Automatic Generation of Kinematics for a Reconfigurable Modular Manipulator System

Laura Kelmar[1] and Pradeep K. Khosla[2]

Abstract

Previous research on the kinematics of robotic manipulators has addressed fixed configuration manipulator systems. In this paper, we present a method of automatically generating the kinematics of a new class of manipulators called Reconfigurable Manipulators. Reconfigurable Manipulators are designed and built, from a system consisting of links and joints of various sizes, as appropriate for particular task. Generation of the kinematic equations that govern a modular manipulator starts with geometric descriptions of the units, or modules, as well as their sequence in the manipulator. We propose an algorithm that automatically generates the Denavit-Hartenberg (DH) kinematic parameters of a reconfigurable manipulator. The DH kinematic parameters are then used to obtain the forward kinematic transformation of the system. We also address the problem of obtaining the inverse kinematics of reconfigurable manipulators. In order to automate the inverse kinematics and to make the procedure as general as possible, we use a numerical approach. In the case of a redundant manipulator, we exploit the extra degrees of freedom to achieve singularity avoidance. We have implemented our algorithms on the prototype Reconfigurable Modular Manipulator System (RMMS) being developed in our laboratory.

1 Introduction

Traditionally, robot manipulators have been developed for specific applications. While this is practical and sufficient for industrial applications where the task can be well defined and constrained, manipulators are also needed in unpredictable and changing environments (such as those at a space station). In theory, robot manipulators are flexible and can be reprogrammed for new tasks. However, each robot's configuration makes it capable of only a limited number of applications. For example, a manipulator well-suited to precise movement across the top of a table, would not be capable of lifting heavy objects in the vertical direction. A solution to this problem is to have a set of joints of various performance characteristics and a set of links of various lengths which can be assembled into a configuration appropriate for each task. This set of joints and links of various specifications is called the Reconfigurable Modular Manipulator System (RMMS) [18].

The first step in configuring a modular manipulator for a specific task is to choose the desired components from the set of joint and link modules of RMMS. The next step is to obtain the forward kinematics to define the position and orientation of the end-effector for a given set of joint angles. After this has been done, an inverse kinematic solution must be found to determine the joint angles necessary to achieve a desired cartesian position and orientation of the end-effector. While the derivation of the kinematic equations of manipulators has been researched in depth for *fixed configuration* manipulators [15, 17], work has not been done for *reconfigurable* manipulators where the creation of the kinematic equations must be done without a priori knowledge of the number and arrangement of joint and link modules that compose the manipulator.

This paper addresses the problem of automatically generating the kinematics of the RMMS. Other researchers [9, 19, 11] have started with a kinematic model, in the form of DH parameters [4], and generated symbolic equations for the forward kinematics. However, our approach is fundamental and addresses the problem of automatically generating the DH parameters [7]. Starting with a geometric description of the modules used, as well as their sequence in the manipulator, we generate the DH parameters to completely define the geometry of the manipulator.

Using the DH parameters, we create the system of kinematic equations which must be solved to find the joint angles necessary to achieve a desired position and orientation of the end-effector of the manipulator. This is the inverse kinematics problem. In this paper we present a method to numerically solve the inverse kinematics for non-redundant, as well as redundant, manipulators. Our approach for kinematics of redundant manipulators is based upon the singularity robust inverse [12]. It exploits the redundancy to achieve singularity avoidance with minimal additional computation. We propose a configuration independent method for choosing the design parameters necessary for the singularity robust method.

This paper is organized as follows: In Section 2, we briefly describe the DH system for representing the kinematics of manipulators. Our approach to obtaining the kinematics of reconfigurable manipulators is described in Section 3. In Section 4, we outline a numerical method for obtaining the inverse kinematics of a modular manipulator and present a method for expanding the inverse kinematics algorithm to solve for the joint variables of redundant manipulators. Finally, we summarize our results in Section 5.

2 Forward Kinematics of Fixed Configuration Manipulators

The forward kinematic transformation for fixed configuration manipulators are typically found by assigning DH link frames [4, 14]. While many such methods exist [5, 17], we chose the DH system which allows for easy derivation of the manipulator Jacobian [14].

