An Adaptive and Efficient System for Computing the 3-D Reachable Workspace

Tarek Alameldin, Norman Badler and Tarek Sobh

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania

Abstract

In this paper, we present an efficient system for computing the 3-D reachable workspace for redundant manipulators with joint limits. In our system, we have decomposed the 3-D reachable workspace problem into two major subproblems: workspace point generation and surface computation. We describe different algorithms that are used to build these modules. Finally, we discuss the advantages offered by our system.

1 Introduction

The reachable workspace of a manipulator is the volume or space encompassing all points that a reference point P on the end effector traces as all the joints move through their respective ranges of motion [4, 8]. The problem of computing the workspace for a redundant manipulator has applications in a variety of fields such as robotics, computer aided design, and computer graphics. Although the workspace problem has long been on the agenda of researchers in robotics, they have not formulated a satisfactory and general solution. A workspace is said to have a hole if there exist at least one straight line which is surrounded by the workspace yet without making contact with it [11, 6]. A workspace is said to have a void if there exist a closed region R, buried within the reachable workspace, such that all points inside the bounding surface of R are not reached by the manipulator [11, 6].

The first efforts to compute the manipulator workspace, based on its kinematic geometry, started in the mid 1970's [8, 4]. They proved that the extreme distance line between a chosen point on the first joint axis and the center point of the end effector intersects all intermediate joint axes of rotation. However, the above result is not valid if any intermediate joint axis is parallel to the extreme distance line, two joint axes intersect, or any joint is not ideal (has limits). Kumar and Waldron [5] presented another algorithm to compute the manipulator's workspace. In their analysis, an imaginary force is applied to the reference point at the end effector in order to achieve the maximum extension in the direction of the applied force. The manipulator reaches its maximum extension when the force's line of action intersects all joint axes of rotational joints and is perpendicular to all joint axes of prismatic joints (since the moment of the force about each joint axis must be zero). Every joint of the

CH2872-0/90/0000-0503 © $1.00 1990 IEEE

manipulator can settle in either of two possible positions under the force action. Hence, the algorithm results in 2^{N-1} different sets of joint variables for a manipulator of N joints in the direction of the applied force. The concept of stable and unstable equilibrium is used to select the set of joints variables that results in the maximum extension in the force direction. This algorithm has exponential time complexity and only deals with manipulators that have ideal joints (without limits). Tsai and Soni [9, 10] developed another algorithm to plot the contour of the workspace on an arbitrarily specified plane. The algorithm suffers from the limitations of not only being restricted to computing 2D workspace cross sections but also high computational cost. The previous published workspace algorithms suffer from one or more of the following drawbacks:

- High computational cost.
- Computing only 2-D cross sections. This is not adequate for manipulators with joint limits since the workspace is not axi-symmetrical. In addition, two dimensional workspace representation forces the user to memorize the data set by looking at multiple displays before making an interpretation or a decision.
- Dealing only with manipulators that have specialized geometry.
- Sensitivity to geometrical and numerical errors or approximations.

2 System Design

In our system, the 3-D reachable workspace problem is decomposed into two major subproblems, workspace point generation and surface computation. The point generation module computes workspace points by direct kinematics based techniques. The surface computation module extracts the workspace contours utilizing a subset of the points generated by the first module.

2.1 Workspace Point Generation

We use direct kinematics based algorithms to compute a dense set of the reachable workspace points. A redundant manipulator is modeled as a series of links connected with either revolute or prismatic joints. We assume without loss of generality that each joint has one degree of freedom. A joint with m degrees of freedom is modeled as m joints connected with links of zero lengths, each of these joints has a lower limit and an upper limit. The first link of the manipulator (link 1) is connected to the base (link 0) by joint 1. The final link, denoted by the end effector (link n) has no joint at its end. A coordinate frame is attached to each link. A homogeneous matrix A is used in order to describe the relationship between consecutive frames [7, 2]. The elements of matrix A are computed by using D-H notations for both prismatic and revolute joints. The description of the end effector (link n) with respect to the base, denoted by T_n, is given by $T_n = A_1 A_2 ... A_{n-1} A_n$. The computational cost of computing each point is O(n), where n is the number of degrees of freedom that are associated with joints in the path from the end effector (distal linkage) to the proximal linkage.

