A Generic Simulator/Controller for Robot Manipulators

Abdelshakour A. Abuzneid and Tarek M. Sobh

Department of Computer Science and Engineering
University of Bridgeport
169 University Avenue
Bridgeport, CT 06601, USA

Abstract. General form application is a very important issue in industrial design. Prototyping a design helps in determining system parameters, ranges and in structuring better systems. Robotics is one of the industrial design fields in which prototyping is crucial for improved functionality. Developing an environment that enables optimal and flexible design using reconfigurable links, joints, actuators and sensors is essential for using robots in the education and industrial fields. We propose a PC-Based software package to control, monitor and simulate a generic SIX-DOF robot including a spherical wrist. This package may be used as a black box for the design implementations or as a white (detailed) box for learning about the basics of robotics and simulation technology.

1 Introduction

To design a complete and efficient robotic system there is a need for performing a sequence of cascaded tasks. The design task starts by determining the application of the robot, the performance requirements, and then determining the robot configuration and parameters suitable for that application. The physical design starts by ordering the parts and assembling the robot. developing the required software (controller, simulator and monitor) and hardware elements is the next task. The following stage includes manipulator testing which determines the performance of the robot and the efficiency of the design. Our aim is to build a complete PC-Based software package for control, monitoring and simulation of a 6-DOF manipulator, including a spherical wrist. The design will be independent of any existing specific robot parameters. The package will be an integration of several packages. Figure 1 shows how such a pc-based robot can be controlled using different schemes [2].
The idea for this work came from a project done in a robotics class at the Department of Computer Science and Engineering, in the School of Science, Engineering and

Technology, University of Bridgeport. The project was to design a full integrated package to control, monitor, and simulate an SIR-1 robot. The SIR-1 robot is a 6-DOF robot with a gripper. While work was done on that project, we were continuously wishing for the existence of such a prototyping package in the market. We did a wide range search and exhaustive market survey for what was available. We searched a variety of papers, books, book chapters and Internet sites. We have also talked to a number of companies that manufacture manipulators and we found out that reasonable progress has been done in this field. Some of the companies introduce prototyping for special or specific manipulators. Others try to design a whole prototyping package introducing mainly numerical solutions rather than closed form solutions. Unfortunately a generic pc-based controller/monitor/simulator package for a generic manipulator does not exist at this time. Initially it looked like it is impossible to find complete closed form solutions for a 6-DOF robot by solving a complicated set of nonlinear equations. This view is changing nowadays. There is a large number of research papers that scientists produce to find a general form solution for a certain configuration of a robot [1,6,7]. If the results of these research papers can be tested and then gathered within a complete and well designed package, the dream of a closed form prototyping controller may be reachable. The variety of powerful mathematical packages available nowadays such as Matlab, Mathematica, Maple, MatCom and others help in achieving our goals. From this point of view, we may be able to find closed form solutions for a 6-DOF robot with a spherical wrist to be the Medicare for the complexity of robot control design.

2 Background

The final design of the software package will be a collection of smaller packages. Each of these packages will be independent of any specific set of robot parameters. This can be done by making all calculations symbolically. Needless to say, that will make the mathematics more difficult. By using mathematical application packages available nowadays such as Maple and Mathematica the job will be easier but not trivial. The next few sections give a theoretical background.

2.1 Forward kinematics

Forward kinematics is used to describe the static position and orientation of the manipulator linkages. There are two different ways to express the position of any link: using the *Cartesian* space, which consists of position *(x,y,z)*, and orientation, which can be represented by a 3x3 matrix called the rotation matrix; or using the joint space, by representing the position by the angles of the manipulator's links. Forward kinematics is the transformation from joint space to Cartesian space. This transformation depends on the configuration of the robot (i.e., link lengths, joint positions, type of each joint, etc.). In order to describe the location of each link relative to its neighbor, a frame is attached to each link, then we specify a set of

parameters that characterize this frame. This representation is called the *Denavit-Hartenberg notation*. Figure 2 shows a physical six-link robot manipulator.
The Denavit-Hartenberg parameters are [1]:

a_i distance along x_i from O_i to the intersection of the x_i and z_{i-1} axes.

d_i distance along z_{i-1} from O_{i-1} to the intersection of the x_i and z_{i-1} axes. d_i is variable if joint *i* is prismatic.

