ornl

ORNL/TM-13402

OAK RIDGE NATIONAL LABORATORY

LOCKHEED MARTIN

An Improved Method for Calculating Self-Motion Coordinates for Redundant Manipulators

D. B. Reister

MANAGED AND OPERATED BY
LOCKHEED MARTIN ENERGY RESEARCH CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

ORNL-27 (3-96)

DISCLAIMER

Portions of this document may be illegible electronic image products. Images are produced from the best available original document.

ORNL/TM-13402

Computer Science and Mathematics Division

AN IMPROVED METHOD FOR CALCULATING SELF-MOTION COORDINATES FOR REDUNDANT MANIPULATORS*

D. B. Reister

DATE PUBLISHED -April 1997

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
Lockheed Martin Energy Research Corp.
for the
U.S. Department of Energy
under Contract Number DE-AC05-96OR22464

CONTENTS

ABSTRACT

For a redundant manipulator, the objective of redundancy resolution is to follow a specified path in Cartesian space and simultaneously perform another task (For example, maximize an objective function or avoid obstacles) at every point along the path. The conventional methods have several drawbacks: a new function must be defined for each task, the extended Jacobian can be singular, closed cycles in Cartesian space may not yield closed cycles in joint space, and the objective is point-wise redundancy resolution (to determine a single point in joint space for each point in Cartesian space).

We divide the redundancy resolution problem into two parts: (1) calculate self-motion coordinates for all possible positions of a manipulator at each point along a Cartesian path and (2) determination of optimal self-motion coordinates that maximize an objective function along the path. This paper will discuss the first part of the problem. Our path-wise approach overcomes all of the drawbacks of conventional redundancy resolution methods: we do not need to define a new function for each task, our extended Jacobian cannot be singular, and closed cycles in extended Cartesian space will yield closed cycles in joint space.

1. INTRODUCTION

A manipulator is a set of links connected in a chain by joints. Usually, each joint has one degree of freedom. The state of the manipulator is described by a vector of joint variables, θ. All points on the manipulator move in Cartesian space. We are usually interested in the position and orientation of the end of the manipulator. The position and orientation of end of the manipulator are described by a vector ($\mathbf{x}$) with six components in 3D space and three components in 2D space. There is a unique mapping from the joint space to the Cartesian space:

$$\mathbf{x} = \mathbf{f}(\theta) \tag{1}$$

A redundant manipulator has more joint variables than position variables. Thus, a redundant manipulator that can move in 3D space has more than six joint variables. The objective of redundancy resolution is to find a trajectory in joint space [θ(t)] that will keep the end of the manipulator on a given path [$\mathbf{x}^G$(t)]. Since the redundant manipulator has extra degrees of freedom, it can follow a given path and simultaneously accomplish other objectives: avoid obstacles, avoid joint limits, minimize joint torques.

Most of the literature on redundancy resolution begins with the time derivative of Eq. (1):

$$\dot{\mathbf{x}} = \mathbf{J}\,\dot{\theta} \tag{2}$$

where $\mathbf{J}$ is the Jacobian of $\mathbf{f}$.

In 1969, Whitney solved the redundancy resolution problem by finding the minimum length vector of joint velocities that satisfied Eq. (2):

$$\dot{\theta} = \mathbf{J}^{\mathbf{p}}\,\dot{\mathbf{x}} \tag{3}$$

where $\mathbf{J}^{\mathbf{p}}$ is the Moore-Penrose generalized inverse of $\mathbf{J}$:

$$\mathbf{J}^{\mathbf{p}} = \mathbf{J}^{\mathbf{T}}(\mathbf{J}\mathbf{J}^{\mathbf{T}})^{-1} \tag{4}$$

In 1983, Klein and Huang discovered that Whitney's solution did not yield repeatable paths in joint space; at the end of a closed cycle in Cartesian space ($\mathbf{x}$), the joints might not return to their initial positions. In 1988, Shamir and Yomdin published a necessary and sufficient condition for repeatable paths in joint space.

In 1985, Baillieul developed the extended Jacobian technique for redundancy resolution. He considers a local optimization problem: maximize an objective function [$g(\theta)$] subject to the constraint provided by Eq. (1). He proves the following Theorem.

Theorem. A necessary condition for θ_0 to define a local maximum of an objective function [$g(\theta)$] subject to the positional constraint [Eq. (1)] is that:

$$G(\theta) = \frac{\partial g(\theta)}{\partial \theta} \psi = 0 \text{ at } \theta_0 \tag{5}$$

for all vectors ψ in the null space of $\mathbf{J}$.

While the end of the manipulator is following a given path [$\mathbf{x}^G(t)$] and simultaneously maximizing the objective function, Eq. (5) will be satisfied for all of the basis vectors of the null space. Thus, Eq. (5) becomes a vector equation:

$$\mathbf{G}(\theta) = \mathbf{0} \tag{6}$$

along the optimal trajectory.

Define a constraint Jacobian ($\mathbf{J_c}$) by:

$$\mathbf{J_c} = \frac{\partial \mathbf{G}(\theta)}{\partial \theta} \tag{7}$$

Baillieul defines the extended Jacobian ($\mathbf{J_e}$) by:

$$\mathbf{J_e} = \begin{bmatrix} \mathbf{J} \\ \mathbf{J_c} \end{bmatrix} \tag{8}$$

Along an optimal trajectory, the joint velocities satisfy:

$$\mathbf{J_e}\,\dot{\theta} = \begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{0} \end{bmatrix} \tag{9}$$

If the extended Jacobian is nonsingular, Eq. (9) can be solved for the optimal joint velocities. Baillieul's extended Jacobian method will yield repeatable paths in joint space.

In 1989, Seraji a developed the configuration control method for redundancy resolution. His method is not based on local optimization. He adds an additional task ($\mathbf{z}$) that will be performed by the manipulator:

$$\mathbf{z} = \mathbf{g}(\theta) \tag{10}$$

The task Jacobian ($\mathbf{J_c}$) is given by:

$$\mathbf{J_c} = \frac{\partial \mathbf{g}(\theta)}{\partial \theta} \tag{11}$$

The extended Jacobian ($\mathbf{J_e}$) is given by Eq. (8) and the joint velocities satisfy:

$$\mathbf{J_e}\,\dot{\theta} = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{z}} \end{bmatrix} \tag{12}$$

If the extended Jacobian is nonsingular, Eq. (12) can be solved for the joint velocities. However, the primary problem with practical applications of Seraji's method is that the extended Jacobian can be singular: when $\mathbf{J}$ is not of full rank, when $\mathbf{J_c}$ is not of full rank, and when the rows of $\mathbf{J_c}$ are not linearly independent of the rows of $\mathbf{J}$.

Since the extended Jacobian can be singular, Seraji developed an improved configuration control method in 1990. The improved method is based upon a damped least-squares solution to a set of linear equations. The method suppresses large joint velocities near singularities, at the expense of small task trajectory errors in Eq. (12).

Examples of tasks are: obstacle avoidance, elbow control, and minimization of the distance between the joint angles and a goal for the joint angles. In general, calculation of a task function and its Jacobian is very difficult for a realistic application. For example, the minimum distance from any point on the manipulator to any obstacle in the environment will be a continuous function of the joint angles. However as the arm moves in the workspace, the point on the arm that is closest to an obstacle can have discontinuous motions. Consequently, the derivative of the function may not exist at many points in joint space. Furthermore, each additional obstacle requires a new definition of the task function.

If $\mathbf{x}$ is fixed, Eq. (1) defines a manifold: the self-motion manifold. In 1989, Burdick discussed the features of the self-motion manifolds. The inverse kinematic solution (or preimage) is a set of points in joint space that are mapped to $\mathbf{x}$ by Eq. (1). In general, the preimage $[\mathbf{f}^{-1}(\mathbf{x})]$ is the union of disjoint s dimensional manifolds:

$$\mathbf{f}^{-1}(\mathbf{x}) = \bigcup_i \mathbf{M}_i(\mathbf{q}) \tag{13}$$

where $k = n - m$, $\boldsymbol{\theta}$ is an n vector, $\mathbf{x}$ is an m vector, and $\mathbf{M}_i(\mathbf{q})$ is an k dimensional self-motion manifold with local coordinates $\mathbf{q}$.

In 1991, Burdick defined two types of redundancy resolution methods: configuration resolution and path-wise resolution. A configuration resolution method finds a point in joint space $[\boldsymbol{\theta}]$ that will keep the end of the manipulator at a given point $[\mathbf{x}^G]$ and maximize a scalar objective function $[g(\boldsymbol{\theta})]$. A path-wise method finds a trajectory in joint space $[\boldsymbol{\theta}(t)]$ that will keep the end of the manipulator on a given path $[\mathbf{x}^G(t)]$ and maximize a scalar objective function that is computed over the entire path. (We believe that point-wise resolution is a better name than configuration resolution.)

Furthermore, there are two types of solutions for the two types of redundancy resolution methods: global and local. A global method yields a solution that has the highest possible value for the objective function. If a solution method is not global, it is called local. After noting that all current redundancy resolution methods are local, Burdick develops a global method for configuration resolution.

We are interested in global path-wise methods for motion planning for a redundant manipulator in a cluttered environment. Consider a manipulator that is performing assembly tasks under the dash of a car. In some regions of task space, the primary objective of redundancy resolution is to avoid joint limits while in other regions the objective is to avoid obstacles. A general motion planning algorithm would be capable of planning transitions from one of the self-motion manifolds to another: to move from an

elbow up configuration in one region of task space to an elbow down configuration in another region.

At each point in task space, the self-motion coordinates define all of the points in the preimage. At each point in the preimage, several scalar objective functions (for obstacle avoidance, joint limit avoidance, minimizing joint torque, and avoiding singular states) can be computed. The path-wise motion planner would find a trajectory in joint space that would maximize a weighted sum of the multiple scalar objective functions over the entire path in task space.

The advantages of using self-motion coordinates for posture control are discussed in Seraji (1990). Since the self-motion coordinates can be difficult to calculate, Seraji has used the arm angle (the angle between the arm plane and a reference plane) for a 7 joint arm [see Seraji (1991)]. However, the extended Jacobian can be singular when the arm angle is the task function.

In this paper, we develop an improved method for calculating self-motion coordinates. Our method does not require the calculation of local coordinates for the self-motion manifold. We work in the tangent space of the self-motion manifold. Given the Jacobian, we use the standard methods of linear algebra to define self-motion parameters and self-motion coordinates. Our approach overcomes all of the drawbacks of conventional redundancy resolution methods: we do not need to define a new function for each task, our extended Jacobian cannot be singular, and closed cycles in extended Cartesian space will yield closed cycles in joint space.

In the next section, we will develop our method to parameterize the self-motion manifold and define a self-motion coordinate. We have developed and demonstrated a control system that allows us to move the manipulator to a position specified by both Cartesian position and orientation and self-motion coordinate. In the third section, we will discuss our experimental results. The fourth section will present our conclusions.

2. THEORY

A manipulator (or a machine tool) is a set of links connected in a chain by joints. Usually, each joint has one degree of freedom. The state of the manipulator is described by a vector of joint variables, $\boldsymbol{\theta}$. All points on the manipulator move in Cartesian space. We are usually interested in the position and orientation of the end of the manipulator, which is described by a vector ($\mathbf{x}$) with six components in 3D space and three components in 2D space. If $\mathbf{x}$ is fixed, the joint variables are constrained to move on the self-motion manifold. If $\boldsymbol{\theta}$ is an n vector, $\mathbf{x}$ is an m vector, and the rank of $\mathbf{J}$ is r, then the dimension (k) of the self-motion manifold is: $k = n - r$. Any point on the manifold can be defined by local coordinates ($\mathbf{q}$), where $\mathbf{q}$ is a k vector. We can express the mapping from $\mathbf{q}$ to the manifold by:

$$\mathbf{q} = \mathbf{g}(\boldsymbol{\theta}) \tag{14}$$

Mathematically, Eq. (14) is identical to Eq. (10). However, we are defining local coordinates for the manifold, while Seraji is defining an additional task for the manipulator. The Jacobian for Eq. (14) is given by Eq. (11). To simplify the notation and make the notation consistent with our previous work, we will call the self-motion Jacobian $\mathbf{B}$:

$$\mathbf{B} = \frac{\partial g(\theta)}{\partial \theta} \tag{15}$$

Since the elements of the **B** matrix are obtained by taking partial derivatives of **g**, the partial derivatives of the elements of the **B** matrix will be symmetrical:

$$\frac{\partial B_{ki}}{\partial \theta_j} = \frac{\partial^2 g_k}{\partial \theta_i \partial \theta_j} = \frac{\partial B_{kj}}{\partial \theta_i} \tag{16}$$

If r (the rank of **J**) is less than m, we will remove redundant rows and reduce the number of rows in **J** (and the components of $\dot{\mathbf{x}}$) to r. We shall call the extended Jacobian **K**:

$$\mathbf{K} = \begin{bmatrix} \mathbf{J} \\ \mathbf{B} \end{bmatrix} \tag{17}$$

From this point on, we will consider **B** to be an arbitrary k×n matrix. We will choose **B** to make the **K** matrix nonsingular. Later we will discuss options for the choice of the **B** matrix. Thus, we may not be able to find a **g** function that is consistent with **B** and **B** may not satisfy Eq. (16). Since the **q** may not be coordinates, we will call them self-motion parameters.

