INTERACTIVE PLANNING USING GRAPHICAL SIMULATION FOR ROBOT TASK PROGRAMMING

Federico Thomas and Luis Basañez
Institut de Cibernètica (UPC – CSIC)
Diagonal 647, 2 planta
08028 Barcelona, Spain

ABSTRACT

Any robot programming environment would be greatly enhanced if the tedious process of specifying a task is performed automatically, allowing the user to state in terms of constraints the properties the task is supposed to have, without the need to manually adjust locations of the robot arm or of the workpieces to handle.

In order to reach this goal a robot programming environment should provide a friendly interactive way to specify a task and a powerful graphic simulation tool to show its execution, as well as the ability to interactively modify the program upon the results obtained.

This paper focuses on novel interactive planning aspects, providing some ideas that facilitate off-line programming of robotic tasks using an interactive graphic system. Some of them have been already implemented using as a test-bed a graphic simulator developed at the Institut de Cibernètica.

INTRODUCTION

The main goal of robot graphical simulation is to facilitate planning, programming and verification of robotic applications by solving, either automatically or with human assistance, some of the geometric and kinematic problems that arise in those applications.

Artificial Intelligence-based task planning systems, such as the one described in [1], with very low requirements for human interaction constitute the more promising approach to robot programming. Nevertheless, while waiting for the actual availability of such a systems, a realistic approach could be to develop interactive computer graphics tools capable of assisting the programming of robot tasks [11].

The already commercially available systems (GRASP [5], ROBCAD 3D [5] and some others) provide useful tools for: (a) robotic cell layout; (b) graphical simulation of robot motions; (c) automatic detection of collisions; (d) specification of initial values for certain parameters; and (e) some limited dynamic analysis.

Nevertheless, a number of difficult problems remains unsolved in CAD-based robot programming environments. These include: (a) representation and analysis of tolerancing specifications and other forms of uncertainty; (b) modelling, analysis and verification of robot programs incorporating sensors; (c) the use of geometric models to represent task constraints in compliant motion control [12]; (d) automatic updating of robot programs upon modifica-

tion of objects design data; (e) full automatic planning of collision free motions; (f) symbolic description of transformations in terms of kinematic constraints; and (g) semantic reference to subparts of workpieces.

Each of these unsolved problems leads to the corresponding subsystem to be developed for assisting the programming of robot tasks, until artificial intelligence based task planning systems be fully implemented.

THE PROGRAMMING ENVIRONMENT

A test-bed environment [13] has been conceived as an useful and practical tool for off-line programming of robots handling parts in highly cluttered robotic cells. Models of the objects are created in ROBMOD [3] and translated to the internal representation of the system and shaded animated hidden-surface-eliminated pictures of the scene are produced.

The package provides key elements for solving most common geometric problems in automatic programming of assembly robots, including fast intersection detection. It also provides an interpreter of VAL-II [15], allowing the user to simulate programs written in this language, and a communications module [6] for terminal emulation of the MK-II PUMA controller and its teach pendant. This module is also able to obtain information about the state of the robot and to download robot programs.

The simulation system, written in the C language, is designed to run on any UNIX machine, except for the representation module and the interface with the user, presently implemented in SunCore [2] and SunWindows respectively, which are the only machine-dependent elements of the package.

