A Binary Paradigm for Robotic Manipulators

Gregory S. Chirikjian
Department of Mechanical Engineering
The Johns Hopkins University
Baltimore, MD 21218

Abstract

Traditionally, kinematics and motion planning paradigms have addressed robots with continuous range-of-motion actuators (e.g. motors, hydraulic cylinders, etc.). Unlike motors, binary actuators have only two discrete states, both of which are stable. As a result, binary manipulators (i.e, those which are actuated with binary actuators) have a finite number of states. Major benefits of binary actuation are that extensive feedback control is not required, task repeatability can be very high, and two-state actuators are generally very inexpensive (e.g., solenoids, pneumatic cylinders, etc.), thus resulting in low cost robots. This paper presents a new paradigm in robotics based on binary actuation, and develops algorithms for the optimal design of binary manipulators for pick-and-place tasks.

1 Introduction

Standard continuous range-of-motion robotic manipulators have not been embraced by many industries because of their relatively high cost, low accuracy, and low payload capability as compared to dedicated machine tools. Thus, there is a need for a new paradigm in robotics which will lead to lower cost and higher reliability.

In principle, an analogy can be made between continuous vs. binary manipulators and analog vs. digital circuits. In the history of electronics and computing, digital devices replaced many of their analog counterparts because of higher reliability and lower cost - exactly the same reasons for developing a binary paradigm for robotics.

Discretely actuated robots have a finite number of states. Binary manipulators are a particular kind of discrete device in which actuators have two stable states. Major benefits of binary manipulators are that they can be operated without extensive feedback control, task repeatability is very high, and two-state actuators are generally very inexpensive (e.g., solenoids, pneumatic cylinders, etc.), thus resulting in low cost robots. While it is true that stepper motors are a commonly used form of discrete actuation, they are prone to losing/slipping steps - a problem not encountered with binary actuation.

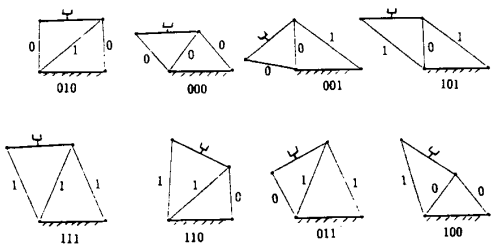

Figure 1: Configurations of a 3-bit Manipulator

Figure 1 illustrates all possible configurations of a '3 bit' planar binary platform manipulator. A finite number of points are reachable by the manipulator's gripper. In this case, 2^3 possible configurations result because there are three actuators. Note that for this design the location of points reachable by the end-effector are a function of the retracted cylinder length, extended cylinder length, and width of the platform. In the general case these kinematic parameters will be divided into joint stop and structural parameters,

1050-4729/94 $03.00

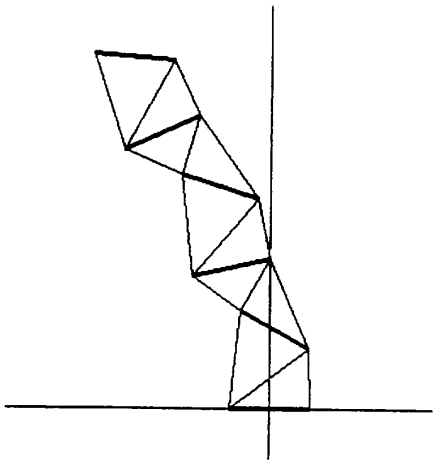

Figure 2: Configuration 110001110001110

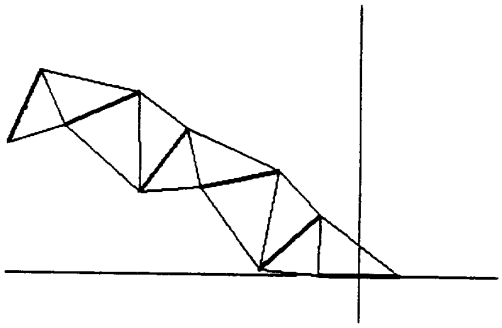

Figure 3: Configuration 001110001110001

which for this case are denoted q^{min}, q^{max}, and w respectively. In Figure 1, $q^{min} = 1$, $w = 1.2$ and $q^{max} = 1.5$. Thus, when an an actuator is in state '1' it is one and a half times its length in state '0'. Section 3 will formalize and generalize notation for the binary manipulator paradigm, but first more insight will be given with another particular example.

