LOCAL SENSORY CONTROL OF A DEXTEROUS END EFFECTOR

A FINAL REPORT
for
GRANT NAG 9-326

Submitted to

Cliff Hess
Larry Li
Automation and Robotics Division
Robotic Systems Technology Branch
Nasa Johnson Space Center
Houston, Texas 77058

by

Victor H. Pinto
Louis J. Everett
Mechanical Engineering Department
Texas A&M University
College Station, Texas 77843

Morris Driels
Mechanical Engineering Department
Naval Postgraduate School
Monterey, California 93943

January 1988 to December 1990

CONTENTS

LIST OF TABLES

LIST OF FIGURES

EXECUTIVE SUMMARY

Task Overview

Grant NAG 9–326 supported 24 months of research comprising two related tasks in grasping using dexterous robot hands. This section of the report summarizes the objectives and status of each task.

Introduction

There are four steps that must be taken to ensure the grasp of an arbitrary object. The first step is the *pre–grasp phase* where approach to the object is planned, but no contact is made. Next the object's position and orientation, or *pose*, within the grasping region of the hand must be known to position and close the hand in an appropriate manner. An added benefit from knowing the object pose has to do with manipulation of the object. Using the sensor data can aid in grasping the object in such a way which will either not require further manipulation or will make the manipulation of the object easier. Third, the hand must close in a way to create an envelope in which the object cannot escape. The forth step is to use the information from tactile, force, and vision sensors to determine whether a stable grasp has been achieved.

It is clearly understood that to achieve the grasp of an object requires low–level control to move, monitor, and compensate the finger motions. To this end, a method to calculate the joint coordinates for prescribed finger positions and a method of determining the pose of the object within the grasping region of the hand are necessary.

The Journal of Robotics and Automation has been used as a model for style and format.

Generalized Inverse Kinematics

Controlling the motion of each finger of a dexterous hand will require the inverse kinematic solution for the hand; i.e., to find the joint coordinates of each finger given a fingertip pose. The desired fingertip positions are pre–selected contact points on the target object. These contact points can be determined based on task constraints. For example, the vision sensors locate and track the target object. Based on the object's geometry, contact points are selected which are inputs to the inverse kinematic algorithm.

Proposed solutions to the inverse kinematic problem have used analytical, iterative, and knowledge–based systems. Numerical iterative techniques are useful because of their generality in the sense that the same basic algorithm can be applied to many different manipulators. This generality allows the user to quickly modify the algorithm for the desired dexterous hand. The solution method consists of writing the finger equations using homogeneous transformations, specifying desired finger contact points, and determination of the joint coordinates by minimizing the difference between the desired fingertip pose and the actual fingertip pose.

The finger equations can be written by treating each finger as an independent manipulator. A finger may be modeled as a series of mechanical linkages connected by prismatic or revolute joints. Each link is then defined in terms of special parameters known as the Denavit–Hartenberg (link) parameters.

An equation may be obtained relating the joint variables in the link matrices to the task space coordinates. Then, through a minimization process, the joint variables are found for the required position and orientation. This minimization is accomplished using a nonlinear least–squares technique.

The algorithm presented has been tested on a three–fingered dexterous hand. Given a target object and selected contact points, feasible solutions were obtained. In general, the limitations in the algorithm are that the fingers must have either revolute or prismatic joints and are open loop kinematic chains. This algorithm can aid in calculating the inverse kinematic solution for the various dexterous hands available on

site, or the ones currently being developed, and will provide an alternative to solving for the solution to these hands analytically.

Pose Estimation of a Pre–Grasped Object

Knowledge about the object, such as the position and orientation, must be obtained that will allow a grasp strategy to be formulated. The problem of grasping objects by dexterous hands has been widely considered. In most studies, the object is well defined by vision sensors and/or by tactile sensors, and the grasp strategy uses the data from these sensors. The use of vision sensors may be limited because the object may be hidden from view by the robot's arms or other objects. In the space environment, tactile sensors cannot be used for the pre–grasp phase because the object cannot be touched prior to grasping or the object will float away. The pose estimation problem is to determine the pre–grasp position and orientation of an object using local, noncontact sensors.

The pose estimation problem is divided into two subtasks: the forward model and the inverse model. The forward model of the pose estimation problem assumes the object pose and finger joint angles are known and solves for the sensor parameters. The *inverse model* assumes sensory data is available, the finger joint angles are known, and uses this information to find the object pose. Results from the forward model will enable verification of the inverse model.

To develop the equations for pose estimation, a mathematical model of the sensor characteristics is necessary for software implementation. The type of noncontact sensors that are planned for use on the EVA Retriever are optical reflectance proximity sensors. Due to the complexity of modelling this type of nonlinear sensor, it was decided to implement an ideal sensor.

The pose estimation equations were written using the concept of triangulation. An example of a two–fingered hand surrounding a sphere is presented. By choosing the object to be a sphere, only the position of the object frame was important. Results show that a minimum of three sensors are needed to find the position of the

sphere. More complex shapes can also be implemented, but this will require more computation to model them.

The algorithms for inverse kinematics and pose estimation were developed assuming several simplifications. In retrospect, the investigation has provided a software shell which can be easily upgraded to take into account more complex shapes and sensor models.

Work In Progress

A strategy termed the *Grasp Algorithm* is proposed which utilizes the inverse kinematics and pose estimation software to provide low–level control of a dexterous hand when grasping an object. This algorithm is currently being implemented on site using computer animation software and should be available for viewing by December, 1990.

Bibliography Generated from the Grant

To date, one paper titled "Pose Estimation of a Pre–Grasped Object Using Local Sensors on a Dexterous Robotic Hand" was generated from grant support. It will be presented at the Fifth International Conference on CAD/CAM Robotics and Factories of the Future that will take place in December, 1990.

INTRODUCTION

Current work at NASA's Johnson Space Center includes developing an autonomous robot to perform tasks such as object rescue and retrieval, space shuttle experimentation, satellite repair, and space station construction. Dexterous end effectors with sensors can help the robot perform this large variety of tasks. Of the tasks previously mentioned, object rescue and retrieval is considered the most challenging and one of the primary reasons for developing an autonomous robot [1]. Object rescue and retrieval requires locating and tracking the object, movement towards the object, and grasping and maintaining a stable grasp of the object [1]. Stable grasp must be ensured to avoid having the object escape and drift into space. The EVA Retriever, shown in Figure 1, supplied with dexterous end effectors is currently under development to perform this task.

There are four steps that must be taken to ensure the grasp of an arbitrary object. The first step is the *pre-grasp phase* where approach to the object is planned, but no contact is made. Next the object's position and orientation, or *pose*, within the grasping region of the hand must be known to position and close the hand in an appropriate manner. An added benefit from knowing the object pose has to do with manipulation of the object. Using the sensor data, the object can be grasped in such a way which will either not require further manipulation or will make the manipulation of the object easier. Third, the hand must close in a way to create an envelope in which the object cannot escape (this may only be possible for objects smaller than the palm width of the hand). Finally, using multiple sensor information, from tactile, force, and vision sensors, determine whether a stable grasp has been achieved.

To achieve the grasp of an object requires low-level control to move, monitor, and compensate the finger motions. To this end, a method to calculate the joint coordinates for prescribed finger positions and a method of determining the pose of the object within the grasping region of the hand are necessary.

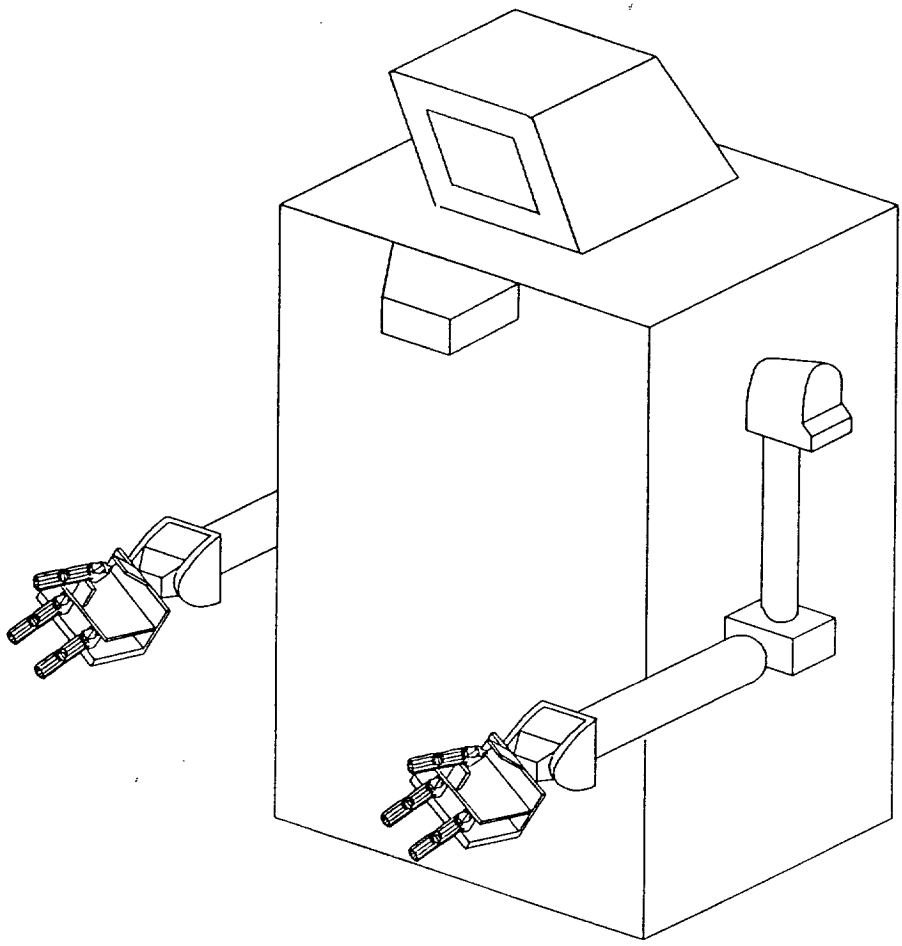


Figure 1. The EVA Retriever [1].

Kinematic Analysis of Manipulators

A manipulator may be modeled as a series of mechanical linkages connected by, but not limited to, prismatic or revolute joints. Each link is defined in terms of special parameters known as the Denavit–Hartenberg (link) parameters. The spatial position of a link frame relative to another frame is expressed as a 4×4 homogeneous transformation matrix [2]. Using this information permits defining a generic manipulator whose configuration is based on user–supplied link parameters.

To control the motion of each finger of a dexterous hand will require the inverse kinematic solution for the hand; i.e., to find the joint coordinates of each finger of a dexterous hand for a given fingertip pose. Proposed solutions to the inverse kinematic problem have used analytical, iterative, and knowledge–based systems. An analytical closed–form solution [2, 3] is advantageous in that it allows the joint variables of a particular manipulator to be easily found; on the other hand since only certain classes of manipulators allow a closed–form solution, it is very difficult to solve explicitly for the joint variables without having some geometric intuition about the manipulator, and the solution only applies to the particular manipulator.

Numerical iterative techniques are useful because of their generality in the sense that the same basic algorithm can be applied to many different manipulators, including kinematically redundant manipulators and manipulators with less than six degrees-of-freedom. Some techniques are more useful than others for certain classes of manipulators. For example, Poon and Lawrence (1988) present an algorithm that is useful for functionally partitionable manipulators, which are manipulators where each link has its own task and the joints are controlled independently. These manipulators, such as the Unimation PUMA and Unimate, Cincinnati Milacron, and the Stanford manipulators, can be partitioned into major and minor linkage sets. The major linkage controls the end effector position and the minor linkage changes the end effector orientation. Furthermore, most iterative techniques utilize the manipulator Jacobian, as in Goldenberg, Benhabib, and Fenton (1985). A necessary condition for this method to function is that the Jacobian must be nonsingular and can

therefore be inverted. A problem arises at certain points in the joint space of the manipulator, termed the *joint-space singularities*, where the Jacobian matrix loses rank and becomes ill-conditioned [6]. Goldenberg, Benhabib, and Fenton overcome the problem of singularities by utilizing the generalized inverse, or *pseudoinverse.*

A knowledge-based solution using neural networks is proposed by Guez and Ziauddin (1988). The drawback to this method is that it is computationally expensive in that the network must be trained to conform to the manipulator used, versus simply changing certain parameters as in most iterative techniques.

Pose Estimation of a Pre-Grasped Object

Knowledge about the object, such as the position and orientation, must be obtained that will allow a grasp strategy to be formulated. The problem of grasping objects by dexterous hands has been widely considered [8, 9, 10, 11]. In most studies, the object is well defined by vision sensors and/or by tactile sensors, and the grasp strategy uses the data from these sensors. The use of vision sensors may be limited by the object being hidden from view by the robot's arms or other objects. In the space environment, tactile sensors cannot be used because the object cannot be touched prior to grasping or the object will float away. The pose estimation problem is to determine the pre-grasp position and orientation of an object using local, noncontact sensors. *Noncontact sensors*, such as proximity sensors, do not require touch to operate.

The sensors that will be modeled are optical reflectance proximity sensors. This type of sensor will be used on the EVA Retriever. The advantages of this type of sensor are: good size/range ratio, low cost, good reliability, simple to use, and low sensitivity to disturbances by using synchronous modulation or pulsed emmission [12]. Balaure describes and gives the mathematical model of an optical proximity sensor (1986). This model will be useful when implementing a real-world sensor in software. Limited research has been performed on using proximity sensors mounted on a dexterous end effector to determine the position and orientation of an object [14]. Furhman and Kanade apply the concept of triangulation to determine the position and

orientation of an object's surface using a multilight source proximity sensor (1984). Romiti and Raparelli (1987) present a method of finding the position and orientation of a dexterous hand based on proximity sensor data.