DH parameters represent a systematic way for the designer to define the position and orientation of adjacent links in a manipulator. The coordinate frames are aligned so that the z-axis for link i is the axis of rotation of the i^{th} DH frame. The following series of transformations describes the position and orientation of the i^{th} link in the i-1^{th} frame:

$$\text{Rotation}(z_{i-1},\theta_i)\ \text{Translation}(a_i,0,d_i)\ \text{Rotation}(x_i,\alpha_i).$$

[1]Graduate Student, Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

[2]Assistant Professor, Department of Electrical and Computer Engineering,Carnegie Mellon University, Pittsburgh, PA 15213

The variables θ_i, a_i, d_i, and α_i are called the DH parameters of the system [14]. The homogeneous transformation matrix created from the rotations and translations defined by the DH parameters is called an **A** matrix of the system. By multiplying successive **A** matrices, $\mathbf{A}_1 \cdot \mathbf{A}_2 \cdot \ldots \cdot \mathbf{A}_N$, where N is the number of degrees of freedom of the manipulator, one obtains the $\mathbf{T}_N$ matrix. The $\mathbf{T}_N$ matrix represents the position and orientation of the end-effector, with respect to the base frame [14], as a function of the joint variables (θ_i, i = 1, . . . ,N).

3 Forward Kinematics of Reconfigurable Manipulators

A RMMS consists of a collection of modules, joints and links, which can be combined as desired to create a manipulator. Given a manipulator, a complete description of the configuration depends upon: the number of modules, the types and shapes of the individual modules, and the relative orientation between successive modules. The module information must be fully specified in order to determine the forward kinematics of a reconfigurable manipulator.

RMMS includes a database containing a geometric description of each joint and link module included in the system. For all modules, the geometric description includes specifying homogeneous transformations that describe the geometry of the link or joint module.

We use one homogeneous transformation matrix to completely specify the geometry of a link. It relates the position and orientation of the connector at one end of the link relative to the connector at the other. The homogeneous transformation for the sample link depicted in Figure 1 is given by:

$$L = \text{Trans}(0,0,w)$$

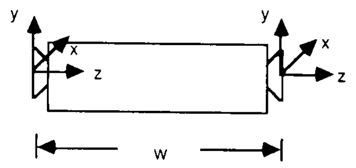

Figure 1: Sample Module Specification for a Link

In order to incorporate the movement (or the degree of freedom) effected by a joint, as well as its shape, into its geometric description we use two homogeneous transformation matrices. We begin our specification of the joint by placing a coordinate frame with its z-axis coincident with the axis of rotation of the joint. The directions of the x and y axes are chosen arbitrarily so long as they form a right-handed orthogonal coordinate system. Also, the origin of the coordinate frame may be located arbitrarily along the z-axis. The origin of this frame designates the origin of the module, or the joint origin. Figure 2 shows a sample module specification for a joint.

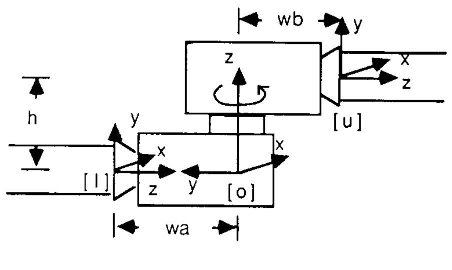

Figure 2: Sample Module Specification for a pivot joint

The complete geometric description of the joint is then specified by two transformations: ${}^{l}J_{o}$ from the lower left connector to the origin of the joint, and ${}^{o}J_{u}$ from the origin to the upper right connector[3]. The transformations:

$${}^{l}J_{o} = \text{Trans}(0,0,w_a)\text{Rot}(x,-90^{\circ})$$
$${}^{o}J_{u} = \text{Trans}(0,-w_b,h)\text{Rot}(x,90^{\circ})^{4}$$

specify the geometry for the sample joint in Figure 2.

In order to allow for the modules to be assembled in any sequence, the coordinate frames for the connectors, as assigned in the database, all have the same orientation. Therefore, the implied transformation between connectors of successive modules is the identity transformation. However, a manipulator may be configured with a twist, or orientation offset, between two modules. We incorporate this into our model with an offset angle about the z-axis at the connector. It represents the angle by which a module's orientation differs from its previously assigned home position. We call the link offset angle σ.

Thus the complete geometric transformation from the origin of joint module *j-1* to the origin of joint module *j* is given by:

$${}^{j-1}M_j = [{}^{o}J_u]_{j-1} \cdot \text{Rot}(z,\sigma_1) \cdot [L]_j \cdot \text{Rot}(z,\sigma_2) \cdot [{}^{l}J_o]_j$$

The above expression allows for the most general case. If two joints are connected in sequence, without a link between them, then the matrix $[L]_j$ is replaced by an identity matrix.