A schematic of a highly actuated prototype is shown in Figures 2 and 3 for two of its almost thirty three thousand (2^{15}) configurations. This particular design is a variable geometry truss manipulator. As currently configured, this manipulator consists of 15 identical prismatic actuators, each with two stable states (completely retracted (0) or completely extended (1)). In these figures each cylinder has $q^{min} = 3/20$ and $q^{max} = 5/20$, with the width of each platform $w = 1/5$.

Actuators are numbered from left to right in each 'bay' of the truss, and from base to tip. Writing these 1's and 0's from left to right, the most significant bit corresponds to the actuator on the left side of the base, and the least significant bit corresponds to the actuator at the right side of the distal end of the manipulator. It is interesting to note that the configurations shown in Figures 2 and 3 are the 1's compliment of each other.

The remainder of this paper is organized as follows: Section 2 reviews the literature. Section 3 formalizes and generalizes the concept of a binary manipulator. Section 4 introduces and solves the optimal design problem for binary manipulators performing pick-and-place tasks. Section 5 illustrates this method with an example.

2 Related Literature

Due to the high cost/performance ratio of sophisticated robotic systems, a recent trend in 'minimalist' robotics has begun to gain momentum. For instance, there are current efforts to develop new paradigms in robotics which parallel the development of reduced instruction set computers (RISC) [CaG]. Related efforts include the investigation of *sensorless* robots [Go92,Ma]. In these efforts the mechanics of contact and pushing are used to formulate planning algorithms which are guaranteed to work provided particular physical constraints are observed. Thus, objects can be manipulated in a precise manner without the need for force feedback.

This trend runs counter to ideas that robots equipped with sophisticated tactile feedback systems provide the solution to all industrial robotics problems. However, the minimalist approach has not included a paradigm for robotic manipulators completely devoid of joint level feedback (including position and velocity) until now.

Nonetheless, if one reviews the literature, sporadic efforts in binary actuation can be found, e.g., [AnH67,RoRS73]. Such efforts resulted when computers were first available to control robotic manipulators. However, despite the seemingly natural parallel between discretely actuated mechanical systems and the development of the computer, these efforts were abandoned for lack of a framework in which to design and plan well-behaved motions of such systems.

Of course, a natural question that one might raise is how different binary manipulators are from current systems which use stepper motors, or pick-and-place machines used in circuit board fabrication, or even flexible automation systems in which technicians set joint stops. The answer is that, just as in electronics, the true benefit of binary mechanisms is not so much a function of their discrete nature as it is the fault tolerance of having only two states. Moreover, simple robots with only a few binary actuators cannot perform complicated tasks such as obstacle avoidance. Therefore, the true benefit of a binary paradigm for robotics can only be exploited if a large number of actuated degrees of freedom are considered. In this way, the current work combines the trend towards minimalist (or sensorless) robots with the author's past and present interest in high-degree-of-freedom manipulators, e.g., [ChB92].

3 Formalizing the Paradigm

The *forward kinematics* of a robotic manipulator completely characterizes the relationship between generalized joint displacements and the position and orientation of the end-effector in space. If $\vec{q} = [q_1, ..., q_n]^T \in \mathbf{R^n}$ denotes a vector whose elements are the joint angles of a robotic manipulator, then the forward kinematic function $\vec{g}(\cdot) \in \mathbf{R^N}$ maps these joint angles to end-effector coordinates. This is written as:

$$\vec{x}_{ee} = \vec{g}(\vec{q}, \vec{a}). \tag{1}$$

$\vec{x}_{ee} \in \mathbf{R^N}$ represents the position and/or orientation of the end-effector with respect to a given reference frame in space. The vector $\vec{a}$ contains condensed information on the kinematic structure of the manipulator. Typically, $\vec{a} \in \mathbf{R^{3n}}$ for serial manipulators because each link has three structural variables given by the Denavit-Hartenberg framework, e.g., link length, offset, and twist. This is usually different for parallel and hybrid parallel-serial manipulators. In essence, it is the choice of $\vec{a}$ which distinguishes one member of a class of manipulators from another, while it is the part of the structure of $\vec{g}(\cdot)$ which is independant of $\vec{a}$ which distinguishes among different classes.

