Robotic arm graphical representation, motion simulation and execution

M. Papamattheou, S. Manesis, A. Grammaticos
Department of Electrical Engineering, Division of Systems and Control, University of Patras, GR 26001 Patras, Greece

Abstract

The classical approach in programming and testing a robotic installation in a workshell is by trial and error. In industry, one has to install the robot and then test it's functionality. In addition, the insertion of movements into memory through a limited keyboard without visualization, is a tedious and time consuming task.

This paper describes a method of programming, simulating, visualizing, and controlling of *any manipulatory, step motor driven, robotic arm through a common personal computer (PC),* thus avoiding the test-after-installation procedure. The software, was developed under Borland C++ v4.0, uses 32-bit instructions and can be run under a multitasking environment such as Microsoft's Windows v3.1 or higher. The hardware link between the computer and the robot can be achieved through any D/A card with sufficient output rate and high level language routines at a very low cost.

1 Introduction

Simulation and animation tools can certainly enhance the design, development, and even the operation of integrated robotic systems. It has been recognized as an important and almost indispensable tool in investigating the performance of complex processes or systems, and has been the subject of extended research during the last few years.

Most of the past research in robot simulation has been focused on simulating the structure and kinematics of a robotic system [1], [2]. One can examine a number of important issues related to robot joint and link behavior using these tools. By incorporating models of a robot workspace one can visualize the operation of a robot in a simulated workspace. Task planning and execution are interleaved, especially in an integrated robotic system [3],[4].

The simulator integrated in the developed system serves as an appropriate tool and medium which allows the robot designers and operators to examine the following specific items: 1) the structural feasibility of the mechanical devices before construction, 2) the response of the robot under extreme situations, 3) the correctness of the robot performing a desired sequence of sensing and actions, and 4) the visualization of changing environment and status of the robot.

Our research is directed towards the development of a powerful graphical environment to provide a suitable medium and mechanism for the design, development and operation of an integrated robotic system (CAD enchanced robotic system). Thus the environment helps the designer find flows in the early development stages, and also helps the operator ensure the safety of robot operation by testing a new procedure in simulation mode first, e.g. *preview* the action in a simulation before it takes effect in the real world. In addition , some extreme conditions should only be simulated but not realized experimentally. Therefore the system can reduce the cost of development and increase the reliability of both robot hardware and software.

The main objectives of the developed system include: 1) integration of planning and simulation; 2) compatibility of operation in real and simulation modes; 3) machine transferability; 4) support of any step-motor driven manipulator robot; 5) minimum hardware requirements, and 6) simple user interface.

These concepts are illustrated by implementation of simulation, animation, visualization, and interactive control on an ATLAS educational robot.

2 System description

Requirements

Most of the systems able to produce real time 3-D rendered graphics in full motion [5], were developed using a high level language under the UNIX operating system on extremely powerful and expensive workstations such as the Silicon Graphics ones.

The objective of minimum hardware requirements resulted in the following ones :

- An i80386 compatible personal computer (PC).
- An I/0 card with 6 analog outputs.
- A very few hours of work.

Although the minimum requirements include an Intel microprocessor compatible personal computer, the system can be implemented using a wide variety of computers, up to multi-processing systems and high performance RISC-based workstations.

Installation procedure

One of the most important goals achieved was the maximum flexibility. The procedure of implementation is actually very simple and consists of the following steps :

- Derive the Denavit-Hartenberg (D-H) representation constants of the simulated robot [6].

- Write down the geometric characteristics of the 3-D model of the robot and its workspace.
- Command the robotic arm to move using its own microcomputer and measure the level and width of the pulses used to trigger the step motors' drives, as long as their maximum frequency.
- Write down these data in simple ASCII files.
- Make the appropriate hardware connection between the card outputs and the robot motors' drives, and between the card inputs and the systems' sensors (if any).

Software

The software was developed under Borland C++ v4.0, uses 32-bit instructions and comes in two identical versions which run under Microsoft Windows v3.1 and Windows NT.

Windows is the most widespread operating system all over the world, and has a very sophisticated graphic user interface. This makes the software very easy to use by utilizing multiple, sizable and movable windows, menus, buttons, dialogue boxes, prompts etc. Windows is also a multi-tasking operating system and this means that the software doesn't have to monopolize a computer machine.

Windows NT, like UNIX is a 32-bit operating system, with enormous machine transferability and advanced features such as symmetric multiprocessing, preemptive multitasking and networking. It can be run under a wide variety of platforms such as Intel processor compatible systems, multi-processing systems based on Intel Pentium CPUs, high performance RISC-based workstations based on the DEC Alpha AXP or MIPS R4400 CPUs and other systems totaling a number of 1.700 different machines. Therefore, the Windows NT version of the program, takes advantage of all the advanced features of the operating system, and is even faster due to the 32-bit instructions it uses.