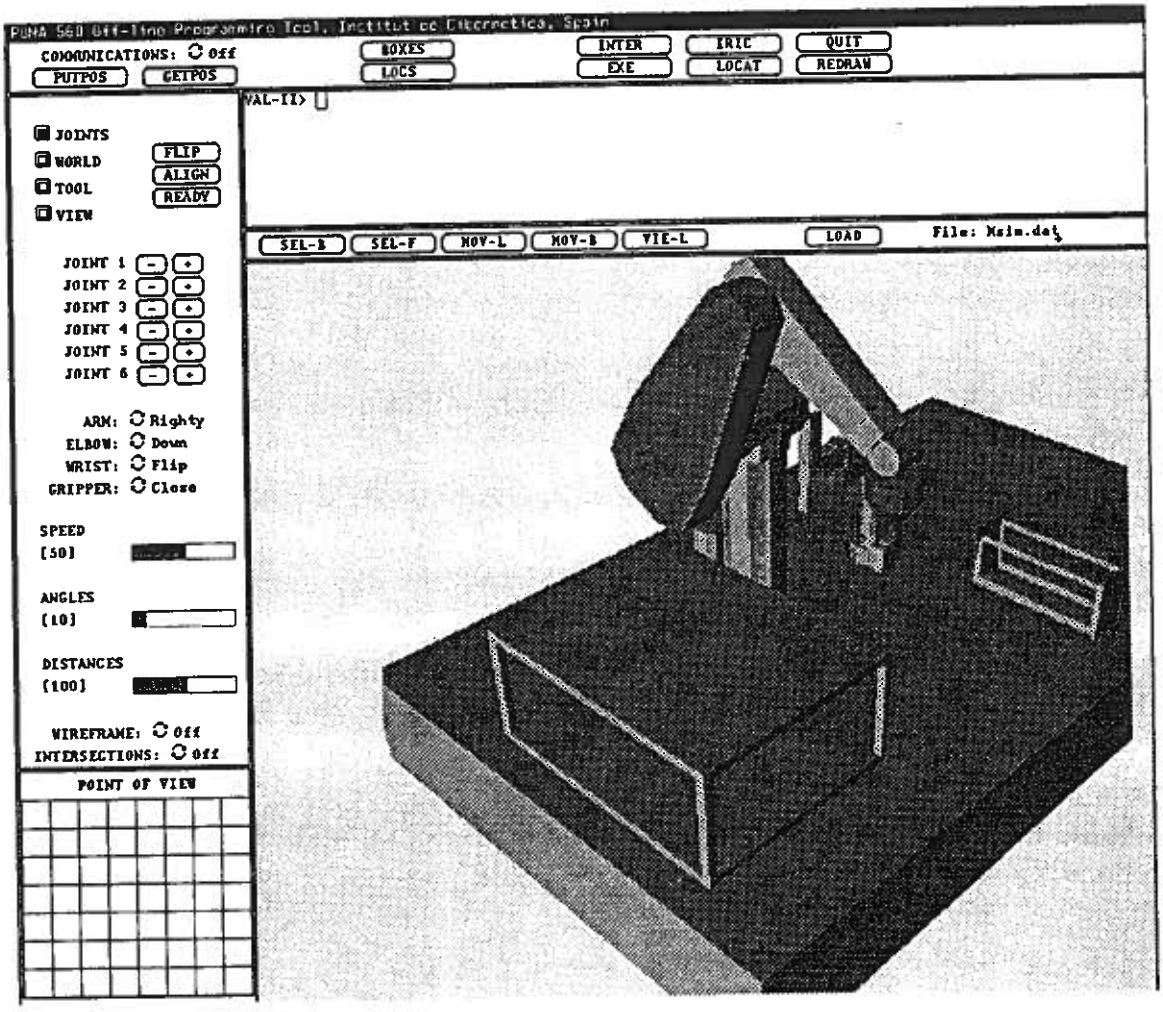
Additional modules allow the system to incorporate real time information from the actual robotic cell as supplied by physical sensors. Till now, two such modules has been developed.

The first one makes use of a Matrox based vision system to determine geometric parameters of parts present in the cell, the model of these parts having been previously defined. Then, the models are instanced with the measured parameters values and included in the simulation. The vision system also provides the position and orientation of the parts.

The second module gets 3D parts information by means of structured laser light, in combination with the Matrox vision system. This module is able to deal with unmodelled parts that, in this way, can be incorporated in the simulation and taken into account in the programming of the task.

INTERACTIVE PLANNING

In general, in any robotic application some elements (either components of the robot or workpieces) have to be located in the available working space following a predefined sequence. This can be done in an easier way by specifying spatial relationships among the components instead of giving absolute transformations. To this end, we must be able of referring to these



Test-bed environment designed for implementing the interactive planning concepts discussed herein.

components and of getting their final location given multiple spatial relationships. We have built a module that accepts a set of geometric relationships between the moving object and a fixed reference and returns, if possible, the corresponding transformation [4].

