Proceedings of the 1996 IEEE
International Conference on Robotics and Automation
Minneapolis, Minnesota - April 1996

A Combinatorial Approach to Trajectory Planning for Binary Manipulators

David S. Lees*
Gregory S. Chirikjian†

Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218

Abstract

Binary manipulators are powered by actuators which have only two stable states. Therefore, they can reach only a discrete (but possibly large) number of locations. Compared to a manipulator built with continuous actuators, a binary manipulator provides reasonable performance, and is relatively inexpensive (up to an order of magnitude cheaper). The number of states of a binary manipulator grows exponentially with the number of actuators. This makes the calculation of its inverse kinematics quite difficult. This paper presents a combinatorial method for computing the inverse kinematics of a binary manipulator that reduces the search space to a manageable size. It also creates reasonably smooth motions that follow a specified trajectory accurately (in both position and orientation), despite the discrete nature of binary actuation.

1 Introduction

Most robots available today are powered by continuous actuators such as DC motors or hydraulic cylinders. Continuously actuated robots can be built to be precise and to carry large payloads, but they usually have high price/performance ratios, as evidenced by the high cost of the industrial robots available today. In contrast, discrete actuators such as solenoids and pneumatic cylinders are relatively inexpensive. "Hyper-redundant," discretely actuated robots are a promising alternative to traditional robots for certain applications. A hyper-redundant robot can be built by stacking variable geometry trusses (VGT's) on top of each other in a long serial chain (See Figure 1). This approach yields a structure with good stiffness and load-bearing capabilities at a low cost, compared to traditional non-redundant robots. Using discrete, rather than continuous, actuators to power a truss-based robot increases the reliability and lowers the cost of the system even further.

The characteristics of binary VGT manipulators make them well suited to a number of tasks. They could be used for inspection or repair in constricted spaces, where the flexibility and compactness of the VGT structure is a distinct advantage. They are also candidates for use in human service applications, where good performance is needed, along with low cost. Finally, they are potentially useful in situations where a very small robot is needed [1], since small discrete actuators are easier to fabricate than small continuous actuators.

While the hardware costs of a binary manipulator are lower than for a continuously actuated manipulator, there is a tradeoff in the complexity of the trajectory planning software. The number of possible configurations of a binary robot grows exponentially with the number of actuators. For example, a binary robot with 30 actuators has 2^{30} (approximately 10^9) distinct states, which makes the exhaustive enumeration of all of its states impractical. The large number of possible states of a binary manipulator prevents us from computing its inverse kinematics with a brute force search except for manipulators with relatively few actuators.

This paper describes an efficient combinatorial method for computing the inverse kinematics and planning the trajectory of binary manipulators. It searches for a solution by changing only a small number of the manipulator's actuators at any given time. This approach reduces the size of the search space considerably, and because only a small number of actuators change state, it produces very smooth robot motions.

2 Previous Work

This section reviews two distinct, but complementary bodies of literature. First, previous work on the kinematics of highly redundant manipulators is dis-

*E-mail: lees@polaris.me.jhu.edu
†E-Mail: greg@polaris.me.jhu.edu
This work was supported by a young investigator award and a presidential faculty fellow award to the second author from the National Science Foundation

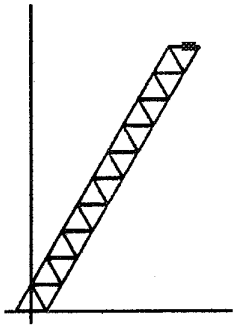

Figure 1: A 10 module binary manipulator built with variable geometry trusses.

cussed. Then, a brief history of the development of binary robots is presented.

2.1 Kinematics of Redundant Manipulators

A number of efforts have been made to study the kinematics and control of hyper-redundant manipulators with continuous actuators, e.g, [2] and references therein. Chirikjian presents the forward kinematics of a truss based manipulator in [3]. He has also demonstrated an approach to computing the inverse kinematics of a binary manipulator by making it adhere to a specified curve [4]. Hughes [5], has analyzed the characteristics of VGT based manipulators, including finding estimates of their stiffness using finite element methods. The kinematics of hyper-redundant VGT truss manipulators embody elements of the kinematics of both serial and parallel mechanisms. This is because an individual module of a VGT manipulator is a parallel mechanism, while the complete manipulator, composed of a stack of VGT modules, looks more like a serial structure. It is also possible to build serial-revolute hyper-redundant manipulators that are structured like mechanical "snakes." The forward kinematic analysis for such a mechanism is simpler than for a VGT based manipulator, but a serial-revolute robot is not nearly as stiff as a VGT based robot.

