Computational Issues in the Planning and Kinematics of Binary Robots

Matthew D. Lichter, Vivek A. Sujan, and Steven Dubowsky

{ lichter | vasujan | dubowsky@mit.edu } Department of Mechanical Engineering Massachusetts Institute of Technology Cambridge, MA 02139 USA

Abstract

To meet the objectives of many future missions, robots will need to be adaptable and reconfigurable. A concept for such a robotic system has been proposed previously based on using a large number of simple binary actuators. Previous researchers have addressed some of the issues brought up by robots with a few binary actuators. This paper examines the computational feasibility of controlling and planning such binary robotic systems with a large number of actuators, including computation of their workspace, forward kinematics, inverse kinematics and trajectory following. Methods are proposed and evaluated by simulation. Detailed error analysis and computational requirements are presented. An example of the planning for a binary walking robot is presented.

1. Introduction

Future robots will be needed to perform complex tasks in difficult environments. For example, missions to Mars will require robots to perform tasks such as scouting, mining, conducting science experiments, and constructing facilities for human explorers and settlers [9]. To accomplish these objectives, robotic systems will need to be lightweight, reliable and robust. Further, the elements of these systems need to be capable of large and fine motion, a large motion workspace, multiple degrees of freedom, and have a small stowed volume. A new design concept has been proposed to meet these challenges [12, 15]. In this concept, robotic systems use tens or hundreds of simple binary actuators embedded in a flexible structures. Each of the binary degrees of freedom contains a bi-stable element so that the actuators simply flip the state of the joint. As the number of binary degrees of freedom in the system increases, the capabilities of the device approach that of a conventional continuous robotic system. This is analogous to the digital computer replacing the analog computer. The control of such devices is rather simple. To achieve a given position a set of joints simply need to be flipped. No feedback is required and theoretically no "servo errors" will exist. Control of such actuators has been classified as sensor-less manipulation [4, 6, 12]. The kinematics and control of such "hyper-redundant" manipulators, both with and without binary actuation have been studied by a number of researchers [1, 3, 7, 8, 12]. However, many of the planning and kinematics issues of binary robots are fundamentally different and more difficult than those of conventional robotics [3, 12, 15]. For example, the inverse kinematics problem for a binary robot involves searching through a discrete set of configurations to find the one that best matches the desired state, rather than solving geometric equations to

determine joint angles or link lengths, as one would do for a continuous systems. Most research has involved binary robotic systems when the number of degrees of freedom is relatively small, in the order of tens of DOF.

This paper describes analysis and simulation studies performed to examine the feasibility of controlling and planning binary-actuated robotic systems in real time when the number of DOF is very large (hundreds or thousands). Such systems are currently under development [16]. The work also suggests planning algorithms that can be used in future systems. The work outlines some of the issues and potential methods for the workspace analysis and optimization, the forward and inverse kinematics, and trajectory planning of binary robotic systems. These methods are then applied to binary systems used for robot locomotion.

2. Workspace Analysis and Optimization

The workspace of a robot generally refers to the locus of all points that a robot's end-effector can reach [2]. With a continuous system, the workspace is usually a region in continuous space (see Figure 1(a)). Many continuous robots are also able to achieve a continuous range of end-effector orientations for a given point in the workspace. Understanding the size of the workspace as well as the "orientability" of the end-effector within this workspace gives some measure of the ability of the robot to perform diverse manipulation tasks.



(a) continuous robot workspace (b) binary-actuated robot workspace Figure 1: The distinction between binary and continuous robot workspace

For binary-actuated robots the notion of a workspace takes on some subtle differences [14]. For a binary system, the workspace in not a continuous volume but rather a finite set of points in space (see Figure 1(b)). For each point there is an associated orientation of the endeffector, indicated by the arrows originating from each point in Figure 1(b). In such a workspace one can guarantee the existence of at least one binary configuration of the robot that achieves a minimum error of end-effector position and orientation. Thus for a binary robot, the density of the points within the workspace can be important, since a dense set of points will generally achieve small end-point errors. The density of points increases as the number of actuators in the system increases. Each additional actuator doubles the number of workspace points.

It is useful to view a discrete workspace cloud from the perspective of a density function map. In designing a binary robot, one might want to optimize its workspace. For example, it may be desirable for repeated pick and place tasks to have a workspace that has a great density of points in the pick and place locations. In other cases, it might be desirable to have a uniform distribution of reachable points over the entire workspace. To deal with such issues the notion of workspace distribution is proposed. For a planar robot, a density map represents the density of points (the z-axis) versus the Cartesian location in space (the x- and y-axes). With a discrete cloud, the density map appears as delta functions at each workspace point, with all other areas of the map having a value of zero density (see Figure 2 (a)). Applying a lowpass filter (such as convolution with a Gaussian function) to the density map, the spikes blend together and provide a continuous approximation of the density of the workspace (see Figure 2(b)).