The above transformation equation expresses the geometry of the modules, or more specifically the shape of the manipulator between successive axes of rotation. It does not include the variable of motion of the joints. We use this transformation equation solely to determine the constant values of the DH parameters. The degree of freedom, or the variable of motion, is included when we form the DH A matrices. If the joint is revolute, we include the parameter θ_i as a variable, else the joint is prismatic and we include the parameter d_i as a variable.

While the modular transformation given above can always be achieved with a homogeneous transformation, it may not be DH in nature. That is, it may not be possible to transform the coordinate frame at the *j-1*th joint origin to that at the *j*th origin with the four DH transformations.

Figure 3 contains an example manipulator configured with RMMS. The modular frames are labelled with [j=n] and the DH frames are labelled with {i=n}, where n is the frame number. For frames 0 and 2 the modular frames and the DH frames are identical.

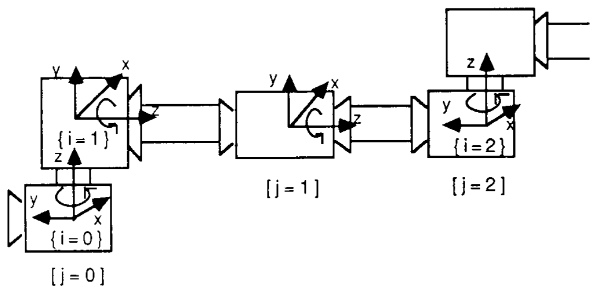

Figure 3: Example: 3 DOF Manipulator

[3]The specification of *lower left* and *upper right* is arbitrary.

[4]Note: This modular transformation is not realized by a Denavit-Hartenberg transformation; it requires a translation along the y-axis of the frame at the joint origin.

However, the modular frame of the rotation joint, frame [j=1] in Figure 3, cannot be reached from the base frame by DH transformations. It requires a translation along the y-axis of the base frame and the DH system only allows for translations in the x and z directions. Therefore, the locations of modular frame [j=1] and DH frame {i=1} must differ. (The disparity between the modular frame [j=1] and DH frame {i=1} cannot be eliminated by simply choosing an alternate orientation for the base axes. This also yields a frame which cannot be transformed to frame [j=1] through DH transformations.)

We cannot define the DH parameters for successive frames solely from the modular transformation ${}^{j-1}M_j$. It is necessary to have a transformation equation which involves the modular transformations for the new joint and for the previous joint, as well as the DH transformation for the previous joint. We define a transformation from the *i-1th* DH frame to the *jth* modular frame. It is called ${}^{i-1}N_j$ and is given by:

$${}^{i-1}N_j = [A_{i-1}]^{-1} \cdot {}^{i-2}N_{j-1} \cdot {}^{j-1}M_j$$

where ${}^{i-2}N_{j-1}$ is the transformation between the *i-2th* DH frame and the *j-1th* modular frame. $[A_{i-1}]^{-1}$ is the inverse of the previous DH transformation. The justification for this equation comes from the transformation diagram in Figure 4.

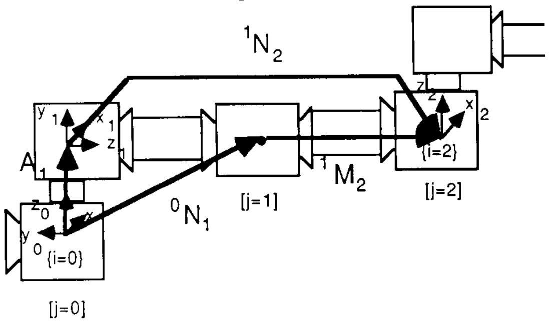

Figure 4: Transform relation between D-H and modular frames

3.1 Forward Kinematics Algorithm

In this section, we present a systematic algorithm to automatically generate the DH parameters for any RMMS manipulator. Our algorithm works without regard for the shapes of the modules or their sequence in the manipulator. We model the degrees of freedom of the manipulator in our model with equations for the axes of rotation. Each joint axis is represented as a line in three dimensional space defined by one point on the line and the line's direction cosines. As an example of our notation, consider the equation of the z-axis of the base frame. It is {(0,0,0) (0,0,1)}, where the first subset is a point on the axis, and the second is the direction cosines.

