Virtual Robots Module: An effective visualization tool for Robotics Toolbox

R. Sadanand
Department of Mechanical Engineering
Indian Institute of Technology Delhi
New Delhi
ratansadan@gmail.com

R. G. Chittawadigi
Department of Mechanical Engineering
Amrita School of Engineering
Bengaluru
rg_chittawadigi@blr.amrita.edu

R. P. Joshi
Department of Mechanical Engineering
Indian Institute of Technology Delhi
New Delhi
ravi2008joshi@gmail.com

S. K. Saha
Department of Mechanical Engineering
Indian Institute of Technology Delhi
New Delhi
saha@mech.iitd.ac.in

ABSTRACT

An introductory level robotics course mainly comprises the topics like geometry, kinematics, and dynamics of serial-chain robots. The description of the robot geometry using the Denavit-Hartenberg parameters and the kinematic and dynamic analyses require advanced mathematical concepts and are computationally intensive for robots with higher degrees-of-freedom. This calls for the use of robotics learning software, which would effectively aid the instructor to explain the concepts lucidly, and help the students in analyzing the mechanics of the robot. Robotics Toolbox is one such commonly used software, which is a collection of MATLAB-based functions that support various dedicated mathematical operations required in mechanical analysis of robots. RoboAnalyzer is another attempt towards the same goal, which focuses on the learning of robotics concepts from the physics of the robot motion. In this paper the integration of the Virtual Robots Module of RoboAnalyzer with the Robotics Toolbox is presented. With multiple number of industrial robot models, the Virtual Robots Module acts as an effective visualization add-in for the analysis performed using the Robotics Toolbox. The proposed visualization add-in can be used from software like MATLAB, MS-Excel, etc. The Virtual Robots Module allows improved visualization and easy simulation of industrial robot models for robotics research and education.

Categories and Subject Descriptors

I.2.9 [**Robotics**]: Kinematics and dynamics, I.3.8 [**Computer Graphics**]: Applications

General Terms

Experimentation, Verification.

Keywords

Robot simulation, robot visualization, robotics toolbox.

1. INTRODUCTION

Robotics based courses are now being introduced even in the undergraduate level curriculum. Majority of the introductory level courses on robotics emphasize on the mechanics of serial-chain robots due to their extensive applications in industry and research. The modelling and analysis of serial-chain robots involves topics that are not intuitive to teach or learn. Understanding the geometry, kinematics and dynamics of serial robots can be made easy and intuitive with the use of robotics teaching software. This would aid the teachers in explaining the concepts in robot mechanics with better clarity. The visualizations and simulations can assist the students in acquiring a better understanding of the concepts and in verifying the results of the simulation. The ability of robotics teaching software to illustrate the concepts in robot geometry and mechanics, using visualizations, is a crucial factor in making it an effective learning aid. At the fundamental level, robot mechanics involves concepts from vector mechanics, coordinate transformations, matrices, etc. Effective visualization of the same, along with the robot model and its motion, can complement an introductory level robotics course, as well as robot simulation used in research to a great extent.

A number of robotics teaching aids are being used nowadays. Robotics Toolbox [1, 2] using MATLAB is a very popular teaching and simulation tool that is being used in robotics courses. It is a collection of MATLAB-based functions dedicated for the mathematical operations and functions used in robot mechanics (e.g. homogeneous transformation matrices, trajectory generation, manipulator Jacobian, etc.). It allows the users to model, simulate, and analyze serial-chain robots using readymade functions and exploit the computational capabilities of MATLAB for further simulation and analysis. Apart from serial-chain robots, it also has modules for mobile robotics and path planning. RoboAnalyzer [3-5] is another attempt at robotics teaching software that allows easy visualization and analysis of kinematics and dynamics of serial-chain robots of various degrees-of-freedom.

Note here that the Robotics Toolbox uses the default plotting environment of MATLAB for the visualization of robot models and simulation of robot motion. By default, the robot models are rendered as schematic skeleton models only. Although the Robotics Toolbox supports rendering of robots when the STL files of the links are available, the method is not intuitive and requires the user to configure the robot geometry and graphical rendering. An effective alternative would be to have a separate visualization module, which can display readymade as well as skeleton robot models with better visualization capabilities.