2.2 Previous Efforts in Minimalist and Binary Manipulation

Since the high price/performance ratio of most robots makes them impractical for many potential applications, efforts to develop inexpensive, but capable, robots have begun to gain momentum. For example, Canny and Goldberg [6] have proposed a reduced complexity paradigm for robotic manipulation. There have also been several efforts to develop reliable sensorless manipulation [7, 8]. In sensorless manipulation, the geometric constraints of a task are exploited to create a manipulation strategy that is guaranteed to succeed even without feedback, within certain broad limits. For example, Erdmann and Mason [9] have demonstrated an algorithm that can force an "L" shaped bracket into a known orientation by placing it on a tray and moving the tray through a pre-determined sequence of motions.

Binary robots are a natural extension of sensorless manipulation. Sensorless manipulation reduces the need to sense a robot's environment, while binary actuators allow us to build a robot without joint-level sensing of position and velocity. There have been a number of efforts in the past to build robots with binary actuators [10, 11, 12]. However, at the time these projects were undertaken, effective algorithms for controlling hyper-redundant manipulators had not yet been developed, nor were computers sufficiently powerful to control robots with many degrees of freedom, even if they could have used the control algorithms that are currently available.

Recent results in binary robotics research include an efficient algorithm to compute the workspace of binary robots [13] and a method to synthesize a binary manipulator that reaches a specific set of points exactly [14].

3 Definitions and Description of Algorithm

This section defines the terminology used to discuss the inverse kinematics algorithm and describes its implementation. The terminology used here was originally defined in [13].

3.1 Definition of Terms

Module A *module* is a kinematically independent element of a manipulator. For example, an individual truss of the variable geometry truss (VGT) manipulator shown in Figure 1 is a module for a truss manipulator.

Macroscopically-serial manipulator
A *macroscopically-serial manipulator* is a manipulator that is serial on a large scale, i.e. it can be represented by a serial collection of modules (where each module is mounted on top of the previous one).

Binary manipulator A *binary manipulator* is a macroscopically-serial manipulator that is composed of a set of modules with two-state actuators, stacked one atop the other. The modules are numbered from $1, \ldots, B$, starting from the base of the manipulator.

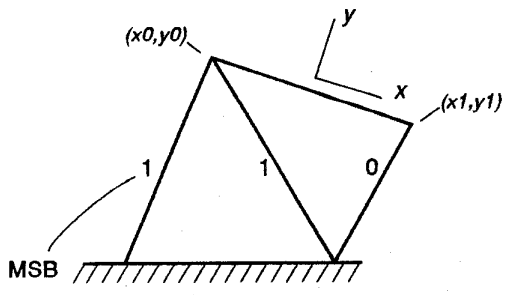

Figure 2: A single binary truss module shown in state 110. The bits within the truss are numbered from left to right.

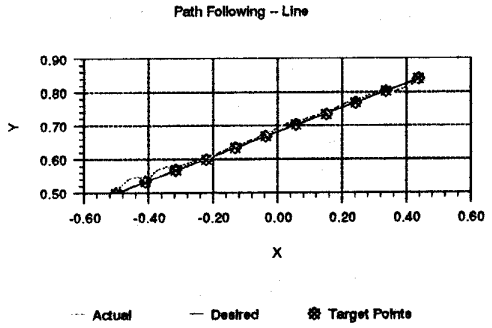

Figure 3: A ten truss binary manipulator following a line.

Each module has a frame attached to its top (Figure 2). The frames are numbered such that frame i is on top of module i, and frame 0 is the frame at the manipulator base. J_i denotes the number of independent binary actuators in module i. Therefore, there are 2^{J_i} different combinations of binary actuator states (and corresponding configurations) for the i^{th} module.

Manipulator state The *state* of a binary manipulator is a binary number, S, whose bits represent the states of each actuator in the manipulator. The state of an individual module in the manipulator is denoted by s_i, which is a J_i bit binary number. Therefore, the total number of bits in S is given by $\mathbf{J} = \sum_{i=1}^{B} J_i$, so the manipulator can achieve $2^{\mathbf{J}}$ different configurations. S is constructed from the individual module states in such a way that the state of the manipulator base module corresponds to the most significant bits of S.