Using the **K** matrix, the kinematic equation relating joint velocities to Cartesian and manifold velocities is:

$$\mathbf{K}\dot{\theta} = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{q}} \end{bmatrix} \tag{18}$$

The inverse of the **K** matrix is partitioned into two matrices (**E** and **F**):

$$\mathbf{K}^{-1} = [\, \mathbf{E} \quad \mathbf{F} \,] \tag{19}$$

The matrices **J**, **B**, **E**, and **F** satisfy:

$$\mathbf{J}\mathbf{E} = \mathbf{I} \tag{20}$$
$$\mathbf{J}\mathbf{F} = \mathbf{0} \tag{21}$$
$$\mathbf{B}\mathbf{E} = \mathbf{0} \tag{22}$$
$$\mathbf{B}\mathbf{F} = \mathbf{I} \tag{23}$$
$$\mathbf{E}\mathbf{J} + \mathbf{F}\mathbf{B} = \mathbf{I} \tag{24}$$

Using the inverse of the **K** matrix, the solution of Eq. (18) is:

$$\dot{\theta} = \mathbf{E}\dot{\mathbf{x}} + \mathbf{F}\dot{\mathbf{q}} \tag{25}$$

The solution has the form of a particular solution plus a null space solution (**F** is in the null space of **J**).

In Seraji's configuration control method, **B** is the Jacobian of an arbitrary task function. The extended Jacobian (**K**) will be singular if $\mathbf{B}^T$ does not have a sufficient number of components in the null space of **J** and Eq. (23) is not satisfied.

For any arbitrary matrix **B** that makes the **K** matrix nonsingular, the self-motion parameters (**q**) are determined by:

$$\dot{\mathbf{q}} = \mathbf{B}\,\dot{\boldsymbol{\theta}} \tag{26}$$

with the initial condition $\mathbf{q}(0) = \mathbf{0}$.

An arbitrary matrix **B** may not satisfy Eq. (16). However, for an arbitrary constant matrix (**C**) the partial derivatives of the elements of the **C** matrix will be zero and will satisfy Eq. (16). Let **C** be the initial value of **B**:

$$\mathbf{C} = \mathbf{B}(0) \tag{27}$$

The self-motion coordinates **p** are determined by:

$$\dot{\mathbf{p}} = \mathbf{C}\,\dot{\boldsymbol{\theta}} \tag{28}$$

with the initial condition $\mathbf{p}(0) = \mathbf{0}$. If **B** is an appropriate matrix, then -**B** could also be used. To overcome this ambiguity, we require that the matrix $1/2(\mathbf{C}^T\mathbf{B} + \mathbf{B}^T\mathbf{C})$ be positive definite. If the smallest positive eigenvalue of $1/2(\mathbf{C}^T\mathbf{B} + \mathbf{B}^T\mathbf{C})$ approaches zero, we choose a new value for **C**: $\mathbf{C} = \mathbf{B}(t^*)$. If we choose **B** (and **C**)to be dimensionless, the units of **p** and **q** are radians.

Our goal is to develop a motion control system that allows us to move the manipulator to a position specified by both Cartesian position and orientation and self-motion coordinates (see Figure 1). For any arbitrary matrix **B** that makes the **K** matrix nonsingular, Eq. (25) can be used to calculate $\dot{\boldsymbol{\theta}}$ from inputs of $\dot{\mathbf{x}}$ and $\mathbf{q}$. The motion control system requires an error calculation system that calculates the changes in **x** and **q** ($\dot{\mathbf{x}}$ and $\dot{\mathbf{q}}$). The inputs to the error calculation system are goals for both the position and orientation of end of the manipulator ($\mathbf{x}^G$) and for the self-motion coordinates ($\mathbf{p}^G$) and the current values of the state variables (**x** and **p**). To completely define the motion control system, we must define **x**, $\dot{\mathbf{x}}$, **J**, **B**, and the error calculation system. We begin with **B**.

2.1. Definition of the B Matrix.

We would like to parameterize all possible choices for the **B** matrix. We need to develop a basis for n vectors. A natural basis is provided by the range of $\mathbf{J}^T$ and a basis for the null space of **J**. To calculate a basis for the null space of **J**, we apply Singular

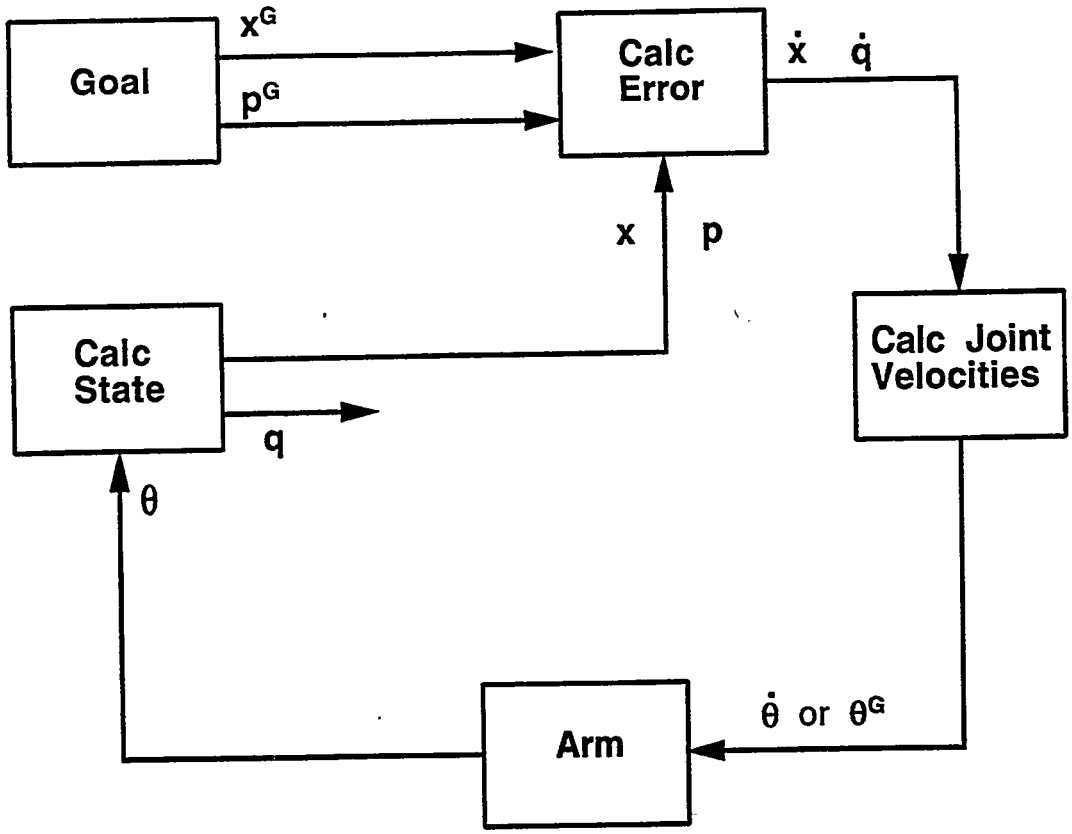

Fig. 1. The goal for the motion control system is specified by both Cartesian position and orientation and self-motion coordinates.

Value Decomposition (SVD) to the matrix **K** with a zero **B** matrix: $\begin{bmatrix} \mathbf{J} \\ \mathbf{0} \end{bmatrix}$. The SVD algorithm provides an orthonormal basis for the null space, **V**:

$$\mathbf{J}\,\mathbf{V} = \mathbf{0} \tag{29}$$

$$\mathbf{V}^{\mathbf{T}}\,\mathbf{V} = \mathbf{I} \tag{30}$$

Using the basis [$\mathbf{J}^{\mathbf{T}}$ and **V**], we can write general expressions for **B**, **E**, and **F**:

$$\mathbf{F} = \mathbf{V}\,\boldsymbol{\alpha} \tag{31}$$

$$\mathbf{B}^{\mathbf{T}} = \mathbf{J}^{\mathbf{T}}\,\boldsymbol{\gamma} + \mathbf{V}\,\boldsymbol{\lambda} \tag{32}$$

$$\mathbf{E} = \mathbf{J}^{\mathbf{T}}\,\boldsymbol{\eta} + \mathbf{V}\,\boldsymbol{\sigma} \tag{33}$$

where $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$, $\boldsymbol{\lambda}$, $\boldsymbol{\eta}$, and $\boldsymbol{\sigma}$ are parameter matrices.

If **E** satisfies Eq. (20), the $\boldsymbol{\eta}$ parameter matrix must satisfy:

$$\boldsymbol{\eta} = (\mathbf{J}\mathbf{J}^{\mathbf{T}})^{-1} \tag{34}$$

If **B** and **E** satisfy Eq. (22), the $\boldsymbol{\gamma}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\sigma}$ parameter matrices must satisfy:

$$\boldsymbol{\gamma}^{\mathbf{T}} + \boldsymbol{\lambda}^{\mathbf{T}}\,\boldsymbol{\sigma} = \mathbf{0} \tag{35}$$

If **B** and **F** satisfy Eq. (23), the $\boldsymbol{\lambda}$, and $\boldsymbol{\alpha}$ parameter matrices must satisfy:

$$\boldsymbol{\lambda}^{\mathbf{T}}\,\boldsymbol{\alpha} = \mathbf{I} \tag{36}$$

A convenient solution for Eq. (36) is obtained when the **F** matrix is orthonormal: $\mathbf{F} = \mathbf{V}$, $\boldsymbol{\alpha} = \mathbf{I}$, and $\boldsymbol{\lambda} = \mathbf{I}$.

The particular solution (**E**) is the sum of the Moore-Penrose generalized inverse of **J** [Eq. (4)] and a component in the null space. The **B** matrix has a component in the null space ($\mathbf{F}^{\mathbf{T}}$) and a component in the range of **J**. The tradeoffs between the two components are expressed by Eq. (35). We will choose **B** to break the connection and place **B** in the null space with **E** in the **J** space. Thus, $\boldsymbol{\gamma} = \boldsymbol{\sigma} = \mathbf{0}$ and our choices are:

$$\mathbf{B}^{\mathbf{T}} = \mathbf{F} = \mathbf{V} \tag{37}$$

$$\mathbf{E} = \mathbf{J}^{\mathbf{T}}(\mathbf{J}\mathbf{J}^{\mathbf{T}})^{-1} = \mathbf{J}^{\mathbf{p}} \tag{38}$$

where $\mathbf{J}^{\mathbf{p}}$ is the Moore-Penrose generalized inverse of **J** [see Eq. (4)].

When we do not have null space motion [$\dot{\mathbf{q}} = \mathbf{0}$], our solution is the Moore-Penrose generalized inverse of **J** and Klein and Huang have demonstrated that the Moore-Penrose solution did not yield repeatable paths in joint space; at the end of a closed cycle in Cartesian space (**x**), the joints might not return to their initial positions. In our experimental results, we will find that closed cycle paths in **x** and **p** space result in

repeatable paths in joint space. However, the final values of **q** may not be equal to the initial values.

Unlike the extended Jacobian of Baillieul and the configuration control method of Seraji, our method cannot have kinematic singularities or algorithmic singularities. If **J** is rank deficient, the null space of **J** becomes larger. The **K** matrix cannot be singular.

We conclude this subsection by developing a simple example to illustrate the differences between local coordinates for the self-motion manifold and our self-motion parameters. Consider a manipulator with one prismatic joint and one revolute joint:

$$x = \theta_1 + 0.5 \cos \theta_2$$

The Jacobian is given by:

$$J = [\, 1 \;\; -0.5 \sin \theta_2].$$

If B^T is in the null space of J:

$$B = [0.5 \sin \theta_2 \;\; 1].$$

and the **K** matrix is never singular: $|\mathbf{K}| = 1 + 0.25 \sin^2 \theta_2$.

Since B does not satisfy Eq. (16), the self-motion parameters are not self-motion coordinates.

Two sets of self-motion coordinates are: θ_1 and θ_2. If $g = \theta_1$, $B = [1 \;\; 0]$ and $|\mathbf{K}| = 0.5 \sin \theta_2$ which is singular when $\theta_2 = 0$.
If $g = \theta_2$, $B = [0 \;\; 1]$ and $|\mathbf{K}| = 1$ which is never singular.

2.2. Definition of the **x** vector.

We have been discussing the position and orientation of the end of the manipulator, the vector **x** and its derivative $\dot{\mathbf{x}}$. However, there are several different ways to define these vectors. In this section, we will discuss our choices. In general, rigid body motions consist of both a rotation and a translation. The representation of rigid body motions requires both a rotation matrix and a translation vector. The standard method for defining the position and orientation of end of the manipulator in 3D space is by a 4×4 homogeneous transformation matrix [**T**(t)] that maps from a fixed coordinate frame (F) to a moving coordinate frame (M):

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix} \tag{39}$$

where **R** is a 3×3 rotation matrix, and **d** is a three vector for translation. In the language of group theory, rotation matrices are in the group SO(3) and transformation matrices are in the group SE(3). We would like to map from **T** to **x** and from **x** to **T**. The vector **d** provides three of the components of **x**. The difficult problem is to find a three vector that

maps to **R**. Options include: Euler angles, quaternions, Rodrigues' parameters and exponential coordinates.