AIR'15, July 02 - 04, 2015, Goa, India
©2015 ACM. ISBN 978-1-4503-3356-6/15/07..$15.00
DOI: http://dx.doi.org/10.1145/2783449.2783475

In this paper, an implementation of Virtual Robots Module of RoboAnalyzer is proposed as a visualization add-in for Robotics Toolbox. The proposed add-in will help in visualization and simulation of six-axis serial-chain industrial robots using the computational capabilities of Robotics Toolbox in MATLAB, as well as other standalone applications like Microsoft Excel. The significance of the add-in lies in the fact that it eliminates the time spent on modelling and rendering a robot model for simulation and research purposes. It aids students and researchers to visualize the simulation results of commonly used industrial robots with ease. The computational capabilities of Robotics Toolbox and the graphical environment of Virtual Robots Module are integrated into a single MATLAB class. An overview of the features and capabilities of Robotics Toolbox and Virtual Robots Module of RoboAnalyzer are given in Section 2 and Section 3, respectively. Section 4 discusses on how the Virtual Robots Module has been integrated with the Robotics Toolbox as an effective visualization add-in. The use of Virtual Robots Module from other standalone applications like Microsoft Excel for robot visualization is explained in Section 5. Finally the conclusions are given in Section 6.

2. ROBOTICS TOOLBOX

The Robotics Toolbox [1] is a computational tool for analysis of mechanics of serial manipulators. Apart from serial-chain robots, the latest version also supports analysis and path planning of mobile robots. It is basically a collection of MATLAB-based functions that support the various mathematical constructs which are pertinent to the kinematic and dynamic analyses of robots. A related Machine Vision Toolbox [2] is also available to aid the image processing and visual servoing in robotics. The Robotics Toolbox can be used for computations related to the following topics in robotics - homogeneous transformations, differential motion, trajectory generation, mechanics of serial link manipulators, path planning and localization of mobile robots, and graphical rendering of results. The interfaces with other robot simulation packages like V-REP [6], ROS [7], etc. are also available. The integration of Robotics Toolbox and V-REP is implemented in the latest available version of the former.

Robotics Toolbox uses the object oriented programming approach for creating the software counterpart of a serial-link robot. A 'SerialLink' class with suitable methods and properties is used to represent any serial-chain robot. The robot geometry is described using the widely accepted convention of Denavit and Hartenberg (DH) parameters [8]. A separate 'Link' class is used to represent a robot link and its associated properties. A number of 'Link' objects can then be combined to create a 'SerialLink' object. The 'Link' objects can be created by specifying the DH parameters of the robot.

A visualization of the schematic skeleton model of KUKA KR 5 Arc robot using the Robotics Toolbox is shown in Figure 1. A teach pendant for joint-level jogging is also shown in the plot window, which allows the user to change each of the joint angles individually. A 'SerialLink' object can be created once the DH parameters of the robot are known.

2.1 Some important functions

The relevant functions, classes, and methods of the Robotics Toolbox, which are used to integrate the Virtual Robots Module of RoboAnalyzer as visualization add-in, are given in Table 1.