KINEMATIC ANALYSIS OF MANIPULATORS

Robot manipulators can be considered to consist of a series of rigid links connected together by a series of joints. The relationship between adjacent links is described by 4×4 homogeneous transformations whose elements are dependent on the link parameters known as the Denavit–Hartenberg parameters. The product of these 4×4 matrices gives the pose of a selected frame relative to a reference frame. This chapter describes the kinematics of a manipulator using the concept of homogeneous transformations.

Homogeneous Transformations

Paul (1981) describes the homogeneous transformations that define the translation and orientation of a coordinate frame.

The transformation corresponding to a *translation* in the direction of vector $\mathbf{v} = a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$ relative to a reference frame is given as a 4×4 homogeneous matrix

$$\text{Trans}(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

Orientation is frequently specified as the product of rotations about the x, y, and z axes, as shown in Figure 2. The Euler transformation describes the orientation of a coordinate frame in terms of a rotation ϕ about the z axis, then a rotation θ about the new y axis, y', and finally a rotation about the new z axis, z'', of ψ. Thus, the Euler transformation is given by the product of the three rotation matrices

$$\text{Euler}(\phi,\theta,\psi) = \text{Rot}(z,\phi)\text{Rot}(y,\theta)\text{Rot}(z,\psi)$$

$$\text{Euler}(\phi,\theta,\psi) = \begin{bmatrix} \cos\phi\cos\theta\cos\psi - \sin\phi\sin\psi & -\cos\phi\cos\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\sin\theta & 0 \\ \sin\phi\cos\theta\cos\psi + \cos\phi\sin\psi & -\sin\phi\cos\theta\sin\psi + \sin\phi\cos\psi & \sin\phi\sin\theta & 0 \\ -\sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2}$$

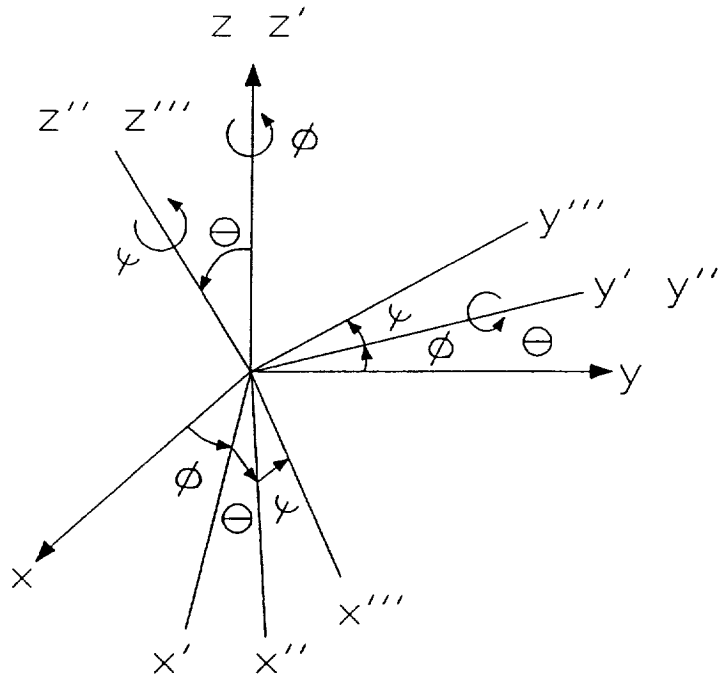


Figure 2. Euler angles [2].

If a coordinate frame is translated in the direction given by $\mathbf{v}$ and then rotated by angles ϕ, θ, and ψ as described above, the composite homogeneous transformation matrix $\mathbf{T}$ which represents the position and orientation of the resulting coordinate frame is

$$\mathbf{T} = \text{Trans}(a, b, c)\text{Euler}(\phi, \theta, \psi). \tag{3}$$

Denavit–Hartenberg Parameters

A serial link manipulator can be considered to consist of a sequence of links connected together by actuated joints, and for an n degree-of-freedom manipulator, there are $n + 1$ links and n joints.

Assigning coordinate frames to each link of the manipulator permits finding the link parameters. Coordinate frames can be assigned to the links according to the

scheme presented in Paul (1981), pp. 50–52. In general, each link of the manipulator will have assigned to it a series of link parameters, known as the Denavit–Hartenberg parameters, which characterize each link and give the relationship between connected links. The parameters are the *distance* d and *angle* θ between adjacent links, and the *length* a and *twist angle* α of a single link (Figures 3, 4, and 5).

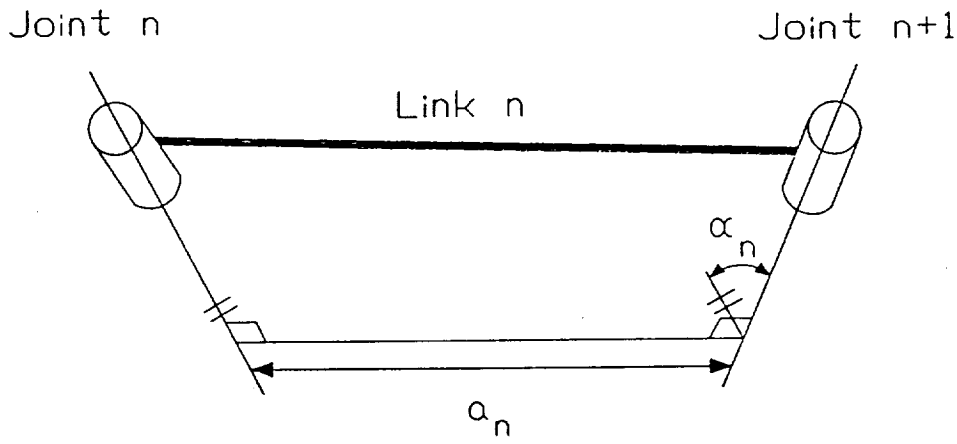


Figure 3. The length a and the twist α of a link [2].

Depending on what type of joint a link has will determine the joint variable. For a revolute joint, the joint variable is θ. For a prismatic joint, the distance d is the joint variable. Figure 4 shows the link parameters for revolute joints and Figure 5 shows the link parameters for a prismatic joint.

Having defined the link coordinate frames, the relation between the successive frames $n-1$ and n is the matrix $\mathbf{A}_n$ defined as

$$\mathbf{A}_n = \text{Rot}(z, \theta_n)\text{Trans}(0, 0, d_n)\text{Trans}(a_n, 0, 0)\text{Rot}(x, \alpha_n) \tag{4}$$

which results in

$$\mathbf{A}_n = \begin{bmatrix} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & a_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & a_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

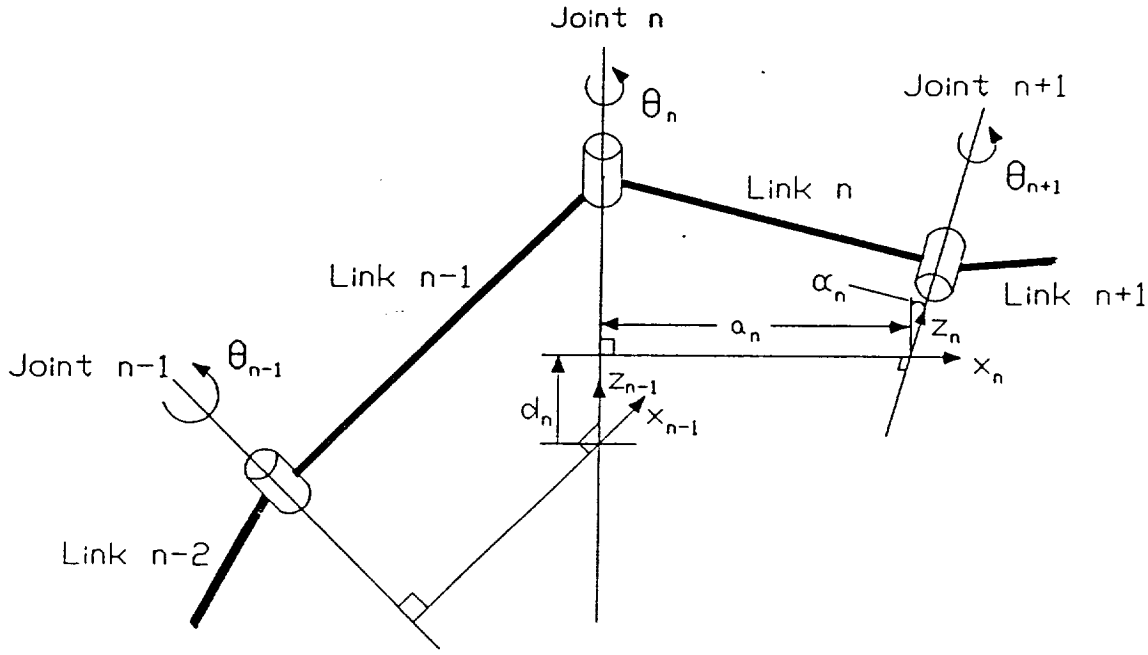


Figure 4. Link parameters of revolute joints [2].

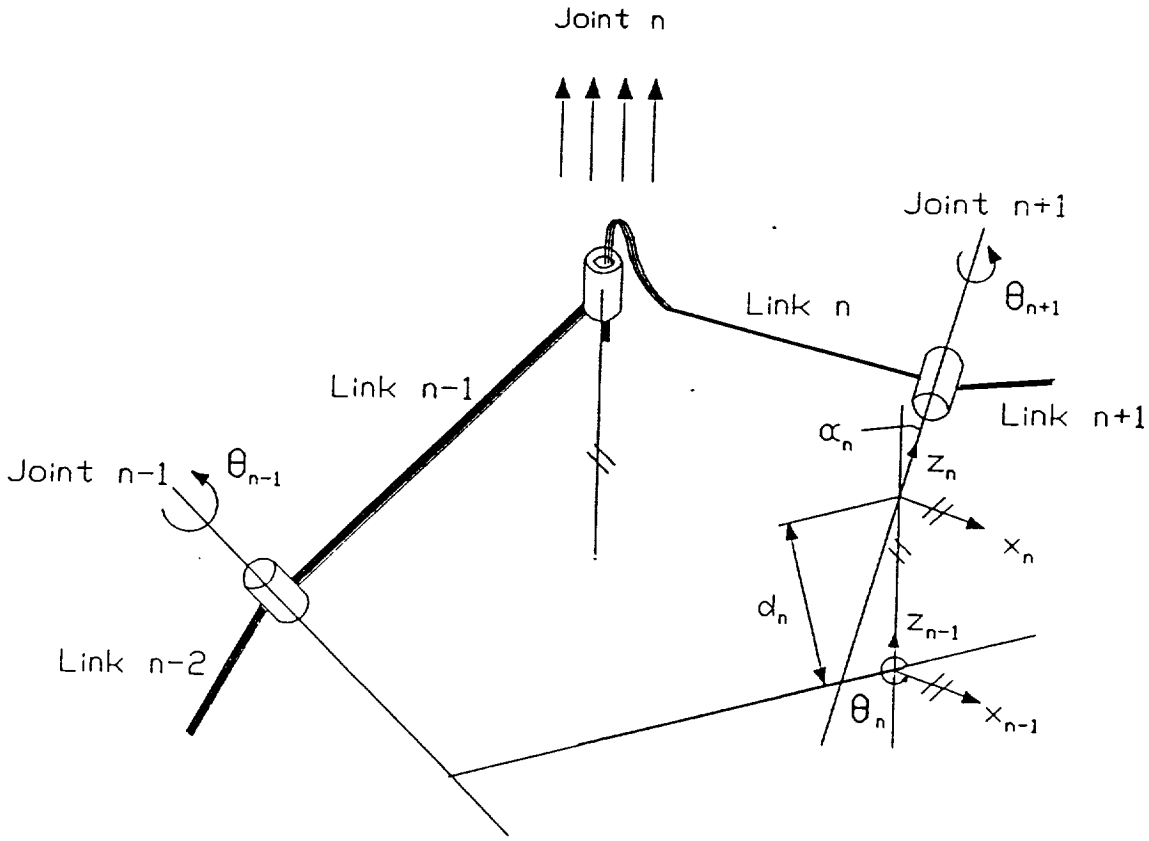


Figure 5. Link parameters of a prismatic joint [2].

Relation Between Joint and Task Space Coordinates

An equation may be obtained relating the joint variables θ and d in the $\mathbf{A}$ matrices to the task space coordinates, which are the required position and orientation of the selected frame given by the $\mathbf{T}$ matrix. For example, if matrix $\mathbf{A}_1$ describes the position and orientation of the first link relative to the base frame and matrix $\mathbf{A}_2$ describes the position and orientation of the second link relative to the first, then the position and orientation of the second link relative to the base frame is the matrix $\mathbf{T}_2$. This is represented as

$$\mathbf{T}_2 = \mathbf{A}_1\mathbf{A}_2.$$

More links may be added, and their position and orientation relative to the base frame can be found. Thus, the transformation representing the position and orientation of the end effector with respect to the base frame for an n degree-of-freedom manipulator is

$$\mathbf{T}_n(\mathbf{q}) = \prod_{i=1}^{n} \mathbf{A}_i(\mathbf{q}) = \begin{pmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6}$$

where $\mathbf{n}$, $\mathbf{o}$, $\mathbf{a}$ are orientation vectors, $\mathbf{p}$ is the position vector, and $\mathbf{q}$ is the vector consisting of the joint variables [5]. Equation 6 in matrix form is

$$\mathbf{T}_n(\mathbf{q}) = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Inverse Kinematic Problem

Generally the target position and orientation of the end effector frame for an n degree-of-freedom manipulator is given as the matrix $\mathbf{T}_n^t$, where the superscript t designates target pose. The problem of finding the joint variables to achieve this pose is known as the inverse kinematic problem (IKP). In other words, the IKP is the determination of the vector $\mathbf{q}$ composed of the joint variables that will yield the end effector target transformation $\mathbf{T}_n^t$.