Euler's theorem [Goldstein (1980)] states that any rotation can be expressed as a rotation about a vector (the axis of rotation). Initially, we used the method of Mladenova (1990) and associated a rotation matrix with a vector **c**. The vector is the axis of rotation and the magnitude of the vector is related to the angle of rotation (ϕ) by: $c = \tan \phi/2$. However, a rotation of 180 degrees results in a vector with an infinite magnitude. A more convenient representation is when the magnitude of the vector is equal to the angle of rotation. Thus, we use exponential coordinates and map the rotation matrix to a vector (**h**) where the direction is determined by the axis of rotation and the magnitude is the angle of rotation. This mapping is well known [see Paul (1981) or Craig (1986) or Murray (1994)]. Using the rotation vector (**h**), the **x** vector is given by:

$$\mathbf{x} = \begin{bmatrix} \mathbf{h} \\ \mathbf{d} \end{bmatrix} \tag{40}$$

McCarthy (1990) demonstrates that the motion of the end of the manipulator is determined by the tangent operator (**S**):

$$\mathbf{S} = \dot{\mathbf{T}}\ \mathbf{T}^{-1} \tag{41}$$

The tangent operator for 3D motion is:

$$\mathbf{S} = \begin{bmatrix} \boldsymbol{\Omega} & \mathbf{v} \\ \mathbf{0} & 0 \end{bmatrix} \tag{42}$$

where $\boldsymbol{\Omega}$ is the angular velocity matrix (and is skew symmetric), and **v** is a 3 component velocity vector. Let **w** be the vector associated with the $\boldsymbol{\Omega}$ matrix:

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \tag{43}$$

and let **Q** be the six component vector that maps to **S**:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} \tag{44}$$

Murray (1994) calls **Q** a twist. In the language of group theory, angular velocity matrices are in the group so(3) and **S** matrices are in the group se(3). There is a one to one exponential map from so(3) to SO(3) and from se(3) to SE(3): $\mathbf{R} = e^{\boldsymbol{\Omega}}$ and $\mathbf{T} = e^{\mathbf{S}}$.

We will identify **Q** with $\dot{\mathbf{x}}$ in Eq. (2):

$$\dot{\mathbf{x}} = \mathbf{Q} \tag{45}$$

Unfortunately, **Q** is not the time derivative of **x** defined by Eq. (40).

2.3. Calculation of the J matrix.

The derivation in this subsection follows McCarthy (1990). Murray (1994) has a similar derivation. In Eq. (2), the Jacobian ($\mathbf{J}$) is the partial derivative of $\mathbf{f}$ with respect to the joint variables ($\boldsymbol{\theta}$). To calculate $\mathbf{J}$, the position and orientation of the end effector must be related to the joint angles. Since the $\mathbf{T}$ matrix depends on the joint variables [$\mathbf{T} = \mathbf{T}(\boldsymbol{\theta})$], the time derivative of the $\mathbf{T}$ matrix is given by:

$$\dot{\mathbf{T}} = \sum_i \frac{\partial \mathbf{T}(\boldsymbol{\theta})}{\partial \theta_i} \dot{\theta}_i \tag{46}$$

The tangent operator $\mathbf{S}$ is defined by Eq. (41):

$$\mathbf{S} = \sum_i \dot{\theta}_i \frac{\partial \mathbf{T}(\boldsymbol{\theta})}{\partial \theta_i} \mathbf{T}^{-1} \tag{47}$$

We define the partial tangent matrices ($\mathbf{S}_i$) by:

$$\mathbf{S}_i = \frac{\partial \mathbf{T}(\boldsymbol{\theta})}{\partial \theta_i} \mathbf{T}^{-1} \tag{48}$$

Using the partial tangent matrices, the $\mathbf{S}$ matrix may be written:

$$\mathbf{S} = \sum_i \mathbf{S}_i \dot{\theta}_i \tag{49}$$

Let $\mathbf{Q}$ be the vector that determines the $\mathbf{S}$ matrix and let $\mathbf{Q}_i$ be the vector that determines the $\mathbf{S}_i$ matrix:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} \tag{50}$$

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{w}_i \\ \mathbf{v}_i \end{bmatrix} \tag{51}$$

McCarthy (1990) demonstrates that:

$$\mathbf{Q} = \sum_i \mathbf{Q}_i \dot{\theta}_i \tag{52}$$

Our goal in this subsection is to define **J** (the Jacobian matrix). In the last subsection, we defined $\dot{\mathbf{x}}$ to be equal to **Q** [Eq. (45)]. Since Eq. (52) relates $\dot{\mathbf{x}}$ to $\dot{\theta}$, it has the same form as the definition of **J** [Eq. (2)]. We define the columns of **J** to be the vectors $\mathbf{Q}_i$. Thus, Eq. (52) may be written:

$$\mathbf{Q} = \mathbf{J}\,\dot{\theta} \tag{53}$$

The **T** matrix is the product of **A** matrices and each **A** matrix depends on a single joint variable:

$$\mathbf{T} = \prod_i \mathbf{A}_i\,(\theta_i) \tag{54}$$

The first partial tangent matrix is given by:

$$\mathbf{S}_1 = \frac{d\mathbf{A}_1}{d\theta_1}(\mathbf{A}_1)_{-1} \tag{55}$$

Thus, $\mathbf{S}_1$ only depends on θ_1.

If we define the partial **T** matrices ($\mathbf{U}_i$) by:

$$\mathbf{U}_1 = \mathbf{A}_1 \tag{56}$$

$$\mathbf{U}_i = \mathbf{U}_{i\text{-}1}\,\mathbf{A}_i \quad \text{for i greater than 1} \tag{57}$$

The partial tangent matrices are given by:

$$\mathbf{S}_i = \mathbf{U}_{i-1}\frac{d\mathbf{A}_i}{d\theta_i}(\mathbf{U}_i)_{-1} \quad \text{for i greater than 1} \tag{58}$$

Each $\mathbf{S}_i$ depends on the first i joint angles: from θ_1 to θ_i.

2.4. Solution of the control problem.

Our goal is to develop a motion control system that allows us to move the manipulator to a position specified by both Cartesian position and orientation and self-motion coordinates (see Figure 1). For any arbitrary matrix **B** that makes the **K** matrix nonsingular, Eq. (25) can be used to calculate $\dot{\theta}$ from inputs of $\dot{\mathbf{x}}$ and $\dot{\mathbf{q}}$. The motion control system requires an error calculation system that calculates the changes in **x** and **q** ($\dot{\mathbf{x}}$ and $\dot{\mathbf{q}}$). The inputs to the error calculation system are goals for both the position and orientation of the end of the manipulator ($\mathbf{x}^G$) and for the self-motion coordinates ($\mathbf{p}^G$) and the current values of the state variables (**x** and **p**). Rather than using the **x** vector directly, we will map to the **T** matrix (see Figure 2).

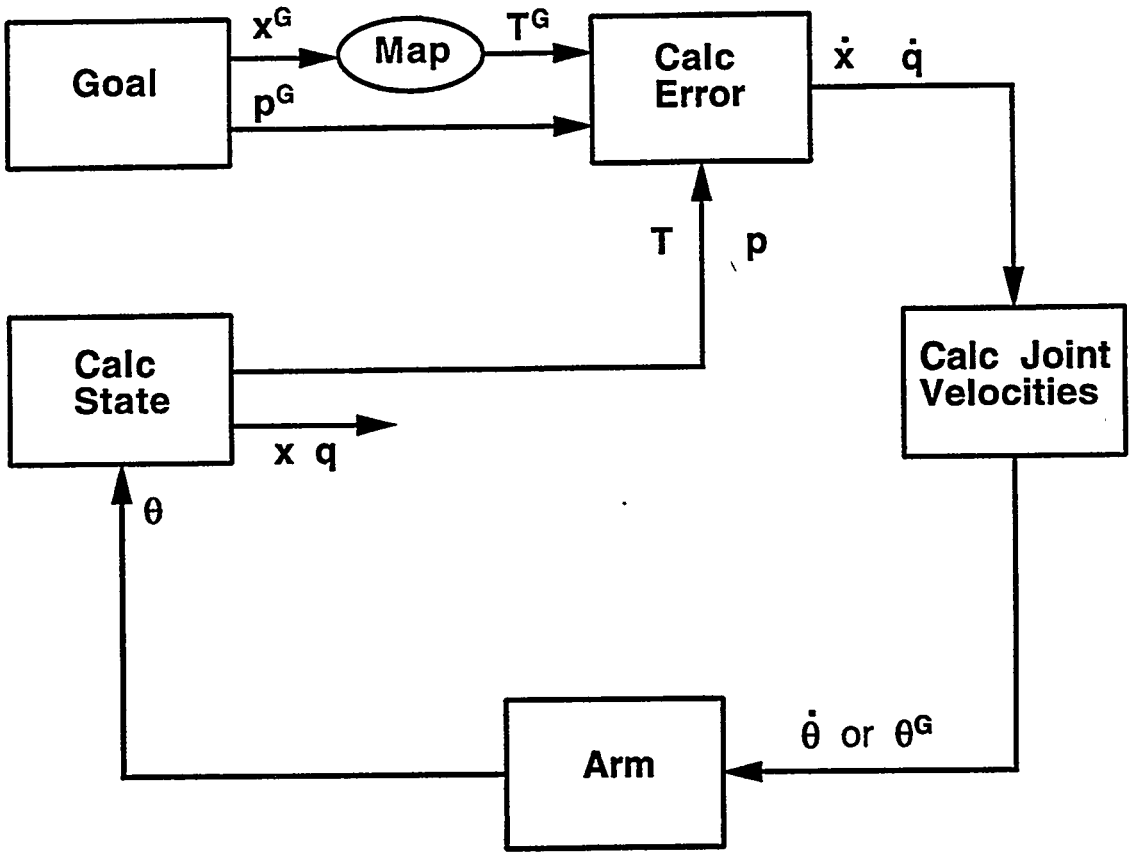

Fig. 2. The Cartesian position and orientation goal ($\mathbf{x}^G$) for the motion control system is mapped to a **T** matrix before calculating the error.

Given a Cartesian goal [$\mathbf{x}^G(t)$], we can calculate a goal for the **T** matrix [$\mathbf{T}^G(t)$]. Let $\mathbf{T}^C$ be a correction **T** matrix that moves the system from its current state [**T**] to the goal:

$$\mathbf{T}^G = \mathbf{T}^C \mathbf{T} \tag{59}$$

Solving Eq. (59):

$$\mathbf{T}^C = \mathbf{T}^G \mathbf{T}^{-1} \tag{60}$$

We can calculate a matrix (**L**) that is the logarithm of $\mathbf{T}^C$:

$$\mathbf{T}^C = \exp \mathbf{L} \tag{61}$$

Brockett (1984) noted that any **T** matrix can be mapped to a logarithm matrix with the same structure as the right side of Eq. (42). Thus, the **L** matrix consists of a skew symmetric matrix and a vector and can be mapped to a six component **Q** vector. If we define a **T** matrix ($\mathbf{\Lambda}$) by:

$$\mathbf{\Lambda} = \exp t\,\mathbf{L} \tag{62}$$

and take the time derivative of $\mathbf{\Lambda}$, we find:

$$\dot{\mathbf{\Lambda}} = \mathbf{L} \exp t\,\mathbf{L} \tag{63}$$

Using the definition of the tangent operator [Eq. (41)]:

$$\mathbf{S} = \mathbf{L} \tag{64}$$

As t increases from 0 to 1, $\mathbf{\Lambda}$ moves from **I** to $\mathbf{T}^C$. The **L** matrix can be mapped to a **Q** vector. The **Q** vector determines $\dot{\mathbf{x}}$ [see Eq. (45)], one of the two terms on the right side of Eq. (25) that determine the joint space move [$\dot{\boldsymbol{\theta}}$]. We calculate the other term [$\dot{\mathbf{q}}$] as the difference between the goal for the null space coordinate ($\mathbf{p}^G$) and the current state (**p**):

$$\dot{\mathbf{q}} = \mathbf{p}^G - \mathbf{p} \tag{65}$$

where **p** is determined by integrating Eq. (28).

If the manipulator moved from **I** to $\mathbf{T}^C$ in one second, we might exceed the limits on the joint space velocities. If any of the calculated joint space velocities violate the speed limits, we use a scale factor to reduce all of the calculated joint space velocities and increase the estimated time required for the motion. Since the system is highly nonlinear, the estimated time of arrival may not be accurate but it should steadily decrease as we near the goal.

