International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:9, 2007

Robot Task-Level Programming Language and Simulation

M. Samaka

***Abstract*—**This paper presents the development of a software application for Off-line robot task programming and simulation. Such application is designed to assist in robot task planning and to direct manipulator motion on sensor based programmed motion. The concept of the designed programming application is to use the power of the knowledge base for task accumulation. In support of the programming means, an interactive graphical simulation for manipulator kinematics was also developed and integrated into the application as the complimentary factor to the robot programming media. The simulation provides the designer with useful, inexpensive, off-line tools for retain and testing robotics work cells and automated assembly lines for various industrial applications.

***Keywords*—**Robot programming, task-level programming, robot languages, robot simulation, robotics software.

I. Introduction

ROBOTIC systems are ever more difficult to design, program, and to operate, due to their growing complexity. Also, the current state of art of powerful robot programming has reflected the requirement for a highly complicated multidiscipline system.

A great deal of such system requirement is based on decision-making, and knowledge accumulation [1], [2]. However, enhancing the system supporting software with appropriate tools is considered as a primary factor in developing a highly efficient and easy to operate robot system [3], [4].

In recent years, a great deal of simulation work has been developed for robotics [5]. A complementary factor to the robot programming media is the graphic simulation. Presentation of shape and trajectories assists in program verification and hence minimizing the real robot mistakes. Also, interactive simulations incorporating robot kinematics can provide a useful method for designing and evaluating robotics work cells both for existing and future industrial applications [6]-[8]. Such graphical simulations permit safe and inexpensive methods for determining robot motion paths without requiring the use of any actual robotics hardware.

The aim of this work is to provide an interactive software tool for robot programming and simulation. The Robot programming language presented in this paper supports the development of interactive commands that are compiled for runtime applications. The technique implemented is based on incremental compilation. An interactive 3D graphical simulation, for a general-purpose multi-link robot manipulator, was also developed and integrated into the robot programming environment. The simulation program includes several algorithms such as: direct and inverse kinematics analysis for multi-link mechanisms, motion trajectory determination, workspace estimation, and collision avoidance.

Manuscript received September 30, 2005.

M. Samaka is with the department of Computer Science and Engineering, College of Engineering, University of Qatar, P. O. Box 2713, Doha-Qatar (phone:+974 5471371; fax: +974 485 2961; e-mail: Samaka.m@qu.edu.qa).

II. System Description

The structure of the robot software system, as shown in Fig. 1, consists of seven modules. Based on the system design strategy, a modular approach in software development was adopted in order to enhance system efficiency and flexibility. In the following paragraph, each module is described briefly. Also, the system inter-module relations and interactions is illustrated in Fig. 2.

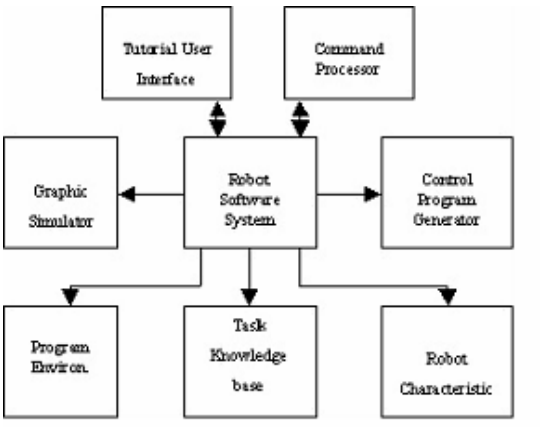

Fig. 1 Modules of the Robot Software System

The tutorial module is a GUI interface. It provides users with useful commands, and other information that are needed for operating the system efficiently. The system requirements can be satisfied when using the menu-based command processor module. Menu options could be standardized for most robot types and applications. Besides, the command processor program is facilitated with built-in editing, and debugging facilities. They were designed especially to simplify the development process in all stages.

The robot characteristics module is a GUI interface that

allows the user to enter robot parameters, and robot geometric data into the robot parameter, and geometric files respectively. This information is needed for a robot kinematics operation for moving the manipulator through the appropriate trajectory of the intended points. The basic steps required to enter a desired multi-link manipulator using this module include:

- Assignment of Denavit-Hartenberg coordinates to each link [9].
- Measurement and entry of necessary robot parameters.
- Determination of the geometric module for each link of robot using graphic elements such as lines, polygons, revolute, circle, etc.
- Interactive entry and storage of the robot type, robot link parameters, and geometric data into appropriate files.