The inverse kinematics of a manipulator can be solved by finding the closed-form solution. For example, for a six degree-of-freedom manipulator such as the

Stanford manipulator, the transformation from the base frame to the end effector frame yields a $\mathbf{T}_n$ matrix of the form

$$\mathbf{T}_6 = \mathrm{A}_1\mathrm{A}_2\mathrm{A}_3\mathrm{A}_4\mathrm{A}_5\mathrm{A}_6. \tag{7}$$

Equation 7 is premultiplied six times by the $\mathbf{A}$ matrix inverses where, after each multiplication, an equation is obtained for a joint variable in terms of the other variables [2].

Conversely, the IKP may be solved iteratively until the actual end effector transformation $\mathbf{T}_n^a$ is coincident with the target transformation matrix $\mathbf{T}_n^t$. The next chapter describes a numerical algorithm based on this idea for solving the IKP.

GENERALIZED INVERSE KINEMATICS

Two of the most widely used methods of solving the inverse kinematic problem (IKP) are either analytical or numerical. One of the major drawbacks to solving the IKP analytically is that the solution pertains to only a particular manipulator, in contrast to the numerical solution which can be easily modified to find the joint variables of different manipulators. This chapter presents a numerical scheme used to solve the IKP.

Numerical Solution

As previously explained, the IKP is the determination of a vector $\mathbf{q}$, consisting of the n joint variables, that will yield the end effector target transformation $\mathbf{T}_n^t$. The objective is to minimize the difference between the actual end effector transformation $\mathbf{T}_n^a$ and the target transformation $\mathbf{T}_n^t$ through minimization of the residual functions of position and orientation. Given the target transformation matrix $\mathbf{T}_n^t$ as

$$\mathbf{T}_n^t = \begin{pmatrix} \mathbf{n}^t & \mathbf{o}^t & \mathbf{a}^t & \mathbf{p}^t \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and $\mathbf{T}_n^a$ as

$$\mathbf{T}_n^a = \begin{pmatrix} \mathbf{n}^a & \mathbf{o}^a & \mathbf{a}^a & \mathbf{p}^a \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

then the functions of residual position are defined as

$$\begin{aligned} r_x &= p_x^t - p_x^a \\ r_y &= p_y^t - p_y^a \\ r_z &= p_z^t - p_z^a \end{aligned} \tag{8}$$

and the residual orientation functions are

$$\begin{aligned} r_\phi &= \phi^t - \phi^a \\ r_\theta &= \theta^t - \theta^a \\ r_\psi &= \psi^t - \psi^a \end{aligned} \tag{9}$$

where, from Paul (1981), ϕ, θ, and ψ are defined as

$$\begin{aligned}
\phi &= \begin{cases} 0, & \text{if } a_x \text{ and } a_y = 0; \\ \text{atan2}(|a_y|, |a_x|) + \pi, & \text{if } a_x \text{ and } a_y < 0; \\ \text{atan2}(a_y, a_x), & \text{otherwise,} \end{cases} \\
\theta &= \text{atan2}(a_x \cos\phi + a_y \sin\phi,\ a_z), \\
\psi &= \text{atan2}(-n_x \sin\phi + n_y \cos\phi,\ -o_x \sin\phi + o_y \cos\phi).
\end{aligned} \tag{10}$$

Goldenberg, Benhabib, and Fenton (1985) define the residual function vector $\mathbf{r} = \mathbf{r}(\mathbf{q})$ as

$$\mathbf{r} = (r_x r_y r_z r_\phi r_\theta r_\psi)$$

where $(r_x r_y r_z)$ and $(r_\phi r_\theta r_\psi)$ stand for the residual position and the residual orientation, respectively. The $\mathbf{r}$ vector represents the six independent constraints on the n unknown components of the vector $\mathbf{q}$. The target vector $\mathbf{q}$ is obtained when the residual functions are a minimum and $\mathbf{T}_n^a$ is equal to $\mathbf{T}_n^t$; i.e.,

$$\mathbf{r}(\mathbf{q}) = 0. \tag{11}$$

Properties of Solutions

Existence of Solutions

There are several conditions necessary for which solutions to the IKP exist. If the desired end effector position is outside the work envelope of the manipulator, then a solution clearly does not exist. The *work envelope* of a manipulator is defined as the locus of points in $\mathbf{R}^3$ that can be reached by the manipulator as the joint variables are swept through their respective limits [6].

In addition, even if the desired position is within the work envelope, the required end effector orientation may be such that it causes one or more of the joint variable limits to be exceeded. Goldenberg, Benhabib, and Fenton (1985) suggest a method to prevent from converging to a nonfeasible solution: 1) impose upper and lower limits on the joint variable displacements such that

$$\mathbf{q}^l \leq \mathbf{q} \leq \mathbf{q}^u \tag{12}$$

where $\mathbf{q}^l$ and $\mathbf{q}^u$ designate the lower and upper limits, respectively, checking the solution at each iteration against the limits and correcting those variable values which are out-of-range to their nearest limit; or 2) reduce the iteration step size with a criteria for reduction which is dependent on the numerical method chosen for finding the joint variables.

In general, for an arbitrary end effector position and orientation to be possible, the number of unknowns in $\mathbf{q}$ must be at least equal to the number of independent constraints, or

$$\text{for a general end effector pose} \quad n \geq 6. \tag{13}$$

Equation 13 is a necessary but not sufficient condition for the existence of a solution to the IKP. Also, the end effector position must be within the work envelope of the manipulator, and the desired orientation must be such that none of the joint variable limits are exceeded.

Uniqueness of Solutions [6]

When solutions are obtained, they are usually not unique. For example, a kinematically redundant manipulator, where $n > 6$, can typically have infinitely many solutions because it has more degrees of freedom than necessary to establish an arbitrary end effector pose. Even when the manipulator is not kinematically redundant, there may be times when the solution to the IKP is not unique. Several distinct solutions can arise when the size of the joint-space work envelope is sufficiently large. Figure 6 shows two solutions for a nonredundant robot placing a tool at point p . The two solutions are referred to as the *elbow-up* and *elbow-down* solution. As shown, both solutions give the same pose for the end effector, but their joint space coordinates are clearly distinct.

Algorithm

The residual functions are minimized using a nonlinear least-squares technique based on the Levenberg-Marquardt method. The least-squares algorithm is part of the IMSL Library of Mathematical Routines and is labeled DUNLSF. Figure 7

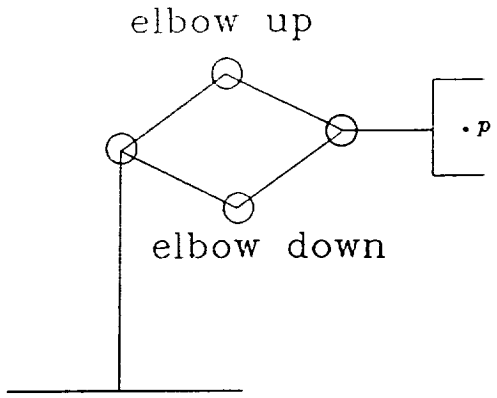


Figure 6. Example showing the multiple solutions for a nonredundant robot [6].

shows a flowchart of the algorithm to solve a general inverse kinematic problem. The first step (box 1) is to determine the link parameters for the manipulator. Next, input the target position and orientation of the manipulator end effector frame as data arrays POS(3) and EULER(3), as shown in box 2. In box 3, input an initial guess of the joint coordinates in the array labeled XGUESS. In box 4, initialize the parameters required by the IMSL software (LDFJAC,M,N,IPARAM). Next, a subroutine labeled FORWARD, which is the driver for DUNLSF, is called by the main program to find the joint coordinates (box 5). FORWARD contains the nonlinear r functions to be minimized, and DUNLSF performs the minimization process. Figure 8 shows a flowchart for subroutine FORWARD. When a solution is obtained, the program verifies whether the joint limits have been exceeded using a user-defined subroutine called CONSTRAINT (box 6). If one or more joint coordinates exceeds their respective limits, this subroutine replaces it with its nearest limit. Then the iteration process begins again. Once all of the joint coordinates have been found, the results are printed (box 7).

The next two sections present examples of using the algorithm.

Stanford Manipulator

The analytical solution is presented and its results will be compared to the results obtained numerically.

Analytical Solution

A sketch of the Stanford Manipulator is shown in Figure 9 with coordinate frames assigned to the links. As shown, the manipulator consists of five revolute joints with joint variables θ_1, θ_2, θ_4, θ_5, and θ_6, and a prismatic joint with variable d_3. The link parameters are shown in Table 1.

Table 1. Link parameters for the Stanford Manipulator [2].

Link	Variable	α	a	d
1	θ_1	$-90°$	0	0
2	θ_2	$90°$	0	d_2
3	d_3	$0°$	0	d_3
4	θ_4	$-90°$	0	0
5	θ_5	$90°$	0	0
6	θ_6	$0°$	0	0

The following abbreviations will be used for the sine and cosine of the angle θ

$$\sin\theta_i = \mathrm{S}_i$$

$$\cos\theta_i = \mathrm{C}_i$$

$$\sin(\theta_i + \theta_j) = \mathrm{S}_{ij}$$

$$\cos(\theta_i + \theta_j) = \mathrm{C}_{ij}.$$

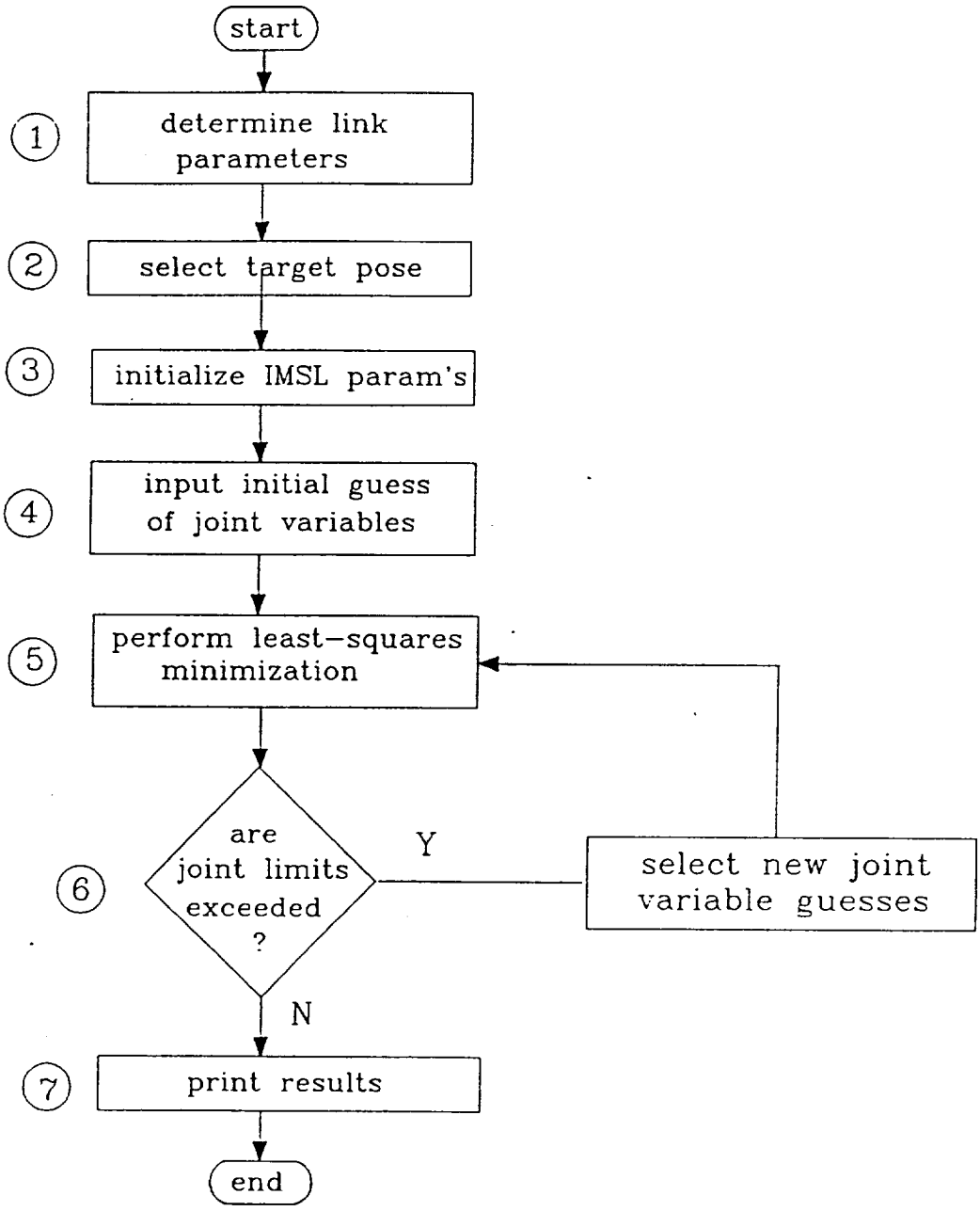


Figure 7. Flowchart for the inverse kinematics algorithm.