α_i the angle between z_{i-1} and z_i measured about x_i.

θ_i the angle between x_{i-1} and x_i measured about z_{i-1} is variable if joint *i* is revolute.

The *Denavit-Hartenberg* parameters for our prototype robot are shown in Table 1. The parameters for the last 3 links are constants with the exception of θ's, the joint variables and d_6 the offset parameter which represents the offset distance between O_3 and the center of the wrist O. a_4,a_5 and a_6 are zeros because the distance along x_i from O_i to the intersection of x_i and z_{i-1} is zero.

The corresponding Transformation matrix is

where

$$A_0^6 = A_1 A_2 A_3 A_4 A_5 A_6 \tag{1}$$

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \tag{2}$$

$$\begin{bmatrix} 0 & s(\alpha_i) & c(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

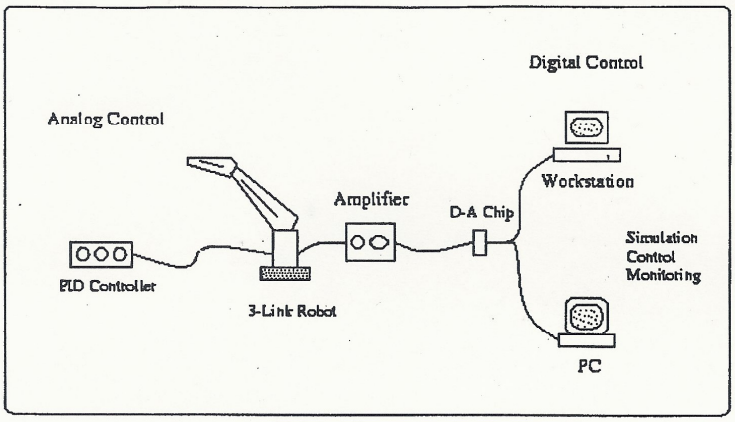

Fig. 1. Controlling the robot using different schemes

2.2 Inverse Kinematics

Inverse kinematics solves for the joint angles given the desired position and orientation in Cartesian space. This is a more difficult problem than forward kinematics. The complexity of inverse kinematics can be described as follows, Given a 4x4 homogeneous transformation which gives the required position and orientation

$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \tag{4}$$

The homogeneous transformation matrix results in 12 nonlinear equations in 16-unknown variables (a_1,a_2,a_3,α_1, α_2, α_3 ,θ_1 .... θ_6, d_1 , d_2, d_3, d_6).

$$T_{ij}(q_1,\ldots,q_6) = H_{ij} \tag{5}$$

where i=1,2,3 , j=1,2,3,4.

Link	a_i	α_i	d_i	θ_i
1	a_1	α_1	d_1	θ_1
2	a_2	α_2	d_2	θ_2
3	a_3	α_3	d_3	θ_3
4	0	-90	0	θ_4
5	0	+90	0	θ_5
6	0	0	d_6	θ_6

Table 1: Symbolic DH parameter for the robot

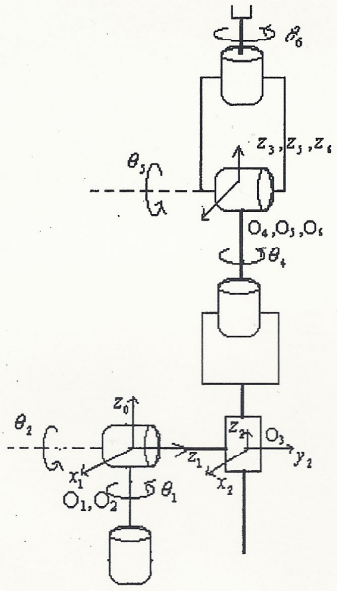

Fig. 2. A physical six-link robot manipulator

For example, to find the corresponding joint variables (θ_1, θ_2,d_3,θ_4,θ_5,θ_6) for RRP:RRR manipulator shown in Figure 2 where

$$\begin{bmatrix} e_{31} & e_{32} & e_{33} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