3. EXPERIMENTAL RESULTS

We have created a software system that performs the calculations described in the last section to control a seven degree of freedom manipulator, our Robotics Research manipulator (see Figure 3). As shown in Figure 4, our software system is modular and communication between the modules is through shared memory using the Helix software [see Jones (1992)]. There are three data structures in shared memory: aGoal, rrGoal, and aState. The aGoal is a Cartesian goal with position and orientation ($\mathbf{x}^G$) and self-motion coordinate (p^G). The rrGoal is a joint goal with either joint velocity or joint position. The aState contains data on the current state of the system and some details of the calculation of the self-motion coordinates. There are four modules in the software system: Control, Control_x, Monitor_x, and Capture. Each of the modules runs at 60 Hz using a real time operating system (VxWorks).

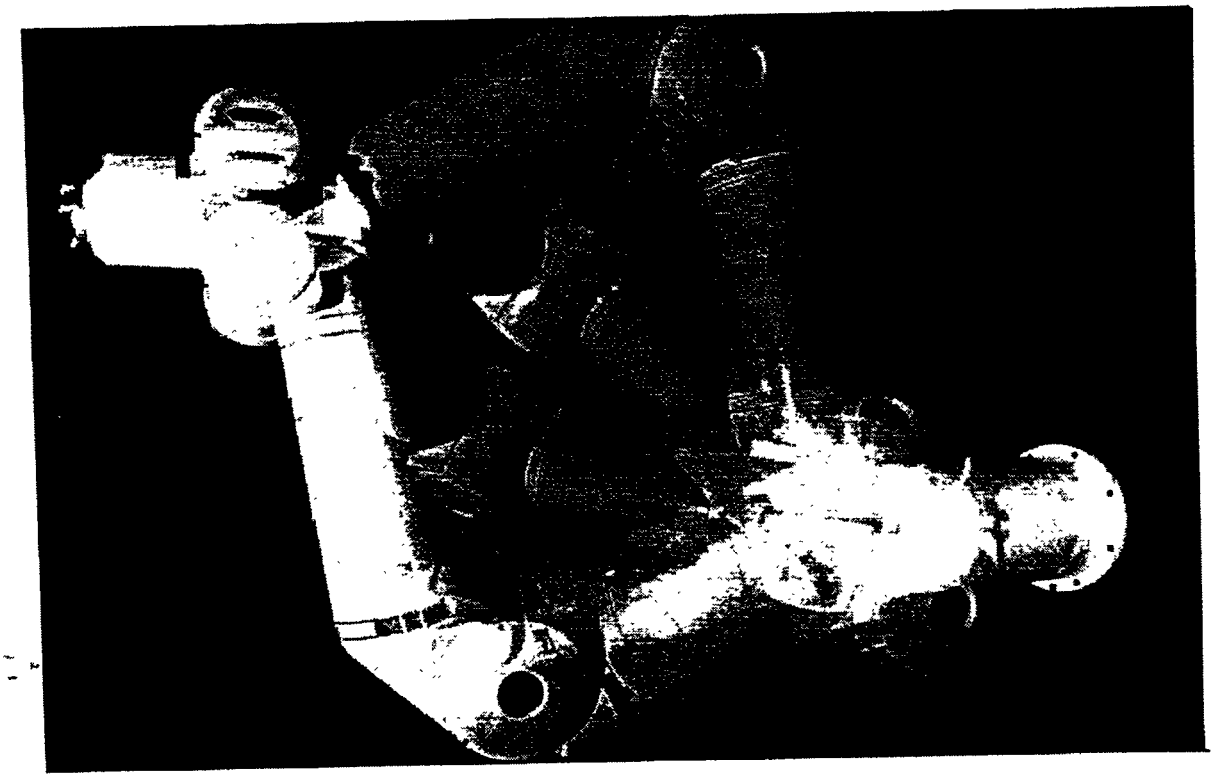

Fig. 3. The ORNL Robotics Research manipulator.

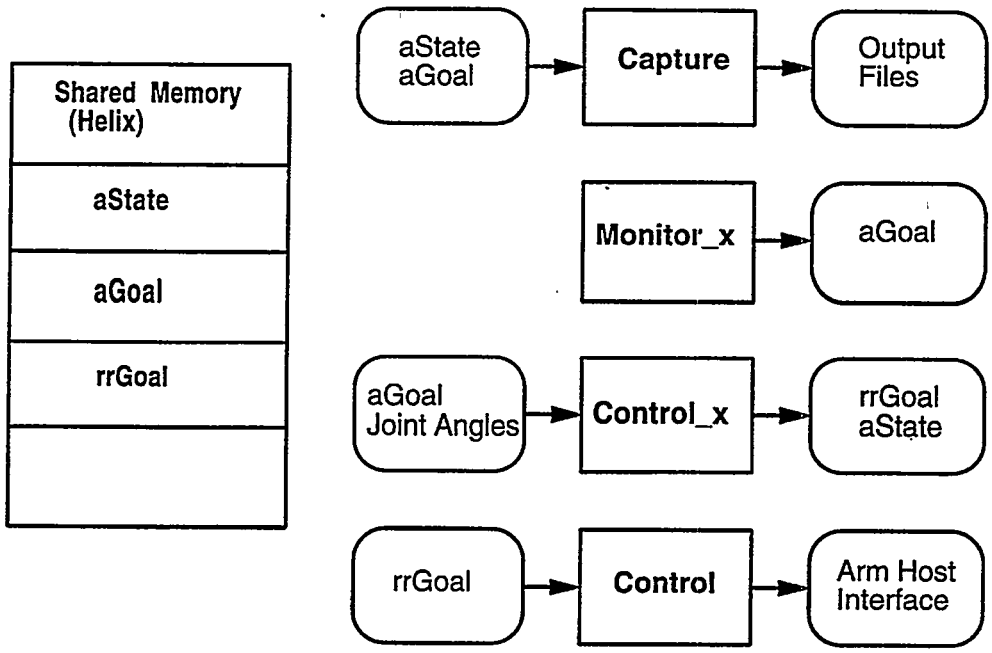

Fig. 4. The architecture of the software system for the manipulator.

The controller for the Robotics Research manipulator has a host interface with a loop time of 36 ms (28 Hz) and a servo interface with a 6ms loop time (167 Hz). The arm controller communicates with joint mounted servo hardware that runs four nested servo loops: position (512 Hz), velocity (1024 Hz), torque (1024 Hz), and current (10,240 Hz). Our software runs on a CPU in a VME rack and communicates with the host interface to the arm controller using a bus to bus connector. We cannot communicate directly with the joint mounted servo hardware.

The Control module reads the joint goal data structure from shared memory and sends an appropriate command to the host interface. The host interface will accept velocity or position commands for both joint angles and Cartesian coordinates. For our experiments, we send joint velocity commands when we are far from the goal and send joint position commands when we are near the goal.

An experiment consists of a sequence of Cartesian goals for predetermined times that are written to shared memory at the appropriate times by the Monitor_x module. The Control_x module reads the Cartesian goal from shared memory and reads the joint angles from the host interface. The Control_x module performs the calculations described in the previous section and writes the joint goal and the current state of the system to shared memory.

The Capture module reads shared memory (at 20 Hz) and writes the state and goal data to output files. The graphs in this section were prepared using the output files produced by Capture.

The experiments described in this section are produced by processes that operate at three distinct frequencies. The high level goals are changed slowly (the durations of the Cartesian goals range from 0.83 seconds to 4.33 seconds). The host interface operates at 28 Hz. The servo loops for the arm controller run at high rates (512 Hz to 10,240Hz). Our experiments demonstrate that our control system can reach a goal. However, we are not tracking a path. During an experiment, the Cartesian positions can have significant errors. We could reduce the errors by giving the system high level goals at a higher frequency using the servo interface.

We will discuss the results of four experiments. The first experiment is a loop in Cartesian position: increase x_5, increase x_6, decrease x_5, and decrease x_6. The second experiment is a loop in Cartesian orientation: increase x_1, increase x_2, decrease x_1, and decrease x_2. The third experiment is a cycle in the self-motion coordinate: increase p from 0 to 1, decrease p to -1, and increase p to 0. The final experiment is a loop in Cartesian position (x_6) and self-motion coordinate.

The first experiment is a loop in Cartesian position. As shown in Table 1, x_5 increases from 0.6514 m to 0.7500 m before returning to 0.6514 m. Simultaneously, x_6 increases from -0.1026 to 0.0 m before returning to -0.1026. Time series values of x_5 and x_6 are plotted in Figs. 5 and 6. The perturbations in x_5 near 1.0 seconds and near 3.5 seconds are caused by the large changes in x_6 at those times. Comparing Table 1 and Figs. 5 and 6, the time values in Table 1 correspond to periods when the position variables are constant.

Table 1. Values of the Cartesian position for the first experiment.

	Time	x_5	x_6
	seconds	meters	meters
1	0.03	0.6514	-0.1026
2	0.73	0.7500	-0.1026
3	1.60	0.7500	0.0000
4	2.63	0.6514	0.0000
5	3.70	0.6514	-0.1026

The second experiment is a loop in Cartesian orientation. As shown in Table 2, both x_1 and x_2 increase from 0.0 radians to 0.5 radians before returning to 0.0 rad. Time series values of x_1 and x_2 are plotted in Figs. 7 and 8. The perturbations in x_1 near 2.0 seconds and near 5 seconds are caused by the large changes in x_2 at those times. Comparing Table 2 and Figs. 7 and 8, the time values in Table 2 correspond to periods when the position variables are constant.

Table 2. Values of the Cartesian orientation for the second experiment.

	Time	x_1	x_2
	seconds	radians	radians
1	0.05	0.0000	0.0000
2	1.65	0.4998	0.0002
3	3.15	0.5001	0.4999
4	4.55	0.0000	0.5000

The third experiment is a cycle in the self-motion coordinate (p). The changes in self-motion coordinate (p), the cosine of the angle between the vectors **B** and **C** (cos μ), and local coordinates (q) are displayed in Table 3 for the third experiment. We will examine plots of each of these variables. Time series values for p are plotted in Fig. 9. The value of p increases from 0 to 1, decreases to -1, and increases to 0. At each step on the control algorithm, we can estimate the time required to get to the goal without exceeding the speed limits on the joint velocities (Goal Time). The values of Goal Time are plotted in Fig. 10 for the third experiment. When a new goal is provided to the control system, the value of Goal Time increases rapidly to a peak and decreases as the arm approaches the goal. At the end of the big swing from p = 1.0 to p = -1.0, the value of p overshoots the goal. When the overshoot occurs (at 4.35 seconds), the Goal Time jumps from 0.05 seconds to 0.12 seconds.

Table 3. Values of the self-motion coordinate (p), cos μ, and local coordinates (q) for the third experiment.

	Time	p	cos μ	q
	seconds	radians		radians
1	0.05	0.0000	1.0000	0.0000
2	2.05	1.0000	0.7699	1.0778
3	5.55	-1.0000	0.7699	-1.0228
4	7.95	0.0000	1.0000	-0.0006

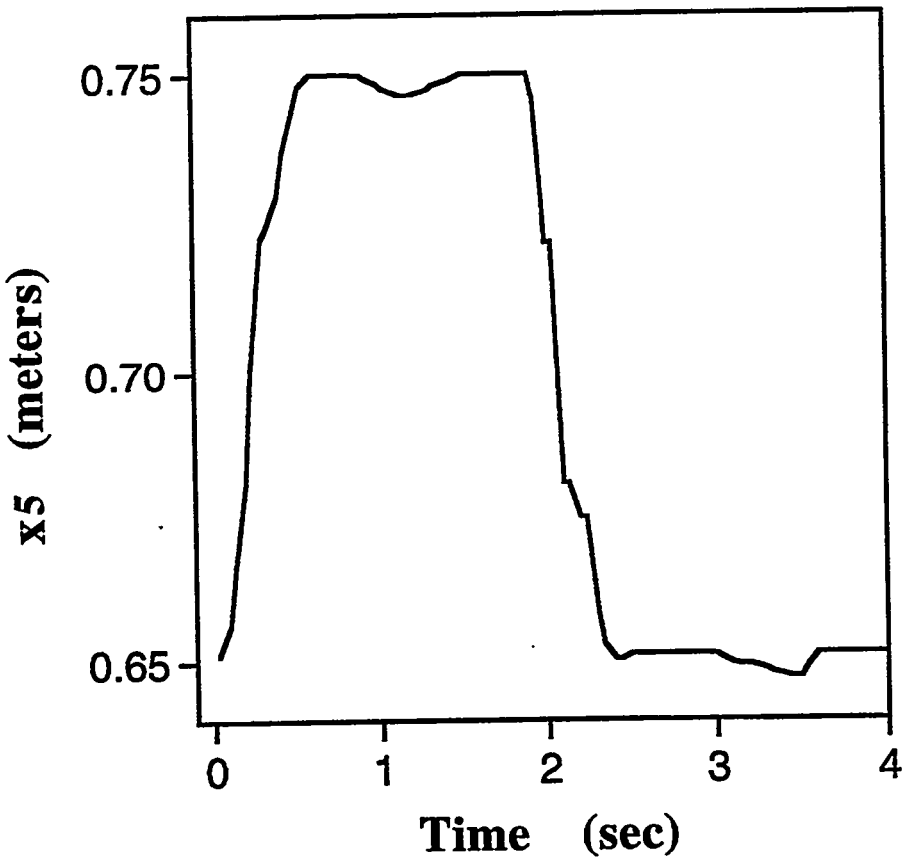

Fig. 5. Changes in x_5 during the first experiment.