The program environment module is a GUI interface that allows the user to enter information that specifies the robot task to be performed; the information includes objects in the work cell, their initial and destination positions, orientations, etc. Also, this module allows the user to determine the desired speed level of the robot motion in real time.

The task knowledge base module accumulates a number of different robot tasks, which are usually created, and built earlier by a user.

The control program generator module is a program that allows the user to build a set of commands. When these commands are executed they direct the tasks that are selected from the task knowledge base module, and therefore controlling the robot manipulator through the defined trajectory. The feedback signals generated from robot controller during the run-time are fed back to the program (see Fig. 2). These signals that reflect the sensors states affect the execution of the running program.

The robot's response to the control program that is constructed within the system can be simulated using the graphic simulator module, which displays 3D animation for the robot motions.

III. Software Development for Robot Control

Within the robot programming environment developed in this study, the robot-operating program can be constructed interactively via two phases. At each phase different levels of commands are being used. The first phase is involved in defining and coding tasks relevant to the intended robot functions. The language used to invoke the process of the task creation is Prolog. During this phase the tasks that are created are then accumulated together in the knowledge base module.

The second phase is concerned with creating of a high-level control robot program. A program of this type usually includes a number of call statements for the tasks that are selected from the knowledge base module to direct a specific robot motion.

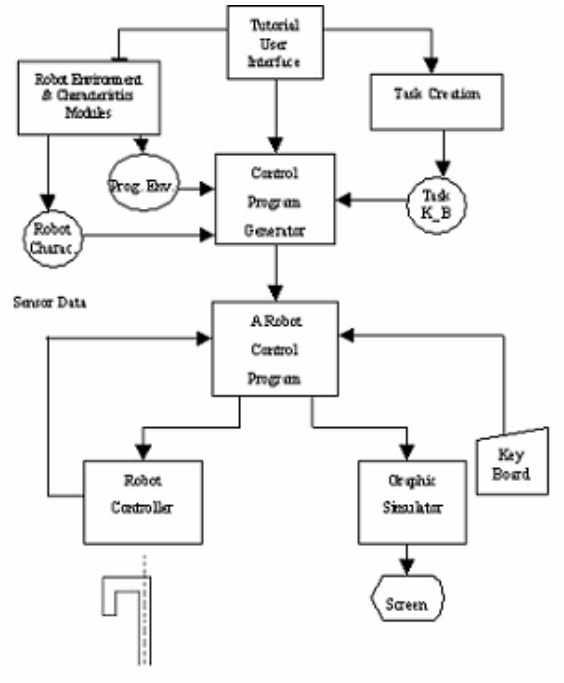

Fig. 2 Control Program Creation And Execution

In the robot programming application of this project, creating of a control program for a certain robot type to carry out a specified function may be achieved interactively. This process starts by first, defining a set of robot characteristics, that should be fed into the robot characteristic module. A user will then be able to utilize the system efficiently and interactively by selecting an appropriate sequence of tasks from the knowledge base using the menu driven facility, presented in Fig. 3.

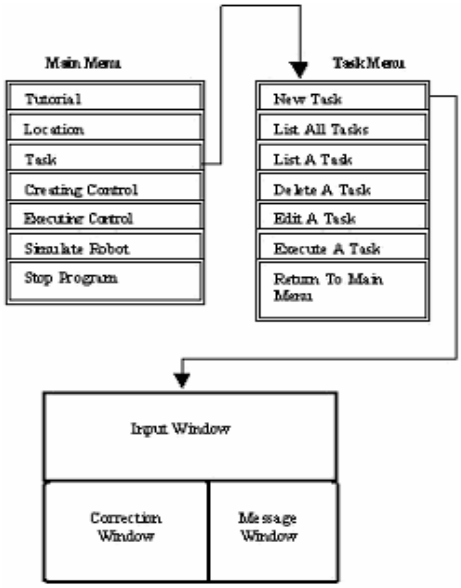

Fig. 3 Menus to create new task

The control program generator will then start by asking the user at each defining task for the related reference points. In that case, the user will respond by either referring to the name of a file created previously, inside the program environment module, or defining a new set of reference locations by feeding them directly. Finally, each task would be checked, and the user would be notified whether the task is being accepted or not as illustrated in Fig. 4.