For standard motor-driven robotic manipulators, each joint angle (or generalized displacement) can be controlled to achieve any desired value within a specified interval. That is,

$$q_i \in [q_i^{min}, q_i^{max}] \in \mathbf{R}, \tag{2}$$

where the notation $x \in [y, z]$ is equivalent to $y \leq x \leq z$. In order for a motor to achieve a desired angle, complex (and expensive) feedback control systems are usually required.

The kinematics of binary manipulators is also described by Equation (1). The difference is that for binary manipulators,

$$q_i \in \{q_i^{min}, q_i^{max}\} \tag{3}$$

where $\{y, z\}$ denotes the set of two real numbers y and z. Another way to write this is for each $i \in [1, ..., n]$:

$$q_i = q_i^{min} + b_{i-1}(q_i^{max} - q_i^{min}) \quad \text{for} \quad b_i \in \mathbf{B} \tag{4}$$

where $\mathbf{B} = \{0, 1\}$ is the set of one-bit binary numbers, or Boolean variables. A binary manipulator configuration is thus described by the n-bit binary number $b_{n-1}b_{n-2}...b_1b_0 \in \mathbf{B^n}$.

The above can be written in matrix form as:

$$\vec{q} = \vec{q}^{min} + \mathbf{B}(\vec{q}^{max} - \vec{q}^{min})$$

by defining the diagonal matrix $\mathbf{B}$ in such a way that each of its diagonal elements are a bit of the binary number with the most significant bit written in the 1,1 place in the matrix, and least significant bit written in the n,n place.

If the manipulator is to be used for pick-and-place tasks with multiple intermediate points defining a discrete trajectory, one possible design criteria could be to set the manipulator kinematic parameters $(\vec{q}^{min}, \vec{q}^{max}, \vec{a})$ so that the end-effector can exactly reach a finite number of points in the workspace. With a priori knowledge of a finite number of points which the end-effector must traverse, an algorithm can be designed so that the kinematic parameters of a binary manipulator guarantee that the end-effector reaches the designated points. In order to solve this problem, it is first necessary to formalize the scenario.

Suppose there are m distinct points which the manipulator's end-effector is to reach, corresponding to m different binary states. Then for each of these different configurations, we can write the expressions:

$$\vec{q}^{\,i} = \vec{q}^{min} + \mathbf{B^i}(\vec{q}^{max} - \vec{q}^{min})$$

and

$$\vec{x}_{ee}^{\,i} = \vec{g}(\vec{q}^{\,i}, \vec{a}).$$

These can all be written together as on big equation:

$$\begin{pmatrix} \vec{x}_{ee}^{\,1} \\ \vdots \\ \vec{x}_{ee}^{\,i} \\ \vdots \\ \vec{x}_{ee}^{\,m} \end{pmatrix} = \begin{pmatrix} \vec{g}(\vec{q}^{\,1}, \vec{a}) \\ \vdots \\ \vec{g}(\vec{q}^{\,i}, \vec{a}) \\ \vdots \\ \vec{g}(\vec{q}^{\,m}, \vec{a}) \end{pmatrix}.$$