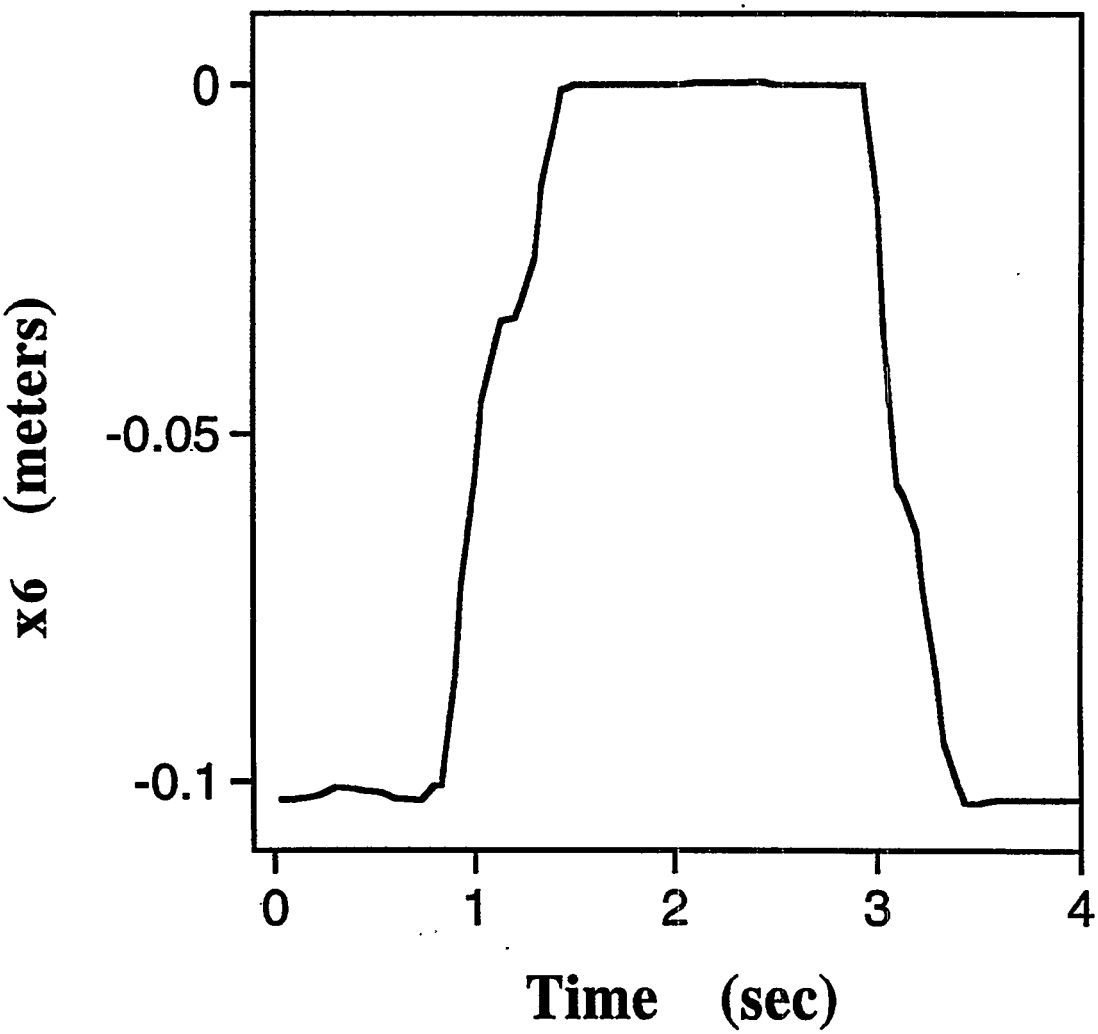

Fig. 6. Changes in x_6 during the first experiment.

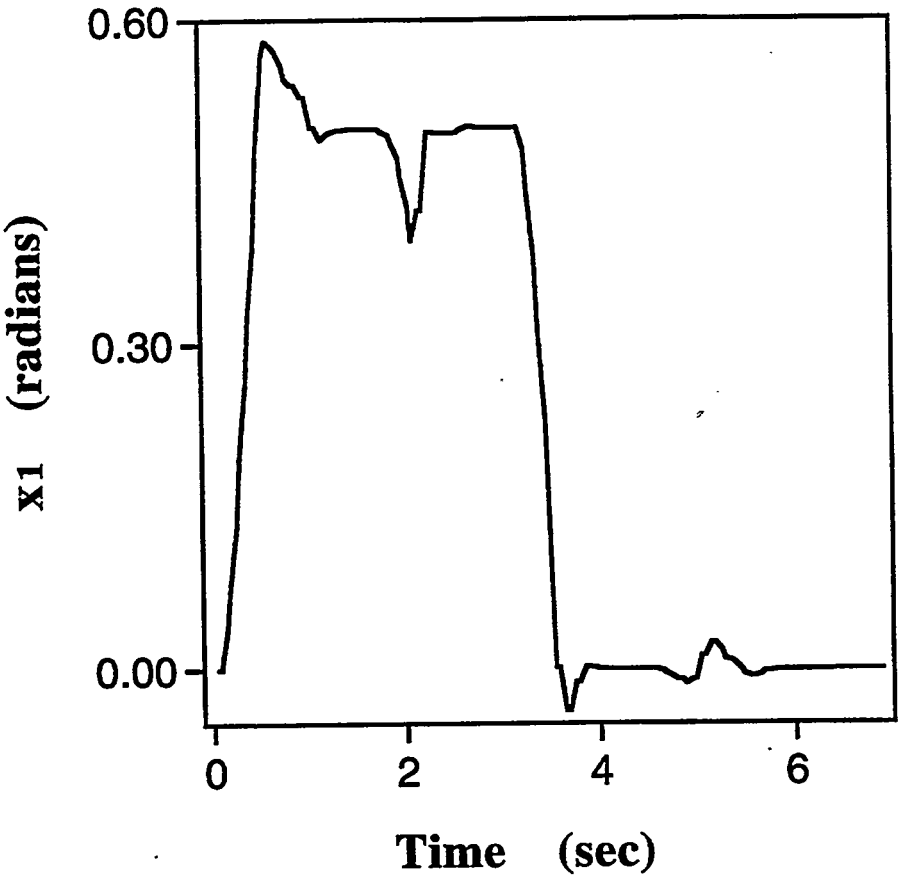

Fig. 7. Changes in x_1 during the second experiment.

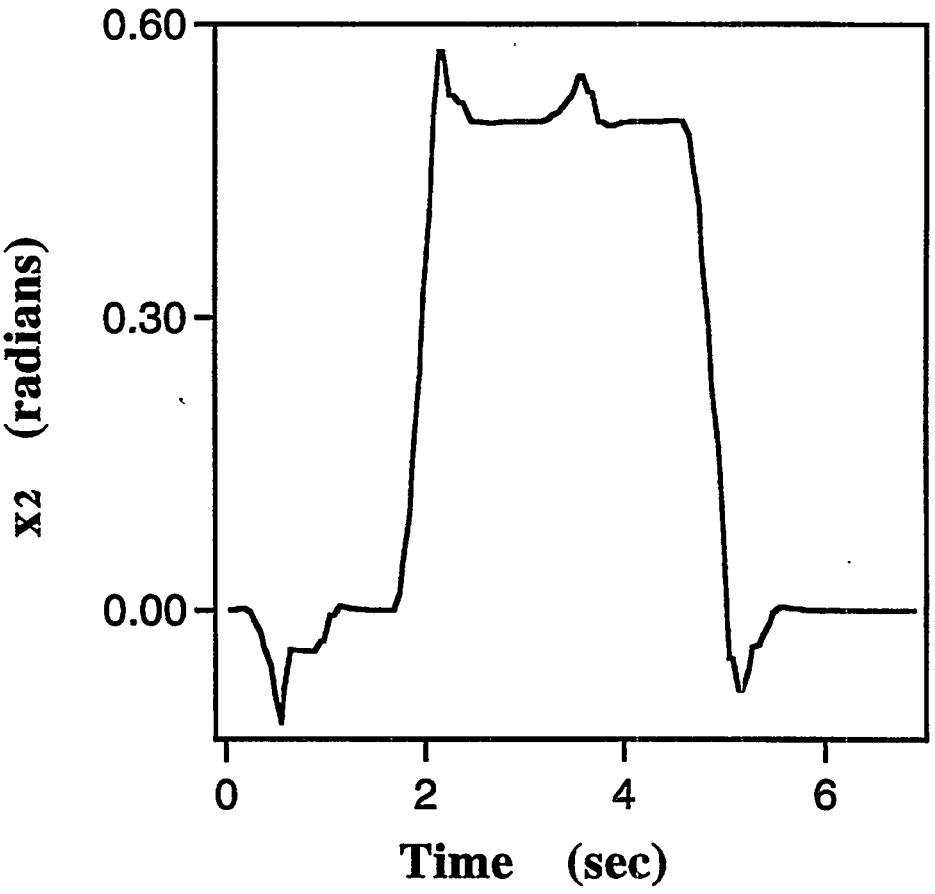

Fig. 8. Changes in x_2 during the second experiment.

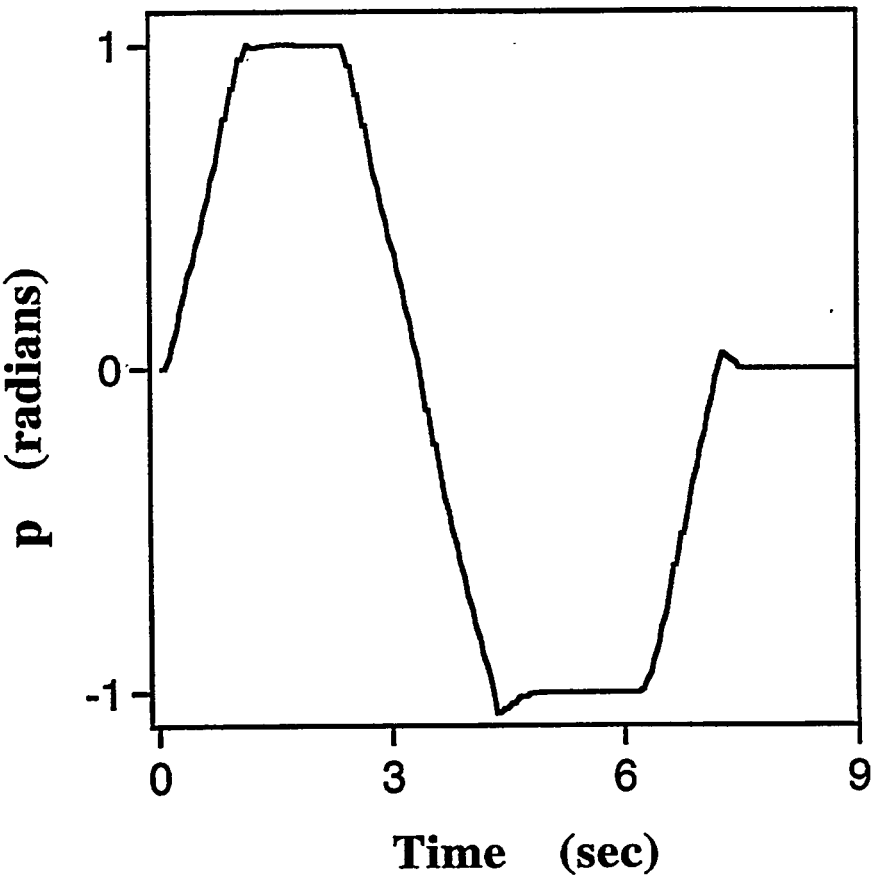

Fig. 9. Changes in p during the third experiment.