Each modular axis of rotation gives rise to a DH axis of rotation; that is, the z-axis for each DH frame has the same direction cosines as its corresponding modular axis of rotation. We incorporate the geometric descriptions into our model with the transformation ${}^{i-1}N_j$. (We use the geometric information to calculate the displacements between the DH frames.)

Actual generation of the DH parameters involves determining the relative position and orientation of successive DH frames. The angle between the z_{i-1} and z_i is the DH parameter α_i. The displacements between the origins of successive DH frames give rise to the parameters a_i and d_i. They represent translations along the x_i and z_{i-1} axes, respectively.

In order to find a_i, we need to know the equation of the x_{i-1} and x_i axes. Before determining the equation for the x_i axis, we classify the *ith* DH frame according to one of two classes: those for which the z_i axis is parallel or skew to the z_{i-1} axis, and those for which the z-axes intersect. To determine into which class the *ith* frame falls, we assume that the z-axes intersect and then check for a contradiction. That is, we assume that there exists a point of intersection between the z_{i-1} axis and the z_i axis. We find a symbolic representation for that point on both z-axes and equate the two representations for the point. If we discover an inconsistency, then our assumption is incorrect and the z-axes must be parallel or skew. Otherwise, they intersect.

If the z-axes intersect, then the direction cosines for x_i are found from the cross product of z_i and z_{i-1}. The distance between the origins of the *i-1th* frame and the *ith* frame defines d_i. No translation along x_i is necessary to get to the *ith* frame. Therefore, a_i is equal to zero.

If the z-axes are parallel, then the direction cosines for x_i are defined so that x_i lies along the common normal between the z axes. To find the equation of the common normal, we represent it by a line and then consider the dot product of the line with the z_i and z_{i-1} axes. If the line is perpendicular to both z-axes, then both dot products must be zero. In this way the direction cosines of the common normal are defined; its length is the DH parameter a_i. Because the z-axes are parallel or coincident, d_i, which is the distance between the coordinate frames along z_{i-1}, must be zero.

The last DH parameter, α_i, is defined to be the angle between z_i and z_{i-1}. It is computed as: $\cos^{-1} [z_{i-1} \cdot z_i]$. However, some configurations require an additional angular displacement. The transformations between the home positions of successive frames, where θ_i is assumed to be 0°, is not possible without a constant rotation about the z_{i-1} axis. Such a rotation, which we call β_i, is typically ±90° because manipulators are built with joints and links at right angles to each other. However, in general, it is equal to the angle between the x_i and x_{i-1} axes and can take any value. It is computed as: $\cos^{-1} [x_{i-1} \cdot x_i]$. Therefore, our DH rotation becomes: $\mathrm{Rot}(z_{i-1}, \theta_i+\beta_i)$. This completes the forward kinematic model and creates the information needed as input to solve for the inverse kinematics.

4 Inverse Kinematics for Reconfigurable Manipulators

In order to do any controlled movement it is necessary to have an inverse kinematic model to determine the joint angles required to achieve a desired position and orientation of the end-effector. Ideally, one derives closed form equations for the inverse kinematics where each joint variable is expressed in terms of other know quantities. However, existence of a closed form inverse kinematics solution depends on the kinematic structure of the manipulator [16, 20]. For example, we know that a closed form solution exists for a manipulator which has six degrees of freedom, three of which have intersecting axes, such as in a spherical wrist [16]. This solvability condition is not necessary, but only sufficient. Because an RMMS manipulator can assume any configuration, including one with more than six degrees of freedom, it may not be possible to find a closed form solution. The standard method of obtaining closed form solutions further inhibits the feasibility of employing them for RMMS. The inverse kinematics equations are typically found by isolating expressions for the individual joint variables in terms of other known quantities in the complex forward kinematic relationships. Since this is not feasible in an automated system, we propose using numerical methods to solve the inverse kinematics equations.

In the forward kinematic equation the vector of position and orientation, $\mathbf{x}$, is related to the joint variables, $\mathbf{q}$, by:

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

where $\mathbf{f}(\mathbf{q})$ is the standard $\mathbf{T}_N$ matrix (for an N degree of freedom manipulator).

