, which is a time consuming process. Therefore the use of dynamics facility in the Toolbox is limited to the predefined Puma560, Twolink and Stanford robots. The robot object construction feature of the toolbox is hard to carry in MATLAB command window. We think that due to this people rarely employed their user defined robots when using the toolbox. Another interesting result is that we could find no GUI application regarding the Robotics Toolbox. Even though GRAK is not fully mature yet, improvements will make the robot education and research easier. Although GRAK requires no exact memorization of commands with all required parameters, its GUI windows provides spaces for the required parameters. A user can easily learn which

parameters are required for a command in a short time by using GRAK.

Our study aimed to make the Robotics Toolbox is a user-friendly toolbox through the GUI windows prepared for each command. This was succeeded for most of the toolbox features. Some features requires extra programming out of MATLAB, and some of them were hard to fully visualize in the limited time of the study. Also we could find no work, which creates its own robot and dynamically analyze it through the Robotics Toolbox. We think that the command for robot creation is in fact a long process, which results a very limited robot object at hand. Due to this, people are discouraged to create and analyze their robots. For this purpose, a robot object constructor window is prepared in GRAK. This GUI window makes the robot construction very easy.

5. CONCLUSIONS

In this paper, a MATLAB GUI based visual robotics toolbox called GRAK is presented. GRAK is based on the Robotics Toolbox for MATLAB [5]. The aim for preparing GRAK is to provide a general simulation and analysis environment in robotics education and research. In present form, GRAK is far from providing a general environment. This is partly, because of our limited time to develop GRAK. Also original Robotics Toolbox has also its own limitations. Later versions of Robotics Toolbox are available on Internet free of charge. However, as an example the robot object construction is not practical in all versions. In GRAK the robot construction is easier. The user can create his robot and make some basic analyses on it using GRAK. However dynamic analysis is still limited only for the predefined robots in the toolbox in GRAK. This is inherited from the Robotics Toolbox. Nevertheless, people generally used the predefined robots of Robotics Toolbox instead of creating their own robots.

Extending the dynamical capabilities of created robots needs sophisticated software that can analyze the geometry and structure of the links. The development of such software is in our future plans.

In GRAK, every command has its own GUI screen, through which the command parameters are entered. Then the required command is called by pressing a button. The user should have no prior knowledge of MATLAB commands and the structure of the Robotics Toolbox commands. In some commands, a simple animation tool is used via the `drivebot` command of the toolbox. This simple animation provides a visual environment for physical structure of the robot. But the robot links are represented by straight lines in the screen. We think that this should be improved by employing better graphical tools.

Future plans for GRAK include the following:

1. Extending the capabilities of GRAK for any mechatronic system.
2. Development of Robot Object Constructor software.
3. Development of better animation tool.
4. Extension of visual GUI application to other MATLAB toolboxes.

An object-oriented language based functional code structure can be developed and integrated in MATLAB. This behaves as a DLL library and users can develop their GUI applications using the object-oriented functional code features. Present GUI objects may not be sufficient for developing visual toolboxes. Developing some new features in GUI makes the development of visual toolboxes easy.

REFERENCES

[1] R. Gourdeau, "Object Oriented Programming for Robotic Manipulator Simulation", *IEEE Robotics and Automation Magazine,* Vol. 4(3), pp. 21-29, 1997.

[2] F. Halsall, *Data Communications, Computer Networks and Open Systems*, 3rd Ed., Addison-Wesley, 1992.

[3] J.N. Pires and J.M.G. Sá da Costa, "Object Oriented and Distributed Approach for Programming Robotic Manufacturing Cells", *IFAC Journal on Robotics and Computer Integrated Manufacturing*, 1999.

[4] J.F.Nethery and M.W Spong,. "Robotica: A Mathematica Package for Robot Analysis", *IEEE Robotics and Automation Magazine*, vol. 1(1): pp. 13-20, 1994.

[5] P.I. Corke , "A robotics toolbox for MATLAB", *IEEE Robotics and Automation Magazine*, vol. 3(1), pp. 24-32, 1996.

[6] *MATLAB Users Guide*, The MathWorks Inc., USA, http://www.mathworks.com.
[7] Peter Corke, *Robotics Toolbox For MATLAB* (Release 6), April 2001.

[8] *User Reference For MATLAB* Ver.6.5, Mathworks Inc., 2003.

[9] M.S. Vural, MATLAB Grafiksel Arabirimi Yardımıyla "Görsel Mikrodalga Araç Kutusu" Tasarımı (Design of a "Visual Microwave Toolbox" via MATLAB Graphical User Interface) , MSc Thesis, Mersin University, Mersin Turkey, 2005.

[10] K.S. Fu, R.C. Gonzalez and C.S.G. Lee, *Robotics Control, Sensing,Vision, and Intelligence,* New York, McGrav-Hill Book Company, 1987.