We must solve 12 simultaneous set of nonlinear equations. The first glance at a simple homogeneous transformation matrix eliminates the possibility of finding the solution by solving those 12 simultaneous set of nonlinear trigonometric equations. These equations are much too difficult to solve directly in closed form and therefore we need to develop efficient techniques that solves for the particular kinematics structure of the manipulator. To solve the inverse kinematics problem, closed form solution of the equations or a numerical solution could be used. Closed form solution is preferable because in many applications where the manipulator supports or is to be supported by a sensory system, the results need to be supplied rapidly (in real-time) [1]. Since inverse kinematics can result in a range of solutions rather than a unique one, finding a closed form solution will make it easy to implement the fastest possible sensory tracking algorithm.

One aim of this work is to try to find closed solutions for a prototype robot which is a general 3-DOF robot having an arbitrary kinematic configuration connected to a spherical wrist. These closed form solutions could be attained by different approaches [3,6,7]. One possible approach is to decouple the inverse kinematics problem into two simpler problems, known respectively, as inverse position kinematics, and inverse orientation kinematics [1,3]. To put it in another way, for a six-DOF manipulator with a spherical wrist, the inverse kinematics problem may be separated into two simpler problems, by first finding the position of the intersection of the wrist axes, the center, and then finding the orientation of the wrist. Lets suppose that there are exactly six degrees of freedom and the last three joints axes intersect at a point O. We express the rotational and positional equations as

$$R_0^6(q_1,\ldots,q_6) = R_{3x3} \tag{7}$$

$$d_0^6(q_1,\ldots,q_6) = d \tag{8}$$

where d and R are the given position and orientation of the tool frame.
The assumption of a spherical wrist means that the axes z_4, z_5 and z_6 intersects at O and hence the origins O_4 and O_5 assigned by the D-H convention will always be at the wrist center O. The importance of this assumption for inverse kinematics is that the motion of the final three links about these axes will not change the position of O. The position of the wrist center is thus a function only of the first three joint variables. Since the origin of the tool frame O_6 is simply a translation by a distance d_6 along the z_5 axes from O, the vector O_6 in the frame $O_0X_0Y_0$ is

$$O_6 - O = -d_6 Rk \tag{9}$$

Note that R is multiplied by *k* because it is a translation along z axes.
Suppose P_c denotes the vector from the origin of the base frame to the wrist center. Thus, in order to have the end-effector of the robot at the point d with the orientation of the wrist center O located at the point

$$P_c = d - d_6 Rk \tag{10}$$

the orientation of the frame $O_0X_0YZ_0$ with respect to the base is given by R. If the components of the end-effector position d are denoted by dx,dy,dz and the components of the wrist center P_c are denoted by Px,Py,Pz then this equation results in the relationship

$$\begin{bmatrix} P_y \\ P_z \end{bmatrix} = \begin{bmatrix} d_y - d_6 r_{23} \\ d_z - d_6 r_{33} \end{bmatrix} \tag{11}$$

Using equation 10 we may find the values of the first three joint variable. Thus for this class of manipulators, the determination of the inverse kinematics can be summarized in 3 steps [1]:

Step 1: Find q_1,q_2,q_3 such that the wrist center P_c is located at $P_c=d-d_6k$
Step 2: Using the joint variables determined in Step 1, evaluate R(0,3).
Step 3: Find a set of *Euler angles* corresponding to the rotation matrix $R_3^6 = (R_3^6)^{-1} R$