A closed-loop method for solving the inverse kinematics equations using the Newton Raphson method proposed in [8] is

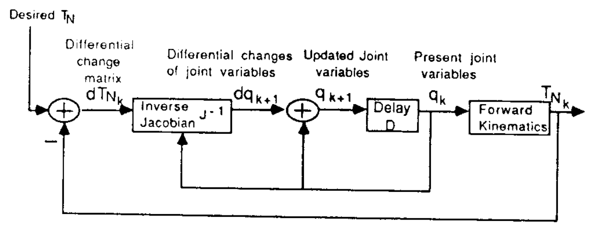

Figure 5: Block diagram of iterative inverse kinematics procedure

depicted in Figure 5. The algorithm, which was derived for a non-redundant manipulator, uses the closed form inverse kinematics expressions to create a *closed form* inverse Jacobian. For RMMS, we solve numerically for the inverse Jacobian and expand upon the algorithm to create a system which is valid for a manipulator with any number of degrees of freedom.

The iterative method determines the necessary changes in the joint angles to achieve a differential change in the position and orientation of the end-effector. The differential changes in the cartesian coordinates ($\mathbf{dx}$) are related to the corresponding differential changes in the joint coordinates ($\mathbf{dq}$) by:

$$\mathbf{dx} = \mathbf{J}(\mathbf{q})\mathbf{dq}, \tag{2}$$

where $\mathbf{J}(\mathbf{q})$ is the position dependent Jacobian of the manipulator. Inverting Equation (2) to obtain an expression for the differential inverse kinematics we get:

$$\mathbf{dq} = \mathbf{J}^{-1}(\mathbf{q})\mathbf{dx}, \tag{3}$$

where $\mathbf{dx}$ is equivalent to $\mathbf{dT}_N$, which represents the differential change in the homogeneous transformation matrix of position and orientation, and $\mathbf{J}^{-1}$ is the closed form inverse Jacobian. It is a function of the $\mathbf{T}_N$ matrix and the differential changes in the joint angles. We rewrite this equation as,

$$\mathbf{dq}_{k+1} = \mathbf{J}^{-1}(\mathbf{q}_k)\mathbf{dT}_{N_k}, \tag{4}$$

where k is the number of the iteration. The joint variables are updated according to:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \mathbf{dq}_{k+1}. \tag{5}$$

We solve Equations (4) and (5) iteratively, until each term in $\mathbf{T}_{N_k}$ is within the required error tolerance, ϵ, of the desired $\mathbf{T}_N$.

4.1 Redundant Manipulators

Although a six degree of freedom manipulator is highly versatile, it is may inadequate for performing a specified task. For example, it may be incapable of avoiding obstacles in its workspace while maintaining the desired position and orientation of the end-effector. Additionally, it may enter a singular configuration while tracing a specified path. Adding extra degrees of freedom to a manipulator enables it to avoid obstacles in its path or to avoid configurations corresponding to internal singularities [6].

Because of the benefits of kinematic redundancy, many manipulators are configured with more than six degrees of freedom. A manipulator is easily made redundant with RMMS. However, introducing redundancy complicates the control algorithm. The Jacobian, which relates differential changes in the joint variables to differential changes in the cartesian variables, is no longer a square matrix. Its dimensions are M x N, where M is the number of degrees of freedom of the workspace and N is the number of degrees of freedom in the manipulator. For a redundant manipulator, M and N are not equal and the Jacobian is not invertible. In this case, we use a generalized inverse of the Jacobian.

Much of the previous research on inverse kinematics for redundant manipulators has focused on the pseudoinverse [1, 3, 10]. The pseudoinverse is a generalized inverse which provides the minimum norm solution [13]. It is defined as follows:

$$\mathbf{J}^{+} = \mathbf{J}^{\mathrm{T}}(\mathbf{J}^{\mathrm{T}}\cdot\mathbf{J})^{-1}. \tag{6}$$

Because standard pseudoinverse control has been shown to be inadequate in the neighborhood of singularities, many methods have been developed which augment the pseudoinverse so as to use the kinematic redundancy to optimize an objective function [1, 2, 10].

In this section we consider a method for control of redundant manipulators within RMMS. While the methods cited above are configuration dependent, computationally intensive, or both, the method we propose for RMMS achieves singularity avoidance and requires marginally more computations than the standard pseudoinverse. It is called the singularity robust inverse [12]. We begin by discussing the background and properties associated with the singularity robust inverse and then we propose a configuration independent technique for automatically choosing the necessary design parameters.

The pseudoinverse solution is problematic in the neighborhood of a singularity. For example, in an effort to converge to an exact solution, the pseudoinverse may generate an infeasible solution. That is, it may generate a solution for which one, or more, of the $\mathbf{dq}$ values is so large that it cannot be physically realized. To circumvent the problem of excessively large joint velocities, we consider the singularity robust inverse [12]. It generates continuous and feasible solutions at, and in the neighborhood of, singular points.