System architecture

As mentioned before, one of the basic objectives was to make the user's control program able to run in either real or virtual (simulation) mode without any modification.

The purpose of using a simulated robot and its workspace is to help the user visualize and examine the system performing a designed sequence of motor actions for a particular task before it is carried out in the real environment. This ability to operate in either real or virtual mode is important for the system not only in research environment but also in production environment, where the cost of shutting down the robots in order to test the new software is very high [7]. Virtual mode makes possible the test of extreme conditions without jeopardizing an expensive installation.

As part of the CAD enchanced robotic system, a simulator of manipulatory robots was developed. It simulates the robot's operation based on the program commands, sensory information and workspace status for a particular task. The simulator enables the user to observe the animation of the operation sequence from various view points.

In order to command a move, the user has to provide the desired end position and orientation. Then the simulator calculates each arm joint exact

path using the inverse kinematics solutions. A selection of consequent moves can then be created, edited, saved and retreived in *movement files*. The integrated path generator is responsible for the many easy ways to program a move [8].

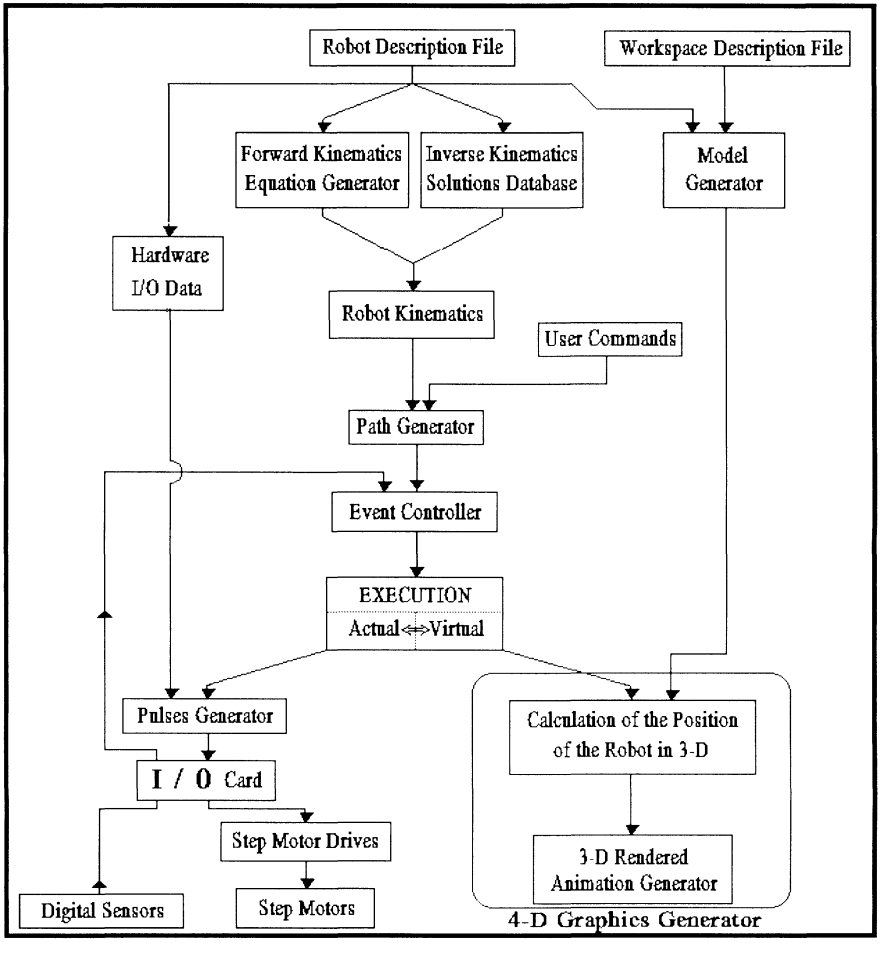

Figure 1: Block diagram of the developed system

There comes another unique feature of the designed system: The D-H representation constants can only give the forward kinematics solution. In all the robotic systems developed so far, the user had to provide the inverse kinematics solutions, which in most cases meant that a whole routine had to be written in a specific programming language. In our suggestion, a database is built in the system's software, and can cope with most manipulatory robotic arms with up to 6 degrees of freedom.

Two control modes are available to control the - real or simulated - robot. The first one, the *Teach Mode*, resembles a "teach-pendant" operation

in which the user interactively "moves" the robot and records the points along the desired trajectory path. In the second one, the *Program Mode*, a set of action commands is executed by the module to display or produce the motion of the robot.

After the path is produced for each and every one of the joints, a 3-D color graphic display is generated for the robot and its workspace. The produced image is based on the geometrical data loaded, the viewer's position, the viewing angle, the light sources and other viewing parameters, using a complicated camera model and the Binary Space Partitioning (BSP) rendering technique [9].