2.3 Velocity and Inverse Velocity Kinematics

In order to move the manipulator at constant velocity, or at any prescribed velocity, we must know the relationship between the velocity of the tool and the joint velocities. To calculate the velocity, the Jacobian matrix should be constructed as follows

$$J = [J_1 J_2 J_3 J_4 J_5 J_6] \tag{12}$$

where

$$J_i = \begin{bmatrix} z_{i-1} \times (O_n - O_{i-1}) \\ z_{i-1} \end{bmatrix} \tag{13}$$

if i is a revolute and

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \tag{14}$$

if i is a prismatic, where z_i is the first three elements in 3rd column of $T_{(0,i)}$ and O_i is the first three elements in the 4th column of $T_{(0,i)}$. Then forward velocity will be

$$\dot{X} = J(q)\dot{q} \tag{15}$$

The inverse velocity problem becomes one of solving the system of linear equations. The Inverse Velocity Kinematics will then be

$$\dot{q} = J^{-1}(q)\dot{X} \tag{16}$$

2.4 Acceleration and Inverse Acceleration Kinematics

Differentiating (15) yields the acceleration equation

$$\ddot{X} = J(q)\ddot{q} + \frac{d}{dt}J(q)\dot{q} \tag{17}$$

By solving 16 for inverse acceleration, we find

$$\ddot{q} = J(q)^{-1}\ddot{X} - J(q)^{-1}\frac{d}{dt}J(q)\dot{q} \tag{18}$$

2.5 Singularities

Singularities represent configurations from which certain directions of motion may be unattainable. It is possible to decouple the determination of a singular configurations for those manipulators with a spherical wrist into two simpler problems. The first is to determine the arm singularities, that is, ingularities resulting from motion of the arm, which consists of the first three or more links, while the second is to determine the wrist singularities resulting from motion of the spherical wrist. Suppose that n=6, that is, the manipulator consists of a 3-DOF arm with a 3-DOF spherical wrist. In this case the Jacobian matrix is a 6x6 matrix and a configuration is singular if and only if

$$\det J(q) = 0 \tag{19}$$

if we now partition the Jacobian matrix into 3x3 blocks as

$$J = \begin{bmatrix} J_p & J_0 \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \tag{20}$$

then, since the final three joints are always revolute

$$J_0 = \begin{bmatrix} z_3 \times (O_6 - O_3) & z_4 \times (O_6 - O_4) & z_5 \times (O_6 - O_5) \\ z_3 & z_4 & z_5 \end{bmatrix} \tag{21}$$

Since The wrist axes intersect at a common point O, if we choose the coordinate frames so that $O_3=O_4=O_5=O_6=O$, then J_0 becomes

$$J_0 = \begin{bmatrix} 0 & 0 & 0 \\ z_3 & z_4 & z_5 \end{bmatrix} \tag{22}$$

and the i-th column J_i of J_p is

$$J_i = \begin{bmatrix} z_{i-1} \times (O - O_{i-1}) \\ z_{i-1} \end{bmatrix} \tag{23}$$

if joint i is revolute and

$$J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \tag{24}$$

if joint i is prismatic. In this case the Jacobian matrix has the block triangular form

$$J = \begin{bmatrix} J_{11} & 0 \\ J_{21} & J_{22} \end{bmatrix} \tag{25}$$

with determinant

$$\det J = \det J_{11} \det J_{22} \tag{26}$$

where J_{11} and J_{22} are each 3x3 matrices. J_{11} has i-th column z_{i-1} x $(O\text{-}O_{i-1})$ if joint i is revolute, and z_{i-1} if joint i is prismatic, while

$$J_{22} = \begin{bmatrix} z_3 & z_4 & z_5 \end{bmatrix} \tag{27}$$

2.6 Dynamics

Manipulator dynamics is concerned with the equation of motion, the way in which the manipulator moves in response to torques applied by the actuators, or external forces. There are two problems related to manipulator dynamics that are important to solve:

- ***inverse dynamics*** in which the manipulator's equations of motion are solved for given motion to determine the generalized forces required for each joint (control stage) and
- ***direct dynamics*** in which the equations of motion are integrated to determine the generalized coordinate response to applied generalized forces (simulation stage).

The equation of motion for an n-axes manipulator are given by

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G)q) \tag{28}$$

Where
The equation may be derived via a number of techniques, including the *Lagrangian* method. Due to the enormous computational cost of this approach it is always difficult to compute manipulator torques for real-time control based on the dynamic equations. To achieve real-time performance many approaches were suggested, including table lookup and approximation [4]. The most common approximation is to ignore the velocity-dependent term *C*, since accurate positioning and high speed motion are exclusive in typical robot application. Practically, a *PID* controller might be a good option to achieve a real-time performance,

$$Q = \ddot{\theta}_d + k_v \dot{E} + k_p E + k_i \int E dt \tag{29}$$

where k_v, k_p and k_i are the derivative, proportional and integral parameters respectively.