3.2 Algorithm Overview

The idea behind the inverse kinematics algorithm presented here is to find a set, or sets, of actuator states that cause the manipulator to reach a certain location in space, and also optimize a certain property, usually the distance between the end-effector and a desired location. Other metrics for measuring error in both position and orientation can be used analogously, e.g., those defined in [15, 16].

To avoid exponential growth in the search space as the number of actuators grows, we solve the inverse kinematics incrementally by changing only a small number of actuators at a time. For example, in the 10 module truss with 30 DOF in Figure 1, we might try to minimize the error between the end-effector and the goal, by changing only three of the actuators at one time. In this case, we need only search $\binom{30}{3}$, or 4060, possible solutions, instead of the 2^{30} we would have had to explore if we searched all possible configurations of the manipulator. The principles on which the algorithm is built are described in more detail in the following sections.

3.2.1 Searching Robot Configurations

Consider the standard definition of the binomial theorem [17]:

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i} \tag{1}$$

If we let $x = 1$ and let $y = 1$, we get the following result:

$$(1+1)^n = \sum_{i=0}^{n} \binom{n}{i} 1^i 1^{n-i} \tag{2}$$

$$2^n = \sum_{i=0}^{n} \binom{n}{i} \tag{3}$$

Therefore, if we have a 10 module truss robot, for example, we can search its entire state set by taking its current state and looking at all zero bit changes from that state, then all one bit changes, etc., until we have considered all 2^n states. Obviously, if we take this approach to its logical conclusion, it is no better than searching all 2^n states in numerical order. However, if we are willing to move the robot toward its goal by searching for changes in only a small number of bits at any given time, we can obtain a substantial performance gain — to the point where we have a practical algorithm, even for robots with many degrees of freedom.

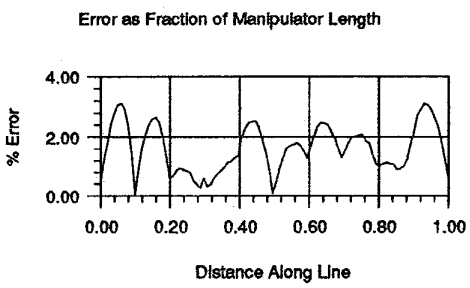

Figure 4: Position error for line tracking task.

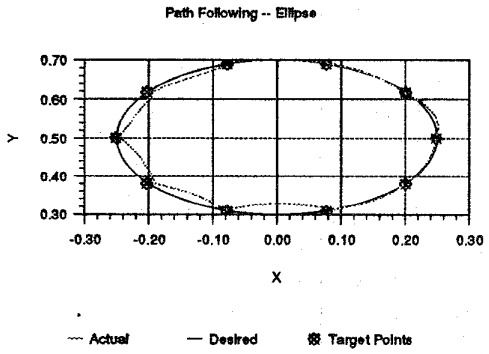

Figure 5: A ten truss binary manipulator following a ellipse.

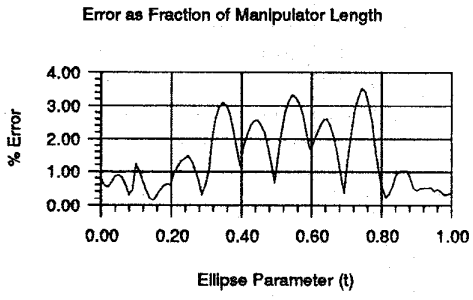

Figure 6: Position error for ellipse tracking task.

3.2.2 Complexity of the Algorithm

While a brute-force search of a binary manipulator's workspace requires computational effort of $\mathcal{O}(2^n)$, the combinatorial algorithm presented in the previous section can be executed in polynomial time, if we fix the number of bit changes that we search, regardless of the number of DOF in the robot. Consider a VGT robot with **J** actuators, which we move toward its target location by changing no more than k of its actuators at a time. To do this we must search through:

$$\binom{\mathbf{J}}{0} + \binom{\mathbf{J}}{1} + \binom{\mathbf{J}}{2} + \ldots + \binom{\mathbf{J}}{k} \tag{4}$$

candidate states to find the one that best moves the robot toward its target position. Note that the operation $\binom{n}{k}$ is defined as follows:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{1}{k!}\,\frac{n!}{(n-k)!} \tag{5}$$

We can expand this equation into:

$$\binom{n}{k} = \frac{1}{k!}\,\frac{n(n-1)\ldots(n-k)\ldots 1}{(n-k)(n-k-1)\ldots 1} \tag{6}$$

$$= \frac{1}{k!}\,n(n-1)(n-2)\ldots(n-k+1) \tag{7}$$

This equation has k terms involving n. Therefore, $\mathcal{O}(n^k)$ time is required to enumerate all the combinations in $\binom{n}{k}$, for all n, with k fixed.