The speed of updating the display depends on the computational power of the system, the graphics hardware, the complexity of the 3-D models, the number of visible objects in the display window, the number of applications running at the same time under Windows, and - in a multi-user machine - the number of users. Of course, in a i80386 based PC, the desired rate of 25 frames per second is very rarely accomplished, therefore there is an option of turning the rendering off which results in a wireframe picture with low quality but much higher screen update rate. If needed, further improvement can be achieved by reducing the display window size.

The developed system, whose block diagram is shown in Figure 1, is based on the following main modules:

- Model generator
- Inverse kinematics solutions database
- 4-D Graphics Generator
- Event controller
- User interface / Help and Tutorial.

Model generator This module reads the data provided in simple ASCII files and builds 3-D models for the robot and its workspace. A file describing a robot must have the structure shown in Figure 2a, containing all the joints' types, the D-H representation constants, the geometrical data, ,the form of the pulses needed to trigger the motors' drives and the initial position . A file describing an workspace must have the structure shown in Figure 2b, and simply contain only the geometrical data of the objects in the workspace.

Inverse kinematics solutions database The robot's joint types and the D-H constants are checked with the ones in the database and, if matched, the appropriate inverse kinematics solutions are adopted. If a match is not found, the user has to provide the inverse kinematics solutions.

4-D Graphics Generator The display and animation are performed by this module (the 4th dimension is the time variable). Upon reading the models' data provided by the model generation module, the graphics library routines are called to generate 3-D motion frames on the display screen, like the one shown in Figure 3. The display window can be set to any size within the screen, and the scene of the robot and its workspace can be translated, rotated, and zoomed in and out. Two sets of information on the status of the simulated robotic system can be displayed on the screen. The first set consists of the values of all the joint positions, and the second one of the end effector's position and orientation.

```
CAD Enhanced
Robotic Station's
ROBOT DESCRIPTION
> GENERAL INFORMATION:
Model       : "xxx"
Joints      :   x
> JOINTS' INFORMATION:
# Type Min Max Step Rate
1:"xx" x.x x.x  x.x  xxx
2:"xx" x.x x.x  x.x  xxx
::  :   :   :    :    :
n:"xx" x.x x.x  x.x  xxx
> D-H REPRESENTATION:
#   d   a  theta  alpha
1  x.x x.x  x.x    x.x
2  x.x x.x  x.x    x.x
:   :   :    :      :
n  x.x x.x  x.x    x.x
> PULSES DESCRIPTION:
'On'  Voltage  : x.x
'Off' Voltage  : x.x
Pulse width    : x.x
> GEOMETRICAL DATA:
Points    : x
Lines     : x
Surfaces  : x
> POINTS PER LINK:
0  :  x
1  :  x
:  :  :
n  :  x
> POINTS' DATA:
0      x      y      z
1      x      y      z
:      :      :      :
n      x      y      z
> LINES' DATA:
0      a      b      c
1      a      b      c
:      :      :      :
n      a      b      c
> SURFACES' DATA:
0      x1    x2  ....  xn0
1      x1    x2  ....  xn1
:      :     :          :
n      x1    x2  ....  xnn
```

(a)

```
CAD Enhanced
Robotic Station's
WORKSPACE DESCRIPTION
> GENERAL INFORMATION:
Name       : "xxx"
Points     : x
Lines      : x
Surfaces   : x
> POINTS' DATA:
0      x      y      z
1      x      y      z
:      :      :      :
n      x      y      z
> LINES' DATA:
0      a      b      c
1      a      b      c
:      :      :      :
n      a      b      c
> SURFACES' DATA:
0      x1    x2  ....  xn0
1      x1    x2  ....  xn1
:      :     :          :
n      x1    x2  ....  xnn
```

(b)

Figure 2: Structure of the robot and workspace description files

Event controller This module checks the status of the sensors (if any) and decides the sequence of the motion commands to follow according to the programmed set of rules.

User interface / Help and Tutorial The user interface is very easy-to-use through multiple, sizable and movable windows, menus, buttons, dialogue boxes, prompts etc. Tutorial information is provided to the user on how to operate the system. At any instance, the user can stop the system operation flow and select an alternative option. The help provided is menu-driven and makes full use of the integrated Windows Help system.