The singularity robust inverse is based upon an evaluation index,

$$d\phi = \begin{pmatrix} \mathbf{dx} - \mathbf{J}\cdot\mathbf{dq} \\ \mathbf{dq} \end{pmatrix}, \tag{7}$$

which simultaneously considers the exactness of the solution, as measured by the top term, and the feasibility of the solution, as measured by the bottom term. When solving the inverse kinematics problem one must find the minimum weighted Euclidean norm of the evaluation index. Through a weighted norm, one is able to control the relative importance of the terms in the evaluation index.

The singularity robust inverse, $\mathbf{J}^{*}$, replaces the pseudoinverse, $\mathbf{J}^{\mathrm{T}}$, in the inverse kinematics algorithm. It is given by the following equation:

$$\mathbf{J}^{*} = \mathbf{J}^{\mathrm{T}}(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{I})^{-1}, \tag{8}$$

where λ is the scale factor between the exactness and the feasibility of the solution. It provides a means for weighting the terms in the evaluation index. In the following section we propose a method for choosing an appropriate value for λ.

4.2 A Method for *Automatically* Choosing the Scale Factor

In order to employ the singularity robust inverse for RMMS, we must develop a method to automatically generate an appropriate scale factor for any manipulator. We would like the scale factor, λ, to have a larger value in the neighborhood of singular points and a small value, or zero, far from singular points. That is, when the manipulator is far from singular points, we would like the solution found with the singularity robust inverse to emulate the solution found with the pseudoinverse. We consider the following equation for λ [12]:

$$\lambda = \begin{cases} \lambda_0(1 - \frac{\omega}{\omega_0}) & \text{if } \omega < \omega_0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where $\omega = \sqrt{determinant(\mathbf{J}\cdot\mathbf{J}^T)}$ is a manipulatability measure for the manipulator [21], λ_0 approximates the magnitude of the scale factor at singular points, and ω_0 is a threshold which identifies the boundary or the neighborhood of singular points. Equation (9) automatically adjusts λ according to the manipulator's distance from a singular point.

In the remainder of this section we propose a technique for selecting the design parameters to determine λ. Since our technique is driven by the need for generality, it should not be configuration dependent. It must be valid regardless of the type or the degree of the redundancy. It must work without a priori knowledge of the location of a manipulator's singularities. Finally, it must operate within a real-time inverse kinematics control loop.

First we consider the problem of detecting the neighborhood of a singularity, and thus specifying ω_0. Just as the location and neighborhood of singularities are manipulator dependent, so is the manipulatability measure, ω. It is a function of the shape of the manipulator. While the value of ω approaches zero as the manipulator approaches a singular point, the absolute magnitude of ω varies with the dimensions and the units of measure of the links and joints of the manipulator. An ω on the order of magnitude of 10^2 may imply that one manipulator is near a singular point, but another manipulator, which has much smaller dimensions, may be far from one. By *scaling* a manipulator we are able diminish the disparity in the magnitudes of ω between different manipulators and eliminate the dependence on the units of measure. Specifically, we propose dividing the a_i and d_i DH parameters of a manipulator by the magnitude of the largest one. The resulting parameters have magnitudes between zero and one, inclusive. However, a disparity in ω values between manipulators remains; different scaled manipulators may still generate significantly different ω values.

Rather than defining an absolute threshold, we propose checking for a sudden drop in the value of ω between iterations. As a manipulator approaches a singular configuration the value of ω decreases dramatically. We detect the neighborhood of a singularity when the ratio $\frac{\omega_{k+1}}{\omega_k}$ falls below a threshold μ. That is, we examine the ratio of ω between the k^{th} and the $k+1^{th}$ iterations of the Newton-Raphson algorithm.

The equation governing the scale factor λ becomes:

$$\lambda = \begin{cases} \lambda_0(1 - \frac{\omega_{k+1}}{\omega_k}) & \text{if } \frac{\omega_{k+1}}{\omega_k} < \mu \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Based upon experimental results [7], we suggest $\mu \approx \frac{1}{\sqrt{10}}$ as a reasonable value for a scaled 7 degree of freedom manipulator.