Workspace Density Representation (c) optimized uniform workspace density Figure 2: Workspace of a 6 DOF serial binary manipulator with optimization

This continuous approximation can be a metric for the uniformity/distribution of the workspace. Here it is defined based on the standard deviation of the workspace density. A small standard deviation of the workspace density indicates a more uniform distribution. This method for quantifying the distribution of the workspace can be extended to three-dimensional workspaces and include endpoint orientation information.

As an example of optimizing a binary robot design to provide uniform workspace density, consider a serial planar manipulator, having between four and ten binary actuators. The lengths of each link, l_i , and the angles of deviation of each binary rotary joint, ϕ_I are to be optimized. This robotic design results in a planar workspace composed of 2^N points, where N is the number of binary actuators.

Using an evolutionary algorithm the design variables of this system can be optimized. The algorithm generates a random set of candidate designs and evaluated them based on their uniformity of workspace. The best candidates (those with the most uniform workspace densities) are evolved in the classical manner with mutation. A few hundred generations result in good solutions to the problem. An example of one such optimization (for an un-optimized system shown in figure 2(a)) is shown in Figure 2(c). Note that the density map in this figure is much more uniform than the one shown in Figure 2(b).

3. Kinematics

3.1. Forward Kinematics

For binary robotic systems, it is convenient to formulate the forward kinematics using four-by-four homogeneous transformation matrices [2]. For example, the transformation matrix $A_{0,M}$ describing the position and orientation of the end-effector relative to the base can be viewed as the product of the M intermediate transformations $A_{i-1,i}$ from module to module within the structure:

$$A_{0,M} = A_{0,1} \cdot A_{1,2} \cdot \dots \cdot A_{M-1,M} = \prod_{i=1}^{M} A_{i-1,i}$$
(1)

where M is the number of intermediate modules comprising the binary robotic system [15].

Due to the discrete nature of binary devices, each term of the intermediate transformation $A_{i-1,i}$ can have only a finite number of possible values. If each module has only a few binary degrees of freedom, all the values that the terms of $A_{i-1,i}$ can be easily enumerated. For example, if a module has three binary DOF, then the module has 2^3 or 8 possible values for $A_{i-1,i}$ (notated by $A_{i-1,i}^{(1)}$, $A_{i-1,i}^{(2)}$, ..., $A_{i-1,i}^{(8)}$). The solution of the module kinematics may require trigonometric or more complex mathematics, but these need only be solved once, and possibly offline. This reduces online computational loads.



An example of such a robot is the Binary Robotic Articulated Intelligent Device (BRAID), developed at the Field and Space Robotics Laboratory, which is a serial stack of identical parallel stages [15] (see Figure 3). Such a design could be used for manipulating instruments, collecting soil samples, or mating two cooperating robots, applications that require only moderate precision (see Figure 4).



(a) mating two rovers (b) instrument maneuvering Figure 4: Potential BRAID applications

In a single parallel link stage of the BRAID system, the three legs are positioned about the vertices of two equilateral triangles. Additionally, based on the joint configuration of each leg, the single stage has only three degrees of freedom—pitch (θ_x) and yaw (θ_y) rotations and a vertical (z) translation (coupling effects lead to nonindependent motions in the x and y directions as well). The four by four transformation matrix A_{i-1,i}, of the ith coordinate frame with respect to the i-1th coordinate frame is defined based on these five motions. The matrix A_{0M} defines the forward kinematics from base to end-effector of the entire system. In this formulation the leg lengths are the control variables. The relationship between these leg lengths and the pitch, yaw, and translation motions of the ith coordinate frame with respect to the i-1th coordinate frame is given below. From Figure 5 the deflection parameters $(\delta^i, \gamma^i, \psi^i)$ give us the coupled x^i and y^i translation of the ith stage:

$$x^{i} = -\delta_{1}^{i}\sin(\gamma_{1}) \quad y^{i} = -\left(\frac{r}{2} - \frac{\delta_{3}^{i}\sin\gamma_{3}^{x}}{\cos\pi/6} - \frac{r}{2}\cos\theta_{x}^{i}\right) \quad (2)$$



