Virtual Robot Simulation in RoboAnalyzer

Ratan Sadanand O. M.
Department of Mechanical Engineering
National Institute of Technology Calicut
Kozhikode, 673601, India
ratan.sadan@gmail.com

Rajeevlochana G. Chittawadigi, Subir K. Saha
Department of Mechanical Engineering
Indian Institute of Technology Delhi
New Delhi, 110016, India
rajeevlochan.iitd@gmail.com, saha@mech.iitd.ac.in

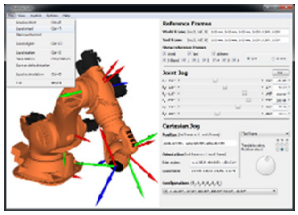
***Abstract*— Robotics is an important area not only in research and development but also from the perspective of industrial automation. As a result, increasing number of fundamental and advanced level robotics courses are being introduced in the undergraduate and postgraduate curricula, particularly in Mechanical and Electrical engineering streams. Robot kinematics is the cornerstone of such courses and it is equally challenging for teachers to teach as well as students to learn, as the concepts such as Denavit-Hartenberg (DH) parameters, robot kinematic and dynamic analyses, trajectory planning, etc. are difficult to understand. Various robotics learning software and tools have been developed by researchers around the world. One such attempt is made here to develop software called RoboAnalyzer. It can show animated DH parameters and performs forward and inverse kinematics, and dynamic analyses on serial robots. In this paper, a new module named "Virtual Robot Module" is reported which consists of 17 CAD models of commercially available industrial robots. Joint-level and Cartesian-level jogging can be performed on these robots. Relative and absolute motion of the end-effector can be achieved in the Cartesian space by controlling the position as well as the orientation of the end-effector. RoboAnalyzer software is freely available for academic purposes from http://www.roboanalyzer.com, and can be used by teachers and students almost instantly. It has a very easy to use interface and lets the user start learning the robotics concepts directly rather than learning CAD modeling, assembly modeling and then simulate a robot, as done using any commercial CAD software such as ADAMS, RecurDyn, Autodesk Inventor, etc.**

Keywords—Robot simulation, Robotics Learning Software, Cartesian motion planning, Jacobian control

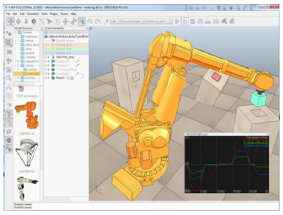
I. INTRODUCTION

With increasing number of applications in industrial and research sector, robotics has grown into a thrust area of research and development. Robotics courses, which were earlier introduced in post-graduate level, are now being offered to the under-graduate students so as to facilitate early entry into research field. Robotics being a multi-disciplinary field, the courseware usually draws topics from mechanics, control, programming, electronics, etc. However, the core of the subject lies in the kinematics and dynamics, which involve mathematical transformations based on the Denavit-Hartenberg (DH) [1] parameters and the geometric and kinematic relationships between the robot links, joints and the end-effector. It is imperative that the student must be thoroughly familiar with the mentioned concepts for gaining effective knowledge in advanced topics. The architecture or the geometric description of serial robot is generally based on the DH parameters and the mathematical transformations involved in position-motion description are done using the same. The related mathematical formulations of robot kinematics and dynamics draw heavily on linear algebra and vectors. It requires more than orthodox teaching methods and rote learning to make the concepts clear, since it involves understanding of the mathematics involved and relating it to the physical motion of the robot. Visualization of robot motion coupled with the underlying mathematics, through real-time demonstration will thus be an effective teaching tool. An overview of various robotics learning software is reported in [2], of which majority use skeleton models to represent serial robots. A skeleton model comprises of primitive shapes such as cylinders, cubes, etc., which are easy to model but do not convey the exact shape of the robot links. For effective and realistic visualization, CAD models of the robots are used in software such as RoKiSim [3] and v-rep [4], shown in Fig. 1(a) and (b), respectively. Apart from robotics education, CAD models of robots are also used in robot off-line programming software to simulate a robot program virtually and if found appropriate, run on actual robot. Examples of offline programming software are ROBOMOSP [5] and WorkSpace [6] shown in Fig. 1(c) and (d), respectively. In [2], RoboAnalyzer as robotics teaching and learning software was introduced with many features typically required in a first-level course on robotics offered to Mechanical and Electrical engineering students.

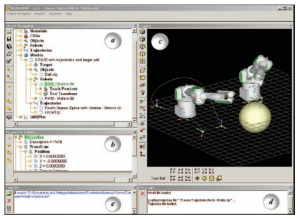
In this paper, a new module named "Virtual Robot Module" (VRM) is explained, which was added to RoboAnalyzer software. It has 17 CAD models of commercially available industrial robots. The VRM can perform joint-level and Cartesian-level jogging. It also can control the robot motion in the Cartesian space by providing position and orientation trajectories of the end-effector. This has been achieved by using Jacobian control. In Section II, the mathematical formulations required for the joint and Cartesian motion planning, and their implementation in the VRM are explained in Sections III and IV, respectively. Finally, the conclusions are given in Section V.