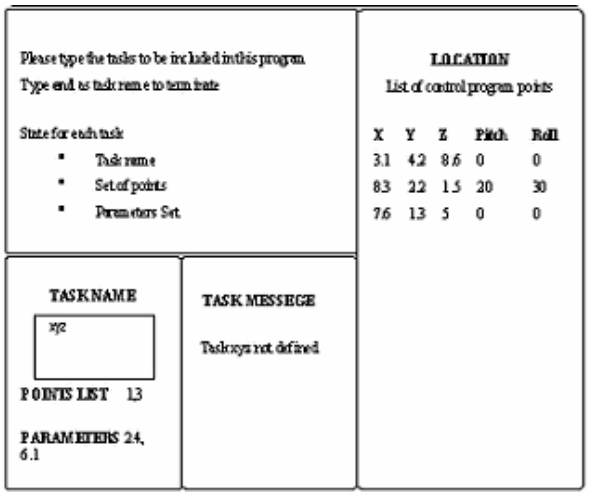

Fig. 4 Creating the control program screen

Since the creation process of a control program is the responsibility of the control program generator module, an important point has to be noted here, is that a robot operator should not necessarily has the knowledge of creating a program using a high level language. All what he/she supposed to know is the appropriate sequence of tasks stored in tasks knowledge-base module, that are associated to his/her robot activity. Also he/she should know the name of files that are held in the program environment module, which contains the required reference location values corresponding to the tasks.

During the robot task creation phase, which is needed for either creating or updating a task, a specially designed flexible screen editor and an incremental compiler programs are being provided. The incremental compiler provides a user with an immediate response, reporting whether the current compiled command line, is syntactically correct or not. Obviously, using the facilities of such software development tools would speed up the process of writing and editing programs, and increase the system reliability. An example of program to perform palletizing is shown in Fig. 5.

IV. Robot Simulation

The aim of developing the software tools for robot simulation in this research study is to assist the user to evaluate the robot performance, and the time sequence of the events by observing the animation on screen. The robot simulator module receives the geometrical and physical characteristics for the robot arm and for other objects in the work cell, from the program environment and robot characteristics modules. However, the feedback sensing signals used to affect the program may also be simulated using the keyboard.

The robot kinematics simulation system designed in this research study consists of several components that are described briefly in the following:

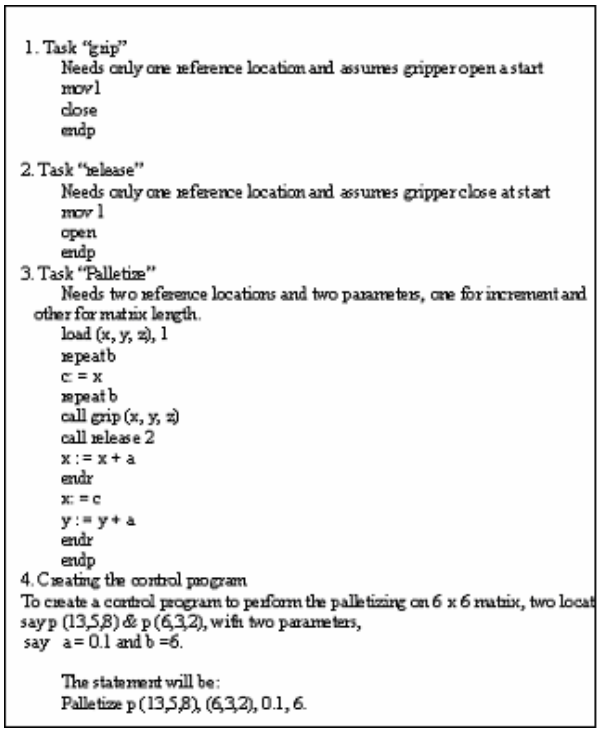

Fig. 5 Example of program to perform palletizing

A. Kinematics Analysis

This component is used to develop the direct and inverse kinematics equations for the desired robot. The direct kinematics problem involves the determination of the position and orientation of the end effectors of a robot manipulator with respect to a reference coordinates system, and the given manipulator joint angle variables. After the Denavit-Hartenberg coordinate system has been established for each link of the robot, a 4x4 homogeneous matrix, which includes translation, rotation and scaling, has to be setup for robot geometric computation.