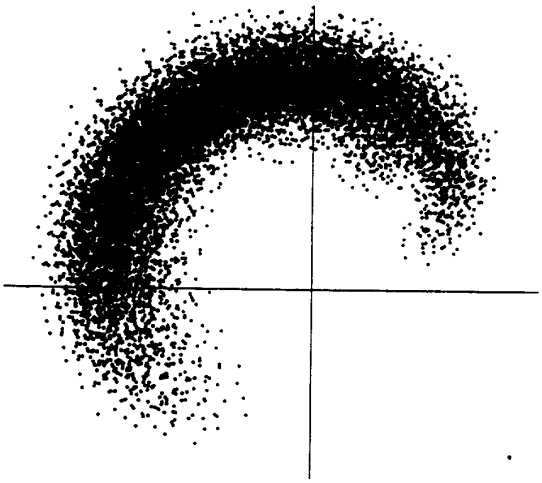

Figure 4: Workspace for $(q^{min}, q^{max}) = (3/20, 5/20)$

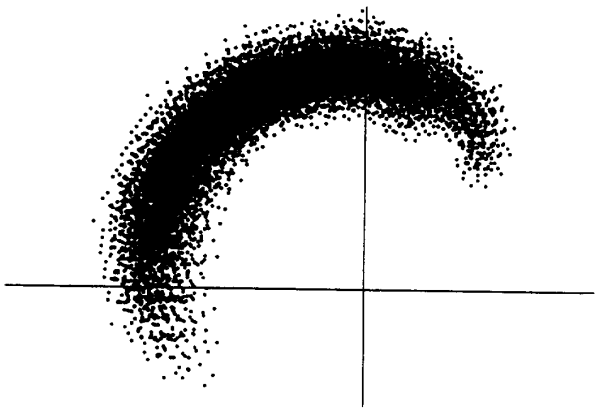

Figure 5: Workspace for $(q^{min}, q^{max}) = (4/25, 6/25)$

Let us assume that the only kinematic parameters which are variable are joint stops. Then, we can rewrite the above as:

$$\vec{x} = \vec{f}(\vec{q}^*) \in \mathbf{R^{mN}}$$

where

$$\vec{q}^* = \begin{pmatrix} \vec{q}^{min} \\ \vec{q}^{max} \end{pmatrix} \in \mathbf{R^{2n}}.$$

Thus, the problem of kinematic synthesis for binary manipulators performing pick-and-place tasks can be stated as an inverse kinematics problem as long as $mN \leq 2n$. In the case of strict inequality, it is desirable to minimize a cost function in order to resolve the redundancy. This is addressed in the next section.

In this mode of operation, a binary manipulator is a semi-dedicated machine, i.e., it is designed to do a very specific pick-and-place task. However, the kinematic parameters determined by the design process can be changed as new tasks are presented to the design algorithm. In practice, these physical parameters would be set by making adjustments to the manipulator structure via changing the joint stops. This in turn specifies the geometry associated with each binary state of the manipulator. Figures 4 and 5 are the workspaces of the same binary manipulator with different joint stops.

Joint Limits can be altered by putting stoppers of different lengths to limit the stroke in a desired way. This idea can be generalized to include *programmable stops* which allow the stroke to be controlled. The benefit of programmable stops is that they could operate at very low bandwidths. They would be fixed while a task is being performed, and adjust themselves over a much larger timescale than the duration of a task, thus adding flexibility of function and form without an explosive increase in cost.

4 Design Optimization

In this section, standard methods for kinematic redundancy resolution are reformulated to derive equations governing the optimal design of binary manipulators. The criteria used here are intuitive: given an initial, or baseline, design and a number of points which the manipulator is expected to reach, alter the original design as little as possible in order to reach the points.

Thus, the vector $\vec{x}$, which is the concatenation of end-effector positions corresponding to each of the m initial configurations, will be made to follow a trajectory to the desired end-effector positions. This trajectory will be parametrized by 'artificial time' (i.e., we are not interested in following a trajectory, but rather the resulting end points, and this calculation is done off line and not in real time). Likewise, the variation of the kinematic parameters (joint stops) in artificial time must not only follow end-effector constraints, but also satisfy optimality criteria which are path independent (since we are only interested in terminal values).