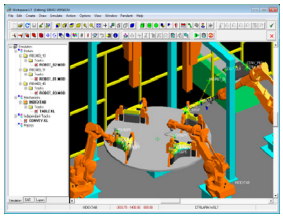
(a) RoKiSim [3]



(b) v-rep [4]



(c) ROBOMOSP [5]



(d) WorkSpace [6]

Fig. 1. Robotics learning and off-line programming software

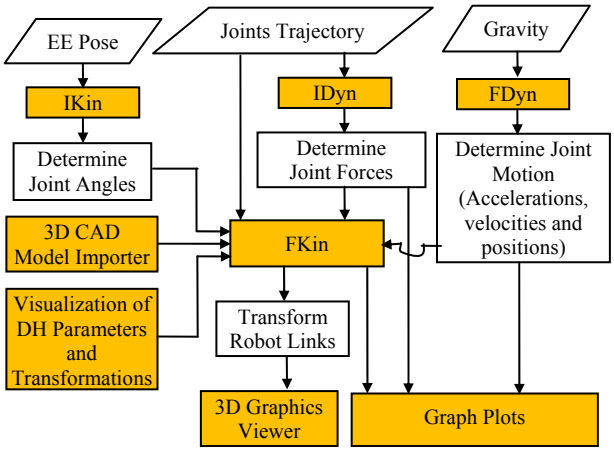
II. OVERVIEW

RoboAnalyzer is a 3D model based robotics learning software developed since 2009 using the concepts of Object Oriented Programming in Visual C# programming language. For the visualization of a robot and its motion in 3D environment, OpenGL, an open-source library, has been used through Tao Framework [7]. RoboAnalyzer has been developed in modules so that modification of existing modules and addition of new modules are easier and do not require major changes in the remaining modules. The module of Forward Kinematics (FKin) of serial robots with revolute joint was reported in [2], which uses skeleton models for the visualization of the robots. The analysis results were in the form of animation of the robot motion. The results of the analysis were also plotted as graphs using ZedGraph [8], an open-source plotting library in C#. Addition of the prismatic joints, Inverse Dynamics (IDyn) and Forward Dynamics (FDyn) analyses were reported in [9]. Modules on "Visualization of DH Parameters and Transformations", "3D CAD Model Importer" and "Inverse Kinematics" (IKin) were reported in [10]. The features of all the modules are briefed in Table I. The interactions of these modules are shown in Fig. 2(a). RoboAnalyzer software, in its present form, is shown in Fig. 2(b).

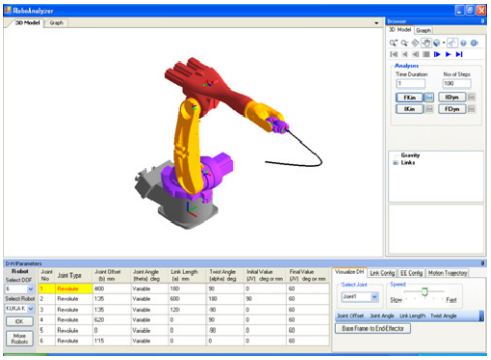
The position and orientation of a robot's end-effector are controlled or manipulated to perform automated industrial tasks like welding, machining, etc. Kinematics involves the position and velocity description of robot links. The standard convention is configuration (position and orientation) description using the DH transformation matrix and the velocity description using the Jacobian matrix. The topics of DH parameters, forward kinematics, Jacobian, etc. which are used to implement the Virtual Robot Module proposed in this paper are available in robotics text books [11-13]. However, an overview of these topics is presented in Appendix for the sake of continuity and to emphasis the nomenclature followed in the implementation of VRM.

TABLE I MODULES OF ROBOANALYZER AND THEIR FEATURES

Module	**Features**
DH Parameter Visualization and Transformations	It lets user select a joint and then select any DH parameter for which, a coordinate frame is drawn at the start configuration. Another coordinate frame is translated or rotated according to the selected DH parameter in the form of an animation. Visualization of transformation between two DH frames is done by drawing them on the link and by displaying the values of the homogeneous transformation matrix.
Forward Kinematics (FKin)	The module takes joints trajectory (i.e., initial and final values of each joint variable and type of joint-level trajectory) as input to determine the configuration of each robot link over the simulation time. The simulation results can be visualized by transforming each robot link for each time step in 3D Graphics Viewer.
Inverse Kinematics (IKin)	The module requires the pose or configuration, i.e., position and orientation, of the frame attached to the end-effector (EE) as input. It determines one or more solutions of the joint angles required to achieve the configuration required. The joint angles are passed on to FKin module for the visualization of the robot configuration. It can also perform animation from one solution of the IKin solution to any other.
Inverse Dynamics (IDyn)	It determines joint forces or torques required to achieve the given joint trajectories, when mass and inertia properties of each link and gravity acting on the robot are known. The motion can be viewed through FKin module and graphs can be plotted for the results.
Forward Dynamics (FDyn)	It determines the joint accelerations for gravity acting on the robot for a free-fall motion. The joint velocities and positional values are found using numerical integrator. The animation of the motion can be viewed through FKin module and the joint motion can be plotted.
3D CAD Model Importer	It imports the 3D CAD files (STL format) of standard robots such as KUKA KR5, PUMA 560 and Stanford Arm in the 3D Graphics Viewer and lets user perform different analyses.
3D Graphics Viewer	It displays skeleton model or 3D CAD model of the selected robot based on the DH parameters. It also shows the animation of the simulation results.
Graph Plots	It plots the results of different analyses.