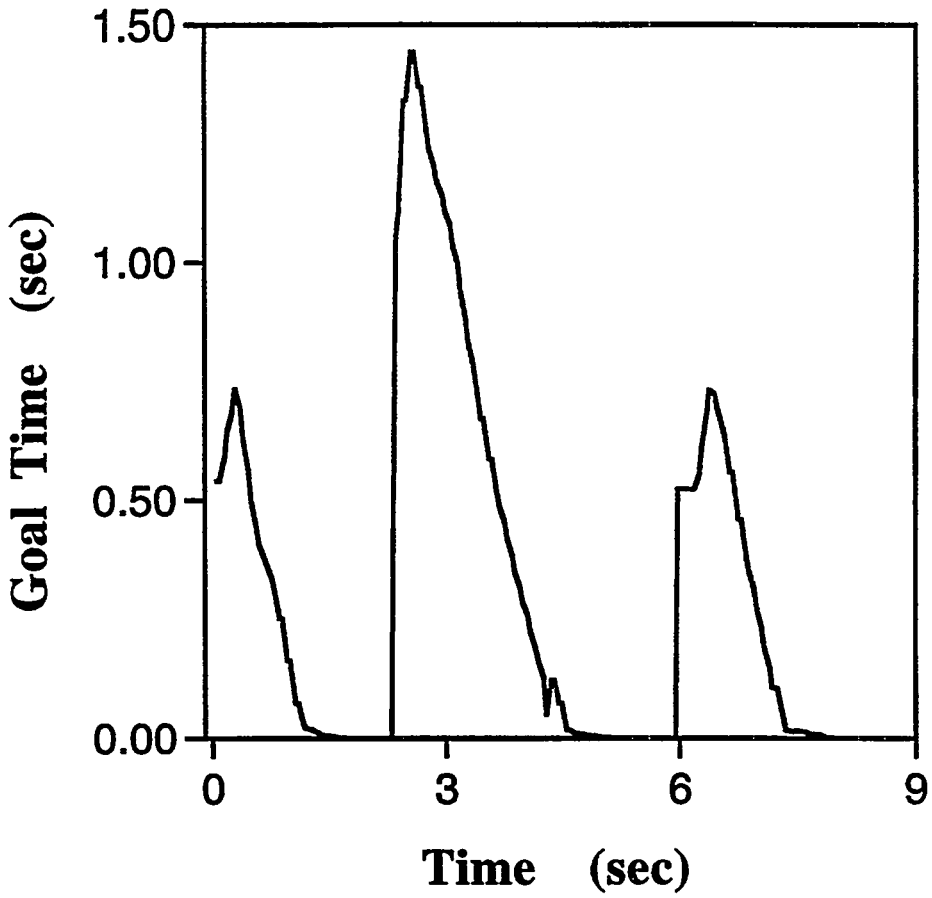

Fig. 10. Changes in Goal Time during the third experiment.

In Table 3, the changes in the local coordinates (q) are a little bit larger than the changes in the self-motion coordinate (p). The difference between q and p is displayed in Fig. 11. We are surprised that the curve is not more symmetrical (we expect positive values when p is positive to be mirrored by negative values when p is negative). In Table 3, the difference when p = 1.0 (0.0778) is more than three times as large as the difference when p = -1.0 (-0.0228). At the end of the experiment, p, q, and the joint coordinates have returned to their original positions.

For the general case, we require that the matrix $1/2(\mathbf{C}^T\mathbf{B} + \mathbf{B}^T\mathbf{C})$ be positive definite. For our experiments, k = 7 - 6 = 1, **B** and **C** are vectors with unit length, and $\mathbf{C}^T\mathbf{B}$ is a scalar. Thus, $\mathbf{C}^T\mathbf{B}$ is the cosine of the angle (μ) between **B** and **C**. To keep the coordinates single valued and invertible, we require that the magnitude of μ be less than 90 degrees. Thus, cos μ should be positive. If cos μ approaches zero, we choose a new value for **C**: $\mathbf{C} = \mathbf{B}(t^*)$, where $\mathbf{B}(t^*)$ is the current **B** vector when we chose a new value for **C**. Values of cos μ are plotted in Fig. 12 for the third experiment. All of the values of cos μ are positive during the experiment. In Table 3, we find that cos μ = 0.7699 when p = $\pm$1.0. Thus, the angle between **B** and **C** is μ = 0.69 radians (40 degrees) when p = $\pm$1.0.

During the third experiment, four of the seven joint angles have substantial changes in their values (see Table 4). The values in Table 4 demonstrate that the joint angles return to their initial values.

Table 4. Values of the joint angles for the third experiment.

	Time	θ_1	θ_3	θ_5	θ_6
	seconds	radians	radians	radians	radians
1	0.05	0.0000	0.0000	0.0000	-0.7854
2	2.05	0.8624	0.4403	-0.8584	-1.3055
3	5.55	-0.8624	-0.4404	0.8583	-1.3052
4	7.95	0.0004	-0.0008	-0.0028	-0.7853

Each of the joints has a speed limit. The targets for the joint velocities are required to be less than the speed limits. Targets and measured values for the velocity of joint one are displayed in Fig. 13. The speed limit for joint one is 1.2 radians per second (70 degrees per second). During periods when the target is at the speed limit, the measured values alternate between a value that is slightly above the speed limit (1.3 radians per second) and zero. In the period from 2.72 seconds to 4.52 seconds, there were 27 measurements of 1.3 r/s and 10 measurements of zero. No other values were measured!

We apply Singular Value Decomposition (SVD) to calculate a basis for the null space of **J**. Since we have added a row of zeros to the matrix, one of the singular values will be zero. If **J** is rank deficient, a second singular value will be zero. We monitor the second smallest singular value (ε) to detect when **J** is becoming rank deficient. Values of ε are plotted in Fig. 14 for the third experiment. The singular value varies from 0.35 to 0.39. Thus, it does not approach zero.

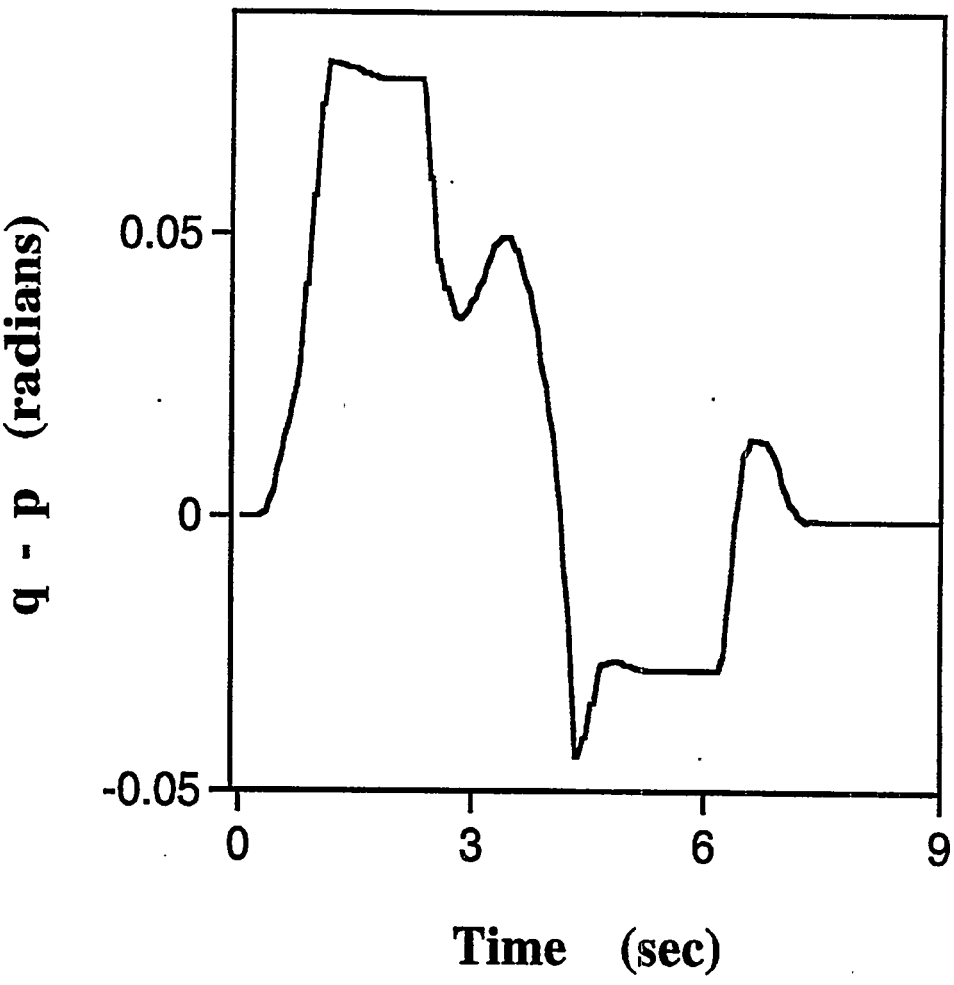

Fig. 11. Changes in q - p during the third experiment.

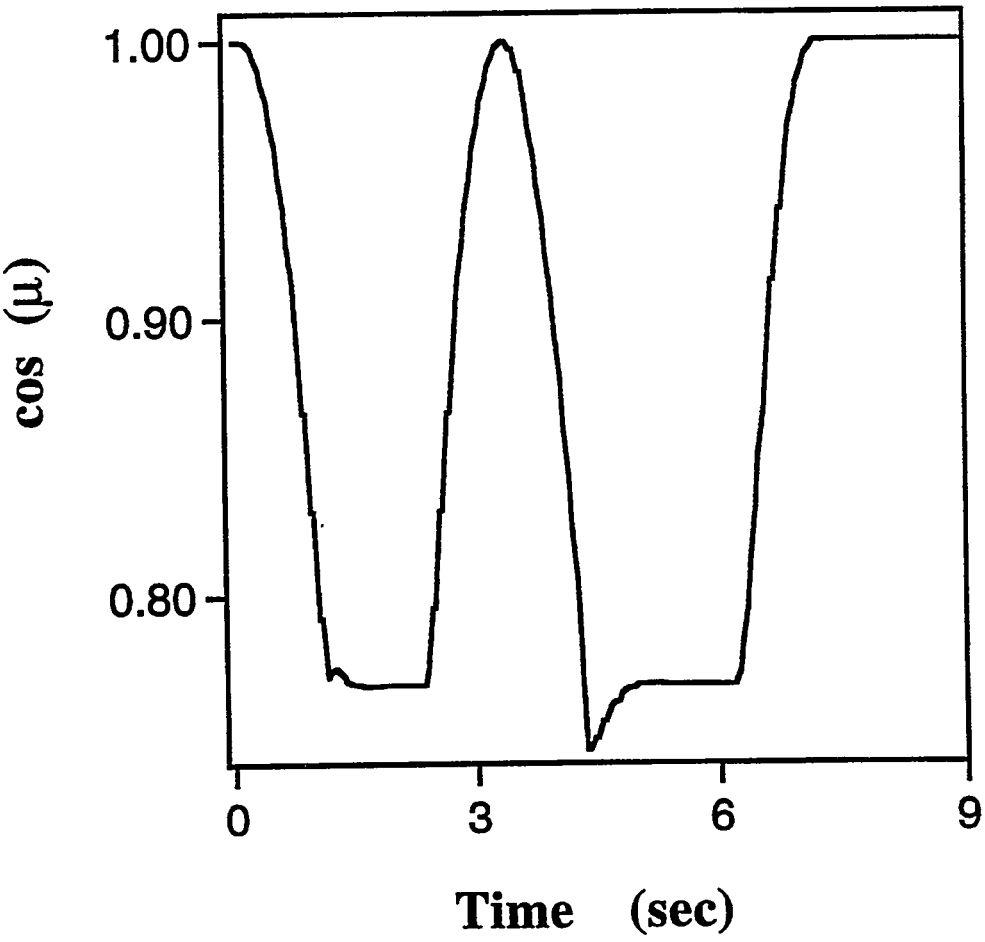

Fig. 12. Changes in cos μ during the third experiment.

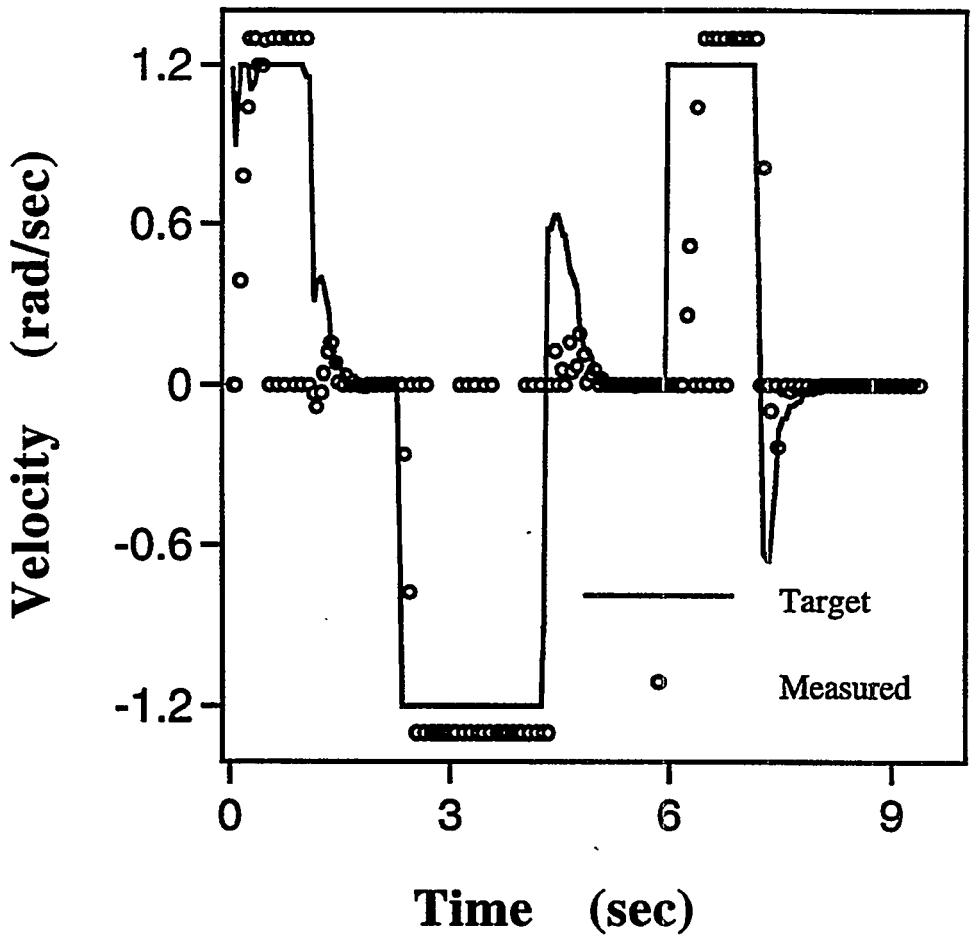

Fig. 13. Velocity of the first joint during the third experiment.