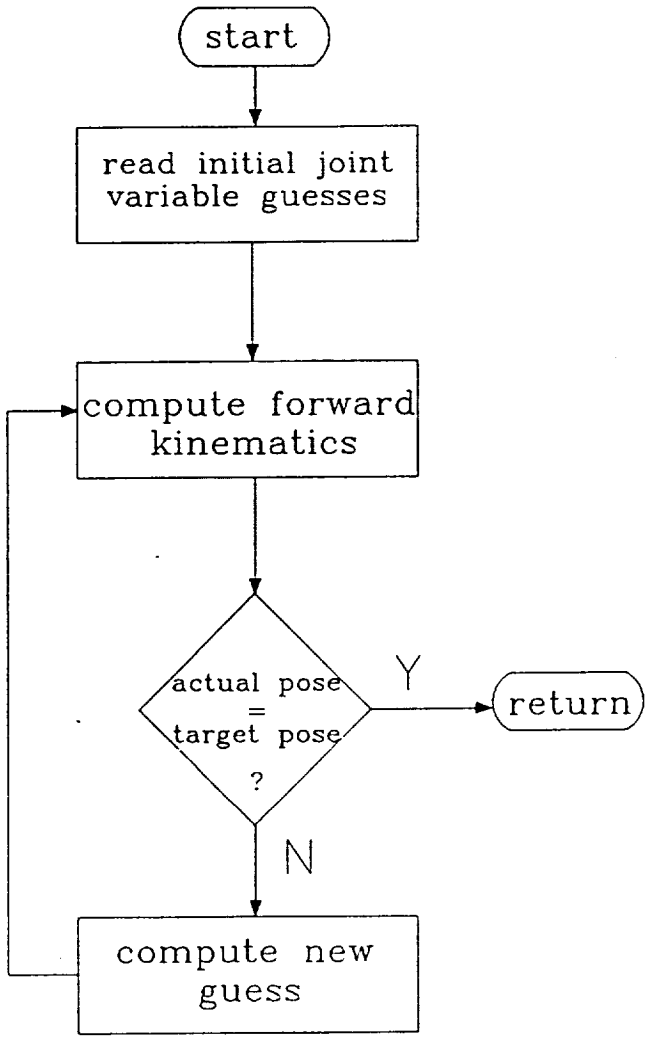


Figure 8. Flowchart of subroutine to minimize functions.

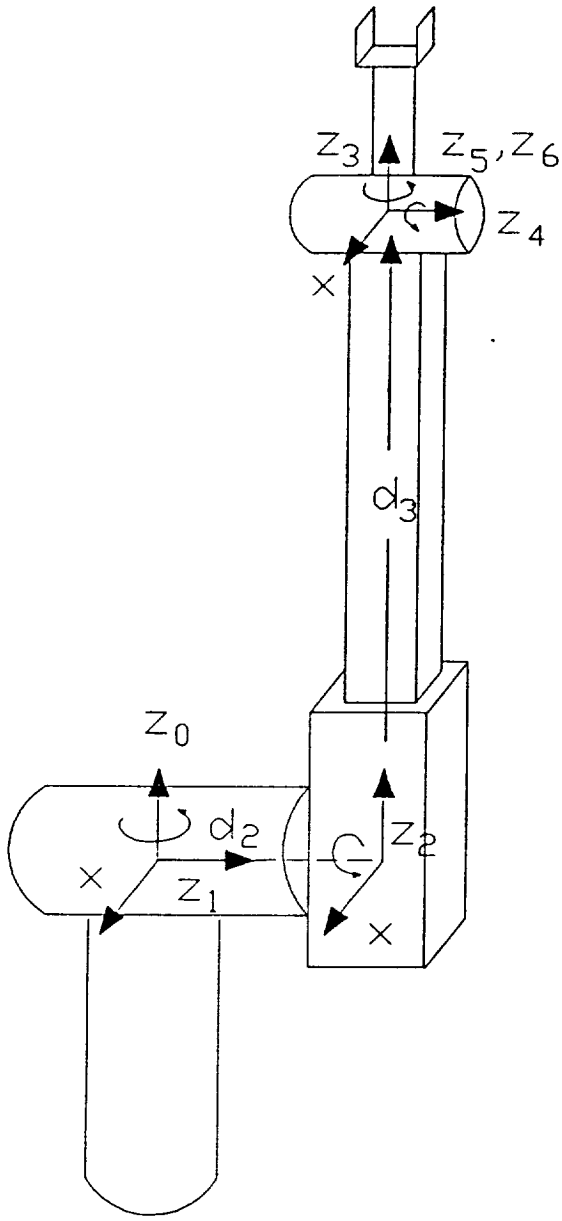


Figure 9. The Stanford Manipulator [2].

Again, the target pose of the end effector frame relative to the base frame is given by equation 7, and is rewritten below

$$\mathbf{T}_6^t = \mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\mathbf{A}_4\mathbf{A}_5\mathbf{A}_6. \tag{14}$$

The procedure Paul uses to solve for the joint variables is to obtain six matrix equations by successively multiplying equation 14 by the **A** matrix inverses. After each multiplication, an equation relating a joint variable to the remaining joint variables is obtained. Paul's solution will be summarized below.

Given the target pose

$$\mathbf{T}_6^t = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the joint coordinate θ_1 is

$$\theta_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right) - \tan^{-1}\frac{d_2}{\pm\sqrt{r^2 - {d_2}^2}}$$

where $+$ is for a right-hand coordinate system and $-$ is for a left-hand coordinate system and

$$\begin{aligned} p_x &= r\ \cos\phi \\ p_y &= r\ \sin\phi \\ r &= +\sqrt{{p_x}^2 + {p_y}^2} \\ \phi &= \tan^{-1}\left(\frac{p_y}{p_x}\right). \end{aligned}$$

These trigonometric substitutions for p_x and p_y are necessary because the equation used to obtain θ_1 is of the form

$$-\mathrm{S}_1 p_x + \mathrm{C}_1 p_y = d_2.$$

The remaining joint coordinates are

$$\begin{aligned} \theta_2 &= \tan^{-1}\frac{\mathrm{C}_1 p_x + \mathrm{S}_1 p_y}{p_z} \\ d_3 &= \mathrm{S}_2(\mathrm{C}_1 p_x + \mathrm{S}_1 p_y) + \mathrm{C}_2 p_z \end{aligned}$$

$$\theta_4 = \tan^{-1} \frac{-S_1 a_x + C_1 a_y}{C_2(C_1 a_x + S_1 a_y) - S_2 a_z} \quad \text{if } \theta_5 > 0$$

or

$$\theta_4 = \theta_4 + \pi \quad \text{if } \theta_5 < 0.$$

For $\theta_5 = 0$, the manipulator becomes degenerate with both the axes of joint 4 and joint 6 aligned. Then, θ_4 can be assigned any value and is usually assigned a value which will result in an overall manipulator configuration which is *close* to the previous configuration in terms of total distance traveled by the manipulator to achieve the new pose. The equation for θ_5 is

$$\theta_5 = \tan^{-1} \frac{C_4[C_2(C_1 a_x + S_1 a_y) - S_2 a_z] + S_4[-S_1 a_x + C_1 a_y]}{S_2(C_1 a_x + S_1 a_y) + C_2 a_z}$$

and finally, θ_6 is

$$\theta_6 = \tan^{-1} \frac{S_6}{C_6}$$

where

$$S_6 = -C_5\{C_4[C_2(C_1 o_x + S_1 o_y) - S_2 o_z] + S_4[-S_1 o_x + C_1 o_y]\}$$

$$+S_5\{S_2(C_1 o_x + S_1 o_y) + C_2 o_z\}$$

$$C_6 = -S_4[C_2(C_1 o_x + S_1 o_y) - S_2 o_z] + C_4[-S_1 o_x + C_1 o_y].$$

Numerical Solution

The left-hand-side of equation 14 is determined using the known target end effector pose with equation 3. Because this manipulator has six degrees-of-freedom, both the position and orientation required for the end effector can be specified. In equation 14, the product of the **A** matrices on the right is the actual transformation matrix $\mathbf{T}_6^a$. These matrices are unknown because the joint variables are unknown.

Example

Given an arbitrary Euler angle set of

$$\phi = 45^\circ$$

$$\theta = 17.6^\circ$$

$$\psi = 27.2^\circ,$$

a position vector with values

$$p_x = 25.1 \text{ inches}$$

$$p_y = 32.7 \text{ inches}$$

$$p_z = 22.6 \text{ inches},$$

and $d_2 = 8$ inches, then the target transformation matrix is

$$\mathbf{T}_6^t = \begin{bmatrix} 0.27626 & -0.93700 & 0.21381 & 25.1 \\ 0.92269 & 0.32083 & 0.21381 & 32.7 \\ -0.26893 & 0.13821 & 0.95319 & 22.6 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Imposing the joint constraints listed below

$$-\pi \leq \theta_1 \leq \pi$$

$$-\pi/2 \leq \theta_2 \leq \pi/2$$

$$d_3 \geq 0$$

$$-\pi \leq \theta_4 \leq \pi$$

$$-\pi/2 \leq \theta_5 \leq \pi/2$$

$$-\pi \leq \theta_6 \leq 0$$

yields the results shown in Table 2, which compares the resulting joint variables for the given $\mathbf{T}_6^t$ matrix using both the analytical and numerical methods. As Table 2 shows, the results for all joint variables using both methods are equal.

Table 2. Analytical versus numerical results for the Stanford Manipulator.

Link	Variable	Analytical Result	Numerical Result
1	θ_1	0.7208 rad	0.7208 rad
2	θ_2	1.0612 rad	1.0612 rad
3	d_3	46.326 in	46.326 in
4	θ_4	3.113 rad	3.113 rad
5	θ_5	0.7548 rad	0.7548 rad
6	θ_6	-2.585 rad	-2.585 rad

In summary, the analytical and numerical solutions for a six degree–of–freedom manipulator have been shown. The numerical solution is a more simple approach and equations are easily obtained using the concept of homogeneous transformations. By selecting appropriate joint variable limits, the solution will converge to the analytical solution.

Dexterous Hand

To solve the inverse kinematics of a dexterous hand, each finger is treated as an independent manipulator. A coordinate frame for the palm, or the *palm frame*, can be selected to serve as the reference frame from which all measurements are made. As a case study, the University of Minnesota hand, shown in Figure 10, is used.

Analytical Solution

The closed–form solution is presented in Koehler and Donath (1988). They assume that the desired location of the fingertips, with respect to the reference frame, are contact points on the object that have been determined based on task constraints; e.g., to provide a stable grasp [3]. Since the fingers have only three degrees-of-freedom, only the required fingertip position (or orientation, but not both) can be specified. Given these three locations, the objective is to find the finger joint angles.

Numerical Solution

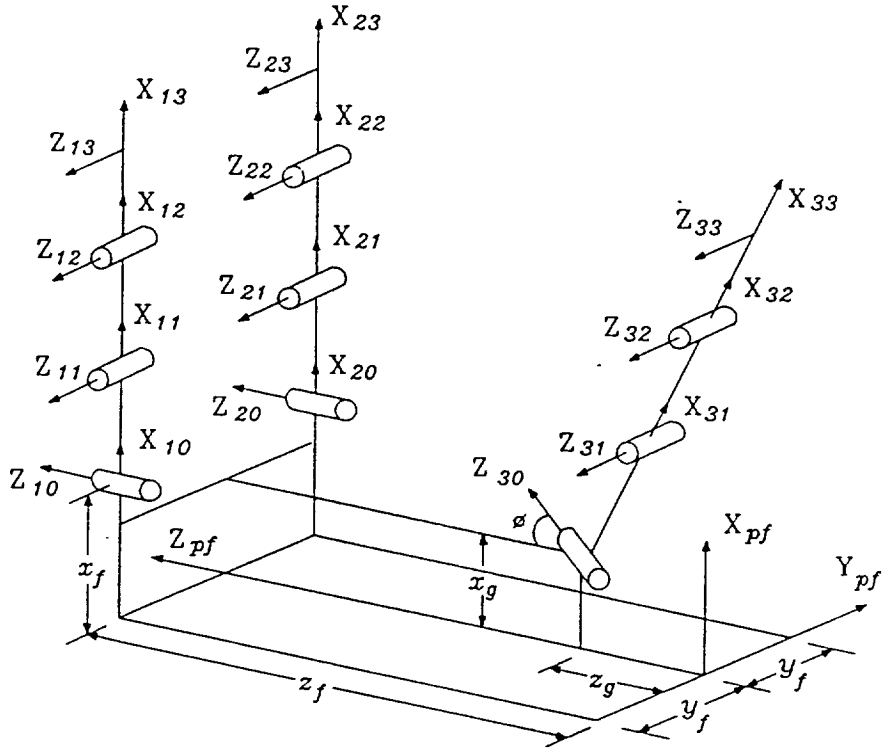
An equation is necessary which relates the desired fingertip positions to the palm frame. As in the analytical solution, the desired fingertip positions are selected contact points on the targeted object.

The transformation from the palm frame to the ith contact point, $\mathbf{T}_{pf}^{c_i}$, is illustrated in Figure 11 and is given by

$$\mathbf{T}_{pf}^{c_i} = \mathbf{T}_{pf}^{fb_i}\, \mathbf{T}_{fb_i}^{c_i} \tag{15}$$

where $\mathbf{T}_{pf}^{fb_i}$ is the transformation from the palm frame to the base frame of finger i, and $\mathbf{T}_{fb_i}^{c_i}$ is the transformation from the base frame of finger i to the contact point corresponding to finger i. The transformation $\mathbf{T}_{fb_i}^{c_i}$ is the product

$$\mathbf{T}_{fb_i}^{c_i} = \prod_{j=1}^{n} \mathbf{A}_j \tag{16}$$



where $\varnothing = 45^{\circ}$. The first subscript designates finger number, the second designates link number.