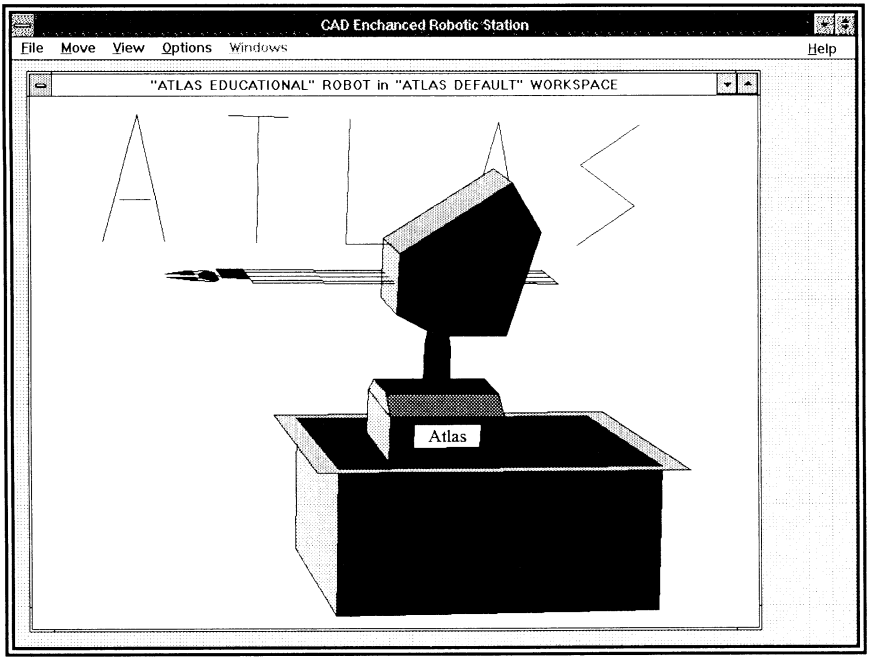

Figure 3: Zoomed side view of the ATLAS robot and its workspace

3 System implementation & performance validation

The simulation, animation, visualization and interactive control environment developed can simulate any kind of manipulatory robot and its actual workspace.

It has already been implemented and thoroughly tested with an ATLAS educational robot and a 32 channel I/O card. This robot has 4 rotary and one prismatic joint totaling 5 degrees of freedom. Since it was manufactured in 1982, it was equipped with a "primitive" microcomputer (6502), which communicated with an external computer named "VISA". The last could edit (in a text display), execute and store small and simple programs written in the built-in BASIC programming language.

The implementation of the developed system provided 4-D graphical visualization and simulation of the robotic installation, advanced features in movement editing, ability to virtual test a sequence of moves, and faster and more accurate control over the robotic arm servos, while dramatically improving the man-machine interface.

The system implementation showed that the developed approach is able to turn an old and unsophisticated robot to a CAD enchanced robotic station, at a very low cost and equally low effort.

4 Concluding remarks

The unique features of this CAD enchanced robotic station approach are the applicability to any step motor driven manipulatory robot, the low requirements in hardware, the machine transferability of the software (due to Windows NT), the fact that the inverse kinematics solutions are in most cases not required, and last but not least the extremely easy, cheap, and fast installation.

Future research is directed towards several enhancements. The first one is the support of mobile [10], and high power DC motor driven robots [11]. The simulation model should also be more complete after including the dynamics models of the robot and the objects in its workspace. Finally integration of voice activated commands and support of AutoCad and 3D Studio files as description files are in our future plans as well.

References

1 . C. Mirolo, E. Pagello *A solid modeling system for robot action planning*, The MIT Press, 1987.
2 . F. Dai, Collision-free motion of an articulated kinematic chain in a dynamic environment, *IEEE Computer Graphics Applications*, Vol. 9, No. 1, pp. 70-74, 1989.
3 . C. Chen, M. Trivedi, Task planning and action coordination in integrated sensor-based robots, *IEEE Transactions on Systems Man and Cybernetics*, Vol. 25, No. 1, 1995.
4 . M. Trivedi, C. Chen, A vision system for robotic inspection and manipulation, *IEEE Computer, Special Issue on Autonomous Intelligent Machines*, Vol. 22, No. 6, pp. 91-98, 1989.
5 . C. Chen, M. Trivedi, C. Bidlack Simulation and animation of sensor-driven robots, *IEEE Transactions on Robotics and Automation*, Vol. 10, No. 5, pp. 684-704, October 1994.
6 . Lee, Gonzalez, Fu *Robotics*, McGraw-Hill, New York, 1994.
7 . T. Lozano-Perez, *AI in the 1980's and beyond*, The MIT Press, 1987.
8 . C. Chen, M. Trivedi, Architecture and automatic task planning for integrated robotic systems, *Proceedings of the 1992 IEEE/RSJ Int. Conf. Intelligent robots and systems (IROS '92)*, North Carolina, July 1992.
9 . L. Heiny Windows graphics programming with Borland C++, Chapter 17, *Rendering Solid Objects*, pp. 499-534, John Wiley & Sons Inc. , New York, 1994.
10 . D. Todd, *Walking machines - An introduction to legged robots*, Chapman & Hall, 1985.
11 . R. Dillmann, M. Huck, A software system for the simulation of robot based manufacturing processes, *Robotics*, Vol. 2, pp. 3-18, 1986.