(a) Existing modules



(b) Graphical User Interface (GUI)

Fig. 2. RoboAnalyzer Software

III. Robot Motion Planning

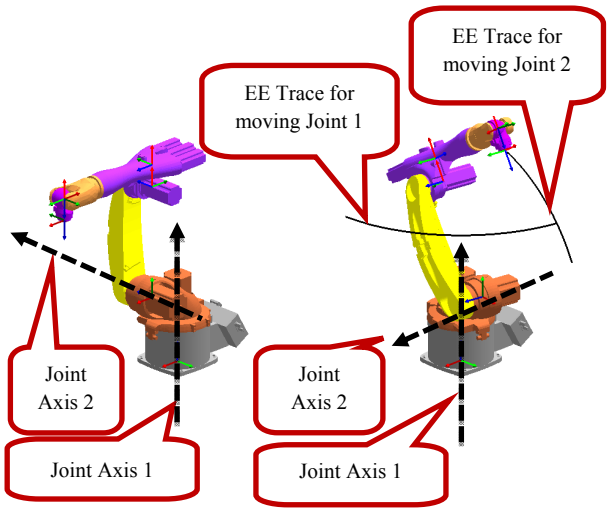
Industrial robots are generally used to perform various repetitive tasks such as pick-and-place operations, arc welding, spray painting, etc. In general, the instructions to the robot can be given in the following modes [13]:

- Lead Through or Teach Mode: In this mode, the robot's joints are moved using the robot's teach-pendent and a set of desired configuration, i.e., position and orientation, of the end-effector is taught to the robot controller. During playback, the controller moves the robot joints to the taught configurations repeatedly. Note that the incremental movement of the joints performed is known as 'joint-level jogging'. In the robots, the movement of the end-effector in the Cartesian space is also possible, which is referred to as 'Cartesian-level jogging'.
- Continuous Walk-through Mode: In this mode, the joints are moved simultaneously and the continuous motion is recorded by the controller. During the playback, the same motion is repeated.
- Software or Program Mode: This is an advanced mode in which the desired configurations can be entered into a robot program or the taught points can be inserted into a program. Different commands such as 'point-to-point', 'linear', and 'circular' can be used in these programs and the controller executes to achieve the desired robot motion.

Note that in all the three modes, motion at joint and Cartesian space are desired, which will be briefly reviewed below.

A. Joint Motion

In the joint-level motion or joint-space approach, the desired trajectory of the end-effector (EE) is specified in terms of the variation of joint position, velocity and acceleration. Motion of the end-effector is obtained by performing forward kinematics as explained in Appendix, i.e., using (4) and (5), as illustrated in Fig. 3 for Joints 1 and 2. The position and orientation data can be obtained from final homogenous transformation matrix (HTM) **T**. A smooth trajectory is obtained for all the joint variables based on the initial and final joint angles. Though it is computationally simpler, useful motion of the end-effector in the task space is difficult to visualize and the trajectory planning may be difficult in an environment with obstacles. Hence, a better approach to define the motion in task space or Cartesian space is required which is discussed next.



(a) Initial Position (b) Final position after moving Joint 1 followed by Joint 2

Fig. 3. Motion in joint space

B. Cartesian Motion

The tasks to be performed by a robot are generally defined in the Cartesian space where as the robot is controlled in joint-space. For any end-effector configuration, the joint angles required to achieve it can be determined by performing inverse kinematics [12]. The trajectory of the end-effector to be followed in the Cartesian space can be divided into a number of infinitesimal segments and then using the position and orientation required at each of these via-points, obtain the joint angle values using the inverse kinematic equations

for the robot. However, inverse kinematics may yield multiple solutions for the joint angle values. A difficulty that is inherent in this approach, along with the intense computations, is choosing the appropriate set of solutions for all the via-points such that the joint angles change smoothly with time without branching.