Any relationship given by the user must involve an element of the moving body, an element of the fixed world, and the kind of relation that holds between them. As elements one can take a point (usually a vertex of a body), a line (an edge), or a plane (a face), and as relationships, coincidence, parallelism, perpendicularity, or angle formed, in any combination that makes sense.

The set of relationships given by the user is transformed into a set of purely rotational constraints and a set of purely translational constraints. Purely rotational constraints are of two types: parallelism of vectors, and angle formed by two vectors. Perpendicularity is treated as a particular case of angle formed. Purely translational constraints express the fact that a point lies on a plane, a line, or another point. The process followed to find the transformation involves first solving the rotational part, and then the translational one [4]. The system is not complete, in the sense that it is not able to solve the problem whenever the set of relationships determines a unique transformation.

It would be also interesting to introduce relationships involving subparts of the workpieces, such as grooves, holes, etc. This problem is closely related to the topic of *feature recognition* and, unfortunately, it is still an open problem [10]. In fact, the number of different processes, other than graphics, which can be actually performed in solid models is still small [17].

Once the sequence of motions have been specified, one should ensure that they are free of collisions. There have been many robot simulators in the past that perform some sort of collision detection. Several of these were designed to perform collision avoidance as well. References [8], and [9] are good examples of early developments. It is important to have a system that performs collision detection with the entire arm, each time it moves. In the implemented system, detection is done by moving the simulated robot small increments along its trajectory and performing a static collision check at each point using a fast method that employs a hierarchy of enclosing objects.

The significance of the algorithm used [14] is that it detects intersections between non-convex polyhedra by simply checking sign changes of some vector triple products, which only involve vertices of the polyhedra. Therefore, no new geometric entities are constructed, contrary to the usual way of dealing with non-convex cases.

Collision avoidance implies the more difficult problem of proposing a path around an obstacle once a potential collision has been detected. The collision avoidance problem can be solved by using a potential field approach [7] by translating non-intersection constraints into "energy" functions on the position parameters, non negative functions with zeroes at the goal location.

Spatial relationships can also be formulated in terms of energy functions, mainly those for which the above mentioned algebraic method is unable to deal with. Then, since energy functions compose by addition, the solution to a system of constraints is the solution to a single equation, the sum of the energy terms. This can be seen of an extension of the potential field approach to path-planning to the more general problem of specifying robot tasks.

While conventional algebraic methods return no solution to an overdetermined system, the energy minimum solution is tolerant to over- and under-determined systems, so it can be trapped in local minima. The solution proposed in [16] to this problem is again user interaction. Since constraints are satisfied by moving through a curve in parameter space using a numerical method, the constraint solving process can itself be animated, permitting the user to assist the solver in escaping local energy minima. As already pointed out in [16] such minima are usually easy to interpret geometrically, then the user can often correct the situation by manually repositioning a part. In fact, the user can literally push or pull on parts of the models with the mouse pointer, introducing a time-varying energy term into the equation, to bump it out of local minima.

Then, as a summary, the basic interaction required for robotic assembly task planning, at the level presented here, is performed through two basic operations: (a) interactive selection of faces and subparts (features) of the elements in the work cell; and (b) interactive application of virtual forces using a pointing device.

Besides these two basic interactive operations, commands from the user are also accepted at any time, which are basically of two kinds: VAL sentences and representation commands.

CONCLUSIONS

We have focused on interactive planning aspects, providing some ideas in order to facilitate off-line programming of robotic assembly tasks using an interactive graphic system. It has been shown that, while the interactive reference to elements and subparts of workpieces are of great interest for easy specification of assembly tasks, the application of virtual forces is useful during the planning process to aid the planner to escape from dead ends.

The aim of the presented environment is twofold: to get deeper insight in the problem of automatic programming of robots, and to provide an efficient and well-structured system on top of which it would be easy to build other environments with higher capabilities. The system briefly described can be seen as an implemented intermediate step towards a more sophisticated one [1] that would allow, for example to deal with knowledge data bases and to apply artificial intelligence tools.