Suppose instead of the usual joint rate norm or kinetic energy cost functions used in kinematic redundancy resolution [KlH83], we seek to minimize:

$$\frac{1}{2}(\vec{q} - \vec{q}_0)^T \mathbf{W} (\vec{q} - \vec{q}_0) \tag{5}$$

subject to the constraints:

$$\vec{x} = \vec{f}(\vec{q}). \tag{6}$$

where $\vec{f}(\cdot)$ is the concatenation of forward kinematic functions corresponding to different binary states of the manipulator, $\mathbf{W} \in \mathbf{R^{2n \times 2n}}$ is a symmetric positive definite matrix which is independent of configuration, and $\vec{q}_0 \in \mathbf{R^{2n}}$ represents the initial joint angle limits, from which we desire to change as little as possible. Note that the superscript $*$ has been dropped from $\vec{q}^*$ because it is cumbersome. This problem is solved as follows.

First, a function is defined as:

$$F = \frac{1}{2}(\vec{q} - \vec{q}_0)^T \mathbf{W}(\vec{q} - \vec{q}_0) + \vec{\mu}^T \left(\vec{f}(\vec{q}) - \vec{x}\right), \quad (7)$$

where $\vec{\mu}$ is a vector of Lagrange multipliers.

The constrained optimization problem then becomes one of solving the following simultaneous equations:

$$\frac{\partial F}{\partial \vec{q}} = \vec{0} \quad \frac{\partial F}{\partial \vec{\mu}} = \vec{0}, \quad (8)$$

which together represent a system of $2n + mN$ scalar equations. These equations are written as (6) together with:

$$\mathbf{W}(\vec{q} - \vec{q}_0) + \mathbf{J^T}\vec{\mu} = \vec{0}, \quad (9)$$

where $\mathbf{J} \in \mathbf{R^{mN \times 2n}}$ is the Jacobian matrix $\partial \vec{f}/\partial \vec{q}$.

Both of these equations can be written in rate form by taking the derivative with respect to artificial time, in which case we get:

$$\begin{pmatrix} \hat{\mathbf{W}} & \mathbf{J^T} \\ \mathbf{J} & \mathbf{O} \end{pmatrix} \begin{pmatrix} \dot{\vec{q}} \\ \vec{\mu} \end{pmatrix} = \begin{pmatrix} \vec{0} \\ \dot{\vec{x}} \end{pmatrix}. \quad (10)$$

where $\hat{\mathbf{W}} = \hat{\mathbf{W}}^{\mathbf{T}}$ is a matrix such that

$$\hat{\mathbf{W}}\dot{\vec{q}} = \mathbf{W}\dot{\vec{q}} + \dot{\mathbf{J}}^{\mathbf{T}}\vec{\mu} + \dot{\mathbf{W}}(\vec{q} - \vec{q}_0). \quad (11)$$

It is well known (e.g. [AdD74]) that this system of equations in (10) is inverted symbolically as:

$$\begin{pmatrix} \dot{\vec{q}} \\ \vec{\mu} \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{B^T} \\ \mathbf{B} & \mathbf{C} \end{pmatrix} \begin{pmatrix} \vec{0} \\ \dot{\vec{x}} \end{pmatrix}. \quad (12)$$

where

$$\mathbf{A} = \hat{\mathbf{W}}^{-1}\left[1 - \mathbf{J^T}(\mathbf{J}\hat{\mathbf{W}}^{-1}\mathbf{J^T})^{-1}\mathbf{J}\hat{\mathbf{W}}^{-1}\right],$$

$$\mathbf{B} = (\mathbf{J}\hat{\mathbf{W}}^{-1}\mathbf{J^T})^{-1}\mathbf{J}\hat{\mathbf{W}}^{-1},$$

and

$$\mathbf{C} = -(\mathbf{J}\hat{\mathbf{W}}^{-1}\mathbf{J^T})^{-1}.$$

Therefore, we can solve the following simultaneously:

$$\dot{\vec{q}} = \hat{\mathbf{W}}^{-1}\mathbf{J^T}(\mathbf{J}\hat{\mathbf{W}}^{-1}\mathbf{J^T})^{-1}\dot{\vec{x}} \quad (13)$$

and

$$\dot{\vec{\mu}} = -(\mathbf{J}\hat{\mathbf{W}}^{-1}\mathbf{J^T})^{-1}\dot{\vec{x}}. \quad (14)$$