We choose λ_0 based on the tradeoff that is the premise for the singularity robust inverse method. By adding a larger scale factor we make the solution less exact, but more feasible or robust. In order to generate a less exact solution we must increase ϵ. (Recall ϵ is the convergence error tolerance for the Newton-Raphson algorithm.) Alternatively, we maintain the error tolerance and increase the number of iterations of the Newton-Raphson algorithm until the error becomes less than ϵ. In order for the Newton-Raphson iteration to converge, the residual error must be less than the error tolerance ϵ. As λ contributes to the residual error, we propose setting λ_0 equal to a value one order of magnitude smaller than ϵ. That is, we set $\lambda_0 = 0.1\epsilon$ as a conservative choice for the scale factor near a singularity.

Based upon experimental results, our algorithm typically requires 3-4 iterations to converge with four decimal place accuracy for large changes in the joint variables. Generally, the singularity robust inverse converges in the neighborhood of a singular point in 4-5 iterations.

The desired effect of implementing the singularity robust inverse according to Equation (10) is the augmentation of the pseudoinverse only in the neighborhood of a singular point. However, if augmenting should occur far from a singular point, it causes very little disturbance to the system. The scale factor, λ (as chosen according to the above method), is significantly smaller than the singular values of the manipulator Jacobian. Therefore, it has very little effect on the system.

4.3 Examples: Pseudoinverse vs. Singularity Robust Inverse

In order to compare the singularity robust inverse with the pseudoinverse we consider examples for both redundant and non-redundant manipulators. Figure 6 shows a 3 degree of freedom planar manipulator under pseudoinverse control. The manipulator starts in a nearly singular configuration, moves to a distant point, and then returns to its start point. Notice that the manipulator returns to a nearly singular configuration with $\omega = 8.82E\text{-}2$. Figure 7 shows the same manipulator under singularity robust control. It returns to a more robust configuration with $\omega = 2.91E\text{-}1$.

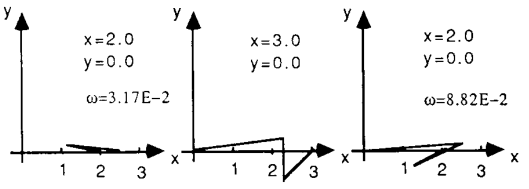

Figure 6: Pseudoinverse Control Near a Singularity

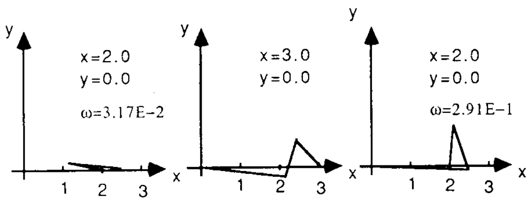

Figure 7: Singularity Robust Control Near a Singularity

While Figures 6 and 7 provide insight for a 3 degree of freedom planar manipulator, they cannot effectively show the improvement of the singularity robust inverse for 6 and 7 degree of freedom manipulators. Rather, we present a table which summarizes the manipulatability results for several examples. In each case, the manipulator begins in a configuration which is nearly singular and then moves in the direction of the singularity. Table 1 lists the initial ω_i and final ω_f values for the scaled manipulators under pseudoinverse control and under singularity robust control.

Manipulator	Pseudoinverse	Singularity Robust
Planar 3-DOF	ω_i=4.0E-2 ω_f=6.2E-3	ω_i=4.0E-2 ω_f=8.4E-3
6-DOF	ω_i=1.6E-3 ω_f=1.1E-7	ω_i=1.6E-3 ω_f=1.9E-6
7-DOF	ω_i=2.5E-3 ω_f=3.5E-5	ω_i=2.5E-3 ω_f=1.1E-4

Table 1: Comparison of Pseudoinverse and Singularity Robust

5 Summary

In this paper we have proposed an algorithm to determine the forward and the inverse kinematic solutions of a Reconfigurable Modular Manipulator System. Our method is not only independent of the number and shape of the modules, but it is also independent of the types of joints used in the manipulator. That is, it accomodates both prismatic and revolute joints. We solve for the inverse kinematics using numerical methods. Our method is completely general and can be applied to a redundant system. In the proposed method the extra degrees of freedom of a redundant manipulator are used to achieve singularity avoidance. Further, in order to make the singularity robust inverse independent of the manipulator link lengths, we have proposed the idea of scaling a manipulator. We have also described techniques for choosing the design parameters for the singularity robust inverse. Finally, we have implemented our method on the prototype RMMS developed in our laboratory. Our implementation requires 35 msec for each iteration of the inverse kinematics algorithm for a seven degree of freedom manipulator.