The advantages of using a PID controller are the following:

- Simple to implement
- Suitable for a real-time control
- The behavior of the system can be controlled by changing the feedback gains

2.7 Simulation

To simulate the motion of a manipulator, we may use the simulation module by manipulating (28)

$$\theta = M \quad (q)\left[Q - C(q,q)q - F(q) - G(q) \right] \tag{30}$$

This represents the direct or integral or forward dynamic formulation giving joint motion in terms of input torques.M(q) is the symmetric joint-space inertia matrix and for a 6-DOF manipulator M is an 6x6 symmetric matrix.C is the manipulator Coiolis/centripetal torque and for 6-DOF manipulator C will be a 6x1 matrix.

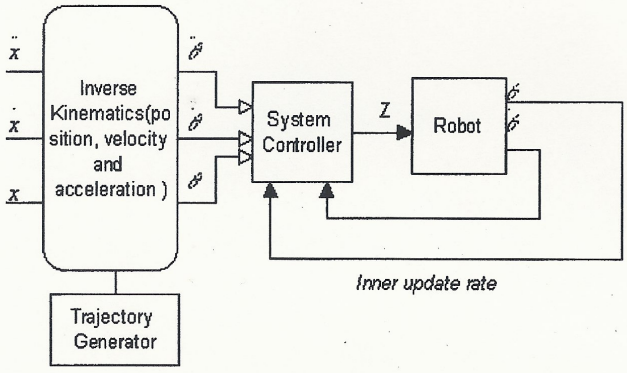

Fig. 3. Trajectory Generator integrated in the control loop

2.8 Trajectory Generator

Trajectory generation describes the position velocity and acceleration of each link. This includes how the front user interfaces to describe the desired behavior of the manipulator. This could be a very complicated problem depending on the desired accuracy of the system. In some applications we might need to specify only the goal position, whereas in some application, we might need to specify the velocity with which the end effector should move. Since trajectory generation occurs at run time on a digital computer, the trajectory points are calculated at a certain rate, called the *path update rate*. One is advantage of using a PID controller is a high update rate is required to achieve reasonable accurcy. Our package role here is to calculate trajectory points which generate a smooth motion for the manipulator. The smoothness of motion is a very important issue due to physical considerations such as the required torques that causes this motion, the friction at the joints, and the frequency of update required to minimize the sampling error. Figure 3 shows how trajectory generator can integrated in the control loop. It also shows two update rates, one is the inner update rate which update the system control with the actual joint position and velocity. The other loop updates the system control with the required joint values. The sampling of the two update rates can be different.

1	RRR:RRR
2	PRR:RRR
3	RPR:RRR
4	PPR:RRR
5	RPP:RRR
6	PRP:RRR
7	RPP:RRR
8	PPP:RRR

Table 2: Possible robot configuration

The black box includes

1. Full control loop implementation (PID & Dynamics based)
2. Full simulation loop
3. GUI with error analysis

3 Project Ideas and Progress

One target of the package is to find closed form solutions such that direct substitutions are made when parameters are entered. This requires determining which parameters should be variables and which should be constants. Variables could be robot parameter configuration variables or state variables. The former are variables that define the structure of the manipulator, so they are constants for the same robot (i.e. a's, α's, dynamic parameters...etc.).

The latter describe the state of the robot (Joint Variable). Thus θ_i may be a state variable if i-th joint is revolute otherwise, it is a configuration variable. When the program is run, it will ask for the configuration of the robot (one of those listed in Table 2. Then the program will decide what the robot configuration variables are and ask the user to enter them one after another. According to the task the program is asked to run, it will ask for the state variables. For example if the program is asked to calculate the Inverse Kinematics, the program will ask for the target Cartesian position and orientation to get the values of q's as an output. When the front user asks to do a task, the program calls the task handler. The task handler is a large set of equations that are invoked when the front user enters the required input, and displays the results rapidly. Figure 4 shows the task flow chart. The next few sections give a few examples of how we managed to do the mathematics for the different tasks .