The inverse kinematics problems involve finding a set of point variables and obtaining the position and orientation of the end effectors of the robot arm at the desired position with respect to a reference coordinates system.

The screen of Fig. 6 shows the input data for the Animation PUMA robot and the corresponding link parameter table after being processed by this component. The simulated robot affected by this set of parameters is shown in Fig. 7.

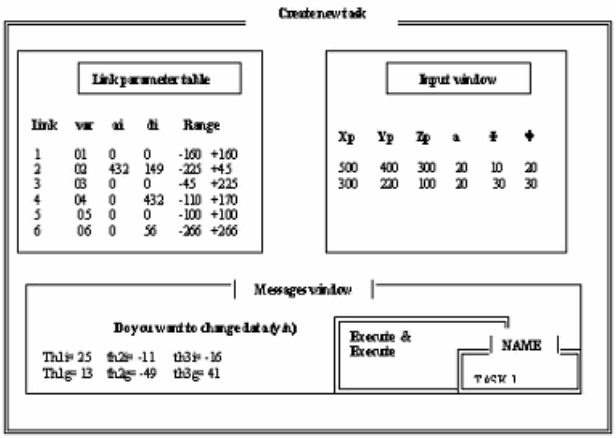

Fig. 6 Input data of the Animation PUMA and the links parameters after being processed

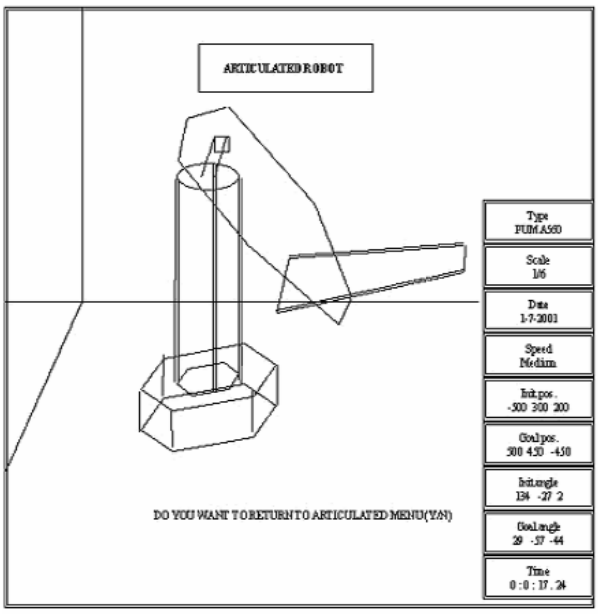

Fig. 7 The simulated robot affected by the input set of parameters

B. Trajectory Planning

This component calculates the robot trajectory according to the given positions and orientation of the desired points, and the type and speed of a robot motion.

The software routines of this component are developed to perform the following:

- Simulating the point-to-point motion of robot. In this type of motion, the robot can be trained manually moving the arm to define the position, orientation and end effectors control for picking, or placing operations. For implementing the point-to-point motion control, the digitize mode method was used in which the user enters the desired position and orientation of the end effectors to determine the kinematics solution for that point.
- Simulating the straight-line motion of a robot for moving the robot end effectors. This routine invokes the following steps:
  (a) The motion file is opened and the user enters the positions and orientations of the two desired points.
  (b) The program uses pools Cartesian path control algorithms to find the drive transform matrices [9].
  (c) The inverse kinematics solution routines are then executed and their solution stored in the motion file.
- Determining of a collision free path for robot motion is an important task that must be determined so that the robot arm can move from one reachable initial position to another reachable final destination position.
- Finally, this component is also responsible to calculate and display the real time duration that would be spent during the robot task.

C. Workspace

The workspace component of a robot manipulator provides information about the three dimensional volumes that can be reached by the robot end effectors. This component can be used to compute the work envelope for a particular robot. It permits the designer to implement optimum layouts for robotics work cells by allowing comparative evaluation of different robots and associated machinery for specific industrial applications. Also, this component is able to check if a robot trajectory would exceed the determined workspace. If so, an interactive scheme of message would be introduced as shown in Fig. 8, as the component discovers an out of range angle value.

