A Hybrid System For Computing Reachable Workspaces For Redundant Manipulators

Tarek K. Alameldin, Norman Badler, and Tarek Sobh

Department of Computer and Information Science
University of Pennsylvania, Philadelphia, PA 19104-6389

Abstract

An efficient computation of 3D workspaces for redundant manipulators is based on a "hybrid" algorithm between direct kinematics and screw theory. Direct kinematics enjoys low computational cost, but needs edge detection algorithms when workspace boundaries are needed. Screw theory has exponential computational cost per workspace point, but does not need edge detection. Screw theory allows computing workspace points in prespecified directions, while direct kinematics does not. Applications of the algorithm are discussed.

1 Introduction

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. 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]. In computer aided design, three dimensional workspace for a human limb or robotic manipulator can be used in designing the interior of cars, tanks, or space vehicles. Different panels and keys can be repositioned within the 3-D reachable workspace. These different configurations can be tested before being manufactured. In robotics, the workspace problem has long been on the agenda of researchers; however, 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 [14,6]. 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 [14,6].

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 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). The reachable workspace boundary for an articulated chain that has three degrees of freedom or less can be represented by explicit equations [10]. Unfortunately, the reachable workspace boundary for an articulated chain that has more than three degrees of freedom is hard to desribe by explicit equations [11]. The computational complexity of the workspace problem was analyzed in [2] and the problem is proved to be at least NP hard. Tsai and Soni [10,11] 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.

In the next two sections, we describe algorithms that use direct kinematics and screw theory in order to compute the reachable workspace. Each class of these algorithms has advantages and disadvantages. Instead of debating the merits of these algorithms, we integrate them as described in section four.

2 Algorithms Based on Direct Kinematics

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 in order to describe the relationship between two consecutive links as illustrated in figure 1. A homogeneous matrix A is used in order to describe the relationship between consecutive frames [7,3]. The elements of matrix A are computed by using D-H notations for both prismatic and revolute joints. The transformation matrix A for a revolute joint is:

$$A_i = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix A for a prismatic joint is:

$$A_i = \begin{bmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i S\alpha_i & 0 \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

a_i: is the common normal distance between the two axes of j_i and of j_{i+1}.

α_i: is the angle between the two axes of j_i and j_{i+1} in a plane perpendicular to a_i.

d_i: is the distance between the normals of joint axis i[1] and is measured along the axis of joint i.

θ_i: is the angle between the normals of joint axis i and is measured in a plane normal to the axis.

In the case of a revolute joint θ_i is called the *joint variable* and the other three fixed quantities (d_i, a_i, α_i) are called the *link parameters*. On the other hand, d_i is the joint variable for a prismatic joint and the other three fixed quantities $(\theta_i, a_i, \alpha_i)$ are called the link parameters.

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.

Workspace points that are computed by direct kinematics 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

[1]The joint axis i has two normals to it, one for link $i-1$ and one for link i.

point, it is marked with one and zero if it does not contain a reachable point. A workspace cell is considered a boundary cell if any of its neighbors is marked with zero.

3 Algorithms Based on Screw Theory

Kumar [4,5] did pioneering work in computing workspace boundary points by using screw theory. He used the fact that the manipulator is in singular configuration when the end effector reference point is positioned at a workspace boundary point. This configuration occurs when all active[2] joint axes are all reciprocal to a zero pitch wrench (force) axis[3] [5,12,13,9]. This is evident because the wrench will create moments about the joint axes that are not reciprocal to the wrench axis if the reciprocal condition is not satisfied. Accordingly, these moments cause those joints to move until the reciprocal condition is satisfied or until they reach one of their limits. For a revolute joint (screw axis of zero pitch), the reciprocal condition is satisfied when the wrench axis has either finite or infinite intersection with the joint axis [5,9]. For a prismatic joint, the reciprocal condition is met when the wrench axis is perpendicular to the joint axis [9].