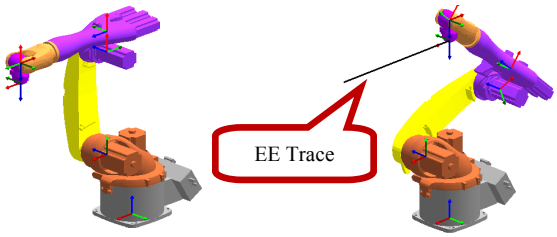
Hence, another method is used here [12] to perform Cartesian-level motion planning with a smooth change in joint variables. The method uses Jacobian-based control and thus the computations associated to the solution of non-linear algebraic equations arising in the inverse kinematics solutions are avoided. The Jacobian-based control is explained here for a straight line motion between two specified points. The straight line is divided into a number of infinitesimal segments and the inverse of the Jacobian matrix, introduced in Appendix, i.e., **J** of (11), is used to generate a continuous set of joint values using (13) which correspond to all the via-points. If the initial joint angle configuration $\boldsymbol{\theta}_0$ is known, then the initial position $\mathbf{p}_0$ and the orientation $\mathbf{Q}_0$ can be calculated from the forward kinematics equations (4) and (5) given in Appendix. The initial orientation in terms of the RPY angles, i.e., $\boldsymbol{\varphi}_0$, can be obtained by solving for them. From the final position ($\mathbf{p}_f$) and orientation in terms of RPY angles ($\boldsymbol{\varphi}_f$) given as input, the infinitesimal changes in the EE position, i.e., $\boldsymbol{\Delta p}$, and the infinitesimal change in the orientation, i.e., $\boldsymbol{\Delta\varphi}$, are determined by interpolating for the required number of via-points. The change in the joint angles, $\boldsymbol{\Delta\theta}_k$, required for the end-effector motion between $(k\text{-}1)^{st}$ and k^{th} via-points is determined by

$$\boldsymbol{\Delta\theta}_k = \mathbf{J}_k^{-1}\begin{bmatrix}\mathbf{L\Delta\varphi}\\ \boldsymbol{\Delta p}\end{bmatrix} \qquad (1)$$

Where $\mathbf{J}_k^{-1}$ is the inverse of the Jacobian matrix evaluated at the $(k\text{-}1)^{st}$ via point. The joint angles required to reach k^{th} via point is evaluated as

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \boldsymbol{\Delta\theta}_k \qquad (2)$$

Thus, a set of continuously varying joint angles required are obtained for all via-points, and a smooth motion can be seen. For a straight line motion with constant orientation, i.e., $\boldsymbol{\Delta\varphi}$=0, illustration of a KUKA KR5 robot is shown in Fig.4 (a) and (b).



(a) Initial Configuration (b) Final Configuration

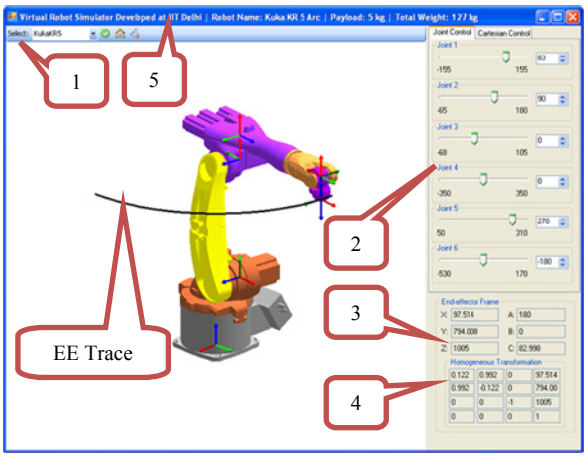
Fig.4. Cartesian motion of KUKA KR5 along a straight line

IV. VIRTUAL ROBOT MODULE

Virtual Robot Module (VRM) is a new module integrated with the RoboAnalyzer software. It allows visualization of 17 commercially available industrial robots using their CAD models. The VRM displays the selected robot model, its specification, DH frames attached to the various links, homogeneous transformation matrix of the EE, and the trace of the EE. The pose or the configuration of the EE is displayed in terms of the Cartesian coordinates and RPY angles with respect to the frame attached to the base-link. It is primarily developed as the motion planning module of RoboAnalyzer software. The following types of robot motions can be planned and visualized using the VRM proposed in this paper.

A. *Joint-level Jogging*

'Joint Control' pane of the VRM allows joint-level motion study of the robot models by movement of one joint at a time between the specified joint limits. This helps to understand the effect of moving the individual joints and is useful for understanding the workspace boundaries. The user interface required for joint-level jogging in the VRM is illustrated in Fig.5.

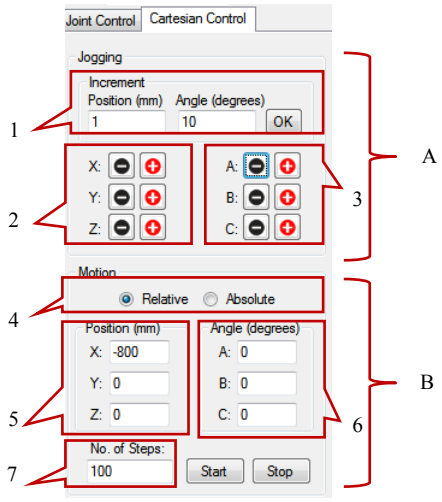


1. Robot Selection and other buttons.
2. Sliders to change joint angles within their limits.
3. EE position and orientation.
4. Homogeneous transformation matrix of the EE.
5. Robot details.