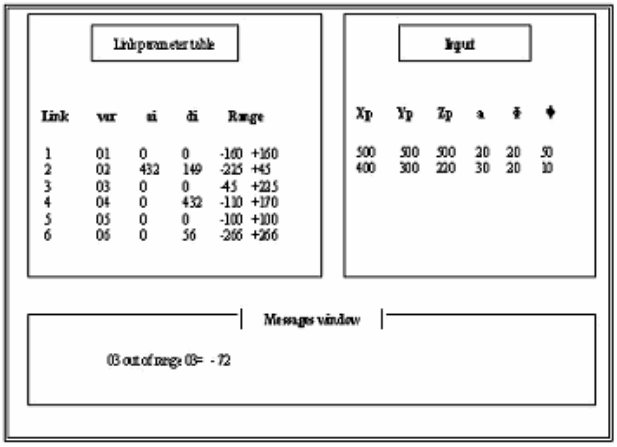

Fig. 8 An out of range angle value

D. Graphics Generation and Processing

By using this module, the values for the window, view ports, view normal vector, and the desired types of projections

will be set. The subroutines to create the graphical primitives such as arcs, circles, revolute, etc., will then be called to display the simulated robot on the desired graphical device. In this regard, a toolbox was built that contains all of these subroutines. Each of them may be called by issuing a special command.

V. Conclusion

The paper discusses the process of developing an off-line robot task programming and simulation environment. This environment may be considered as an ideal media for robot task accumulation and manipulation activities.

The modularity in system design allows for integrating the task programming media with the robot simulation module that was also developed in this project. The capabilities of this robot simulation system have been discussed and illustrated in this paper using a six-degree of freedom revolute robot. An industrial designer can use this simulation to analyze and evaluate the robot manipulator performance for a particular application, to check for positioning of the orientation, as well as to determine collision free paths for robot motion. The modularity and interactive nature of this simulation permit the addition of other similar robots into this simulation as well as its extension to include other types of robots. Future work in this area can involve the extension of this simulation to include robot dynamics, incorporation of various algorithms for optimizing robot path control, and multi-sensory feedback to make this robot simulation more realistic.

The incremental compilation approach implemented in this project reduces the compilation process into a modular activity that results in reducing the compilation time greatly. Also, the use of screen editor and simulator play an important role in assisting the programmer to visualize the program integration. Further enhancement to the automation of fast compilation processes may be achieved by the use of multilevel incremental compilation.

References

[1] R. Willgoss, and J. Iqbal, "*Neurofuzzy Learning of Mobile Robot Behaviors*", Proceeding of the12th Australian Joint Conference on Artificial Intelligence, AI '99, Sydney, Australia, December 6-10, 1999, pp. 278-290.

[2] S. Lopes, and J. Connell, "*Sentience in Robots: Applications and Challenges*", IEEE Intelligent Systems, Computer Society, 5(16), 2001, pp. 66-84.

[3] T. Längle, T. Lüth, E. Stopp, and G. Herzog, "*Natural Language Access to Intelligent Robots: Explaining Automatic Error Recovery*", In: A. M. Ramsay (ed.), Artificial Intelligence: Methodology, Systems, Applications, Amsterdam, IOS Press, 1996, pp. 259-267.

[4] W. Suwannik, and P. Chongstitvatana, "*Improving the robustness of evolved robot arm control programs with multiple configurations*", Proceeding of the 2nd Asian Symposium on Industrial Automation and Robotics, Bangkok, Thailand, May 17-18, 2001, pp. 87-90.

[5] H. Cheng, and K. Gupta, "*A Study of Robot Inverse Kinematics Based upon the Solution of Differential Equations*", Journal of Robotic Systems, 8(2), 1991, pp. 159-175.

[6] J. Buhmann, W. Burgard, A. Cremers, D. Fox, T. Hofmann, E. Schneider, J. Strikos, and S. Thurn, "*The Mobile Robot RHINO*", AI Magazine, 1(16), 1995, pp. 31-38.

[7] EASY-ROB *"3D Robot Simulation"*, CARAT robotic innovation, Germany. Available: http://www.easy-rob.de/product.html.

[8] ROBOT3D- "*Robot Offline Programming & Simulation*", Portugal, Available: http://clientes.netvisao.pt/fnavegan/example2.htm.

[9] J. Denavit, and R. Hartenberg, "*A kinematics notation for lower pair mechanisms based on matrices*", ASME Journal of Applied Mechanics, June, 1955, Volume 22, pp. 215-221.