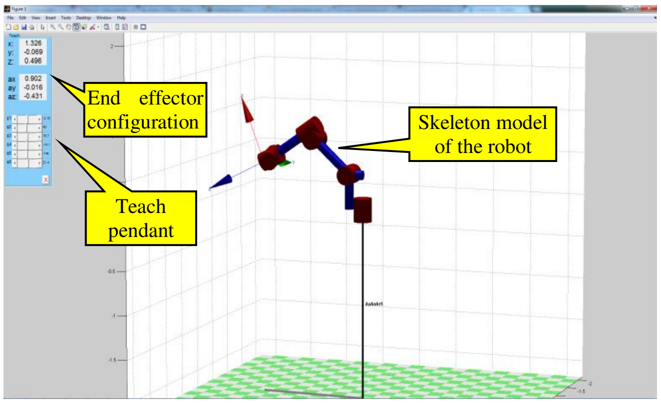


Figure 1. Skeleton Model of KUKA KR5 Arc in Robotics Toolbox

Table 1. Relevant classes and methods of Robotics Toolbox

Class / Method	**Description**
SerialLink	Class that represents a serial-chain robot
Link	Class that represents a robot link
SerialLink.theta, SerialLink.d, SerialLink.alpha, Serialink.a	Access the values of the DH parameters of the SerialLink object
SerialLink.fkine(q)	Returns the HTM[a] of the robot end-effector corresponding to joint state (q)
SerialLink.jacob0(q)	Returns the manipulator Jacobian corresponding to joint state (q)
transl(x, y, z)	HTM corresponding to a translation of the coordinate frame by x, y, and z along the X,Y, and Z axes
trotx(a), troty(b), trotz(c)	Returns the HTMs corresponding to the rotation of the coordinate frame by a given angle along the axes
jtraj(q_i, q_f)	Returns the joint trajectory between the initial (q_i) and final (q_f) joint states
tr2rpy(**H**)	Returns the Roll-Pitch-Yaw angles from the HTM (**H**)

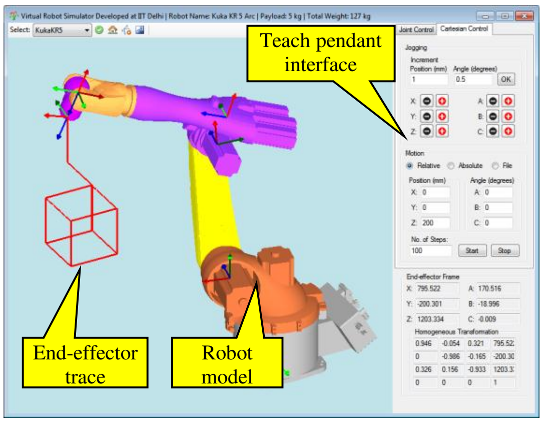
[a]HTM: Homogeneous Transformation Matrix

2.2 Visualization of Robots in Robotics Toolbox

The 'SerialLink' object of the Robotics toolbox has a 'plot' method, which uses the MATLAB's plot window to render the robot model, as shown in Figure 1. However, the 'plot' method of the toolbox has a limitation that it can render only a schematic model of the robot. The coordinate frames (DH frames) attached to the robot are not displayed by default, except for the frame at the end-effector. The other DH frames attached to the robot links have to be manually configured by the user. In order to fill these gaps in visualization aspect of Robotics Toolbox, the Virtual Robots Module of RoboAnalyzer is modified and integrated as an add-in for the toolbox. The details of the implementation are presented in Section 4.

3. VIRTUAL ROBOTS MODULE

The Virtual Robots Module (VRM) is a part of the RoboAnalyzer teaching software. It is primarily developed as a module of RoboAnalyzer, which would allow the user to visualize and simulate the virtual models of commonly used industrial robots. The VRM's capabilities include visualization of 18 industrial robots, joint level motion of robots, and straight line point-to-point motion of the robot in the Cartesian space. The user interface of VRM in RoboAnalyzer software is shown in Figure 2.


Virtual Robot Simulator Developed at IIT Delhi | Robot Name: Kuka KR 5 Arc | Payload: 5 kg | Total Weight: 127 kg
Select:
Joint Control
Cartesian Control
Teach pendant interface
Jogging
OK
Motion
Start
Stop
End-effector trace
Robot model

Select: KukaKR5

Jogging — Increment: Position (mm) 1; Angle (degrees) 0.5

Motion: [✓ Relative] Absolute File

Position (mm): X: 0, Y: 0, Z: 200; Angle (degrees): A: 0, B: 0, C: 0

No. of Steps: 100

End-effector Frame: X: 795.522, Y: -200.301, Z: 1203.334; A: 170.516, B: -18.996, C: -0.009

Homogeneous Transformation

0.946	-0.054	0.321	795.52
0	-0.986	-0.165	-200.30
0.326	0.156	-0.933	1203.3
0	0	0	1

Figure 2. User Interface of Virtual Robots Module in RoboAnalyzer

The primary objective of the VRM in RoboAnalyzer software is to provide a virtual environment, where the model of an actual industrial robot can be simulated using a teach pendant like user interface. The joint and Cartesian motion panels allow the motion of the robot model, similar to the jogging done using a robot teach pendant [5]. The VRM also updates and displays the HTM corresponding to the robot end-effector, as well as the end-effector configuration in terms of its position and Roll- Pitch-Yaw angles.