Due to numerical inaccuracy, this method can be augmented so that the right hand of (10) is replaced by:

$$\begin{pmatrix} -\alpha\frac{\partial F}{\partial \vec{q}} \\ \dot{\vec{x}} \end{pmatrix}, \quad (15)$$

where for this particular case, $\frac{\partial F}{\partial \vec{q}}$ is the left side of (9). In general, $0 < \alpha < 1$ is a scalar 'gain.' Thus, if the solution is accurate, this term will vanish because (9) holds, but if numerical errors are present, this term will provide corrective feedback to converge to the solution. This result is known in the optimization literature as the Lagrange-Newton method.

5 An Example of the Optimal Design Problem

In this section, the optimal design scheme outlined in the previous section is demonstrated with a variable geometry truss manipulator (VGTM). In order to solve this problem, the forward kinematics for this device must first be derived. This is performed in Subsection 5.1. Section 5.2 applies the general formulation of Sections 3 and 4 to the particular problem of the VGTM.

5.1 Variable Geometry Truss Forward Kinematics

As has been documented in numerous works, the forward kinematics problem for parallel manipulators is generally much more difficult than the inverse kinematics problem. This is the reverse of the serial manipulator case in which the inverse kinematics is more complicated than the forward kinematics. For manipulators such as the variable geometry truss, which is a cascade of parallel modules, the complexity of the forward kinematics problem is a hybrid of the parallel and serial cases.

Figure 6 shows one module of a variable geometry truss manipulator. The forward kinematics problem for each module is the determination of the function $\mathbf{H^i_{i-1}}(q_{3i}, q_{3i+1}, q_{3i+2})$, which maps the truss leg lengths to the position and orientation of the end-effector relative to the base frame. This can be calculated using trigonometric and/or geometric construc-

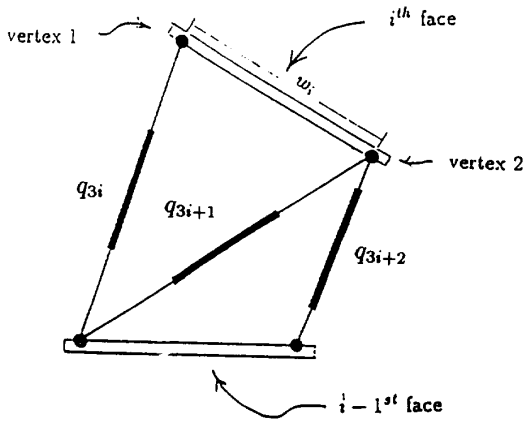

Figure 6: Numbering Convention for a VGT Module

tions. Here, a simple graphical method will be used for the case when the truss width is the same for each module, i.e., $w_i = w$.

Consider the legs with lengths q_j for $j \in [3i, 3i+1, 3i+2]$ in Figure 6. Our goal is to find the positions of 'vertex 1' and 'vertex 2' as a function of leg lengths. The relative position of these vertices with respect to a frame fixed to the left corner of the base of the i^{th} module are denoted $(x_{1,i}, y_{1,i})$ and $(x_{2,i}, y_{2,i})$ respectively. Finding the position and orientation of the top plate with respect to the bottom follow trivially once we have this information.

The following constraints apply:

$$x_{2,i}^2 + y_{2,i}^2 = q_{3i+1}^2 \quad (x_{2,i} - w)^2 + y_{2,i}^2 = q_{3i+2}^2. \quad (16)$$

$$x_{1,i}^2 + y_{1,i}^2 = q_{3i}^2 \quad (x_{1,i} - x_{2,i})^2 + (y_{1,i} - y_{2,i})^2 = w^2. \quad (17)$$

