17th International Research/Expert Conference
"Trends in the Development of Machinery and Associated Technology"
TMT 2013, Istanbul, Turkey, 10-11 September 2013

SIMULATION OF A 5-AXIS RV-2AJ ROBOT

Dipl.-Ing. Adnan Šljivo
Doc. dr. sci. Malik Čabaravdić
University of Zenica, Faculty of Mechanical Engineering,
Fakultetska 1, Zenica
Bosnia&Herzegovina

ABSTRACT
The simulation of motion and work of industrial robots is being used more frequently in industry because it enables generating, testing and optimization of robot programming. In this work the development of an application for simple RV-2AJ robot simulation in C++ programming language for Windows OS, with use of OpenGL is presented. This application enables simulation of robot motion with direct and inverse kinematics calculations, and basic off-line programming of robot. This work is a foundation for further development of the application for RV-2AJ robot simulation, that should enable dynamics simulation and development and testing of control algorithms.
Keywords: robot, simulation, RV-2AJ, kinematics

1. INTRODUCTION
Industrial robots are essential component of automated manufacturing systems because of their versatility and flexibility. On the other side, these characteristics make robots very complex and fragile machines. Because of this, robot simulation is extremely valuable in employing robots on factory floor as it allows testing of solutions for complex tasks in virtual environment where it is easier to manipulate the robot and examine different layouts, scenarios and situations without danger of damaging real equipment or human injury.

Figure 1. Robot RV-2AJ

In this article the development of an application is presented, which enables five axis RV-2AJ robot motion simulation and provides basic off-line programming support. Robot RV-2AJ, shown in Figure 1., was built by Mitsubishi. It has five rotational joints giving it five degrees of freedom.

2. ROBOT KINEMATICS

In robot kinematic analysis one needs to assign frame to each link. In this work frames have been assigned according to Denavit-Hartenberg convention which is shown in Figure 2.

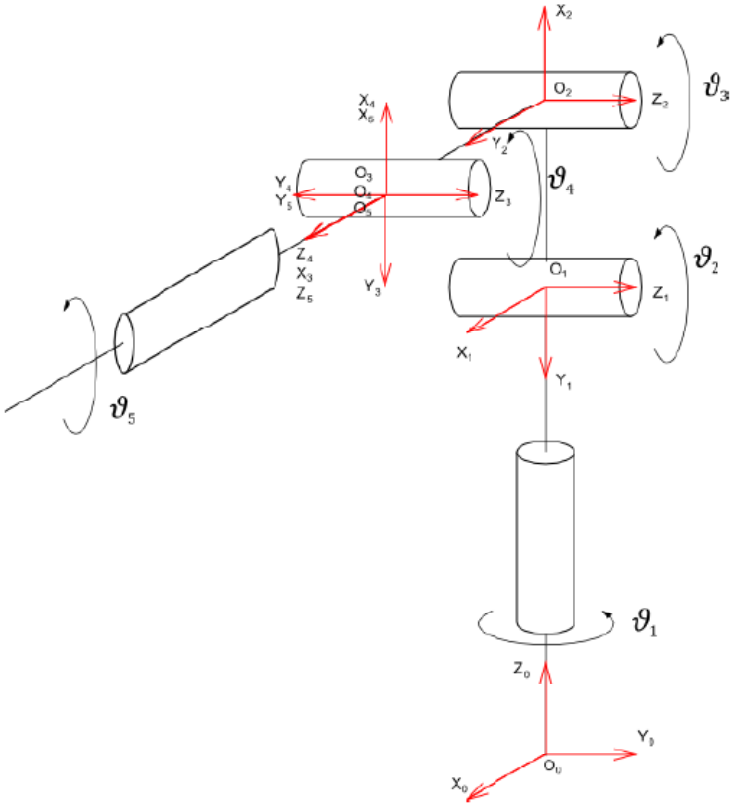

Figure 2. Frames assigned to links

Now it is possible to determine Denavit-Hartenberg parameters, and they are presented in Table 1. Since all joints are rotational, the only non-constant parameter is the angle ϑ_i.

Table 1. Denavit-Hartenberg parameters

i	$a_i[m]$	$\alpha_i[°]$	$d_i[m]$	$\vartheta_i[°]$
1	0	-90	0,3	0
2	0,25	0	0	-90
3	0,16	0	0	90
4	0	-90	0	-90
5	0	0	0	0

2.1. Direct Kinematics

In direct kinematics the goal is to find end-effector coordinates given joint coordinates. When Denavit-Hartenberg parameters are known, the transformation matrix is given by:

$$\boldsymbol{T}_i^{i-1} = \begin{bmatrix} c_{\vartheta_i} & -s_{\vartheta_i}c_{\alpha_i} & s_{\vartheta_i}s_{\alpha_i} & a_i c_{\vartheta_i} \\ s_{\vartheta_i} & c_{\vartheta_i}c_{\alpha_i} & -c_{\vartheta_i}s_{\alpha_i} & a_i s_{\vartheta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{... (1)}$$

Now the transformation matrix from base link to end-effector can be obtained as a product of matrices for every joint given in equation (1):

$$\boldsymbol{T}_5^0 = \boldsymbol{T}_1^0 \times \boldsymbol{T}_2^1 \times \boldsymbol{T}_3^2 \times \boldsymbol{T}_4^3 \times \boldsymbol{T}_5^4 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{... (2)}$$

Here $\boldsymbol{p}$ is position vector of end-effector, and vectors $\boldsymbol{n}$, $\boldsymbol{o}$ and $\boldsymbol{a}$are orthogonal unit vectors that define the orientation of end-effector frame.