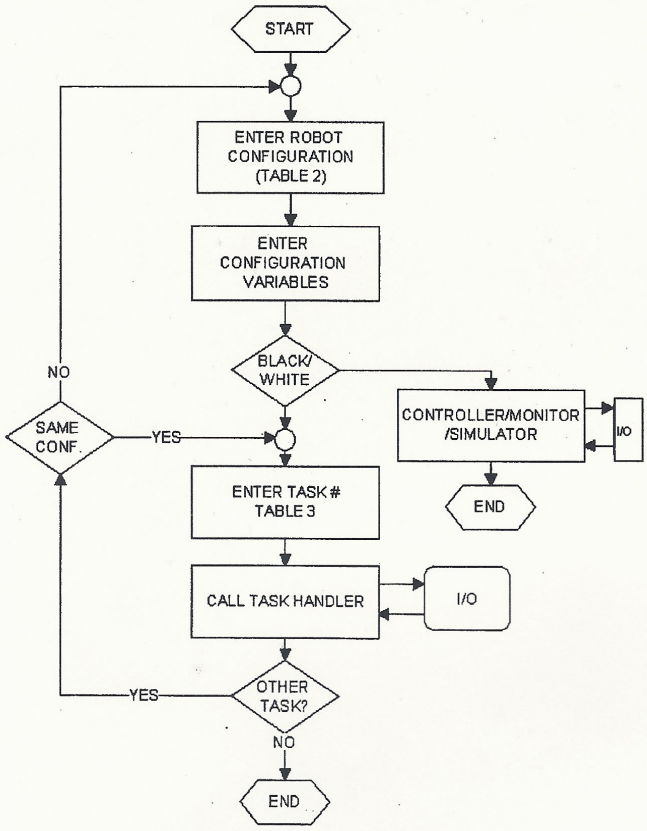

Fig. 4. Task flow chart

References

1. Spong,M. W.,Vidyasagar,M. , Robot dynamics and control,Join Wiley & Sons (1989).
2. Dekhil,M. ,Sobh,T. M. ,Henderson,T. C. ,Sabbavarpu,A., Mecklenburg,R. , Robot manipulator prototyping (Complete design review, University of Utah,7-17.
3. Ho,C. Y. ,Sriwattanathamma, J., Robot kinematics, Symbolic Automation and Numerical Synthesis, Ablex Publishing Corporation,
4. Corke, P.I., Robotics Toolkit, CSIRO, Division of manufacturing technology, (February 1994)
5. T. M. Sobh, M. Dekhil, T. C. Henderson, and A. Sabbavarapu, ``Prototyping a Three-link Robot Manipulator," in ASME Press Series on Robotics and Manufacturing; Recent Trends in Research and Applications, volume 6, pp. 781-786, 1996.
6. Herrea-Beneru,L. ,Mu,E. ,Cain,J. T., Symbolic Computation of Robot ManipulatorKinematics, Department of Electrical Engineering, University of Pittsburgh
7. Rieseler,H., Wahl,F. M., Fast symbolic computation of the inverse kinematics of robots, Institute for Robotics and computer control, Technical University of Braunschweig.
8. M. Dekhil, T. M. Sobh, T. C. Henderson, and R. Mecklenburg, ``UPE: Utah Prototyping Environment for Robot Manipulators." in the Journal of Intelligent and Robotic Systems, 17: 31-60, 1996.
9. M. Dekhil, T. M. Sobh, T. C. Henderson, and R. Mecklenburg, ``UPE: Utah Prototyping Environment for Robot Manipulators". I n proceedings of the IEEE International Conference on Robotics and Automation, Nagoya, Japan, May 1995.
10. M. Dekhil, T. M. Sobh, and T. Henderson, ``URK: Utah Robot Kit – A 3-link Robot Manipulator Prototype.". In proceedings of the IEEE International Conference on Robotics and Automation, San Diego, May 1994