These equations are solved simultaneously to yield

$$x_{2,i} = -\frac{q_{3i+2}^2 - q_{3i+1}^2 - w^2}{2w}$$

$$y_{2,i} = \left(q_{3i+1}^2 - \left(\frac{q_{3i+2}^2 - q_{3i+1}^2 - w^2}{2w} \right)^2 \right)^{\frac{1}{2}}$$

and

$$x_{1,i} = \frac{-b - (b^2 - 4ac)^{\frac{1}{2}}}{2a} \quad y_{1,i} = (q_{3i}^2 - x_{1,i}^2)^{\frac{1}{2}}$$

where

$$a = 4x_{2,i}^2 + 4y_{2,i}^2 \quad b = 4(w^2 x_{2,i} - q_{3i}^2 x_{2,i} - q_{3i+1}^2 x_{2,i})$$

$$c = q_{3i+1}^4 + q_{3i}^4 + 2q_{3i}^2 q_{3i+1}^2 - 2q_{3i}^2 w^2 - 2q_{3i+1}^2 w^2 + w^4 - 4y_{2,i}^2 q_{3i}^2$$

which is written symbolically as

$$x_{j,i} = \hat{x}_{j,i}(q_{3i}, q_{3i+1}, q_{3i+2}) \quad y_{j,i} = \hat{x}_{j,i}(q_{3i}, q_{3i+1}, q_{3i+2})$$

for $j = 1, 2$ and $i = 1, ..., n$.

The forward kinematics of the whole truss structure is then expressed in terms of a sum of contributions from each truss module. The angle of inclination of the i^{th} face with respect to the $i-1^{st}$ face is given by

$$\cos(\Delta\theta_i) = \frac{x_{2,i} - x_{1,i}}{w} \quad \sin(\Delta\theta_i) = \frac{y_{2,i} - y_{1,i}}{w} \quad (18)$$

and so:

$$\Delta\theta_i = \text{Atan2}(y_{2,i} - y_{1,i}, x_{2,i} - x_{1,i}). \quad (19)$$

In order to represent the absolute position of each vertex of the truss in base frame coordinates, the absolute angular displacement of the i^{th} plate is defined as :

$$\theta_i = \sum_{j=0}^{i} \Delta\theta_j \quad (20)$$

where by definition $\Delta\theta = 0$ for the base. The absolute position of each vertex with respect to the center of the base of the truss is then written as:

$$\vec{X}_j^i = \sum_{k=1}^{i} \text{ROT}[\vec{e}_3, \theta_{k-1}] \left[\vec{x}_{k,j} + (1 + (-1)^j) \frac{w}{2} \vec{e}_1 \right]. \quad (21)$$

In the above, $\vec{e}_i$ for $i = 1, 2, 3$ are the natural basis vectors for $\mathbf{R}^3$, and $\text{ROT}[\vec{e}_3, \alpha]$ is the rotation matrix which rotates vectors counterclockwise about the $\vec{e}_3$ axis by an amount α.

With this kinematic information, the manipulator Jacobian matrix is defined by simply taking the appropriate partial derivatives:

$$\mathbf{J}(\vec{q}) = \frac{1}{2} \frac{\partial}{\partial \vec{q}} \left(\vec{X}_1^n + \vec{X}_2^n \right). \quad (22)$$

5.2 Example of the Method

Let us suppose that our goal is to move a single point of the initial workspace to a desired point in a new workspace without regard to any other points. Applying the algorithm from Section 4 (with $\mathbf{W} = \mathbf{1}$) starting with the configuration in Figure 2 as the baseline, and altering kinematic parameters so that the end-effector reaches the point $\vec{x}_{ee} = (0, 0.8)$ while retaining the same binary values, the resulting configuration is that shown in Figure 7. The new joint stops are:

$$\vec{q}^{min} = \begin{pmatrix} 0.150, 0.150, 0.129, 0.154, 0.151, 0.150, 0.150, 0.150, \\ 0.132, 0.148, 0.149, 0.150, 0.150, 0.150, 0.137 \end{pmatrix}^T$$