Figure 10. Coordinate frames for the University of Minnesota Hand [3].

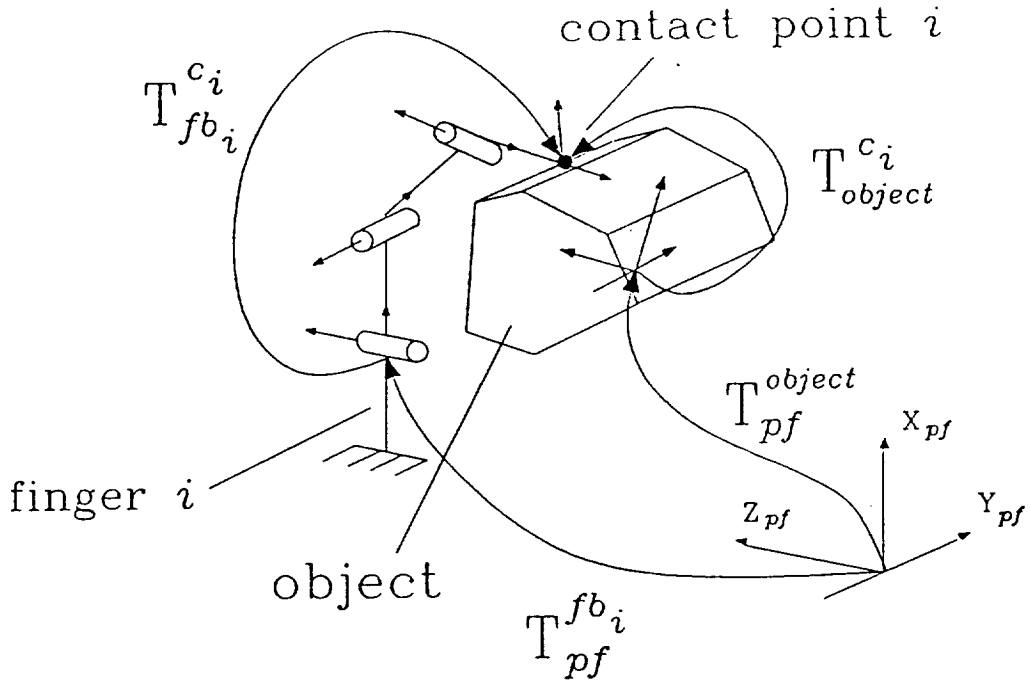


Figure 11. Transformation from the palm frame to the contact point.

where n designates the degrees-of-freedom of finger i.

Another equation for $\mathbf{T}^{c_i}_{pf}$ is

$$\mathbf{T}^{c_i}_{pf} = \mathbf{T}^{object}_{pf}\ \mathbf{T}^{c_i}_{object} \tag{17}$$

and is also illustrated in Figure 11. The transformations given in equation 17 are known assuming the pose of the object frame relative to the palm frame is known and contact points on the object, one for each finger, have been specified. By equating (15) to (17), the equation for finding the joint variables of each finger is

$$\mathbf{T}^{fb_i}_{pf}\ \mathbf{T}^{c_i}_{fb_i} = \mathbf{T}^{object}_{pf}\ \mathbf{T}^{c_i}_{object}. \tag{18}$$

The objective is then to minimize the difference between both sides of the equation above.

Example

If the target object is a rectangular bar with the base frame aligned to the palm frame as shown in Figure 12, the matrix which describes the transformation from palm frame to the bar frame $\mathbf{T}_{pf}^{bar}$ is

$$\mathbf{T}_{pf}^{bar} = \text{Trans}(2,0,2) = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

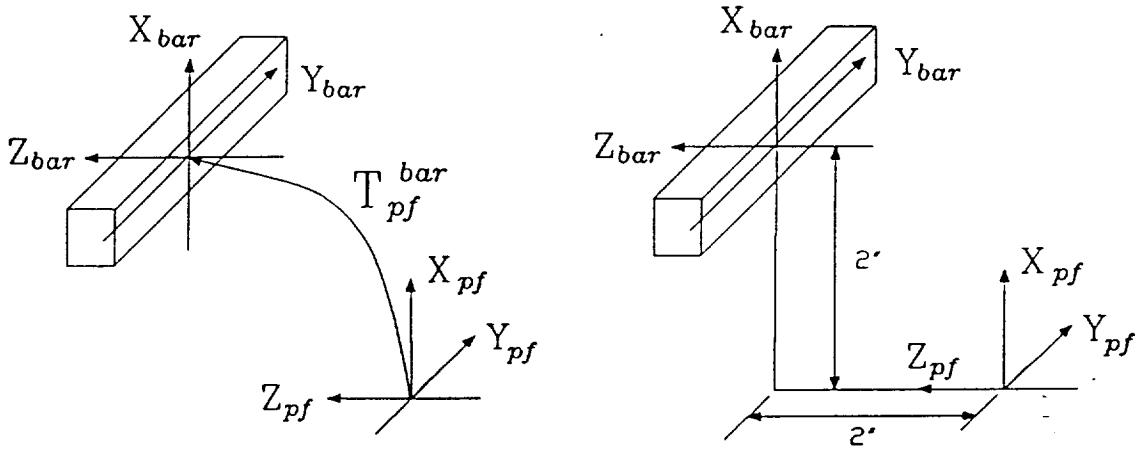


Figure 12. Bar frame with respect to the palm frame.

Selecting contact points on the bar as shown in Figure 13, the transformations

from the bar frame to the each contact point, $\mathbf{T}_{bar}^{c_i}$, are

$$\mathbf{T}_{bar}^{c_1} = \text{Trans}(0, -1, .5) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & .5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{bar}^{c_2} = \text{Trans}(0, 1, .5) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & .5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{bar}^{c_3} = \text{Trans}(0, 0, -.5) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

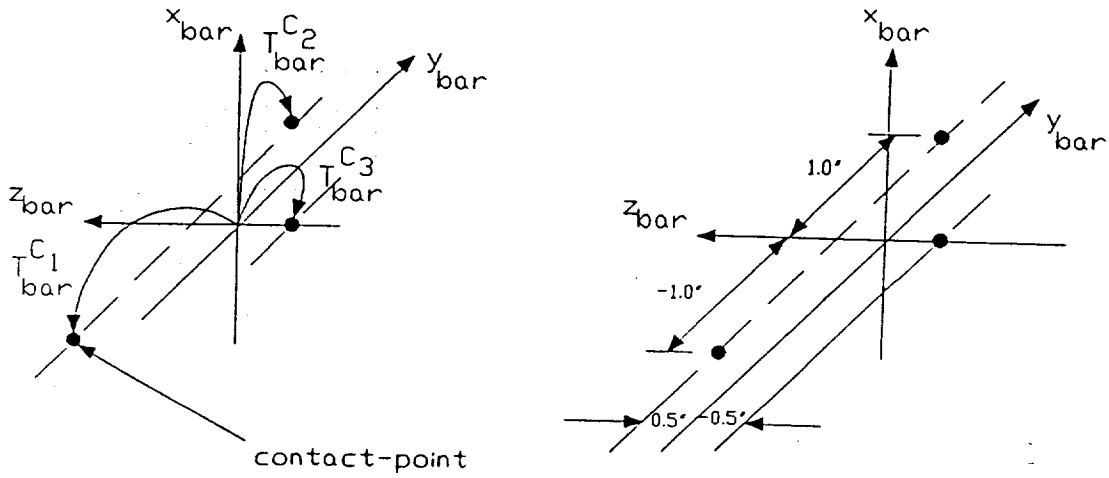


Figure 13. Finger contact points.

Figure 14 gives a graphical representation of the link parameters and Table 3 lists the D–H parameters for the fingers. Note that all of the fingers have been chosen

Table 3. Link parameters for the Minnesota hand.

Link	Variable	α	a	d
1	θ_1	90°	1"	0
2	θ_2	0°	1"	0
3	θ_3	0°	1"	0

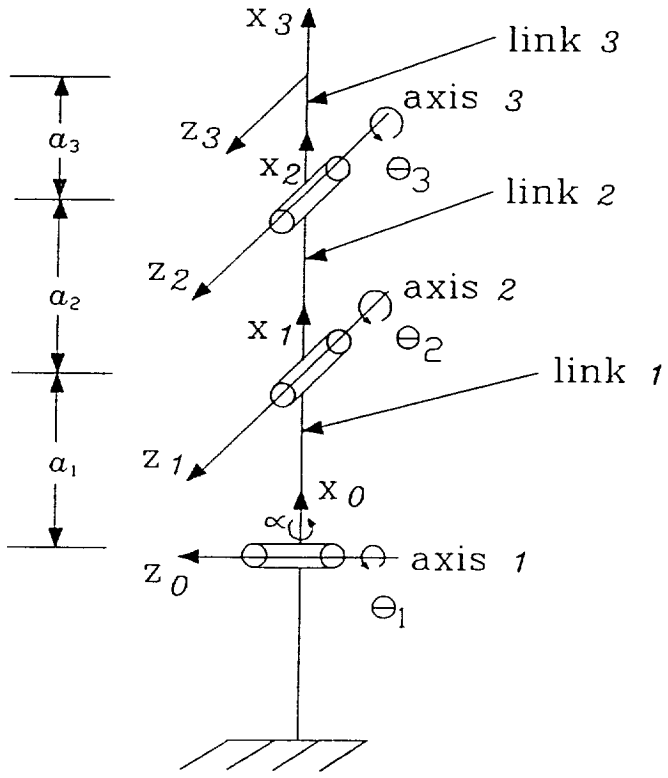


Figure 14. Link parameters for a finger on the Minnesota hand.

to have the same D–H parameters.

Using these parameters, the A matrix for each link of a finger can be determined

Table 4. Assumed values for the dexterous hand.

Variable	Value
x_f	1"
y_f	1"
z_f	4"
x_g	1"
z_g	1"
ϕ	45°

using equation 4. For example, the **A** matrix for link 1 is

$$\mathbf{A}_1 = \mathrm{Rot}(z_0,\theta_1)\mathrm{Trans}(0,0,0)\mathrm{Trans}(a_1,0,0)\mathrm{Rot}(x_0,90^\circ)$$
$$= \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & a_1\sin\theta_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Next the transformation from the palm frame to the base frame of each finger $\mathbf{T}_{pf}^{fb_{1,2,3}}$ is determined using Figure 10. For fingers 1 and 2, this transformation is the result of a translation from the palm frame to the finger base frame (0 link frame) a distance x_f along the x–axis, $\pm y_f$ along the y–axis, and z_f along the z–axis. Mathematically, this is expressed as

$$\mathbf{T}_{pf}^{fb_{1,2}} = \mathrm{Trans}(x_f, \pm y_f, z_f)$$
$$= \begin{bmatrix} 1 & 0 & 0 & x_f \\ 0 & 1 & 0 & \pm y_f \\ 0 & 0 & 1 & z_f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $-y_f$ corresponds to finger 1, and $+y_f$ corresponds to finger 2. For finger 3, the transformation is the result of a translation x_g along the x–axis, a translation z_g along the z–axis, and a rotation ϕ about the y–axis. This is expressed as

$$\mathbf{T}_{pf}^{fb_3} = \mathrm{Trans}(x_g, 0, z_g)\mathrm{Rot}(y,\phi)$$
$$= \begin{bmatrix} \cos\phi & 0 & \sin\phi & x_g \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & z_g \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The values for x_g, z_g, x_f, y_f, z_f, and ϕ are listed in Table 4. Finally, impose the

joint limits listed below: for fingers 1 and 2

$$-\pi/6 \leq \theta_1 \leq \pi/6$$

$$-\pi/3 \leq \theta_2 \leq 0$$

$$-\pi/2 \leq \theta_3 \leq 0$$

and for finger 3

$$-\pi/6 \leq \theta_1 \leq \pi/6$$

$$0 \leq \theta_2 \leq \pi/3$$

$$0 \leq \theta_3 \leq \pi/2$$

The resulting hand configuration is shown in Figure 15.

Discussion

A numerical scheme was presented to solve the inverse kinematic problem for a general manipulator and adopted to a dexterous hand. The scheme was based on minimization of the residual functions of position and orientation. As examples, the inverse kinematics for the Stanford Manipulator and the University of Minnesota Hand were determined.

A benefit of using a numerical scheme versus an analytical method is that it allows quick modification to conform to the desired hand. The scheme may be modified to correspond to most recently developed dexterous hands; i.e., that have revolute or prismatic joints, and fingers that are open kinematic chains. The routine can be modified by changing the number of fingers, the number of links per finger (not counting link 0), the D–H parameters, and the joint variable constraints. Contact points on the object corresponding to each finger must also be defined. Depending on the number of degrees-of-freedom for the fingers will determine the number of residual functions that are minimized.