2.2 Surface Computation

Workspace points that are computed by the above module don't necessarily lie on the surface boundary. We use an edge detection algorithm in order to obtain the workspace boundary as well as the holes and the voids that are buried inside the reachable workspace. This can be achieved by computing the dimensions of the cube that encompasses the workspace points. This cube is divided into cells according to the required resolution of the application. If the cell contains a workspace point, it is marked by one and zero otherwise. A workspace cell is considered a boundary cell if any of its neighbors is marked by zero. A geometric primitive (triangle or polygon) is used to interconnect the detected contours, and finally the resulting geometric representation is rendered. This can be achieved by using optimal surface reconstruction mechanisms developed by Fuchs [3] for planar contours. Those techniques utilize both graph theory and optimization procedures for solving the triangulation problem. We use the "shortest diagonal" algorithm developed by Christiansen [1] since it is very fast and works well for consecutive contours which are mutually centered and of similar size and shape. This algorithm lends itself easily to the workspace problem since the workspace boundaries exhibits similar properties.

3 Conclusions

Our system offers the following advantages:

- **Modularity:** Decomposing the workspace problem into independent modules allows incorporating new techniques into the system, specially in the surface fitting module which is a very active area of research. In addition, it becomes easier to maintain and debug the implemented system based on modular design.

- **Flexibility:** Our system allows the user to select between low and high workspace surface quality based on the application type and deadline. This is achieved by controlling the number of workspace points that are used by the surface computation module. None of the existing approaches offers this capability.

- **Parallel Implementation:** Our system lends itself easily to parallel implementation. The workspace point generation module can be implemented in parallel by dedicating each processor to compute different workspace points. The surface computation module can also be implemented in parallel by assigning each processor to compute a contour, and finally assigning each processor to compute different inter-slice connection.

- **Adaptability:** According to the nature of the application, the system allows the user to compute the reachable workspace envelope with or without holes and voids, based on the application requirements.
- **Generality:** This system uses efficient and adaptive algorithms for computing the 3-D reachable workspace of articulated linkages, not only those with redundant degrees of freedom, but also those with joint limits. The computational complexity of the system is linear in terms of the number of degrees of freedom.

References

[1] H. Christiansen and T. Sederberg. *Complex Contour Line Definitions into Polygonal Element Mosaics. Computer Graphics*, 187–192, August 1978.

[2] J. Craig. Introduction to Robotics Mechanics & Control. Addison Wesley, 1986.

[3] H. Fuchs, Z. Kedem, and S. Uselton. *Optimal Surface Reconstruction from Planar Contours. Communications of The ACM*, 693–702, October 1977.

[4] A. Kumar. Characterization of Manipulator Geometry. PhD thesis, University of Houston, 1980.

[5] A. Kumar and K. Waldron. *The Workspace of a Mechanical Manipulator. ASME Journal of Mechanical Design*, 103:665–672, July 1981.

[6] T. Lee and D. Yang. *On the of Manipulator Workspace. Journal of Mechanisms, Transmissions, and Automation in Design*, 105:70–77, March 1983.

[7] R. Paul. Robot Manipulators: Mathematics, Programming, and Control. MIT Press, Cambridge, Mass., 1981.

[8] B. Roth. *Performance Evaluation of manipulators from a Kinematics Viewpoint. NBS Special Publication*, 39–61, 1975.

[9] Y. Tsai and A. Soni. *Accessible Region and Synthesis of Robot Arms. ASME Journal of Mechanical Design*, 103:803–811, October 1981.

[10] Y. Tsai and A. Soni. *An Algorithm For the Workspace of a General n-R Robot. ASME Journal of Mechanical Design*, 105:52–57, July 1983.

[11] D. Yang and T. Lee. *On The Workspace of Mechanical Manipulators. Journal of Mechanisms, Transmissions, and Automation in Design*, 105:62–69, March 1983.