A MATLAB add-in was later released for the simulation of KUKA KR5 Arc robot model, using which the motion of the robot can be simulated after generating the joint trajectories and supplying them as input from MATLAB. This was possible by developing the Virtual Robots Module as a Component Object Model (COM) Server, which can interact with other software, which acts a COM Client.

3.1 Virtual Robots Module as COM Server

The Virtual Robots Module was implemented as a COM Server in Microsoft .NET Framework 2.0 using Visual C# programming language. OpenTK, an OpenGL wrapper for C# was used to generate the visualization of robot models using the CAD files of the robot links in STL format. At the software level, Object Oriented Programming was used to realize the visualization and simulation in the Virtual Robot Module. The schematic of the implementation in .NET and the C# classes are shown in Figure 3. The CAD files of the robot links are supplied in the STL format. The CAD files are rendered based on the robot geometry specified in an XML file. The classical method of the DH parameters [8] is used to describe the robot architecture.

Using the DLL (Dynamic Link Library) files of VRM and OpenTK, the COM Client can access the visualization and analysis capabilities of VRM and use it to display the robot models from the available files. In the typical style of object oriented programming, a ‘VirtualRobotServer’ object can be created by the COM clients, which will have the necessary

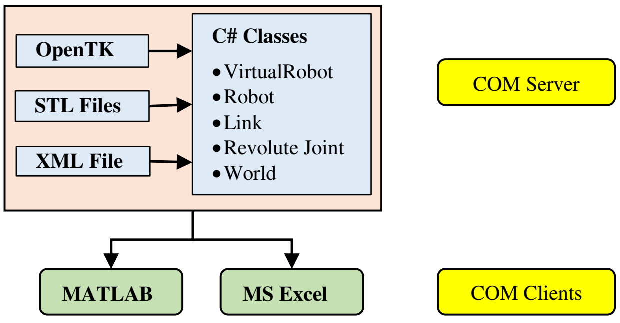


Figure 3. Implementation of Virtual Robots Module in C# .NET Framework

methods for loading and visualizing a robot. The methods of Virtual Robot Module that are exposed to the COM Clients like MATLAB and MS Excel are listed in Table 2 along with the relevant description.

Table 2. Methods of VirtualRobotServer Object

Method	Description of the Method
ShowAvailableRobots	List the robots whose files are available for visualization
LoadRobot	Load the parameters and properties of a required robot model on the server from the available ones
DisplayRobot	Display the robot model currently loaded in the server
UpdateRobot	Change the joint variables of the loaded robot and display the updated robot configuration
GetNoofJoints	Get the total no. of robot joints (Degrees of Freedom)
GetJointAngle	Get the value of the joint angle for a given robot joint
GetJointOffset	Get the value of the joint offset for a given robot joint
GetLinkLength	Get the value of the link length for the required robot joint
GetTwistAngle	Get the value of the twist angle for the required robot joint
GetJointVariableLimitMax	Access the lower limit of value for the required joint variable of the robot
GetJointVariableLimitMin	Access the upper limit of value for the required joint variable of the robot
GetEETransformation	Return the HTM of the robot end-effector for a given joint configuration
DisposeRobot	Delete the reference to the loaded robot object from the COM server
PlotTrace	Plots the trace of the end effector by default. Can be enabled/disabled as required.

The ‘VirtualRobotServer’ object can be created, following which, the methods listed in Table 2 can be used to load and visualize the available robots from any COM Client. The VRM server reported in this paper can visualize up to 18 six-axis serial-chain robots that are used in industrial applications. The robot geometry and specifications are described in an XML file, based on which the VRM server renders the robot model for visualization. New robot models can be added to the server by supplying the CAD files of the links in the STL format and mentioning the DH parameters in the XML file corresponding to the robot model.