Figure 5: Critical parameter representation of BRAID system

The following relations can be obtained from figure 5:

$$h_1^i \sin \theta_2 - l_1^i \sin \theta_1 = \frac{3}{2} r \sin \theta_x \tag{3}$$

$$h_2^i \sin \alpha_2 - h_3^i \sin \alpha_1 = \frac{2}{\sqrt{3}} r \sin \theta_y \tag{4}$$

where $\theta_1 = \angle DAB$, $\theta_2 = \angle ADC$, $\alpha_1 = \angle EHG$, and $\alpha_2 = \angle FEH$, can be found from the leg lengths l_1^i , l_2^i , and l_3^i . Equations 3 and 4 give two independent equations in two unknowns. However, both are highly non-linear transcedental equations and can only be solved numerically to give the BRAID forward kinematics. A Newton-Raphson algorithm was implemented to solve for the unknowns, θ_x and θ_y . Finally, the vertical translation can be solved using solutions for θ_x , θ_y and equation 5.

$$z^{i} = h_{2}^{i} \sin \alpha_{2} - \frac{1}{\sqrt{3}} r \sin \theta_{y} = h_{1}^{i} \sin \theta_{2} - \frac{1}{2} r \sin \theta_{x} \quad (5)$$

3.2. Inverse Kinematics

The planning and execution of practical tasks generally requires the solution to the inverse kinematics problem. The complexity of the trajectory planning and inverse kinematics software of binary devices are more complex than that of continuous systems. The inverse kinematics problem cannot be expressed in a closed form solution. Brute force or exhaustive search methods may prove appealing for systems with few stages (less than 5), but become impractical for larger systems. As the number of degrees of freedom increase, the complexity of the workspace grows exponentially. For example, for every additional stage in the BRAID there is about an order of magnitude increase in the number of states in the workspace. Hence, for large systems more efficient search methods are required to find "optimum" solutions. In this study two search methods for the inverse kinematics problem are studied. The first is a combinatorial search algorithm and the second is a genetic search algorithm. The search metric for both algorithms is to minimize the error between the true endeffector and desired position. Both algorithms are briefly described below.

3.2.1. Search Algorithms

The combinatorial search algorithm was first described in [12]. To avoid exponential growth of the search space as the number of actuators grow, the inverse kinematics are solved by changing the state of only a few actuators at a time. This is perceived as a k-bit change to the given system state, where any system state is defined by an mbit word (m is the number of binary actuators). At any state all possible changes (of up to k-bits, where $k \le N$) are evaluated to determine the one that optimizes the search metric (i.e. reduces the error between the endeffector position and the desired position). This optimal change forms the new state of the system and the search procedure repeats until convergence. This reduces the computational complexity from $O(2^N)$ to $O(N^k)$ or from exponential computational time to polynomial computational time [12].

The genetic algorithm used, is a classical one, where each generation consists of N-bit binary words describing the manipulator state (where m is the number of binary actuators). A full description of genetic algorithms can be found in [5]. Comparing the genetic algorithm to the others discussed, the size of the search space explored by the genetic algorithm is given by:

$$search_space_size = E \cdot G \cdot P \tag{6}$$

where E is the number of populations separately evolved, G is the number of generations for each population, and P is the number of individuals within the population. In studies done here, E, G, and P were kept constant relative to the number of degrees of freedom, N. For more advanced algorithm development, these values could be made a function of N. Within the algorithm, several computations take place that are linearly proportional to N (such as forward kinematics computations) and therefore computation time of the inverse kinematics using a genetic algorithm grows approximately linearly with the number of DOF of the system.

3.2.2. Algorithm Comparisons

Performance of the two search methods is quantified on a stochastic basis using a Monte Carlo method. One thousand target points with random orientations are selected randomly within the volume of a binary workspace cloud of a multi-staged BRAID system. The targets are chosen from within a spherical region, whose radius is roughly 90% of the radius of the actual point cloud. The inverse kinematics for each target pose is then solved for and the solution times and pose errors are computed and recorded. For comparison, results from exhaustive searching are also presented.

Figure 6 shows the times for solving the inverse kinematics problem for the two algorithms described above. The times were computed from simulations performed on a 600 MHz Pentium III processor. In these studies, the exhaustive search was observed to be the fastest for systems with less than 12 DOF, the combinatorial algorithm was the fastest for systems having between 12 and 40 DOF, and the genetic algorithm was the fastest for larger systems.



Figure 6: Inverse kinematics algorithms solution times

Errors in position and orientation for the algorithms are also quantified. An example of the distribution of the errors is shown in Figure 7 for the case of a 30 DOF (a 10 stage) BRAID. The shapes of the error distributions for any given BRAID are very similar for each of the algorithms and most closely resemble a gamma distribution. The outliers are generally near the boundaries of the workspace, and in practice tasks should be planned to avoid these regions.