Fig. 5. User interface for 'Joint Control' in VRM

B. *Cartesian-level Jogging*

The 'Jogging' controls in the 'Cartesian Control' pane of the VRM allows changing the position and orientation of the EE (in terms of Cartesian coordinates and RPY angles, respectively) using buttons. Continuous motion of robot joints are achieved as the EE is jogged along the required direction or rotated by required angles. Jogging in the Cartesian-space is usually performed to teach configurations to the robot controller as it is intuitive and easy to manipulate the motion of the robot. The various available controls in the VRM for the Cartesian motion which has Cartesian-level jogging and Cartesian motion, described later in Section IV-C, are detailed in Fig.6, where *X, Y, Z* are the Cartesian coordinates in mm and *A, B, C* are the roll(ψ), pitch (θ), and yaw (φ) angles in degrees, respectively.



A. Controls for Cartesian-level Jogging

1. Increments in the position coordinates (*X, Y, Z* in mm) and RPY angles (*A, B, C* are roll(ψ), pitch (θ), and yaw (φ), respectively).
2. Buttons for position increments.
3. Buttons for RPY angle increments.

B. Controls for Cartesian Motion

4. Set the motion as Relative or Absolute
5. Required values for Position depending on Relative/Absolute motion required.
6. Required values for RPY angles depending on Relative/Absolute motion required.
7. Specify the number of steps or via-points between initial and final configurations.

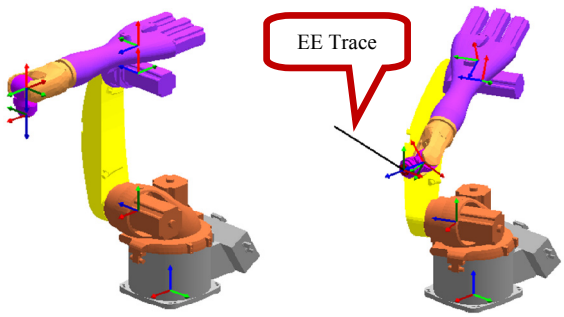
Fig. 6. User interface for 'Cartesian control' in VRM

C. Cartesian Motion

The VRM allows the EE to move from its initial configuration to the specified configuration, along a straight line in space. The final configuration specified maybe relative to the current position and orientation of the EE or with respect to the World frame attached to the base-link. They are explained below.

a) Relative:

It refers to specifying the final configuration of the EE with respect to its initial configuration. The increments desired in each of the position and orientation coordinates has to be supplied as an input. The VRM determines the via-points and calculates the increment in joint angles to be provided to the robot for the motion between any two consecutive via-points and updates the joint angles given by (2). For straight line motion with constant orientation, the *A, B, C* values are set to zeros as there is no relative change in the EE orientation. Similarly, for a pure rotation of the EE at the initial position, *X, Y, Z* values are set to zeros. As an illustration of straight line motion, the initial configuration of KUKA KR5 robot is shown in Fig.7 (a). The EE coordinates for the initial configuration, the desired changes in the coordinates and the actual coordinates of the final configuration, obtained using different number of via-points or steps are reported in Table II. The final configuration obtained using 500 via-points are shown in Fig. 7(b). Note that the values of the position and orientation coordinates of the EE for the final configuration reached do not match the desired coordinates. This is because (1) and (2) are valid for infinitesimal small change in the position and orientation between consecutive via-points. Hence, using a larger number of steps or via-points would result in reduced error upon reaching the final configuration. Moreover, control strategy for each coordinate (*X*, *Y*, *Z*, φ, θ and ψ) should be implemented to reduce the error at every via-point, which will be implemented in future.



(a) Initial Configuration (b) Final Configuration

Fig. 7. Straight line motion of KUKA KR5 robot in Cartesian space

TABLE II INITIAL AND FINAL CONFIGURATION OF KUKA KR5 ROBOT IN THE CARTESIAN SPACE

EE Coordinates	Initial Configuration	Change in coordinates	Final Configuration (Desired)	Final Configuration (Actual)			
				100 steps	200 steps	500 steps	1000 steps
X (mm)	800	-200	600	598.535	599.261	599.696	599.892
Y (mm)	0	200	200	201.818	200.38	200.38	200.188
Z (mm)	1005	-200	805	803.696	804.77	804.77	804.875
A (°)	180	-90	90	90.825	90.407	90.158	90.078
B (°)	0	0	0	0.734	0.363	0.145	0.073
C (°)	0	90	90	91.505	90.505	90.207	90.1

b) Absolute:

The final configuration of the EE is specified as the absolute coordinates of the position (*X, Y, Z*) and the orientation (RPY angles) in the World frame attached to the base-link of the robot. The increments required in the position and orientation coordinates are determined in the VRM using the difference of these coordinates in the initial and final configurations. Then, the procedure followed in "Relative" motion above is used to reach the final configuration.