The VRM Server (i.e., the 'VirtualRobotServer' object) can be called from any COM supported client and hence, the methods mentioned in Table 2 are generic. This is a powerful feature, as basic functions like display of robot model, calculation of HTMs, etc. can be done from commonly used applications like MS Excel, even in the absence of a computational tool like MATLAB.

4. INTEGRATION OF VRM WITH ROBOTICS TOOLBOX

In order to combine the computational features of Robotics Toolbox and the visualization capabilities of VRM, a MATLAB class was implemented using object oriented programming. A 'VRM_Robot' class was created in MATLAB, which can access both the Robotics Toolbox and VRM, thus integrating the capabilities of both the software. The computations involved in robot analysis can be performed using the methods of Robotics Toolbox mentioned in Table 1, while the visualization of robot and its motion can be done using the methods of VRM presented in Table 2. For the ease of the user, the 'VRM_Robot' class has readymade functions for creating the 'SerialLink' object of the robot, performing the forward and inverse kinematics, and joint and Cartesian space motion of the robot. The details of the implementation and use are given in the following subsections.

4.1 VRM_Robot class in MATLAB

The important methods and fields of the 'VRM_Robot' class in MATLAB that a user can access for robot analysis and simulation are shown in Table 3 and Table 4, respectively. After creating an instance of the 'VRM_Robot' class in MATLAB environment, the relevant fields and methods can be accessed to load, simulate and visualize the robot models. The schematic of the working is shown in Figure 4. Using MATLAB scripts, the functionalities of Robotics Toolbox and Virtual Robot Module are made available to the user.

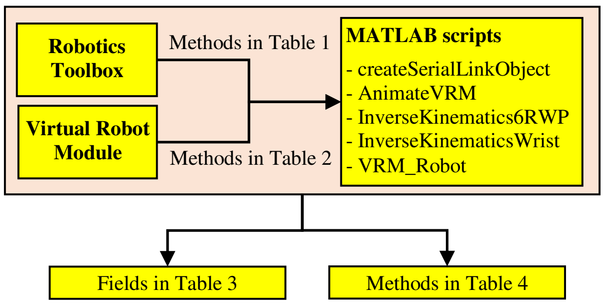


Figure 4. Schematics of implementation of VRM_Robot class in MATLAB

The use of 'VRM_Robot' class for analysis and application can be done in MATLAB. From the available robot models, the user can load one of the robot models using the 'LoadRobot' method, in which the XML file corresponding to the robot is read and a 'SerialLink' object from Robotics Toolbox is configured as per the DH parameters and the joint limits. Following this, the user may perform the forward and inverse kinematics, display the robot model, visualize the motion of the robot, and move the robot model in the joint space as well as Cartesian space. These can be done by making use of the methods mentioned in Table 3 and Table 4.

Table 3. Fields of VRM_Robots Class

Fields	Description
jointState	Current joint configuration of the robot
initialJointState	Initial joint configuration of the robot
finalJointState	Final joint configuration of the robot
jointPositionList	List of joint variable values for a trajectory
rtbSerialLinkObject	SerialLink object of Robotics Toolbox
vrmServer	VirtualRobotServer object of VRM

Table 4. Methods of VRM_Robot Class

Methods	Description
AvailableRobots	Display the robot models installed in VRM
CartesianMotionAbsolute	Move the end-effector of the robot in the Cartesian space with respect to the world coordinate frame
CartesianMotionRelative	Change the pose of the end-effector relative to its current pose.
DisplayRobot	Visualize the robot model
ForwardKinematics	Forward kinematics simulation of the robot between two joint configurations
InverseKinematics	Multiple solutions for the inverse kinematics of a wrist-partitioned robot
LoadRobot	Load an available robot model
MoveRobot	Move the robot in the joint space, between two specified joint configurations

4.2 Application of VRM_Robot class

The user interface of the VRM add-in for visualization is shown in Figure 5, in which a KUKA KR5 Arc robot is displayed.