An interesting implication of Equation 6 is that a search for a state change of a single actuator (i.e. one bit) in a VGT truss robot can be accomplished in linear time. Changing the position of only a single actuator is unlikely to make the robot reach its goal position, but iterating the search several times can make the end-effector approach its target with more and more accuracy.

3.2.3 Smoothness of Motion

It can be difficult to make a binary manipulator follow a smooth trajectory, because the path that it follows between any two discrete states cannot be specified precisely. The algorithm presented here addresses this problem in two ways. First, because it explicitly controls the number of bits that can change at one time, we can limit changes to only a small number of bits, which reduces the overall change in the manipulator's configuration. Second, we can try to minimize our position error by examining changes to the least significant bits of the manipulator's state first (i.e. the top of the manipulator), which gives preference to relatively small motions of the manipulator. To implement this behavior we must generate the state combinations in lexicographic order, from least to most significant. The lexicographic ordering algorithm used for the work in this paper is described in [18].

3.3 Algorithm Description

This section presents a detailed description of the combinatorial inverse kinematics algorithm.

Inputs to the Algorithm:

1. $\mathbf{EE}_{des}$, the desired position of the manipulator's end effector.
2. $\mathbf{EE}_{now}$, the current position of the manipulator's end effector.
3. p_{now}, the manipulator's current state vector.
4. n_{max}, the maximum number of bits allowed to change in the manipulator's state vector.
5. B, the number of degrees of freedom (same as number of bits) of the manipulator.
6. The geometry of the manipulator modules for computing the forward kinematics (using, for example, the method in [3]).

Implementation:

```
/* d_min is the distance from the old to new location. */
d_min = cost(EE_des, EE_now)
/* p_min is the closest state vector to the desired position. */
p_min = p_now
/* b_min is the number of bits we changed to get to p_min. */
b_min = 0
for i = 1 to n_max
    for j = 1 to (B choose i)
        /* Get combo j in lexicographic order. */
        c = combo(B, i, j)
        p_test = c ⊕ p_now
        /* fwdKin(x): fwd. kinematics for state x. */
        d_test = cost(EE_des, fwdKin(p_test))
        if d_test < d_min then
                d_min = d_test
                p_min = p_test
                b_min = i
        end
    end
end
return d_min, p_min, b_min
```

4 Results

The inverse kinematics algorithm was run on a number of sample trajectories with the ten module VGT manipulator shown in Figure 1. Three representative examples are shown here. The top and bottom links of each module in the manipulator were 0.08 units wide, and the upper and lower actuator travel limits were 0.12 and 0.08 units respectively for all actuated links. In all cases, the robot started out with its actuators contracted (i.e. in state 0). It was moved to the initial point on the trajectories by iterating the combinatorial algorithm three times, with a maximum of three bit changes at each iteration. The desired trajectory was broken up into ten segments (twelve for the curve with cusps), and was followed by moving the robot to the end of each segment with only one iteration of the algorithm with a maximum of three bit changes. Therefore, between any adjacent target points on the robot's path, the configuration of the manipulator differs by no more than three bits. Figure 3 shows the manipulator following a straight line. Figure 5 shows the manipulator tracking an elliptical path, and Figure 7 shows the manipulator tracking a path with sharp corners or cusps. The cusp curve was generated by reflecting one-quarter cycle of a sine wave about both the x and y axes. In all of these examples, the cost function was simply the cartesian distance from the current end-effector position to the desired position (i.e. $\text{cost}(p_1, p_2) = \|p_1 - p_2\|$). Figures 4, 6, and 8 show the corresponding errors from the desired trajectory (as a percentage of the manipulator's shortest length) for each of the examples presented. In these plots, it is assumed that all actuators in the manipulator travel from one stop to the other at a constant rate. This assumption is used only to illustrate that the deviation is rather small.