Figure 8 shows that the median errors drop as a function of the number of DOF for the combinatorial and genetic search algorithms. After about 30 DOF, there are only marginal improvements as the number of DOF increases. For systems with 30 DOF (a ten-stage BRAID), displacement errors are within a few percent of the characteristic manipulator length and angular errors are within fifteen degrees. Such a system is unsuitable for precision work, but may be acceptable for such tasks as camera placement, crude instrument manipulation, and sample collection.



Figure 7: Error distribution for a 30-DOF BRAID: displacement error (1000 samples)



Figure 8: Median errors vs. number of DOF for different algorithms: displacement error (1000 samples per DOF)

4. Trajectory Following

The trajectory following problem is also quite different for a binary device than for continuous ones. Instead of using Jacobian matrices to compute actuator commands [2], the problem is solved by a repeated search through the configuration space to find the configuration whose end-effector most closely matches a moving target [12]. Hence, this problem is very close to the methods described above and can be directly applied. For low-DOF systems, the exhaustive search may prove to be the easiest and most robust method for trajectory following. For systems with higher DOF, genetic algorithms or combinatorial searches would be more effective. However, it was found that the genetic algorithm used is not well suited for trajectory following. A genetic algorithm, given the same target and the same initial conditions, will produce different solutions because of its randomly selected initial population and mutation component. Since the high DOF system is highly redundant, there can be a large number of configurations that will produce nearly the same end-effector position yet will have greatly different configurations. A relatively smooth path planned in Cartesian space may have an erratic path in configuration space (see Figure 9). For power consumption, reliability, and transient behavior, this is very undesirable.

Hence, the combinatorial search algorithm is found to be the most effective method for trajectory following. This algorithm searches only the subset of neighboring configurations, and generates a path that is relatively smooth in Cartesian and configuration space. Between time steps, only a few (specifically defined) actuators will be actuated at a time. The combinatorial algorithm runs much faster in the trajectory planning problem than for the inverse kinematics problem described in Section 3.2.1, since it only makes one iteration per time step (see Figure 10).



Figure 10: Inverse kinematics solution times for each trajectory following step

Simulations showed that the errors maintained during trajectory following were acceptable for a number of applications such as maneuvering a camera or instrument, or manipulating an object with low precision (see Figure 11). Typical errors during manipulation were found to be of roughly the same size as those discussed in Section 3.2.2.



Figure 11: Simulation of a camera maneuvering task—desired trajectory: lighter path; actual trajectory: darker path.

5. Locomotion Planning

The trajectory following problem can be extended to the locomotion planning problem, where binary devices are used as legs. Simulations were done to explore the feasibility of planning actuator sequences in real-time for a robot having six binary-actuated legs walking in rough terrain. Each of the six legs is modeled as a BRAID and has 21 binary DOF (see Figure 12), yielding a total of 126 DOF for the system. Desired ground contact points for the legs are chosen and the configurations and actuator sequences are planned to achieve these contact points and body motions.



Figure 12: Simulation of a 6x21-DOF walking robot composed of six BRAIDs for legs, walking in rough terrain.

Several issues arise with a binary system for this problem. First, the multiple legs in contact with the ground form a closed kinematic chain that is overconstrained due to the discrete nature of the leg motions. If the ground contact points are rigidly held, it will be impossible for the contacting legs to change configurations due to the incompatibilities between each leg's workspace (see Figure 13(a)). Thus, it is impossible to shift the body while keeping the feet planted, as required for walking. Here a small amount of local compliance in the ground contact is permitted and the limited effects of the incompatibilities between the workspaces of planted legs are ignored (see Figure 13(b)).



A second issue arises in finding the binary configuration of the legs that allows the body to move in a prescribed manner. The system can be considered as a single parallel 126 DOF system with ground contact points modeled as continuous revolute joints with displacement compliance. With the requirement that trajectories are smooth in both Cartesian and configuration space, this system becomes computationally difficult to solve quickly. To simplify this locomotionplanning problem, each leg was viewed as an independent trajectory planning problem. First, the desired position of the body for the next small time step is selected. Then, the inverse kinematics for each leg is solved using the one-pass combinatorial algorithm to make the leg move to the desired ground contact point.