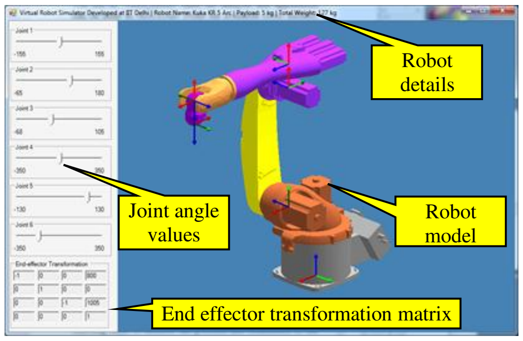


Figure 5. VRM Visualization add-in launched from MATLAB

The necessary data regarding the robot and the current HTM for the end-effector are in-built into the VRM add-in for the convenience of the users, enabling them to monitor the position and orientation of the robot end-effector. Important details of the robot such as the payload, weight, etc., are also indicated in the add-in. In order to give a brief idea about using the 'VRM_Robot' class, brief code snippets are given below for different types of analysis.

4.2.1 *Loading and Visualizing a robot*

An instance of the 'VRM_Robot' object is created first. Then, the 'LoadRobot' method is used to load one of the available robots, by passing the name of the robot as the parameter:

>> robotServer = VRM_Robot ( ) ;
>> robotServer.LoadRobot ('kukakr5')
>> robotServer.DisplayRobot ( )

The 'AvailableRobots' method may be used to query the robot models that are available for visualization and simulation. The loaded robot can be displayed in a required joint configuration by passing the joint angles as a row vector to the 'DisplayRobot' method. The angles must be specified in radians:

>> newRobotJointState = [120 60 45 0 70 120]*pi/180;
>> robotServer.DisplayRobot (newRobotJointState)

4.2.2 *Forward and Inverse Kinematics*

In the forward kinematics simulation, the HTM corresponding to the robot end-effector is to be determined, and this information is displayed in the visualization window. The forward kinematic simulation between any two joint configurations can be performed after loading a required robot:

>> initialConfig = [120 60 30 60 0 130]*pi/180
>> finalConfig   = [-120 90 -45 90 0 -130]*pi/180
>> timesteps = 150;
>> robotServer.ForwardKinematics (initialConfig, finalConfig, timesteps)

The joint trajectory is generated using the 'jtraj' method of Robotics Toolbox, mentioned in Table 1. It calculates and populates the 'jointPositionList' field of the 'VRM_Robot' class.

The inverse kinematics problem involves determining the joint angles of the robot for a given end-effector position and orientation. The proposed MATLAB class has the ability to calculate the multiple inverse kinematics solutions (up to 8) for a valid end-effector pose as per the method presented in [9]. The inverse kinematics maybe done as:

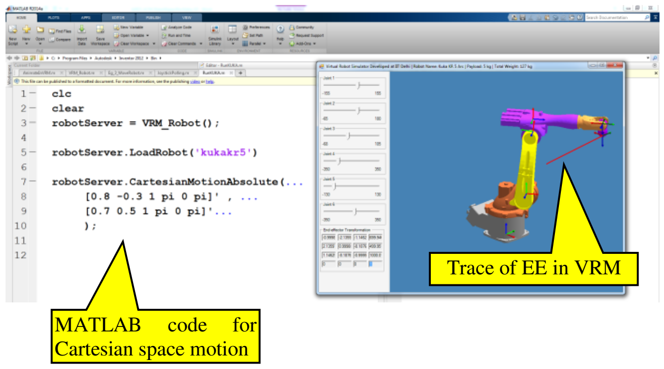
>> eeHTM = [-1, 0, 0, 0.8; 0, 1, 0 0; 0, 0, -1, 1.005;   0, 0, 0, 1] ;
>> robotServer. InverseKinematics (eeHTM)

4.2.3 *Cartesian space motion*

In Cartesian space motion of the robot, the motion is defined by specifying the change in the position and orientation of the end-effector with respect to time. In the 'VRM_Robot' class, the robot end-effector can be moved from one configuration to another, similar to the point-to-point (PTP) operation available in industrial robots. The PTP operation along a straight line can be implemented by calculating the joint angles at every intermediate point on the straight line path, using inverse kinematics. A smooth motion can be generated, which can then be visualized:

>> robotServer. CartesianMotionAbsolute ([0.8 -0.4 1 0 0 0]', [0.8 0.4 1 0 0 pi/2]')

The input is generally given as a column vector, whose first three elements describe the end-effector position, and the last three elements are the Roll-Pitch-Yaw angles describing the orientation as per [5]. The Cartesian space motion of the robot along with the MATLAB script is shown in Figure 6.



```
clc
clear
robotServer = VRM_Robot();

robotServer.LoadRobot('kukakr5')

robotServer.CartesianMotionAbsolute(...
     [0.8 -0.3 1 pi 0 pi]' , ...
     [0.7 0.5 1 pi 0 pi]'...
     );
```

Figure 6. Visualization of robot motion using VRM in Robotics Toolbox

4.3 Advantages of the VRM_Robot class

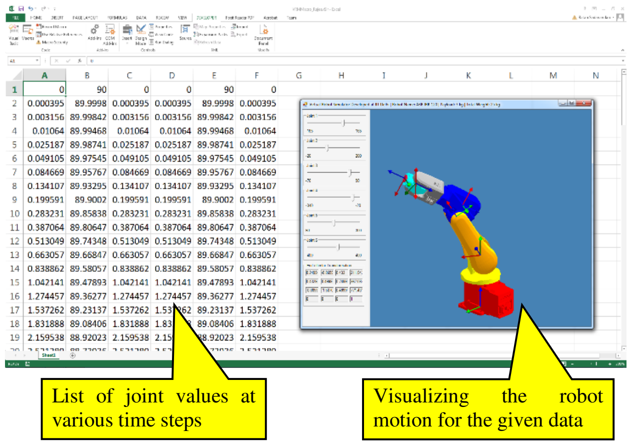
The primary advantage of having the 'VRM_Robot' class is that the user can easily load any serial robot model and get started with basic kinematic simulation and visualization. The 'SerialLink' object corresponding to the chosen robot model is automatically set-up when the 'VRM_Robot' class is instantiated, and the user can access the methods presented in Table 1 to perform the computations in Robotics Toolbox.

Having a readymade robot object eliminates the need to configure it manually. The VRM add-in can configure the 'SerialLink' object of the Robotics Toolbox by automatically reading the DH parameters and other robot properties from the XML files. This is an easy method, especially for beginner level students of robotics who can get started with readymade functions in Robotics Toolbox and use VRM for easy visualization.

5. INTEGRATION OF VRM WITH MICROSOFT EXCEL

VRM can also be used from other COM based clients like MS Excel, etc. The methods mentioned in Table 2 are accessible for such applications. When native programs are developed for research and educational purposes, emphasis is often placed on the analysis at hand, and visualization has to be programmed separately. In such cases, the VRM will serve as a useful add-in for visualization of the robot and its motion, since time need not be spent on programming the graphics and visualizations. For COM enabled platforms, the joint values of the robot can be directly sent to the VRM and the robot configuration can be visualized using the methods in Table 2.

One such example is shown in Figure 7, where the joint values for the ABB-IRB-120 robot is provided in the Excel file and the corresponding robot motion is animated. The robot joint trajectory was obtained externally and stored in the columns of MS Excel. A macro can be written in Excel VBA to connect to a VRM application. The macro reads the values of joint variables for each time step and sends to VRM to display the corresponding robot configuration. If the list of joint values is continuously read, then the motion of the robot can also be seen.