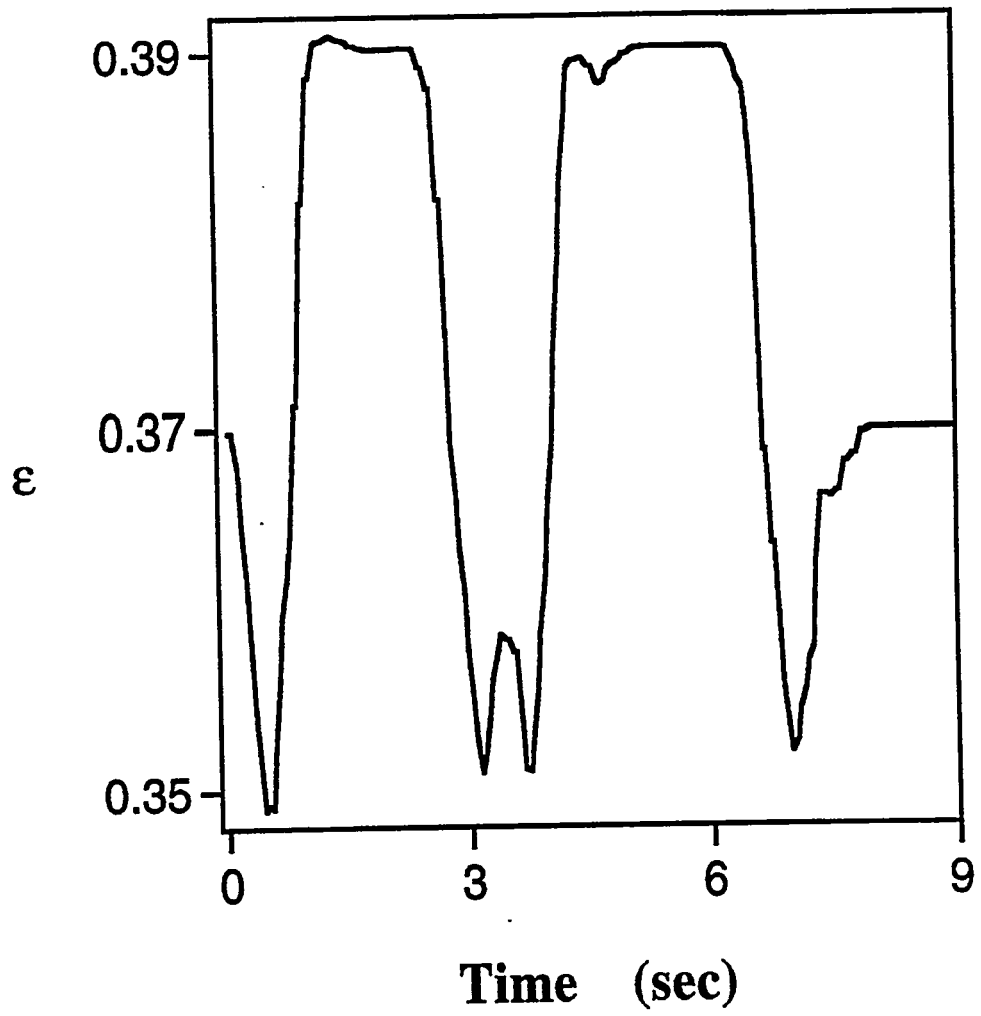

Fig. 14. Changes in the second smallest singular value (ε) during the third experiment.

The final experiment is a loop in Cartesian position (x_6) and self-motion coordinate. The changes in self-motion coordinate (p), the Cartesian position (x_6), and local coordinates (q) are displayed in Table 5 for the fourth experiment. Time series values for p are plotted in Fig. 15. The value of p increases from 0 to 1 and then decreases to 0. Time series values for x_6 are plotted in Fig. 16. The value of x_6 increases from -0.1 m to 0.0 and then decreases to -0.1 m. Small changes in p occur during large changes in x_6 and vice versa.

The difference between q and p is displayed in Fig. 17. As p increases from 0 to 1, q increases from 0.0 to 1.08 radians (the change is the same as for the third experiment). However, the increase in x_6 requires an increase in q from 1.08 radians to 1.14 radians. As p decreases from 1 to 0, q decreases from 1.14 radians to -0.09 radians. During the final decrease in x_6 there is no change in q. At the end of the experiment, p, x_6, and the joint coordinates have returned to their original positions. However, the local coordinate (q) does not return to zero. In the third experiment, the local coordinate returned to zero. In the third experiment the position and orientation (**x**) was constant, while the position changed in the fourth experiment. In Table 5, a 1.0 radian change in p requires a 1.08 radian change in q at the lower value of x_6 and a 1.23 radian change in q at the upper value of x_6.

Table 5. Values of the self-motion coordinate (p), Cartesian position (x_6), and local coordinates (q) for the fourth experiment.

	Time	p	x_6	q
	seconds	radians	meters	radians
1	0.05	0.0000	-0.1026	0.0000
2	1.85	1.0001	-0.1026	1.0772
3	4.05	1.0004	0.0000	1.1395
4	6.65	0.0002	0.0000	-0.0885
5	7.55	0.0000	-0.1026	-0.0887

During the fourth experiment, four of the seven joint angles have substantial changes in their values (see Table 6). The values in Table 6 demonstrate that the joint angles return to their initial values.

Table 6. Values of the joint angles for the fourth experiment.

	Time	θ_1	θ_3	θ_5	θ_6
	seconds	radians	radians	radians	radians
1	0.05	0.0000	0.0000	0.0000	-0.7854
2	1.85	0.8625	0.4403	-0.8586	-1.3056
3	4.05	0.9689	0.3341	-1.0410	-1.2680
4	6.65	0.0002	0.0000	-0.0006	-0.5953
5	7.55	0.0000	0.0000	0.0000	-0.7854

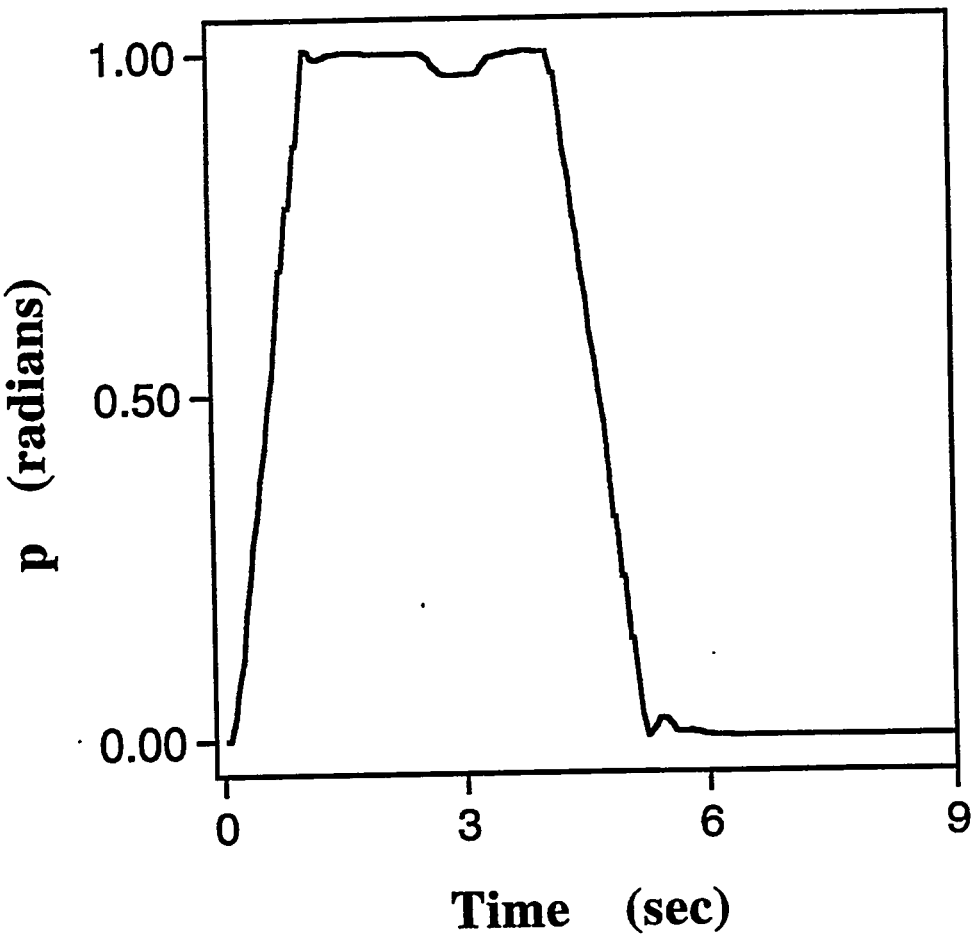

Fig. 15. Changes in p during the fourth experiment.

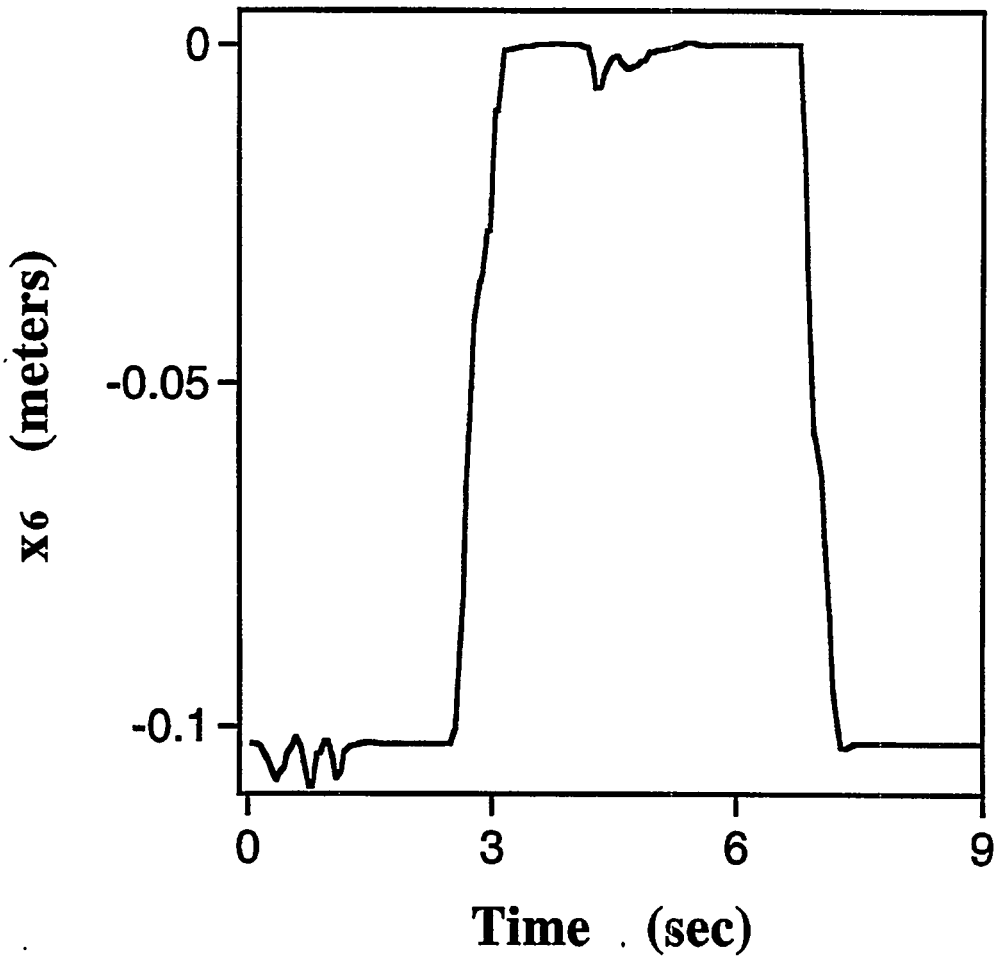

Fig. 16. Changes in x_6 during the fourth experiment.