A wrench of zero pitch (force) is applied to a reference point on the end effector in order to compute a workspace boundary point in the force direction. Then, the direction of the force is changed to sweep either the entire workspace boundary. A closed form algorithm that computes the joint variables satisfying the above conditions is used. The algorithm generates 2^{n-1} different surfaces which bound different workspaces for a manipulator of n joints. These different surfaces result from the fact that each joint can assume one of two positions under the applied force. One of these positions corresponds to a stable equilibrium while the other corresponds to unstable equilibrium. These two positions can be distinguished by computing the work done by the applied force to the end effector when the joint is disturbed from its equilibrium position. If the work done is positive, the disturbed joint is in unstable equilibrium since the force continues doing work until the joint takes up a stable equilibrium position. On the other hand, if the work done is negative, the disturbed joint is in equilibrium position since the applied force causes the joint to return to its initial position when the disturbing torque is removed. Hence, this type of algorithms traces re-entrant surfaces as well as the boundary surfaces. These re-entrant surfaces are non-crossable when the manipulator is positioned in the configuration that traces those surfaces, i.e., they represent barriers inside the workspace and affect the manipulator's controllability. The following example illustrates this approach.

Example

Consider a planar manipulator that has three ideal revolute joints with parallel axes as shown in figure 2. A force is applied to a reference point at the end effector. The manipulator will be in its

[2] A joint is termed inactive when it reaches one of its limits.

[3] In screw theory, two screw axes are called reciprocal to each other when the wrench applied about one screw axis does no work about the other screw axis.

extended positions when the force line of action intersects all joints' axes. The contours which bound the workspace boundary as well as the interior surfaces result from rotating these extended positions about the first axis. Figure 3 illustrates four different workspace contours. The workspace envelope results when all joints are in equilibrium positions.

Finally, this class of algorithms provide surface classification based on the stability of different joints under the force application; however, it cannot detect holes and voids in the workspace. The ability to distinguish holes and voids that are buried inside the workspace can be achieved by integrating this class of algorithms with the direct kinematics based algorithms as described in the next section.

4 Hybrid Algorithms

We have discussed the use of direct kinematics based algorithms in computing reachable workspace points. The computational cost of computing each workspace point is linear; however, it is impossible to predetermine the number of workspace points that has to be computed in order to generate the workspace. A cell marked with zero does not necessarily mean that it is unreachable since the direct kinematics based algorithm might not had computed enough workspace points. On the other hand, screw theory based algorithms have exponential computational cost and should not be used solely in computing the workspace. In the hybrid algorithm, we use direct kinematics to compute the workspace points as an initial approximation. Then, we compute the reachable cube and mark the reachable cells with one. Screw theory based algorithms are used to decide the reachability of cells marked with zero. This can be achieved by applying a force in the direction of the line joining the base and the cell marked with zero. Finally, an edge detection algorithm is used to extract the workspace boundary as well as the holes and voids that are buried inside the workspace. This boundary is used to construct the 3D reachable workspace without user intervention as described in [1].

5 Conclusions

Algorithms based on direct kinematics enjoy low computational cost but can not be used solely to compute the workspace. On the other hand, screw theory based algorithms have exponential computational cost per workspace point and they can not distinguish holes and voids that are buried inside the workspace. We illustrated how to integrate both classes of algorithms into a "hybrid algorithm". The hybrid algorithm is efficient and can detect holes and voids.

References

[1] T. Alameldin, N. Badler, and T. Sobh. *An Adaptive and Efficient System for Computing the 3-D Reachable Workspace.* In *Proceedings of The 1990 IEEE International Conference on Systems Engineering*, pages 503–506, 1990.

[2] T. Alameldin, M. Palis, M. Rajasekaran, and N. Badler. *On the Complexity of Computing Reachable Workspaces for Redundant Manipulators.* In *Proceedings of SPIE Intelligent Robots and Computer Vision IX: Algorithms and Techniques*, 1990.

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

[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] M. Tsai. Workspace Geometric Characterization and Manipulability of Industrial Robots. PhD thesis, Ohio State University, 1986.

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

[11] 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.

[12] R. Vijaykumar. Robot Manipulators - Workspaces and Geometrical Dexterity. Master's thesis, Ohio State University, 1985.

[13] R. Vijaykumar, K. Waldron, and M. Tsai. *Geometric Optimization of Serial Chain Manipulator Structures for Working Volume and Dexterity. International Journal of Robotics Research*, 5:91–104, 1986.