D. Working of VRM

The methodology used to include CAD models of industrial robots in VRM is explained here. The CAD model of an industrial robot was imported in Autodesk Inventor software and the constraints were defined between the robot links. Using the methodology proposed in [14], the DH parameters of the robot were extracted and the DH frames were attached on each robot link using an Autodesk Inventor addin. CAD model of each robot link was then modified such that the DH frame attached to the link coincides with the part origin frame of the link. This is done to simplify the process of visualization and animation. The modified CAD model was then exported as STL (stereolithography) file. An XML (extensible Markup Language) file was created with the DH parameters, robot information and other details. The VRM module reads the XML file of the robot and generates the 3D CAD model in OpenGL using the STL files of the robot links. Users can add CAD models of any robot to VRM if CAD assemblies are available following the above approach.

V. CONCLUSIONS

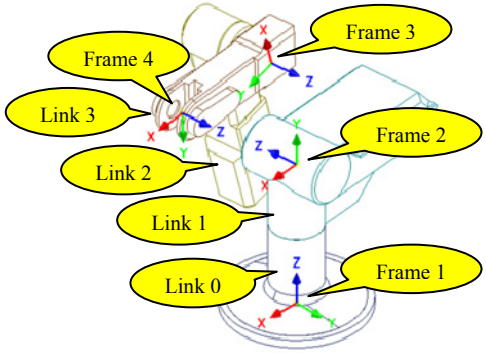
The "Virtual Robot Module" (VRM) for RoboAnalyzer is presented in the paper and its significance in classroom teaching is explained. A computationally simpler approach for Cartesian motion planning is implemented to achieve jogging and straight line motion in the Cartesian space. Trajectory planning with a good degree of accuracy is achieved. Further functionalities like teaching points using joystick and having Cartesian motion between them, interface to write robot programs such as VAL-II, circular trajectory planning, etc., will be implemented in the future versions of VRM module of RoboAnalyzer software. The latest version of the VRM module can be downloaded as Version 6.1 of the software from http://www.roboanalyzer.com. The readers are encouraged to use the same and give feedback to the authors for further improvement of the software.

APPENDIX

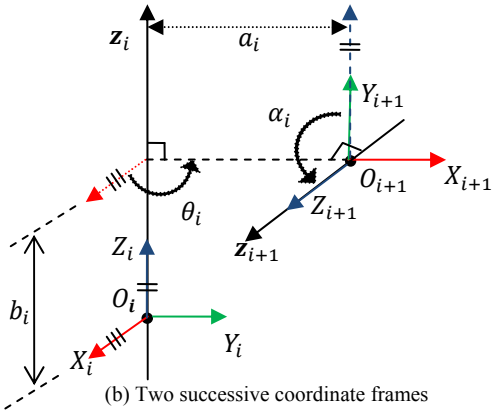
A. Denavit-Hartenberg(DH) Parameters

The robot architecture is described in terms of the Denavit-Hartenberg [1] parameters. A robot usually consists of a number of links connected by joints generally having one degree-of-freedom (DOF). The configuration of the end-effector (i.e., the coordinate frame attached to the end-effector) with respect to the World coordinate frame (the frame attached to the base-link) is obtained through a series of transformations.

These are based on the relative location of the coordinate frames (DH frames) attached to the links of the robot. The conventions used for attaching the DH frames are outlined in [11] and the description of a DH frame with respect to a previous DH frame is done using the four DH parameters, as illustrated in Fig. 3(a) and (b), and explained in Table III.



(a) Coordinate frames attached to links



(b) Two successive coordinate frames

Fig. 8. Denavit Hartenberg (DH) parameters

TABLE IIII DESCRIPTION OF DH PARAMETERS

Parameters	Description
Joint Offset (b_i)	Distance between X_i and X_{i+1} along Z_i
Joint Angle (θ_i)	Angle between X_i and X_{i+1} about Z_i
Link Length (a_i)	Distance between Z_i and Z_{i+1} along X_{i+1}
Twist Angle (α_i)	Angle between Z_i and Z_{i+1} about X_{i+1}

B. Forward Kinematics

Forward kinematics involves determining the position and orientation of the robot end-effector for a given values of joint variables. End-effector (EE) configuration can be obtained through the closure equations as described below:

- For a robot with n+1 links, connected through n joints, coordinate frames are attached to each link

as per the conventions followed in [11]. Frame 1 is attached to the base-link and Frame n+1 is attached to the n^{th} link, i.e., the end-effector.