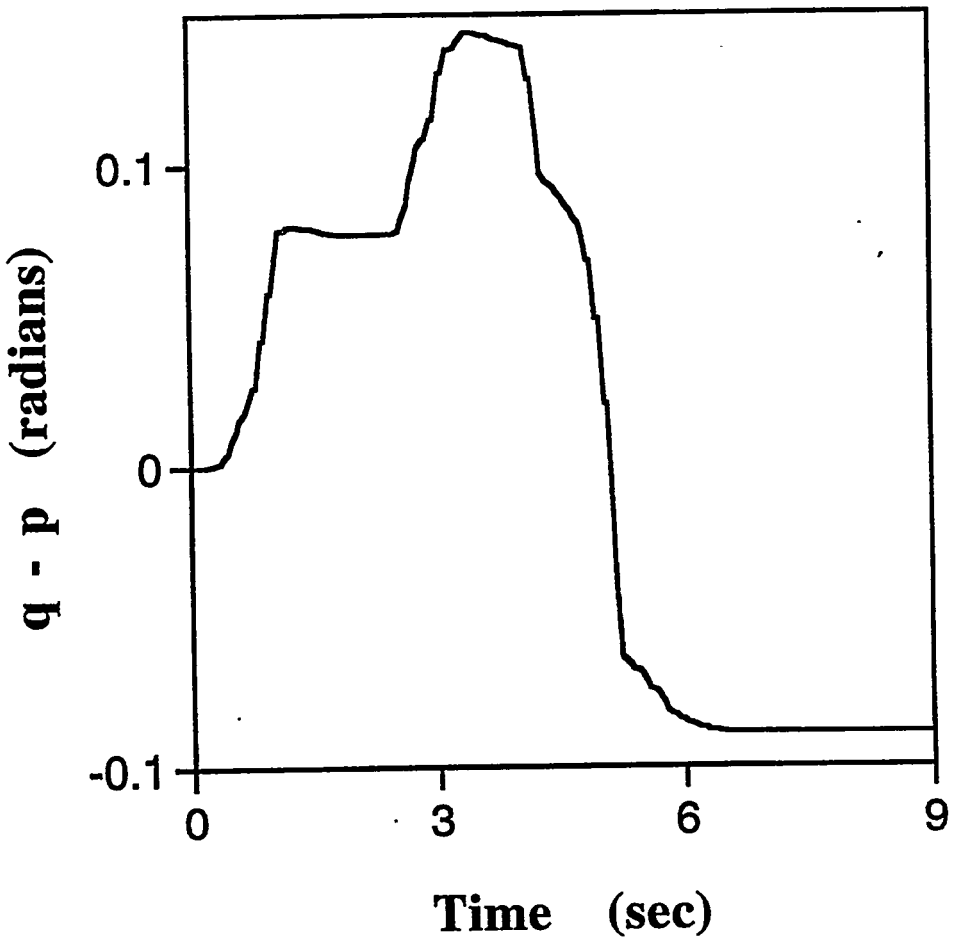

Fig. 17. Changes in q - p during the fourth experiment.

Seraji's configuration control method for redundancy resolution adds an additional task that will be performed by the manipulator. One example of a task is control of the elbow angle (ψ), the angle between the plane containing the shoulder, elbow, and hand of the arm and a reference frame containing the line that joins the shoulder to the hand [see Seraji (1991)]. We conclude this section by comparing the elbow angle with both the self-motion coordinate and the angle between **B** and **C** for several experiments.

Values of the three angles for the third experiment are displayed in Table 7. Although the magnitudes of the three angles are different, the angles move together (all are at their maximum at 2.05 seconds and at their minimum at 5.55 seconds.

Table 7. Values of the self-motion coordinate (p), the angle (μ) between **B** and **C**, and the elbow angle (ψ) for the third experiment.

	Time	p	μ	ψ
	seconds	radians	radians	radians
1	0.05	0.0000	0.0000	0.0000
2	2.05	1.0000	0.6921	0.8332
3	5.55	-1.0000	-0.6921	-0.8330
4	7.95	0.0000	0.0000	0.0000

The position and orientation of the hand are constant for the third experiment, while the position of the hand changes during the fourth and fifth experiments. Values of the three angles for the fourth experiment are displayed in Table 8. The first two rows of Table 8 have the same values as the first two rows of Table 7. In the third row of Table 8, x_6 increases while p is constant and both μ and ψ increase. In the fourth row of Table 8, p decreases to zero while x_6 is constant, ψ decreases to zero, but μ only decreases to 0.25 radians.

Table 8. Values of the self-motion coordinate (p), the angle (μ) between **B** and **C**, and the elbow angle (ψ) for the fourth experiment.

	Time	p	μ	ψ
	seconds	radians	radians	radians
1	0.05	0.0000	0.0000	0.0000
2	1.85	1.0001	0.6923	0.8331
3	4.05	1.0004	0.7513	0.8922
4	6.65	0.0002	0.2472	0.0002
5	7.55	0.0000	0.0000	0.0000

Values of the three angles and the Cartesian position for the fifth experiment are displayed in Table 9. The first two rows of Table 9 have the same values as the first two rows of Tables 7 and 8. In the third row of Table 9, x_6 increases by three times the amount as in the fourth experiment while p is constant and both μ and ψ increase by more than in Table 8. In the fourth row of Table 9, p decreases to 0.02 radians while x_6 is constant, ψ decreases to 0.02 radians, but μ increases slightly to 1.00 radians. Table 9 demonstrates that the values of the three angles can have independent changes.

Table 9. Values of the self-motion coordinate (p), Cartesian position (x_6), the angle (μ) between **B** and **C**, and the elbow angle (ψ) for the fifth experiment.

	Time	p	x_6	μ	ψ
	seconds	radians	meters	radians	radians
1	0.07	0.0000	-0.1026	0.0000	0.0000
2	2.30	1.0000	-0.1026	0.6921	0.8330
3	5.80	0.9999	0.2000	0.9350	1.0736
4	10.13	0.0244	0.2000	1.0042	0.0220

4. CONCLUSIONS

There is a large literature on the control of redundant manipulators. Whitney solved the redundancy resolution problem by using the Moore-Penrose generalized inverse. Klein and Huang discovered that Whitney's solution did not yield repeatable paths in joint space. Shamir and Yomdin published a necessary and sufficient condition for repeatable paths in joint space. Baillieul developed the extended Jacobian technique for redundancy resolution. He maximizes an objective function at every point on the specified path in Cartesian space. If the extended Jacobian is nonsingular, the method will yield repeatable paths in joint space. Seraji a developed the configuration control method for redundancy resolution. His method is not based on local optimization. He adds an additional task that will be performed by the manipulator. The kinematic Jacobian can be combined with the task Jacobian to define an extended Jacobian. If the extended Jacobian is nonsingular, the joint velocities can be determined. The extended Jacobian will be singular if either the kinematic Jacobian or the task Jacobian are not of full rank.

At every point on the specified path in Cartesian space, the arm is constrained to move on the self-motion manifold. Burdick has discussed the features of the self-motion manifolds. The inverse kinematic solution is the union of disjoint manifolds. A general path planning algorithm for a redundant manipulator should plan transitions from one of the self-motion manifolds to another. Burdick has defined two types of redundancy resolution methods: configuration resolution and path-wise resolution. A configuration resolution method finds a point in joint space that will keep the end of the manipulator at a given point in Cartesian space and maximize a scalar objective function. A path-wise method finds a trajectory in joint space that will keep the end of the manipulator on a given path in Cartesian space and maximize a scalar objective function that is computed over the entire path. Our long term objective is to develop a global method for path-wise resolution.

In this paper, we have defined and demonstrated a self-motion coordinate system. For the extended Jacobian methods of Baillieul and Seraji, either the objective function or additional task is specified by a predetermined function of the joint variables. There are three problems with this approach: how to choose a function, the function may be difficult to calculate, and the extended Jacobian can become singular. Calculation of an obstacle avoidance function for a realistic problem may be difficult. As the arm moves in the workspace, the point on the arm that is closest to an obstacle can have discontinuous motions. The Jacobian may not be defined at the discontinuties. Adding or moving an obstacle will require the calculation of a new function. Our method does not require a predetermined function; we numerically determine a basis for the null space of the kinematic Jacobian. Our method cannot have kinematic singularities or algorithmic singularities.

We have developed and demonstrated a control system that allows us to move the manipulator to a position specified by both Cartesian position and orientation and self-motion coordinates. When we do not have null space motion, our solution is the Moore-Penrose generalized inverse of **J** and Klein and Huang demonstrated that the Moore-Penrose solution did not yield repeatable paths in joint space. In our experimental results, we find that closed cycle paths in **x** and **p** space result in final values of the joint angles that are equal to the initial values however the final values of **q** may not equal to the initial values. We have demonstrated that our self-motion coordinates are not the same as Seraji's elbow angle.

5. REFERENCES

J. Baillieul, 1985. "Kinematic programming alternatives for redundant manipulators", Proc. IEEE International Conference on Robotics and Automation, pp. 722-728.

J. Baillieul and D. P. Martin, 1990. "Resolution of kinematic redundancy", Proc. of Symposia in Applied Mathematics, Vol. 41, pp. 49-90.

R. W. Brockett, 1984. "Robotic manipulators and the product of exponentials formula", Proc. of the International Symposium on Mathematical Theory of Networks and Systems, Springer-Verlag, Berlin, pp. 120-129.

J. W. Burdick, 1989. "On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds", Proc. IEEE International Conference on Robotics and Automation, pp. 264-270.

J. W. Burdick, B. C. Cetin, and J. Barhen, 1991. "Efficient global redundant configuration resolution via sub-energy tunneling and terminal repelling", Proc. IEEE International Conference on Robotics and Automation, pp. 939-944.

R. Colbaugh, H. Seraji, and K. L. Glass, 1989. "Obstacle avoidance for redundant robots using configuration control", Journal of Robotic Systems, Vol. 6, pp. 721-744.

J. J. Craig (1986). Introduction to robotics: Mechanics and Control, Addison-Wesley, Reading, Massachusetts.

H. Goldstein (1980). Classical Mechanics, Addison-Wesley, Reading, Massachusetts.

J. P. Jones, P. L. Butler, S. E. Johnson, and T. G. Heywood (1992). Hetro Helix: Synchronous and asynchronous control systems in heterogeneous distributed networks, Robotics and Autonomous Systems, Vol. 10, pp. 85-99.

C. A. Klein and C. Huang, (1983). "Review of pseudoinverse control for use with kinematically redundant manipulators", IEEE Trans. on Systems, Man, and Cybernetics, Vol. 13, pp. 245-250.

J. M. McCarthy (1990). Introduction to Theoretical Kinematics, The MIT Press, Cambridge, Massachusetts.

C. Mladenova (1990). "A contribution to the modeling and control of manipulators", Journal of Intelligent and Robotic Systems, Vol. 3, 349-363.

R. M. Murray, Z. Li, and S. S. Sastry, (1994). A mathematical introduction to robotic manipulation, CRC Press, Boca Raton, FL.

F. C. Park, J. E. Bobrow, and S. R. Ploen (1995). A Lie group formulation of robot dynamics, The International Journal of Robotics Research, Vol. 14, pp. 609-618.

R. P. Paul (1981). Robot manipulators: Mathematics, programming, and control, The MIT Press, Cambridge, Massachusetts.

H. Seraji, 1989. "Configuration control of redundant manipulators: theory and implementation", IEEE Trans. on Robotics and Automation, Vol. 5, pp. 472-490.

H. Seraji and R. Colbaugh, 1990. "Improved configuration control for redundant robots", Journal of Robotic Systems, Vol. 7, pp. 897-928.

H. Seraji, M. Long, and T. Lee, 1991. "Configuration control of 7 DOF arms", Proc. IEEE International Conference on Robotics and Automation, pp. 1195-1200.

T. Shamir and Y. Yomdin, 1988. "Repeatability of redundant manipulators: mathematical solution of the problem", IEEE Trans. on Automatic Control, Vol. 33, pp. 1004-1009.

D. E. Whitney, 1969. "Resolved motion rate control of manipulators and human prostheses", IEEE Trans. on Man-Machine Systems, MMS-10, pp. 47-53.

This page left blank intentionally

INTERNAL DISTRIBUTION

1-5.	J. Barhen
6.	C. W. Glover
7.	W. C. Grimmell
8.	H. E. Knee
9.	E. M. Oblow
10.	C. E. Oliver
11-15.	L. E. Parker
16.	N. Peterfreund
17.	F. G. Pin
18.	V. Protopopescu
19.	N. S. Rao
20-24.	D. B. Reister
25-29.	R. F. Sincovec
30.	M. S. Smith
31.	E. C. Uberbacher
32.	M. A. Unseren
33-37.	CSMD Reports Office
38.	Laboratory Records-RC
39.	Document Reference Section
40.	Central Research Library
41.	ORNL Patent Office

EXTERNAL DISTRIBUTION

42-43. Office of Scientific and Technical Information, P. O. Box 62, Oak Ridge, TN 37831.

44. Dr. Fred Aminzadeh, UNOCAL, 14141 SW Freeway, Suite 301-225, Sugarland, TX 77478.

45. Dr. John Baillieul, Aerospace & Mechanical Engineering Department, Boston University, 110 Cummington Street, Boston, MA 02215.

46. Dr. John Blair, JBX Technologies, 25 Moore Road, Wayland, MA 01778.

47. Dr. Roger W. Brockett, Harvard University, Pierce Hall, 29 Oxford Street, Cambridge, MA 02138.

48. Dr. Oscar P. Manley, 13212 Wye Oak Drive, Gaithersburg, MD 20878.

49. Dr. Robert E. Price, ER-15, Office of Basic Energy Sciences, Department of Energy, Washington, DC 20585.

50. Dr. Charles R. Weisbin, Robotics and Mars Exploration Technology Program, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109.