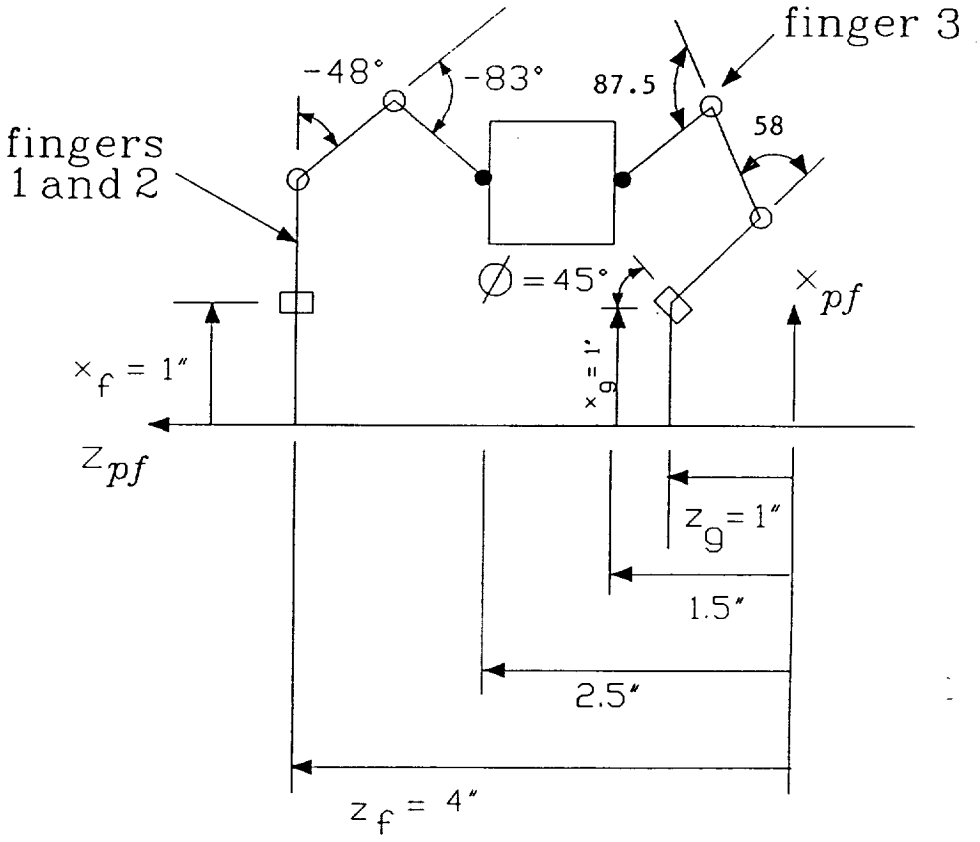


Figure 15. Resulting hand configuration.

POSE ESTIMATION OF A PRE-GRASPED OBJECT

This chapter presents a method of calculating the pre-grasp pose of an object using data from local, noncontact sensors. *Local* refers to sensors that are mounted on the end effector and for the case of dexterous hands, the sensors are mounted on the fingers. *Noncontact* sensors are sensors that do not require physical contact to operate, such as proximity sensors.

The pose estimation problem is divided into two subtasks: the forward model and the inverse model. The *forward model* is analogous to the forward kinematic solution of a manipulator where the joint coordinates are known and the question is to find the end effector pose. The forward model of the pose estimation problem assumes the object pose and finger joint angles are known and solves for the sensor parameters. The *inverse model* assumes sensory data is available, the finger joint angles are known, and uses this information to find the object pose. Results from the forward model will enable verification of the inverse model.

A discussion on a real-world sensor follows which exhibits the mathematical equations for a fiber optic reflectance sensor. Due to the complexity of the real-world model, an ideal sensor is defined and used in the pose estimation algorithm.

Pose Estimation Analysis

Forward Model

The forward model determines the sensor parameters based on a known object pose. The sensor parameters to determine are: r, the radial (sensor-to-trigger point) distance, and the angles the trigger point may be offset from the sensor axes.

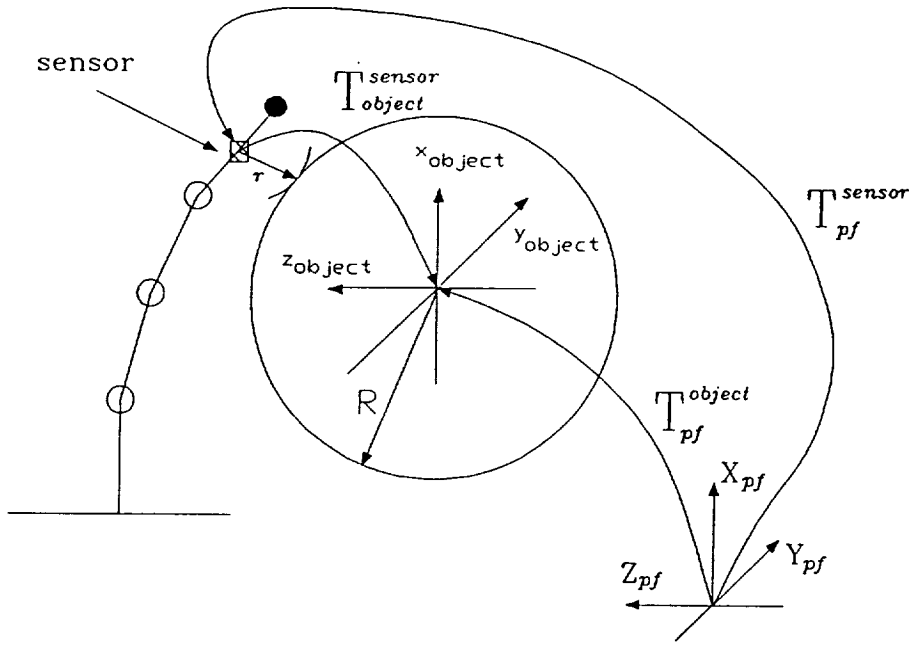


Figure 16. Transformations from the palm frame to the sensor frame.

Using Figure 16 the following equation can be written

$$\mathbf{T}^{sensor}_{pf}\mathbf{T}^{object}_{sensor} = \mathbf{T}^{object}_{pf}$$

which becomes

$$\mathbf{T}^{object}_{sensor} = \mathbf{T}^{pf}_{sensor}\mathbf{T}^{object}_{pf}. \tag{19}$$

In equation 19, $\mathbf{T}^{object}_{sensor}$, the transformation from the sensor frame to the object frame, is unknown because r and the angular offsets are not known. For example, if the ideal

sensor model is used (this model will be discussed in the next section) the offsets γ and β are unknowns in the $\mathbf{T}_{sensor}^{object}$ matrices.

$\mathbf{T}_{sensor}^{pf}$ is the transformation from the sensor frame to the palm frame given by

$$\mathbf{T}_{sensor}^{pf} = [\mathbf{T}_{pf}^{fb}\mathbf{T}_{fb}^{link}\mathbf{T}_{link}^{sensor}]^{-1}. \tag{20}$$

$\mathbf{T}_{pf}^{fb}$ is the transformation from the palm frame to the finger base frame, $\mathbf{T}_{fb}^{link}$ is the transformation from the finger base frame to the link frame, and $\mathbf{T}_{link}^{sensor}$ is the transformation from the link frame to the sensor frame. These transformations are known because the finger joint angles and the sensor locations are known.

The transformation from the palm frame to the object frame, $\mathbf{T}_{pf}^{object}$, is

$$\mathbf{T}_{pf}^{object} = \text{Trans}(X_O, Y_O, Z_O)\ \text{Euler}(\phi, \theta, \psi).$$

This matrix is known because the position and orientation of the object are known.

Inverse Model

The inverse model determines the object pose given the sensor data. Using a scheme based on triangulation, which is a method used to optically locate points in space [15], the necessary equations are developed. From Figure 17 the following transformation equations are written

$$\mathbf{T}_{pf}^{object} = \mathbf{T}_{pf}^{sensor_1}\mathbf{T}_{sensor_1}^{object} = \ldots = \mathbf{T}_{pf}^{sensor_i}\mathbf{T}_{sensor_i}^{object}$$

for i sensors. It then follows that

$$\begin{aligned}
\mathbf{T}_{pf}^{sensor_1}\mathbf{T}_{sensor_1}^{object} &= \mathbf{T}_{pf}^{sensor_2}\mathbf{T}_{sensor_2}^{object} \\
\mathbf{T}_{pf}^{sensor_1}\mathbf{T}_{sensor_1}^{object} &= \mathbf{T}_{pf}^{sensor_3}\mathbf{T}_{sensor_3}^{object} \qquad (21) \\
&\vdots \\
\mathbf{T}_{pf}^{sensor_1}\mathbf{T}_{sensor_1}^{object} &= \mathbf{T}_{pf}^{sensor_i}\mathbf{T}_{sensor_i}^{object}.
\end{aligned}$$

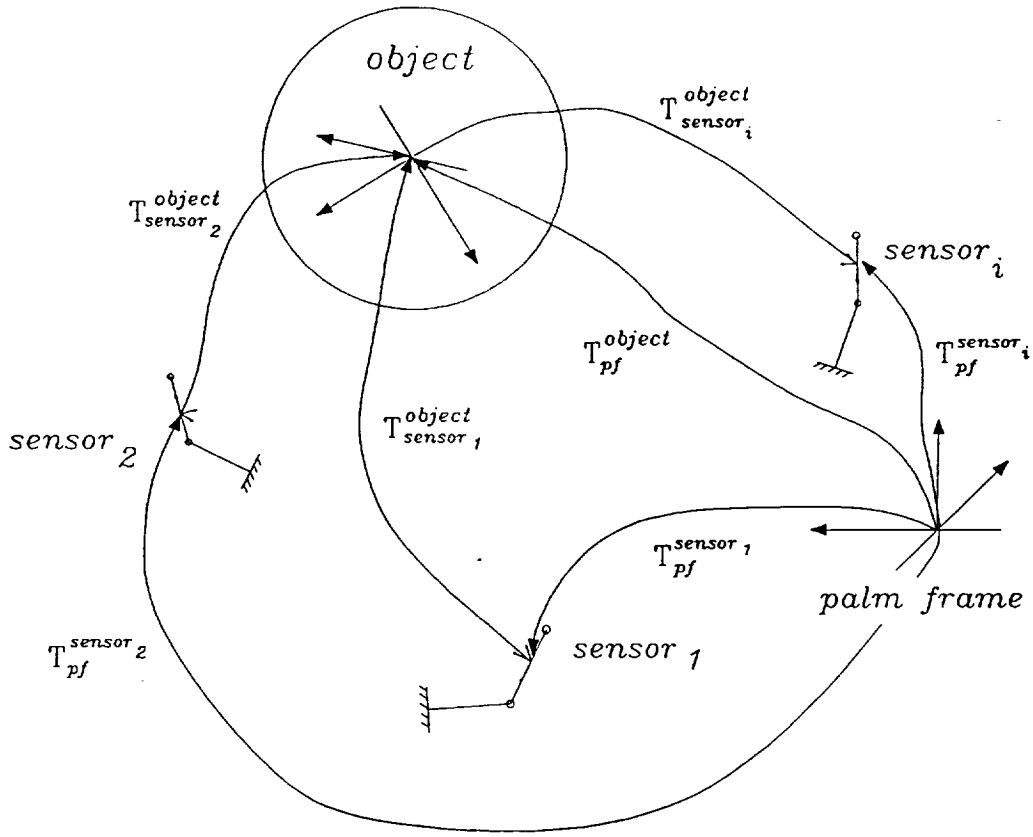


Figure 17. Triangulation scheme.

The known and unknowns in the matrices above are: the $\mathbf{T}^{sensor}_{pf}$ matrices are known because the inverse kinematics are known, r (sensor-to-trigger point distance) for each sensor is known because the sensors have been triggered. However, the angular offsets are not known. Again, if the sensors used are ideal sensors, then for each equation in (21) there are four unknowns (γ and β are unknown on both sides). Therefore *at least two equations must be available*; i.e., at least three individual sensors must trigger to obtain six equations with six unknowns with the assumption that only the object position is desired but not the object orientation.

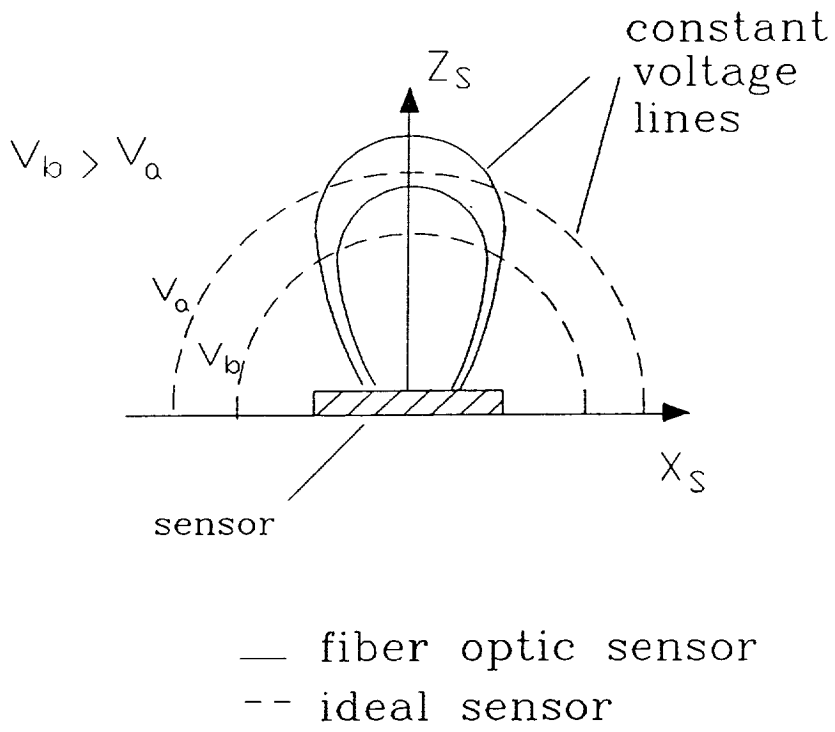


Figure 18. Response curves for an optical reflectance sensor and an ideal sensor.

Noncontact Sensor Models

Fiber Optic Reflectance Sensor

There are many different types of noncontact sensors available, each corresponding to a particular need. Inherent to each sensor is the type of output that it gives. For example, the typical output curve for an optical reflectance sensor is shown in Figure 18. This is compared to an ideal sensor, which will be discussed in the next section. The output response of an optical reflectance sensor represents diffused light intensity from the object to the receiver, which is dependent on the distance, the

photometric properties of the object surface, and the local orientation of the object [17].