Other capabilities, such as performance analysis involving time variables (dynamic behaviour of the robot, cycle time of a repetitive task, etc.), seem much more difficult to be introduced. Presently, the current implementation, as many other early simulators, do not account for dynamics. They repond ideally with no slop or backslash in the joints. Feedback control dynamics is considered instantaneously. There are no gravitational or inertial effects. The links have infinite acceleration and deceleration, therefore having no overshoot errors.

Finally, as with any simulator, the accuracy of the model limits the extent that the system can be used. The graphic system introduces the errors associated with the definition of the workpieces and their actual location in the workcell. The automatic generation of sensory strategies is an important point for future developments.

ACKNOWLEDGEMENTS

This work has been partially supported by the CICYT under the project ROB 89-0287 (*Sistema de percepción con integración multisensorial para robótica y automatización*).

REFERENCES

[1] L. Basañez, C. Torras, J. Ilari, and A. Sanfeliu, "Operation Specialists for Automatic Programming and Monitoring of Robotic Assembly Cells", *Journal of Robotics and Computer Integrated Manufacturing*, Vol. 6, No. 4, pp. 269-276, 1989.

[2] R.D. Bergeron, P.R. Bono, and J.D. Foley, "Graphics Programming Using the Core System," *ACM Computer Surveys*, Vol. 10, No. 10, December 1978.

[3] S. Cameron, and J. Aillet, "ROBMOD: A Geometric Engine for Robotics," *IEEE International Conference on Robotics and Automation*, pp. 880-885, Philadelphia, U.S.A. 24-29 April 1988.

[4] E. Celaya, "LMF: A Program for Positioning Objects Using Geometric Relationships," *AIENG 92 (Applications of Artificial Intelligence in Engineering)*, July 1992, University of Waterloo, Ontario (Canada).

[5] S. Derby, "GRASP. From Computer Aided Design to Off-lina Programming," *Robotics Age*, February 1984.

[6] A. Izaguirre, "A C library to Control a PUMA 560 Using the DDCMP Protocol," *GRASPLab Report*, CIS Dep., Univ. of Pennsylvania, 1984.

[7] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The Int. Jour. of Robotics Research*, Vol. 5, No. 1, 1986.

[8] J.K. Myers, "RCODE: The Robotic Simulator with Collision Detection," *Tech Note SRI International*, December 1984.

[9] A. de Pennington, M.S. Bloor, and M. Balila, "Geometric Modelling: A Contribution Towards Intelligent Robots," 13^{th} *ISIR*, pp. 7.35-7.54, Chicago, April 7-21, 1983.

[10] M.J. Pratt, "Solid Modeling and the Interface Between Design and Manufactures," *IEEE Trans. on Comp. Graphics and Applications*, July 1984.

[11] P. Sjolund and M. Donath, "Robot Task Planning: Programming Using Interactive Computer Graphics," 13^{th} *ISIR*, pp. 7.122-7.135, Chicago, April 17-21, 1983.

[12] P. Simkens, J. de Schutter and H. Van Brussel, "Graphical Simulation of Compliant Motion Robot Tasks," *INCOM 1989*.

[13] F. Thomas, J. Ilari and L. Basañez, "Paquete gráfico para la programación fuera-de-línea y simulación de un robot PUMA 560," *First Congress of the Spanish Robot Association*, November 1989.

[14] F. Thomas, and C. Torras, "Exact Interference Detection between Non-Convex Polyhedra," *submitted for publication.*

[15] Unimation Inc., "User's Guide to VAL-II", Unimation Inc., Danbury, Conn., Version 1.1, August 1984.

[16] A. Witkin, K. Fleischer and A. Barr, "Energy Constraints on Parametrized Models," *Computer Graphics*, vol. 21, No. 4, July 1987.

[17] J.R. Woodwark, "Some Speculations on Feature Recognition," *Computer-Aided Design* 20, 4, May 1988.