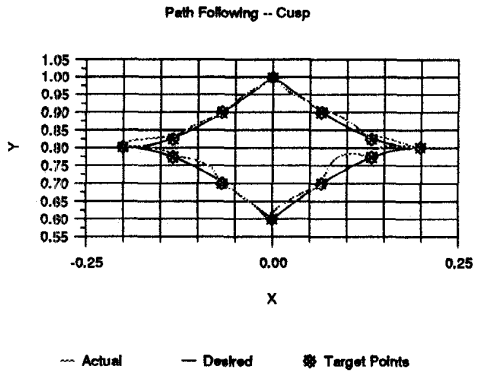

Figure 7: A ten truss binary manipulator following a path with cusps.

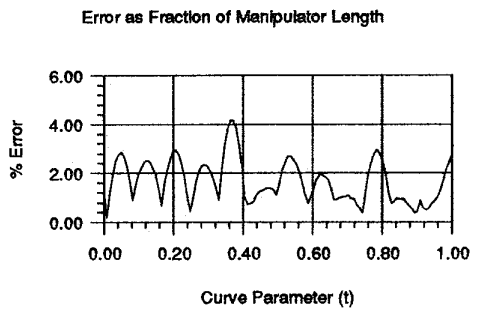

Figure 8: Position error for cusp tracking task.

5 Discussion and Conclusion

Binary actuation offers us a way to create manipulators that are simultaneously fast, rigid and inexpensive. The biggest obstacle to the use of binary manipulators has been the challenge of planning the trajectory of a highly redundant manipulator effectively. In this paper, we presented an algorithm for computing the inverse kinematics and generating smooth trajectories for manipulators with binary actuators. The algorithm executes quickly and produces precise and smooth motions.

References

[1] P. L. Bergstrom, T. Tamagawa, and D.L. Polla, "Design and fabrication of micromechanical logic elements", in *Proceedings of the IEEE Micro Electro Mechanical Systems Workshop*, Napa, CA, February 1990, pp. 15–20.

[2] G.S. Chirikjian and J.W. Burdick, "A modal approach to hyper-redundant manipulator kinematics", *IEEE Transactions on Robotics and Automation*, June 1994.

[3] G. S. Chirikjian, "A binary paradigm for robotic manipulators", in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, pp. 3063–3069.

[4] G. S. Chirikjian and D. Lees, "Inverse kinematics of binary manipulators with applications to service robotics", in *IROS '95*, August 1995.

[5] P.C. Hughes, "Trussarm - a variable-geometry-truss manipulator", *J. of Intelligent Materials, Systems and Structures*, vol. 2, pp. 148–160, April 1991.

[6] J. Canny and K. Goldberg, "A risc paradigm for industrial robotics", Tech. Rep. ESRC 93-4/RAMP 93-2, Engineering Systems Research Center, University of California at Berkeley, 1993.

[7] M.T. Mason, "Kicking the sensing habit", *AI Magazine*, Spring 1993.

[8] K. Goldberg, "Orienting polygonal parts without sensors", *Algorithmica*, 1992, Special robotics issue.

[9] Michael A. Erdmann and Matthew T. Mason, "Exploration of sensorless manipulation", *IEEE Journal of Robotics and Automation*, vol. 4, pp. 369–379, August 1988.

[10] D.L. Pieper, *The Kinematics of Manipulators under Computer Control*, PhD thesis, Stanford University, Stanford, CA, 1968.

[11] B Roth, J. Rastegar, and V. Scheinman, "On the design of computer controlled manipulators", *First CISM-IFTMM Symp. on Theory and Practice of Robots and Manipulators*, pp. 93–113, 1973.

[12] A. Sh. Koliskor, "The l-coordinate approach to industrial robot design", in *IFAC '86*, Suzdal, USSR, 1986.

[13] I. Ebert-Uphoff and G. S. Chirikjian, "Efficient workspace generation for binary manipulators with many actuators", *Journal of Robotic Systems*, June 1995.

[14] G.S. Chirikjian, "Kinematic synthesis of mechanisms and robotic manipulators with binary actuators", in *1994 ASME Mechanisms Conference*, Minneapolis, MN, Sept 1994.

[15] F.C. Park, "Distance metrics on the rigid-body motions with applications to mechanism design", *ASME Journal of Mechanical Design*, December 1995.

[16] J.M.R. Martinez and J. Duffy, "On the metrics of rigid body displacements for infinite and finite bodies", *ASME Journal of Mechanical Design*, vol. 117, pp. 41–47, March 1995.

[17] Sheldon Ross, *A First Course in Probability*, MacMillan, New York, NY, 1984.

[18] Bradley W. Jackson and Dmitri Thoro, *Applied Combinatorics with Problem Solving*, Addison-Wesley, Reading, MA, 1990.