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

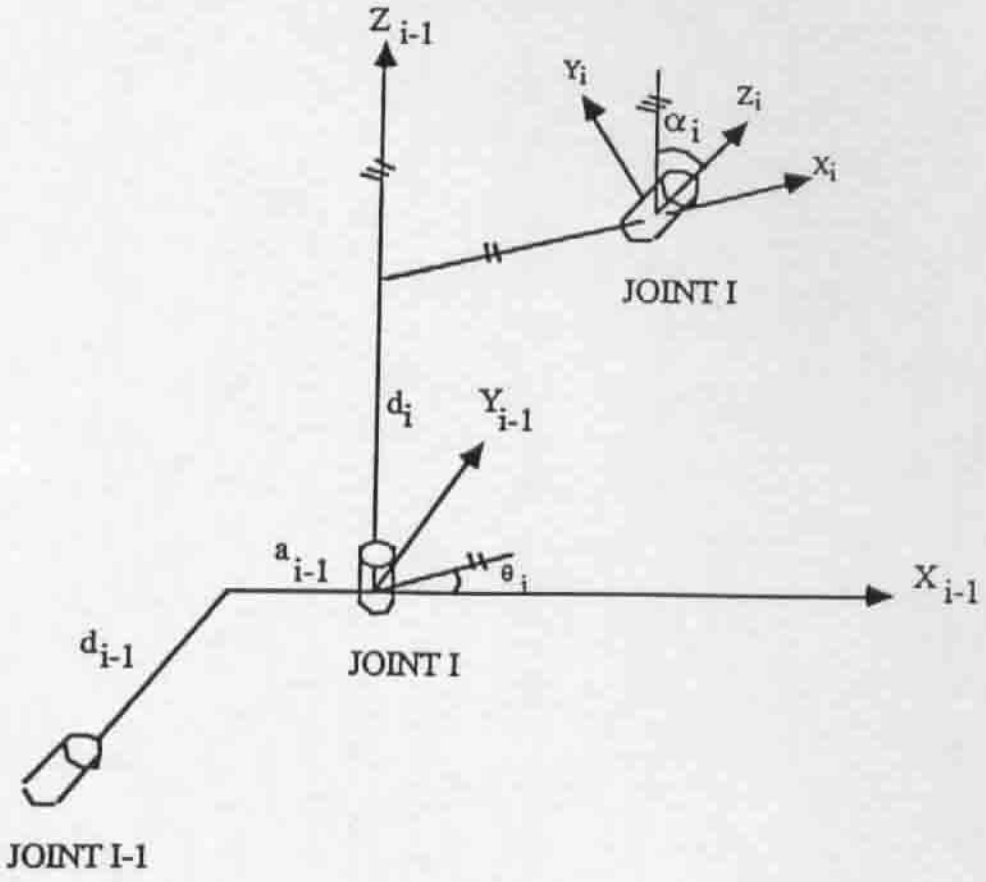

Figure 1: The relationship between consecutive links.