With the configuration of each leg being solved independently, the actual body position does not coincide exactly with the desired body position. The actual body position is obtained from the equilibrium condition of the compliant contact elements on the fixed configuration robot. This problem is solved by minimizing the potential energy stored in the compliant elements as a function of final body position. The error requiring adjustments is small, roughly the size of the errors in the legs themselves, generally a few percent of the characteristic size (see Section 3.2.2). For most applications, these errors would be acceptable.

This planning method is found to be effective and fast. Using a Pentium III 933 MHz processor, the simulated robot plans and executes a stride at a rate of once per second. The robot is able to execute sidestepping and turning motions in rough terrain. It maintains static stability while walking on slopes up to 20 degrees. Static stability is only lost occasionally ascending, descending, and traversing very steep slopes of around 45. This loss of stability would need to be addressed by the high-level planner of the robot [11]. This requirement would be the same for robots with continuous degrees of freedom.

6. Conclusions

This paper considers some of the computational challenges for the planning of binary robotic systems. The notion of a binary workspace optimization was described. The forward kinematics of binary systems was discussed, and the computational simplicity of this operation relative to continuous systems was shown. The methods to solve the inverse kinematics and trajectory planning were addressed and compared to those of continuous systems. The positioning errors of binary systems were also quantified in a probabilistic manner. The methods were applied to a walking system that might be used for future space exploration.

Acknowledgements

The authors would like to acknowledge the NASA Institute of Advance Concepts (NIAC) for their support in this work.

References

[1] Chirikjian, G.S.; Burdick, J.W. *The kinematics of hyper-redundant robot locomotion*. IEEE Transactions on Robotics and Automation. Volume: 11 Issue: 6, Dec. 1995 Page(s): 781–793

[2] Craig J J. Introduction to Robotics: Mechanics and Control. Second ed, Addison-Wesley, Reading, MA, 1989.

[3] Ebert-Uphoff, I.; Chirikjian, G.S. *Inverse kinematics* of discretely actuated hyper-redundant manipulators using workspace densities. Proceedings of the IEEE International Conference on Robotics and Automation, 1996, Volume: 1, Page(s): 139 -145 vol.1.

[4] Erdmann, M.A. and Mason, M.T. *Exploration of sensor-less manipulation*. IEEE Journal of Robotics and Automation, Vol. 4, pp 369-379, August 1988.

[5] Goldberg, D. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA 1989.

[6] Goldberg, K. *Orienting polygonal parts without sensors*. Algorithmica, 1992, Special Robotics Issue.

[7] Huang, M.Z., Shou-Hung Ling. *Kinematics of a class of hybrid robotic mechanisms with parallel and series modules*. Proceedings of the 1994 IEEE International Conference on Robotics and Automation, Page(s): 2180 - 2185 vol.3.

[8] Hughes, P.C. *Trussarm – a variable geometry truss manipulator*. Journal of intelligent materials, systems and structures, vol. 2, pp. 148-160, April 1991.

[9] Huntsberger, T.L., G. Rodriguez, and P. S. Schenker. *Robotics: challenges for robotic and human Mars exploration*. Proceedings of ROBOTICS2000, Albuquerque, NM, Mar 2000.

[10] Kwon, S., Youm, Y. *General algorithm for automatic generation of the workspace for n-link redundant manipulators*. Proceedings of the International Conference Advanced Robotics, 1991. 'Robots in Unstructured Environments', Page(s): 1722 -1725 vol.2.

[11] Latombe J. *Robot Motion Planning*. Kluwar Academic Publishers, Boston, MA 1991.

[12] Lees, D.S. and Chirikjian, G.S. *A combinatorial approach to trajectory planning for binary manipulators*. Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, April 1996.

[13] Lichter, M.D., Sujan, V.A., Dubowsky, S. *Experimental Demonstrations of a New Design Paradigm in Space Robotics*. Proceedings of the Seventh International Symposium on Experimental Robotics, ISER 00. Dec 10-13, 2000, Honolulu, Hawaii.

[14] Sen D, Mruthyunjaya T S. A Discrete State Perspective of Manipulator Workspaces. Mech. Mach. Theory, Vol. 29, No.4, 591-605, 1994.

[15] Sujan, V.A., Lichter, M.D., and Dubowsky, S. *Lightweight Hyper-redundant Binary Elements for Planetary Exploration Robots*. Proceedings of the 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM '01). 8–11 July 2001, Como, Italy.

[16] Hafez, M., Lichter, M.D., and Dubowsky, S. *Optimized Binary Modular Reconfigurable Robotic Devices*. Proceedings of the 2002 IEEE International Conference on Robotics and Automation. Washington, D.C., May 11-15, 2002.