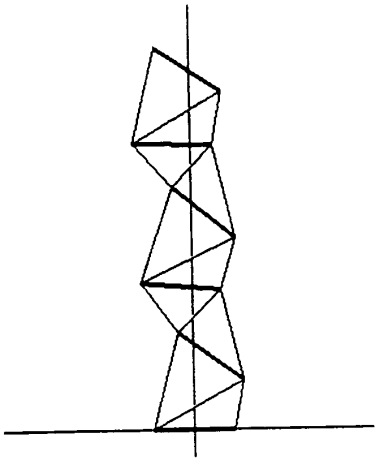

Figure 7: Altered Configuration

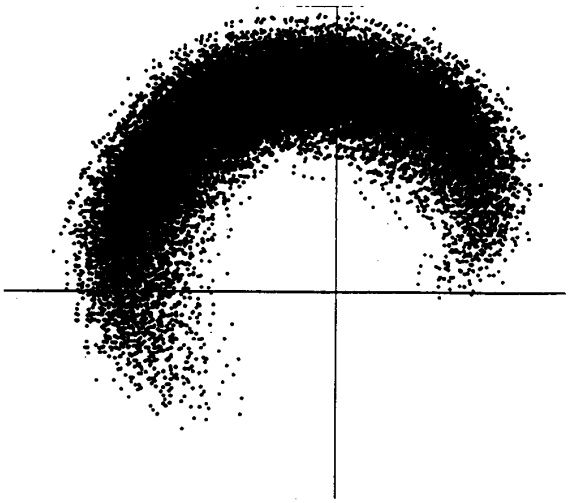

Figure 8: Altered Workspace

and

$$\vec{q}^{max} = \begin{pmatrix} 0.253, 0.258, 0.250, 0.250, 0.250, 0.231, 0.248, 0.260, \\ 0.250, 0.250, 0.250, 0.238, 0.244, 0.258, 0.250 \end{pmatrix}^T$$

Note that for each element of the above vectors, if the ith element has been changed in one, it is not changed in the other.

By changing the joint stops, the workspace in Figure 4 is now altered. The new workspace is shown in Figure 8.

6 Conclusions

This paper has presented a binary paradigm in robotics and has developed one method for solving the problem of optimal design for pick-and-place tasks. The method is based on looking at the kinematic parameters of a manipulator as the variables in the problem, and using methods of constrained optimization to yield a solution. While this method works for relatively low degree-of-freedom manipulators, there is a 'cross over' point beyond which the problem becomes overdetermined, and an exact solution cannot be guaranteed. This occurs when the number of desired points to be reached multiplied by the dimension of the task space exceeds the number of kinematic parameters which can be varied.

Acknowledgements

This work was made possible by the NSF New Young Investigator Award IRI-9357738 from the Robotics and Machine Intelligence Program. Thanks go to Joshua Houck for his development of binary machanical hardware, and Derek Seabury for developing a parallel port interface to run the prototype from a PC.

References

[AdD74] Adby, P.R., Dempster, M.A.H., *Introduction to Optimization Methods*, Chapman and Hall, Ltd., London, 1974.

[AnH67] Anderson, V.C. and Horn, R.C., "Tensor Arm Manipulator Design," ASME paper 67-DE-57.

[CaG] Canny, J., Goldberg, K., "A RISC Paradigm for Industrial Robotics," Technical Report ESRC 93-4/RAMP 93-2. Engineering Systems Research Center, University of California at Berkeley.

[ChB92] Chirikjian, G.S., Burdick, J.W., "Design, Implementation, and Experiments with a Thirty-Degree-of-Freedom 'Hyper-Redundant' Robot" *ISRAM'92*, November, 1992.

[Go92] Goldberg, K., "Orienting Polygonal Parts Without Sensors," Algorithmica, 1992 (special robotics issue).

[KlH83] Klein, C.A., Huang, C.H., "Review of the Pseudoinverse for Control of Kinematically Redundant Manipulators," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-13, No. 2, pp. 245-250, March 1983.

[Ma] Mason, M.T., "Kicking the Sensing Habit," AI magazine, Spring 1993.

[RoRS73] Roth, B, Rastegar, J., and Scheinman, V., "On the Design of Computer Controlled Manipulators," First CISM-IFTMM Symp. on Theory and Practice of Robots and Manipulators, pp. 93-113, 1973. (see pp. 106-108)