2.2. Inverse Kinematics

The problem of inverse kinematics is determination of joint coordinates when end-effector position and orientation are given. Although there are different methods for solving this problem, in this work an iterative algorithm based on jacobian transpose matrix was used. In this method joint coordinates are updated in iteration *k*as follows:

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \Delta\boldsymbol{q} \quad \text{... (3)}$$

Here $\Delta\boldsymbol{q}$ is calculated as:

$$\Delta\boldsymbol{q} = \boldsymbol{J}^T\boldsymbol{K}\boldsymbol{e} \quad \text{... (4)}$$

Matrix $\boldsymbol{K}$ is a suitable (5x5) positive definite diagonal weighting matrix. Matrix $\boldsymbol{J}$is jacobian (6x5) matrix that is configuration dependant and whose each column corresponds to a joint*i*:

$$\boldsymbol{J_i} = \begin{bmatrix} \boldsymbol{z}_{i-1} \times (\boldsymbol{p}_e - \boldsymbol{p}_{i-1}) \\ \boldsymbol{z}_{i-1} \end{bmatrix} \quad \text{... (5)}$$

Vector $\boldsymbol{e}$ is the (6x1) error vector that can be calculated as:

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{p}_d - \boldsymbol{p}_e \\ \frac{1}{2}(\boldsymbol{n}_e(\boldsymbol{q}) \times \boldsymbol{n}_d + \boldsymbol{o}_e(\boldsymbol{q}) \times \boldsymbol{o}_d + \boldsymbol{a}_e(\boldsymbol{q}) \times \boldsymbol{a}_d) \end{bmatrix} \quad \text{... (6)}$$

When the error is sufficiently small iterative procedure stops.

3. SIMULATION APPLICATION

The application for robot motion simulation is built based on presented kinematics analysis of robot. It is capable of solving direct and inverse kinematics, and thanks to used methods it can be used to simulate different types of robots and not just RV-2AJ.

This application is programmed in C++, where OpenGL is used for visualization and WinAPI for GUI. Robot 3D model is read from an STL file.

As it is shown in Figure 3., the application enables motion specification in joint coordinates when it solves direct kinematics, and in world and in tool coordinates, when it solves inverse kinematics. Also, it displays both world and tool frames, which helps at defining end-effectors motion.

This application also offers a basic support for off-line programming. One can save position in position list, and then use them in program for robot motion. A limited set of commands is supported,

such as a command for moving to a desired point with joint ("MOV") and linear interpolation ("MVS"), wait command (DLY), etc.

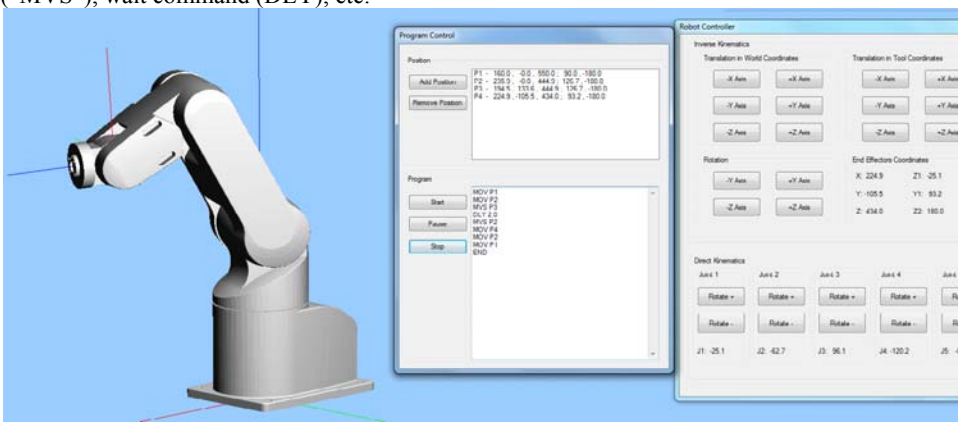

Figure 3. Simulation application window

4. CONCLUSION

In this work the development of an application is presented, that enables robot motion simulation. In such environment it is much easier to manipulate robot, since user can easily pan, zoom and rotate the view, define robot movement. Off-line programming support makes it possible for user to try simple robot programs.

This application should serve as a basis for development of a more robust robot simulation application. The application structure and used methods for kinematics solving allow relatively easy addition of different robot models. Also, off-line programming can be expanded to allow much larger amount of commands, which should allow user to do program tests. In the end, the application in future should support dynamic simulation, which will be a basis for testing different control algorithms.

5. REFERENCES

[1] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics Modelling,Planning and Control, Springer-Verlag, London, 2009

[2] S. R. Buss, Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods, University of California, San Diego, 2009

[3] L. Žlajpah, Robot Manipulators, Trends and Development - Robot Simulation for Control Design

[4] S. Anton, T. Fries, T. Horsch, F. W. Schröer,C.Willnow, C. Wolf, A Framework for Realistic Robot Simulation and Visualisation

[5] Mitsubishi Industrial Robot, RV-1A/RV-2AJ Series, Standard Specifications Manual

[6] M. Coman, S. Stan, M. Manic, R. Bălan, Design, Simulation and Control in Virtual Reality of a RV-2AJ robot

[7] M. S. Alshamasin, F. Ionescu, R. T. Al-Kasasbeh, Kinematic Modeling and Simulation of a SCARA Robot by Using Solid Dynamics and Verification by MATLAB/Simulink, EuroJournals Publishing, Inc. 2009

[8] M. A. Qassem, I. Abuhadrous, H. Elaydi, Modeling and Simulation of 5 DOF Educational Robot Arm