To realize the complexity of modeling a sensor, Balaure (1986) presents the mathematical equations which are described below for the fiber optic reflectance sensor shown in Figure 19. The output from the sensor will depend on parameters: h, α, φ, δ, the slant angle of the object, the whiteness of surface of the object, and the type of optics used. The received flux $\Phi(h,x,y)$ is expressed as

$$\Phi(h,x,y) = \frac{L_o}{4}\int\int_D \frac{A(x,y)\ B(x,y)}{(h^2+u^2)\ (h^2+v^2)}dx\ dy$$

The variable L_o is known as the source luminance. The functions $A(x,y)$ and $B(x,y)$ are defined as

$$A(x,y) = [\cos(2\arctan\frac{u}{h} - \alpha) + \cos\alpha][\cos(2\arctan\frac{v}{h}\varphi) + \cos\alpha]$$
$$B(x,y) = [\cos(\arctan\frac{u}{h} - \alpha)][\cos(\arctan\frac{v}{h} - \alpha)]$$

where

$$u = \sqrt{(\frac{\delta^2}{2}+x)^2+y^2}$$
$$v = \sqrt{(\frac{\delta^2}{2}-x)^2+y^2}$$

Of more concern is the detection range of the sensor. The integration area, or detection range, of the fiber optic sensor is D, the surface located within the intersection of two ellipses. The equations of the ellipses are:

$$y^2 + x^2(\cos^2\alpha + \tan^2\varphi\sin^2\alpha) + x[-\delta\cos^2\alpha + h\sin^2\alpha + (\delta\sin^2\alpha$$
$$+h\sin^2\alpha)\tan^2\varphi] + (\frac{\delta}{2}\cos\alpha - h\sin\alpha)^2 - \tan^2\varphi(\frac{\delta}{2}\sin\alpha + h\cos\alpha)^2 = 0$$

and

$$y^2 + x^2(\cos^2\alpha + \tan^2\varphi\sin^2\alpha) + [\delta\cos^2\alpha - h\sin^2\alpha - (\delta\sin^2\alpha +$$
$$h\sin^2\alpha)\tan^2\varphi] + (-\frac{\delta}{2}\cos\alpha + h\sin\alpha)^2 - \tan^2\varphi(\frac{\delta}{2}\sin\alpha + h\cos\alpha)^2 = 0.$$

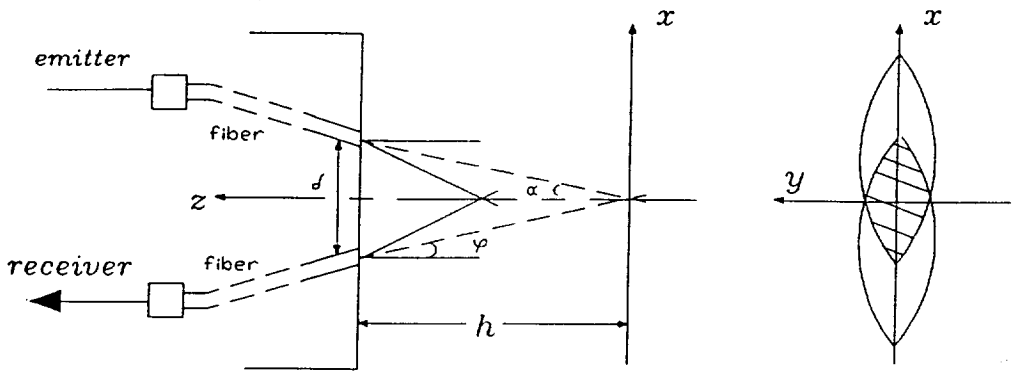


Figure 19. Fiber optic reflectance sensor [13].

Ideal Sensor

An *ideal sensor* is defined with a spherical detection volume instead of a "tear-shaped" volume as for the fiber optic sensor (Figure 18). This sensor is possible if the emitter and receiver lie exactly on top of each other [12]. Another simplification is that the target object is chosen as a sphere. Later, a method of implementing the procedure for other types of objects, such as cylinders or cubes, is discussed.

Using Figure 20, the transformation from the sensor frame to the object is

$$\begin{aligned}\mathrm{Sph}(\gamma,\beta,r+R) &= \mathrm{Rot}(Z_S,\gamma)\mathrm{Rot}(Y_S,\beta)\mathrm{Trans}(Z_S,r+R)\\ &= \begin{bmatrix} \cos\gamma\ \cos\beta & -\sin\gamma & \cos\gamma\ \sin\beta & (r+R)\cos\gamma\ \sin\beta \\ \sin\gamma\ \cos\beta & \cos\gamma & \sin\gamma\ \sin\beta & (r+R)\sin\gamma\ \sin\beta \\ -\sin\beta & 0 & \cos\beta & (r+R)\cos\beta \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (22)$$

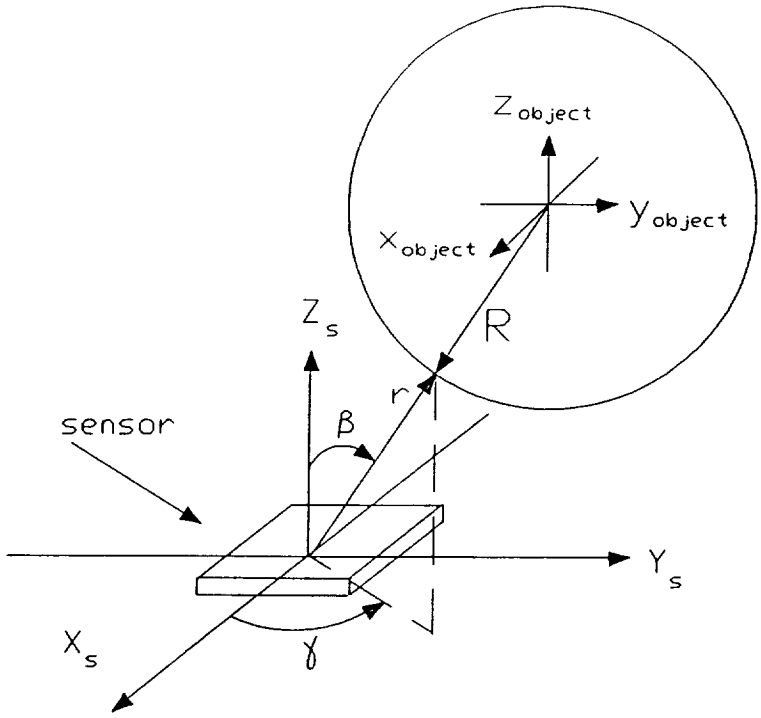


Figure 20. Parameters for the ideal sensor.

where r is the sensor-to-trigger point distance, R is radius of the sphere, γ is the offset angle in the X_S–Y_S plane, and β is the offset angle from the Z_S axis.

Examples

Forward Model

To find the sensor parameters r, γ, and β of the ideal sensor, it is necessary to calculate the transformations required by equation 19, which is rewritten below

$$\mathbf{T}_{sensor}^{object} = \mathbf{T}_{sensor}^{pf}\mathbf{T}_{pf}^{object}.$$

Figure 21 shows one finger of the University of Minnesota hand supplied with an ideal sensor in close proximity to a sphere. The following matrices can then be computed

$$\mathbf{T}_{pf}^{sensor} = \begin{bmatrix} 1 & 0 & 0 & 1.5 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

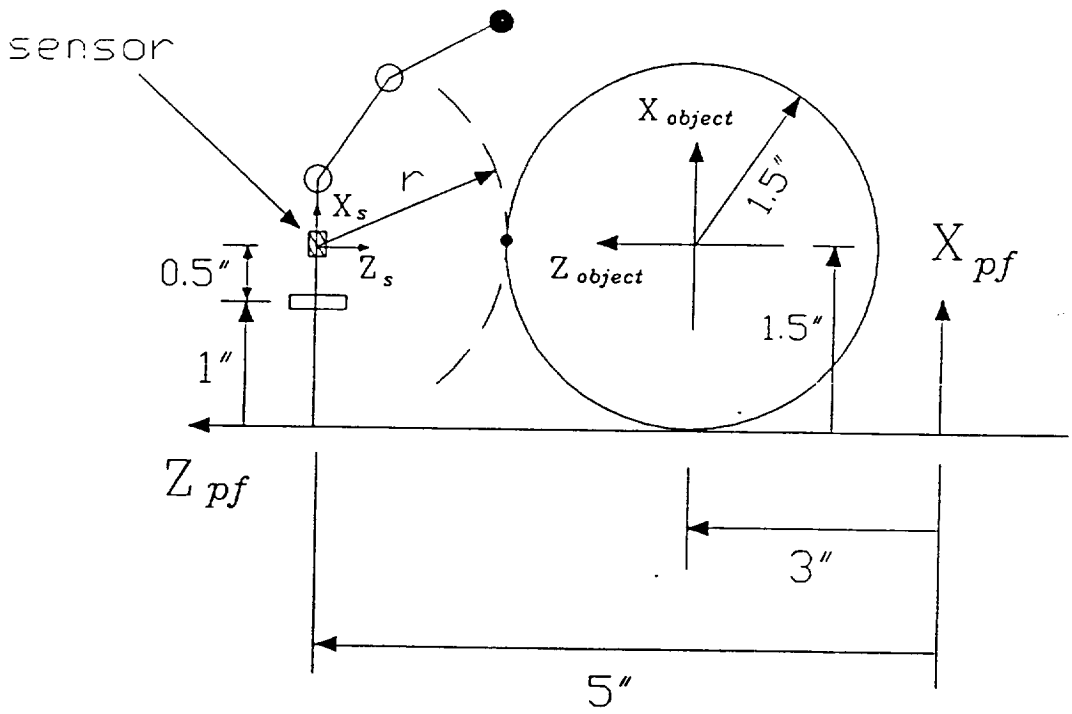


Figure 21. Forward model example using one finger of a dexterous hand.

and the inverse of the matrix above is

$$\mathbf{T}^{pf}_{sensor} = \begin{bmatrix} 1 & 0 & 0 & -1.5 \\ 0 & 0 & 1 & -5 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The transformation from the palm frame to the object frame is

$$\mathbf{T}^{object}_{pf} = \begin{bmatrix} 1 & 0 & 0 & 1.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and the product $\mathbf{T}_{sensor}^{pf}\mathbf{T}_{pf}^{object}$ is

$$\mathbf{T}_{sensor}^{pf}\mathbf{T}_{pf}^{object} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The last column of the matrix above is the position vector, as shown below

$$\mathbf{T}_{sensor}^{pf}\mathbf{T}_{pf}^{object} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -2 \\ 0 \\ 1 \end{bmatrix}. \tag{23}$$

The transformation $\mathbf{T}_{sensor}^{object}$ is

$$\mathbf{T}_{sensor}^{object} = \mathbf{R} \cdot \mathrm{Sph}(\gamma, \beta, r + R) \tag{24}$$

where $\mathbf{R}$ is a matrix which reorients the x, y, z frame at the sensor to the X_S, Y_S, Z_S frame required to find the sensor parameters. The matrix $\mathbf{R}$ is then

$$\mathbf{R} = Rot(x, 90^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{25}$$

Equation 24 becomes

$$\mathbf{T}_{sensor}^{object} = \begin{bmatrix} \cos\gamma\ \cos\beta & -\sin\gamma & \cos\gamma\ \sin\beta & (r+R)\cos\gamma\ \sin\beta \\ \sin\beta & 0 & -\cos\beta & -(r+R)\cos\beta \\ \sin\gamma\ \cos\beta & \cos\gamma & \sin\gamma\ \sin\beta & (r+R)\sin\gamma\ \sin\beta \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{26}$$

Equate the position vector of the matrix in equation 26 to equation 23 results in three equations with three unknowns

$$(r + R)\cos\gamma\ \sin\beta = 0 \tag{27}$$

$$-(r + R)\cos\beta = -2 \tag{28}$$

$$(r + R)\sin\gamma\ \sin\beta = 0. \tag{29}$$

The solution of equations 27–29 yield $\beta = 0^\circ$, $\gamma = 0^\circ$, and

$$r = \frac{2}{\cos\beta} - R = 2 - R = 2 - 1.5 = 0.5 \text{ inches.}$$

This procedure for finding r, γ, and β for a general three-dimensional case can be programmed using a nonlinear least-squares algorithm.

Inverse Model

Figure 22 shows two fingers of a dexterous hand surrounding a circle. Recall that at least three sensors must trigger for this scheme to work if using ideal sensors.