- The DH parameters are assigned as described in Appendix A.
- The homogeneous transformation matrices $\mathbf{T}_1$, $\mathbf{T}_2$,.. $\mathbf{T}_i$,.. $\mathbf{T}_n$ are computed, where $\mathbf{T}_i$ stands for the transformation of Frame i+1 (attached to Link i) with respect to Frame i (attached to Link i-1), using the DH parameters assigned previously. $\mathbf{T}_i$ is given by:

$$\mathbf{T_i} = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & b_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

- The homogeneous transformation matrix of the end-effector frame with respect to the base-link is then obtained by post multiplication of the individual transformations in the order as

$$\mathbf{T} = \mathbf{T}_1\mathbf{T}_2\mathbf{T}_3...\mathbf{T}_n \quad (4)$$

This is the closure equation of the robot. Substituting (3) in (4) for the required number of links, the final homogeneous transformation matrix is obtained as

$$\mathbf{T} = \begin{bmatrix} \mathbf{Q} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \quad (5)$$

Where $\mathbf{Q}$ is the 3×3 orientation matrix of the EE, and $\mathbf{p}$ is the 3-dimensional vector that specifies the position of the EE (i.e., Frame n+1), both with respect to the base-link (Frame 1).

C. Orientation Description using RPY Angles

Roll-Pitch-Yaw (RPY) angles are a set of Euler angles used for orientation representation. If these angles are denoted with φ, θ and ψ, the array of RPY angles is denoted by the 3-dimensional vector $\boldsymbol{\varphi}$ as

$$\boldsymbol{\varphi} = [\ \varphi \quad \theta \quad \psi\ ]^T \quad (6)$$

The RPY angles are defined as the rotation of a frame with respect to the reference frame by an angle φ first about Z axis (yaw), followed by a rotation by an angle θ about Y axis (pitch), and finally a rotation by an angle ψ about X axis (roll). All rotations are made with respect to the frame attached to the base-link. For motion control of industrial robots, roll, pitch and yaw are denoted by angles A $(=\psi)$, B $(=\theta)$ and C $(=\varphi)$, respectively. The orientation matrix representing the resultant frame with respect to the fixed frame is then given by

$$\mathbf{Q}_{RPY} = \begin{bmatrix} C\varphi C\theta & C\varphi S\theta S\psi - S\theta C\psi & C\varphi S\theta C\psi + S\theta S\psi \\ S\varphi C\theta & S\varphi S\theta S\psi + C\varphi C\psi & S\varphi S\theta C\psi - C\theta S\psi \\ -S\theta & C\theta S\psi & C\theta C\psi \end{bmatrix} \quad (7)$$

Where $C\varphi = \cos\varphi$, $S\varphi = \sin\varphi$ and similarly for the other angles θ and ψ.

D. Jacobian

The manipulator Jacobian matrix relates the angular and linear velocities of the end-effector to the joint velocities [11, 12]. The Jacobian matrix depends on the current manipulator configuration given by the joint variables $\boldsymbol{\theta}$, i.e., $\boldsymbol{\theta} = [\theta_1\ \theta_2\ ...\ \theta_n]^T$. The position vector $\mathbf{p}$ and the direction cosine matrix (DCM) $\mathbf{Q}$ are functions of the joint variables. The angular velocity of the end-effector $\boldsymbol{\omega}_e$ can be correlated to the time rate of change in RPY angles and the end-effector's linear velocity $\mathbf{v}_e$ as the time rate of change of the Cartesian coordinates with respect to the base frame. The vectors $\boldsymbol{\omega}_e$ and $\mathbf{v}_e$ can be expressed in terms of rate of change of the joint variables, i.e., $\dot{\boldsymbol{\theta}}$ as

$$\boldsymbol{\omega}_e = \mathbf{J}_\omega \dot{\boldsymbol{\theta}} \quad (8)$$

$$\mathbf{v}_e = \mathbf{J}_v \dot{\boldsymbol{\theta}} \quad (9)$$

Where $\mathbf{J}_\omega$ and $\mathbf{J}_v$ are the $3\times n$ matrices relating the contribution of joint velocities $\dot{\boldsymbol{\theta}}$ to the angular velocity $\boldsymbol{\omega}_e$ and linear velocity $\mathbf{v}_e$, respectively. These matrices are functions of joint variables $\boldsymbol{\theta}$. Equations (8) and (9) can be expressed in a compact form

$$\mathbf{t}_e = \mathbf{J}\dot{\boldsymbol{\theta}} \quad (10)$$

Where, $\mathbf{t}_e = [\boldsymbol{\omega}_e^T\ \mathbf{v}_e^T]^T$ is 6-dimensional vector defined as the twist of the end-effector, whereas, $\mathbf{J} = [\mathbf{J}_\omega^T\ \mathbf{J}_v^T\ ]^T$ is the $6\times n$ matrix called the manipulator 'Jacobian'. With reference to Fig.9, for a manipulator or serial robot having all joints of revolute type, the manipulator Jacobian can be expressed as

$$\mathbf{J} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_n \\ \mathbf{e}_1\times\mathbf{a}_{1e} & \mathbf{e}_2\times\mathbf{a}_{2e} & \cdots & \mathbf{e}_n\times\mathbf{a}_{ne} \end{bmatrix} \quad (11)$$