	A	B	C	D	E	F
1	0	90	0	0	90	0
2	0.000395	89.9998	0.000395	0.000395	89.9998	0.000395
3	0.003156	89.99842	0.003156	0.003156	89.99842	0.003156
4	0.01064	89.99468	0.01064	0.01064	89.99468	0.01064
5	0.025187	89.98741	0.025187	0.025187	89.98741	0.025187
6	0.049105	89.97545	0.049105	0.049105	89.97545	0.049105
7	0.084669	89.95767	0.084669	0.084669	89.95767	0.084669
8	0.134107	89.93295	0.134107	0.134107	89.93295	0.134107
9	0.199591	89.9002	0.199591	0.199591	89.9002	0.199591
10	0.283231	89.85838	0.283231	0.283231	89.85838	0.283231
11	0.387064	89.80647	0.387064	0.387064	89.80647	0.387064
12	0.513049	89.74348	0.513049	0.513049	89.74348	0.513049
13	0.663057	89.66847	0.663057	0.663057	89.66847	0.663057
14	0.838862	89.58057	0.838862	0.838862	89.58057	0.838862
15	1.042141	89.47893	1.042141	1.042141	89.47893	1.042141
16	1.274457	89.36277	1.274457	1.274457	89.36277	1.274457
17	1.537262	89.23137	1.537262	1.53 62	89.23137	1.537262
18	1.831888	89.08406	1.831888	1.83	89.08406	1.831888
19	2.159538	88.92023	2.159538	2.15	8.92023	2.159538

List of joint values at various time steps
Visualizing the robot motion for the given data
Sheet1

Figure 7. VRM used as a visualization module for ABB-IRB-120 robot from MS Excel VBA

A perceived advantage of the VRM in such cases is that, if a student or researcher has developed a program for analysis of six-axis serial chain robots without a visualization module, then the program can be made as a COM client and VRM can be used for visualizing the robot model.

6. CONCLUSION

A visualization add-in for serial-chain wrist-partitioned robots was presented in this paper. The visualizer has been integrated with the Robotics Toolbox of MATLAB. It is capable of animating the robot in different configurations, simulating the forward kinematics, joint-level motions, and motion of the robot end-effector in the Cartesian space. The Virtual Robots Module visualizer can also be used as a part of applications like MS Excel, etc., which can acts as a COM client. The proposed visualizer has the advantage that, it would allow students and researchers to obtain easy visualizations of the serial-chain robots and save precious time which is otherwise spent on programming the graphics and visualizations. Virtual Robots Module is available for free from http://www.roboanalyzer.com.

7. ACKNOWLEDGMENTS

The first and third authors would like to acknowledge the financial support received from the project "Adaptive Force Control of an Industrial Robot equipped with force-torque sensor", sponsored by the BRNS/BARC. The authors sincerely thank the developer of Robotics Tool Box (RTB), Prof. Peter Corke of Queensland University of Technology, for his encouragement to interface RoboAnalyzer with his RTB. His quick feedback on the contents of this paper is also acknowledged.

8. REFERENCES

[1] Corke, P., 1996. A Robotics Toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, vol. 3 (March 1996), 24-32.

[2] Corke, P., 2007. MATLAB Toolboxes: Robotics and Vision for Students and Teachers. *IEEE Robotics and Automation Magazine*, (December 2007) 16-17.

[3] Rajeevlochana, C. G. and Saha, S. K., 2011. RoboAnalyzer: 3D Model Based Robotic Learning Software. *International Conference on Multi Body Dynamics*, 3-13.

[4] Bahuguna, J., Chittawadigi, R. G., and Saha, S. K., 2013. Teaching and Learning of Robot Kinematics Using RoboAnalyzer Software. *International Conference on Advances in Robotics*.

[5] Sadanand, R., Chittawadigi, R. G., and Saha, S. K., 2013. Virtual Robot Simulation in RoboAnalyzer. *1st International and 16th National Conference on Machines and Mechanisms* (December 2013).

[6] Denavit, J., and Hartenberg, R. S., 1955. A Kinematic Notation for Lower pair mechanisms based on Matrices. *ASME Journal of Applied Mechanisms*, 22(2), 215 – 221.

[7] Freese, M., et al., 2000. Virtual robot experimentation platform v-rep: a versatile 3d robot simulator. *Simulation, Modeling, and Programming for Autonomous Robots,* Springer Berlin Heidelberg, 51-62.

[8] Quigley, M., et al., 2009. ROS: an open-source Robot Operating System. *ICRA Workshop on Open Source Software*, 3(3.2).

[9] Angeles, J., 2002. *Fundamentals of robotic mechanical systems*, vol. 2. New York, Springer-Verlag.