Acknowledgements: The authors wish to thank Professor Takeo Kanade, Director of the Robotics Institute, for his support and cooperation throughout the course of this research. We would also like to express our appreciation to Mark DeLouis, Regis Hoffman, Jim Moody, and Don Schmitz for their hardware and software support of the RMMS project.

This research was supported in part by a GTE Fellowship (for L. Kelmar) and the Department of Electrical and Computer Engineering.

References

1. J. Baillieul, J. Hollerbach, R. Brockett, "Programming and Control of Kinematically Redundant Manipulators", *Proc. 23rd Conference on Decision and Control*,December 1984, pp. 768-774.

2. J. Baillieul, "Kinematic Programming Alternatives for Redundant Manipulators", *IEEE International Conference on Robotics and Automation*,Vol. 1,March 1985, pp. 722-728.

3. P.H. Chang, "A Closed-form Solution for the Control of Manipulators with Kinematic Redundancy", *IEEE Internaltional Conference on Robotics and Automation*,Vol. 1,April 1986, pp. 9-14.

4. J. Denavit and R.S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices", *ASME Journal of Applied Mechanics*,Vol. 2, 1955, pp. 215-221.

5. K.C. Gupta, "Kinematic Analysis of Manipulators Using the Zero Reference Position Description", *The International Journal of Robotics Research*,Vol. 5,No. 2,Summer 1986, pp. 5-13.

6. J.M. Hollerbach, "Optimum Kinematic Design for a Seven Degree of Freedom Manipulator", *2nd International Symposium on Robotics Research*,August 20-23 1984, pp. 349-355.

7. L. Kelmar, "Automatic Generation of Kinematics for a Reconfigurable Manipulator System", Master's thesis, Carnegie Mellon University, January 1988.

8. P.K. Khosla, C.P. Neuman, and F.B. Prinz, "An Algorithm for Seam Tracking Applications", *The International Journal of Robotics Research*,Vol. 4,No. 1,Spring 1985, pp. 27-41.

9. M. Kircanski and M. Vukobratovic, "A New Programe for Generating Symbolic Kinematic Models of Arbitrary Serial-Link Manipulators", *Proceedings 16th International Symposium on Industrial Robots*,September 30 1986, pp. 249-258.

10. C. A. Klein and C-H Huang, "Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators", *IEEE Trans. on Systems, Man, and Cybernetics*,Vol. SMC-13,No. 3,March/April 1983, pp. 245-250.

11. A. Liegeois et al, "Mathematical and computer models of interconnected mechanical systems", *Second International Symposium on Theory and Practice of Robots and Manipulators*,September 1976, pp. 5-17.

12. Y. Nakamura and H. Hanafusa, "Inverse Kinematic Solutions with Singularity Robustness for Robot Manipulator Control", *Journal of Dynamic Systems, Measurement, and Control*,Vol. 108,September 1986, pp. 163-171.

13. B. Noble and J.W. Daniel, *Applied Linear Algebra,* Prentice Hall, N.J., 1977, Second Edition

14. R.P. Paul, *Robot Manipulators: Mathematics, Programming and Control,* MIT Press, Cambridge, MA, MIT Press Series in AI, 1981.

15. R.P. Paul and H. Zhang, "Computationally Efficient Kinematics for Manipulators with Spherical Wrists Based on the Homogeneous Transformation Representation", *The International Journal of Robotics Research*,Vol. 5,No. 2,Summer 1986, pp. 32-44.

16. D. Pieper, *The Kinematics of Manipulators under Computer Control,* PhD dissertation, Stanford University, 1968.

17. B. Roth, *Screws, Motors, and Wrenches That Cannot Be Bought in a Hardware Store,* MIT Press, Mass., 1984, pp. 679-693.

18. D.E. Schmitz, T. Kanade, and P.K. Khosla, "Design of a Reconfigurable Modular Manipulator System", *18th ISIR*, 1988.

19. N. Sreenath and P.S. Krishnaprasad, "DYNAMAN: A Tool for Manipulator Design and Analysis", *IEEE International Conference on Robotics and Automation*,Vol. 2,April 1986, pp. 836-842.

20. W.A. Wolovich, *Robotics: Basic Analysis and Design,* Holt, Rinehart and Winston, New York, 1987.

21. T. Yoshikawa, "Analysis and Control of Robot Manipulators with Redundancy", *Reprints from the 1st International Symposium of Robotics Research*, 1983, pp. 735-747.