Where $\mathbf{e}_i$ is the unit vector parallel to the axis of the i^{th} joint and $\mathbf{a}_{ie}$ is the vector drawn from the origin of Frame i to the origin of Frame $n+1$ attached to the end-effector. Vector $\mathbf{a}_{ie}$, for i = 1, 2,...n, can be computed as $\sum_{j=i}^{n} \mathbf{a}_j$. Note that the concept of manipulator Jacobian given by (11) is useful to find infinitesimally small change in position and orientation. For that, the constant relationship between the angular velocity of the end-effector, i.e., $\boldsymbol{\omega}_e$, and the rate of change of RPY angles $\dot{\boldsymbol{\varphi}} = [\dot{\varphi}\ \ \dot{\theta}\ \ \dot{\psi}]^T$ is given by [12] as

$$\boldsymbol{\omega}_e = \mathbf{L}\dot{\boldsymbol{\varphi}} \quad (12)$$

$$\text{Where } \mathbf{L} = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi & 0 \\ \sin\psi\cos\theta & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{bmatrix}$$

From (8), (9) and (12), the changes in the RPY angles, i.e., $\Delta\boldsymbol{\varphi}$, and the position of the end-effector, i.e., $\Delta\mathbf{p}$ can be related to the change in joint angles, i.e., $\Delta\boldsymbol{\theta}$ as

$$\begin{bmatrix} \mathbf{L}\Delta\boldsymbol{\varphi} \\ \Delta\mathbf{p} \end{bmatrix} = \mathbf{J}\Delta\boldsymbol{\theta} = \begin{bmatrix} \mathbf{J}_\omega \\ \mathbf{J}_v \end{bmatrix} \Delta\boldsymbol{\theta} \quad (13)$$

The expression in (13) is used in Section III-B for Cartesian Motion Planning of serial robots.

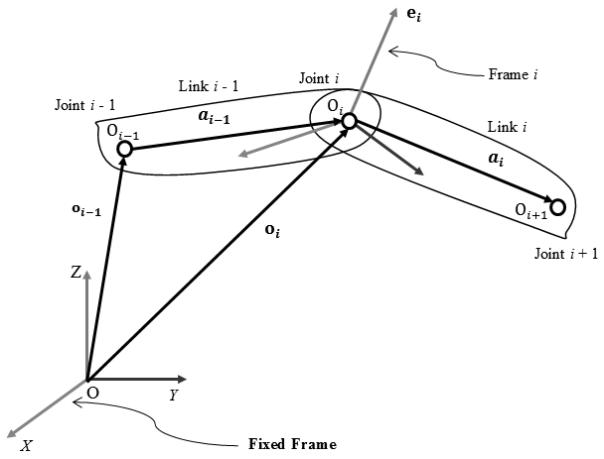


Fig. 9. Coupled links of a robot

REFERENCES

[1] J. Denavit and R. S. Hartenberg, "A Kinematic Notation for Lower pair mechanisms based on Matrices," ASME Journal of Applied Mechanisms, Vol. 22, No. 2, pp. 215 - 221, 1955.

[2] C. G. Rajeevlochana and S. K. Saha, ""RoboAnalyzer: 3D model based robotics learning software," International Conference on Multibody Dynamics, pp. 3 – 13, 2011.

[3] RoKiSim - http://www.parallemic.org/RoKiSim.html.

[4] v-rep - http://www.coppeliarobotics.com

[5] A. Jaramillo-Botero, A. Matta-Gomez, J. F. Correa-Caicedo, and W. Perea-Castro, "ROBOMOSP," IEEE Robotics & Automation Magazine, Vol.13, No.4, pp.62-73, 2006.

[6] WorkSpace - http://www.workspacelt.com.

[7] Tao Framework - http://sourceforge.net/projects/taoframework.

[8] ZedGraph Library - http://zedgraph.sourceforge.net/index.html.

[9] C. G. Rajeevlochana, A. Jain, S. V. Shah, and S. K. Saha, "Recursive Robot Dynamics in RoboAnalyzer," 15th National Conference on Machines and Mechanisms, pp. 482-490, 2011.

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

[11] S. K. Saha, Introduction To Robotics, Tata McGraw-Hill, 2008.

[12] R. Manseur , Robot Modelling and Kinematics, Thomson-Delmar Learning, 2006.

[13] S. B. Niku, Introduction to Robotics: Analysis, Systems, Prentice Hall, 2010.

[14] C. G. Rajeevlochana, S. K. Saha, and S. Kumar, "Automatic Extraction of DH Parameters of Serial Manipulators using Line Geometry," The 2nd Joint International Conference on Multibody System Dynamics, 2012.