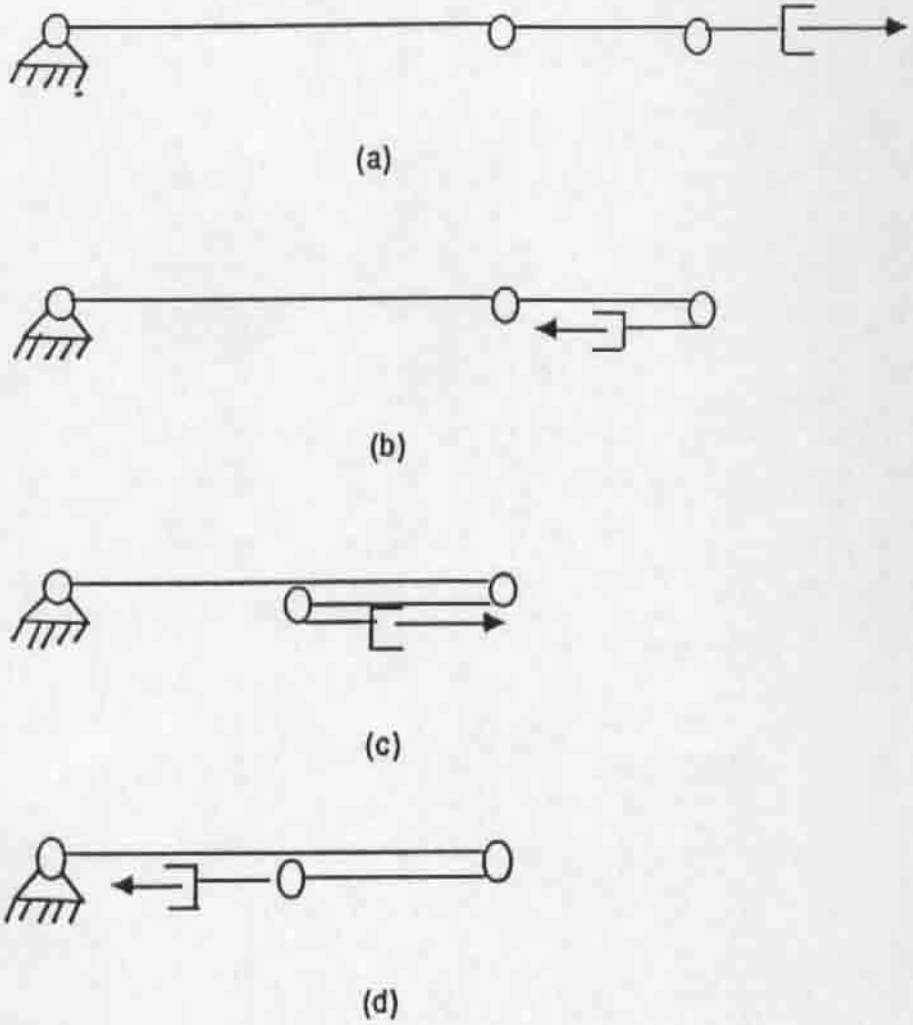

Figure 2: Different equilibrium positions for a three links manipulator.

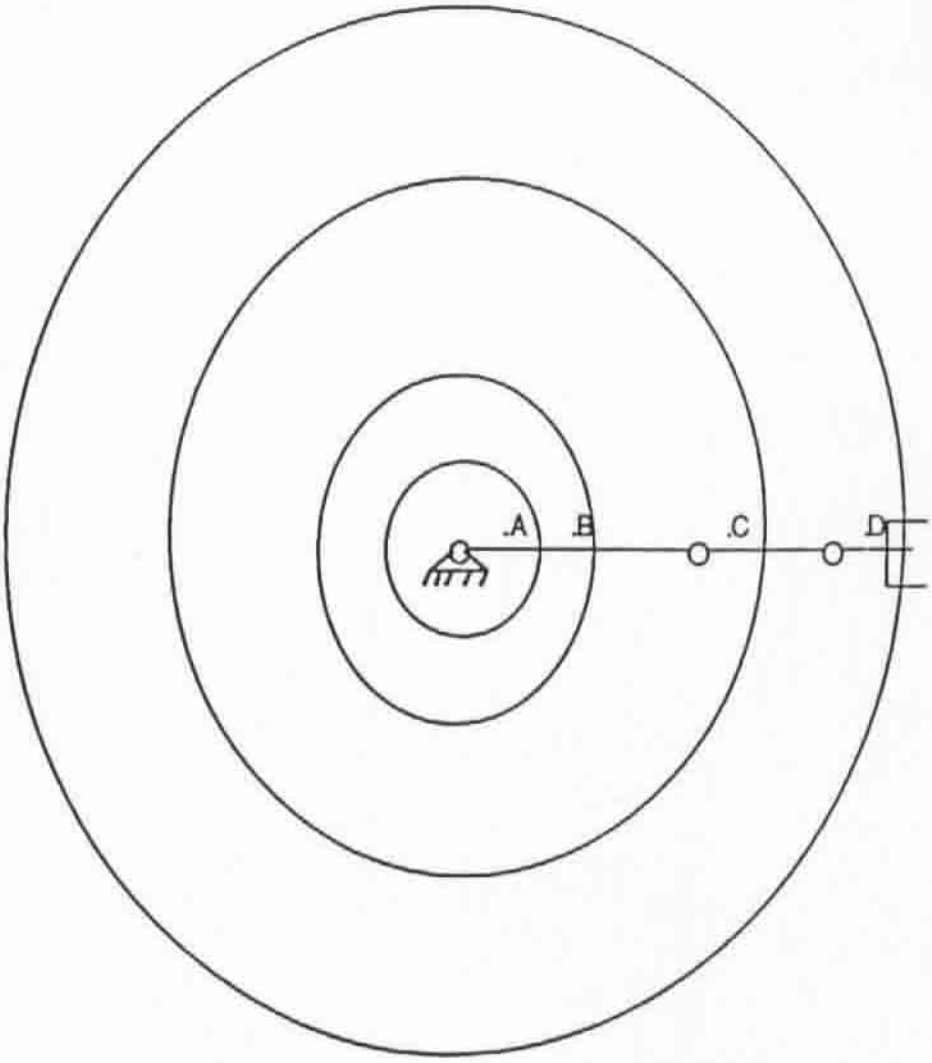

Figure 3: Different workspace contours.