The three sensors used to demonstrate the procedure are a sensor from the first link and a sensor from the second link of finger 1, and a sensor from the first link of finger 2. Thus equation 21 for this example becomes

$$\begin{aligned} \mathbf{T}_{pf}^{sensor_{11}} \mathbf{T}_{sensor_{11}}^{object} &= \mathbf{T}_{pf}^{sensor_{12}} \mathbf{T}_{sensor_{12}}^{object} \\ \mathbf{T}_{pf}^{sensor_{11}} \mathbf{T}_{sensor_{11}}^{object} &= \mathbf{T}_{pf}^{sensor_{21}} \mathbf{T}_{sensor_{21}}^{object} \end{aligned} \tag{30}$$

where the first subscript stands for the finger number and the second subscript stands for the link number. The following matrices are found: for the first link of finger 1, $\mathbf{T}_{pf}^{sensor_{11}}$ is

$$\mathbf{T}_{pf}^{sensor_{11}} = \begin{bmatrix} 1 & 0 & 0 & 1.5 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

For the second link of finger 1, $\mathbf{T}_{pf}^{sensor_{12}}$ is

$$\mathbf{T}_{pf}^{sensor_{12}} = \begin{bmatrix} 0.85264 & 0.5225 & 0 & 2.42632 \\ 0 & 0 & -1 & 0 \\ -0.5225 & 0.85264 & 0 & 4.73875 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

For the first link of finger 2, $\mathbf{T}_{pf}^{sensor_{21}}$ is

$$\mathbf{T}_{pf}^{sensor_{21}} = \begin{bmatrix} 0.70711 & 0.70711 & 0 & 1.35356 \\ 0 & 0 & -1 & 0 \\ -0.70711 & 0.70711 & 0 & 0.64645 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Using equation 24, the transformations $\mathbf{T}_{sensor_{11}}^{object}$ and $\mathbf{T}_{sensor_{12}}^{object}$ are

$$\mathbf{T}_{sensor_{1j}}^{object} = \mathbf{R}_1 \cdot \mathrm{Sph}(\gamma_{1j}, \beta_{1j}, r_{1j} + R) \tag{31}$$

where j is the link number 1 or 2. The $\mathbf{R}_1$ matrix is

$$\mathbf{R}_1 = Rot(x, 90^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

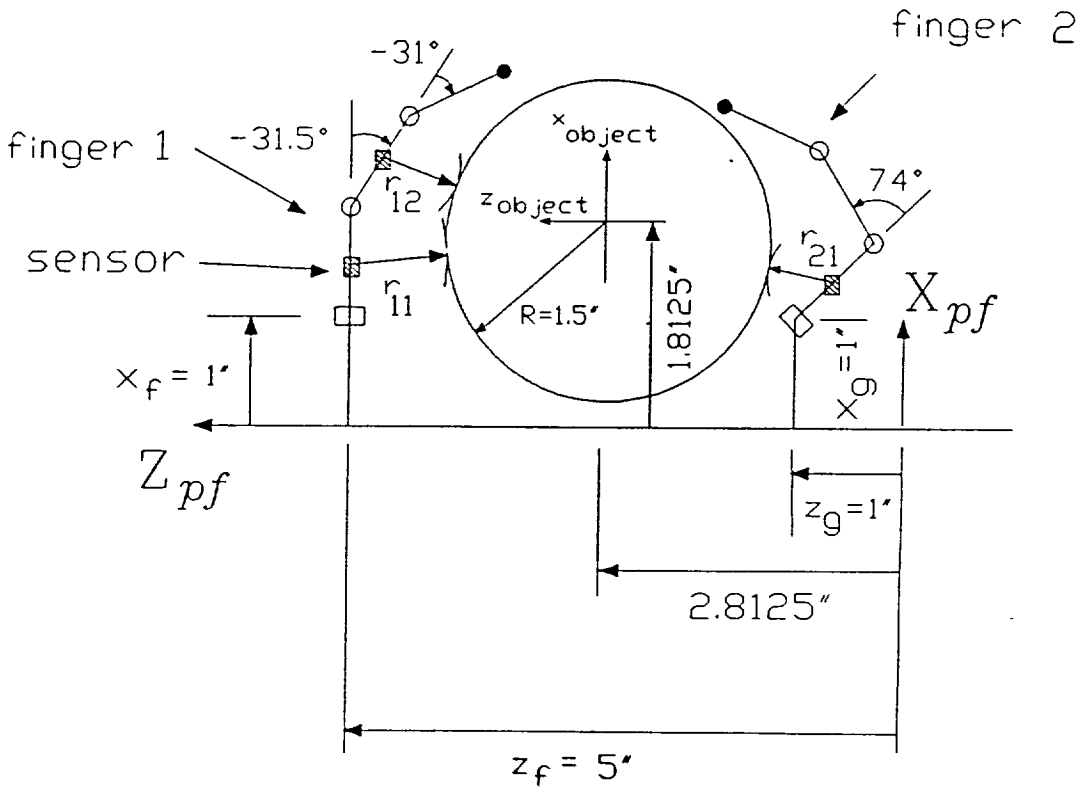


Figure 22. Pose estimation of a circle using a two-fingered hand.

For the first link of finger 2, $\mathbf{T}^{object}_{sensor_{21}}$ is

$$\mathbf{T}^{object}_{sensor_{21}} = \mathbf{R}_2 \cdot \mathrm{Sph}(\gamma_{21}, \beta_{21}, r_{21} + R) \tag{32}$$

where the $\mathbf{R}_2$ matrix is

$$\mathbf{R}_2 = Rot(x, -90^\circ)\ Rot(z, 180^\circ) = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Then equation 31 becomes

$$\mathbf{T}^{object}_{sensor_{1j}} = \begin{bmatrix} \cos\gamma_{1j}\ \cos\beta_{1j} & -\sin\gamma_{1j} & \cos\gamma_{1j}\ \sin\beta_{1j} & (r_{1j}+R)\cos\gamma_{1j}\ \sin\beta_{1j} \\ \sin\beta_{1j} & 0 & -\cos\beta_{1j} & -(r_{1j}+R)\cos\beta_{1j} \\ \sin\gamma_{1j}\ \cos\beta_{1j} & \cos\gamma_{1j} & \sin\gamma_{1j}\ \sin\beta_{1j} & (r_{1j}+R)\sin\gamma_{1j}\ \sin\beta_{1j} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and (32) becomes

$$\mathbf{T}^{object}_{sensor_{21}} = \begin{bmatrix} -\cos\gamma_{21}\cos\beta_{21} & \sin\gamma_{21} & -\cos\gamma_{21}\sin\beta_{21} & -(r_{21}+R)\cos\gamma_{21}\sin\beta_{21} \\ -\sin\beta_{21} & 0 & \cos\beta_{21} & (r_{21}+R)\cos\beta_{21} \\ \sin\gamma_{21}\cos\beta_{21} & \cos\gamma_{21} & \sin\gamma_{21}\sin\beta_{21} & (r_{21}+R)\sin\gamma_{21}\sin\beta_{21} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Taking the products required by equation 30 and looking at only the last column of each, yields

$$\mathbf{T}^{sensor_{11}}_{pf}\mathbf{T}^{object}_{sensor_{11}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} (r_{11}+R)\cos\gamma_{11}\ \sin\beta_{11} + 1.5 \\ -(r_{11}+R)\sin\gamma_{11}\ \sin\beta_{11} \\ -(r_{11}+R)\cos\beta_{11} + 5 \\ 1 \end{bmatrix} \tag{33}$$

$$\mathbf{T}^{sensor_{12}}_{pf}\mathbf{T}^{object}_{sensor_{12}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0.853(r_{12}+R)\cos\gamma_{12}\ \sin\beta_{12} - 0.523(r_{12}+R)\cos\beta_{12} + 2.426 \\ -(r_{12}+R)\sin\gamma_{12}\ \sin\beta_{12} \\ -0.523(r_{12}+R)\cos\gamma_{12}\ \sin\beta_{12} - 0.853(r_{12}+R)\cos\beta_{12} + 4.739 \\ 1 \end{bmatrix} \tag{34}$$

$$\mathbf{T}^{sensor_{21}}_{pf}\mathbf{T}^{object}_{sensor_{21}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} -0.707(r_{21}+R)\cos\gamma_{21}\ \sin\beta_{21} + 0.707(r_{21}+R)\cos\beta_{21} + 1.354 \\ -(r_{21}+R)\sin\gamma_{21}\ \sin\beta_{21} \\ 0.707(r_{21}+R)\cos\gamma_{21}\ \sin\beta_{21} + 0.707(r_{21}+R)\cos\beta_{21} + 0.646 \\ 1 \end{bmatrix}. \tag{35}$$

Table 5. Radial distances, r.

Variable	Distances (in)
r_{11}	0.7097087
r_{12}	0.5216860
r_{21}	0.7141319

Also, from equation 30, the three equations above must all be equal to each other. Equating them yields six equations with six unknowns

$$\gamma_{11},\ \beta_{11},\ \gamma_{12},\ \beta_{12},\ \gamma_{21},\ \beta_{21}.$$

Software using a nonlinear least-squares technique was written to solve for the unknowns. Table 5 lists the radial distances obtained from the forward model. These distances were calculated using the known circle position and finger joint angles. Then, the r values were input to the inverse model to calculate the offset angles and compared to the values obtained in the forward model, as shown in Table 6. The sensor parameters obtained match those given by the forward model with slight differences due to roundoff error.

Once γ and β are obtained for any sensor, the object frame position with respect to the palm frame is the position vector from the product $\mathbf{T}_{pf}^{sensor}\mathbf{T}_{sensor}^{object}$ for that particular sensor. For example, using equation 33 with values obtained from Table 6 for the sensor on link 1 of finger 1, the object frame position with respect to the palm frame is

$$\mathbf{T}_{pf}^{sensor_{11}}\mathbf{T}_{sensor_{11}}^{object} = \begin{bmatrix} 1.8125 \\ -2 \times 10^{-5} \\ 2.8125 \\ 1 \end{bmatrix}$$

which agrees with the input values to the forward solution.

Table 6. Pose estimation results.

sensor parameters	forward solution (degrees)	inverse solution (degrees)	percent error (%)
γ_{11}	≈ 0	0.38080	-
β_{11}	8.1301	8.1302	≈ 0
γ_{12}	≈ 0	0.24633	-
β_{12}	13.8251	13.8251	0
γ_{21}	≈ 0	≈ 0	-
β_{21}	33.0371	33.0373	≈ 0

The overall procedure can be programed using the nonlinear least-squares technique to solve a general three-dimensional problem.

Discussion

There are some limitations in the pose estimation algorithm: the fact that only the position is calculated versus both the position and orientation, and that the object and sensor characteristics are modeled as being spherical.

To estimate the orientation of the object is a more complex problem. Furhman and Kanade (1984) describe a method of estimating the orientation of a surface using a multilight proximity sensor. The three-dimensional locations of the spots of light on the surface of the object are computed using triangulation. Then, by fitting a surface to a set of points, the orientation and curvature of the surface are calculated.

The pose estimation algorithm used an ideal model for the sensor. A more realistic model of the sensor characteristics is the "tear shaped" model as shown in Figure 18. This shape must be modeled mathematically and the equation development must take the difference of sensor models into account.

The pose estimation algorithm also used a spherical model for the target object. Other types of objects can be used, but it will require more computation to model them. One possible way of representing objects mathematically is to use Fourier

Descriptors, which is a method used in imaging processes to represent a shape. The problem with using a method such as Fourier Descriptors is that the computation time required may make the program run too slow for a real-time environment. An alternative would be to treat objects as geometric primitives and develop the algorithm to handle these specific primitives.

FUTURE RESEARCH

This chapter discusses future extensions of the investigation. The first section presents a grasp strategy using the inverse kinematic and pose estimation algorithms to provide low-level control of a dexterous hand when grasping an object. The second section lists some ideas for research, software development and simulation, and hardware design and implementation of the grasp strategy.

Grasp Algorithm

Tomovic, Bekey, and Karplus (1987) proposed a method for grasping arbitrary objects using a multifingered hand. They defined the basic elements involved in grasping based on the philosophy of reflex control (i.e.; each aspect of the grasping task is initiated and terminated using sensory data and rules of behavior derived from human expertise). These basic elements are

1. Represent target objects as geometrical primitives.
2. Preshape and align the hand to conform with the selected geometrical primitive.
3. Determine the most suitable hand configuration for the primitive (1, 2, or 3 fingered grasp).
4. Separation of the grasping task into a target approach phase and a shape adaptation phase while applying reflex control philosophy.

Keeping these basic elements in mind, a strategy termed the *Grasp Algorithm* is proposed which utilizes the inverse kinematic and pose estimation software to provide control of a dexterous hand when grasping an object. Figure 23 shows a flowchart for the proposed strategy. First, the vision system will locate and identify the object as some geometric primitive and determine its coordinate frame, as shown in box 1. A strategy similar to the one proposed by Rao, et al (1988) could be used to accomplish the task of object identification. Using this information, the robot will move towards the object and preshape the hand (box 2). High-level reasoning will select the appropriate preshape based on parameters such as object size and shape, and the location of the selected grasping points.

Next, the robot will position the hand close enough to the object such that the object is within the grasping region of the hand (box 3). Data obtained from the proximity sensors should be able to answer the question of whether the object is within the grasping region. Chammas (1989) discusses a method of obtaining the *grasping pre-image* of a hand, which is the region the object must be to be grasped successfully.

Once the object is within reach, position commands should be sent to the hand based on the type of grasp that is required. These position commands are input to the inverse kinematics software which determines the joint angles necessary (box 4). Now, the object pose is calculated from the proximity sensor data using the pose estimation software (box 5), and any adjustments to the finger positions are made if necessary. Once sensor outputs reach some threshold, the hand is closed and the object is grasped. Vision or tactile sensors can then verify if the object has been successfully grasped through measurement of force and slip (boxes 6 and 7).

Implementation

This work presented algorithms to determine the inverse kinematics of a manipulator and the pose estimation of a pre-grasped object. These algorithms were developed assuming several simplifications such as: the inverse kinematics did not take into account the mechanical constraints of the manipulator and, as a result, multiple solutions were available; the pose estimation algorithm used a spherical sensor model which may or may not be a close estimation of an actual sensor model. Likewise, the object to be grasped was limited to a sphere, and thus orientation did not matter. In retrospect, this investigation provided a software shell which can be easily upgraded to take into account the factors listed above.

At present, software is being developed based on the inverse kinematic and pose estimation algorithms to simulate the movement of a three–dimensional model of the University of Minnesota hand. The modeling software that is being used is called TOPAS (AT&T Bell Laboratories).

Other subjects which should be considered in the future are:

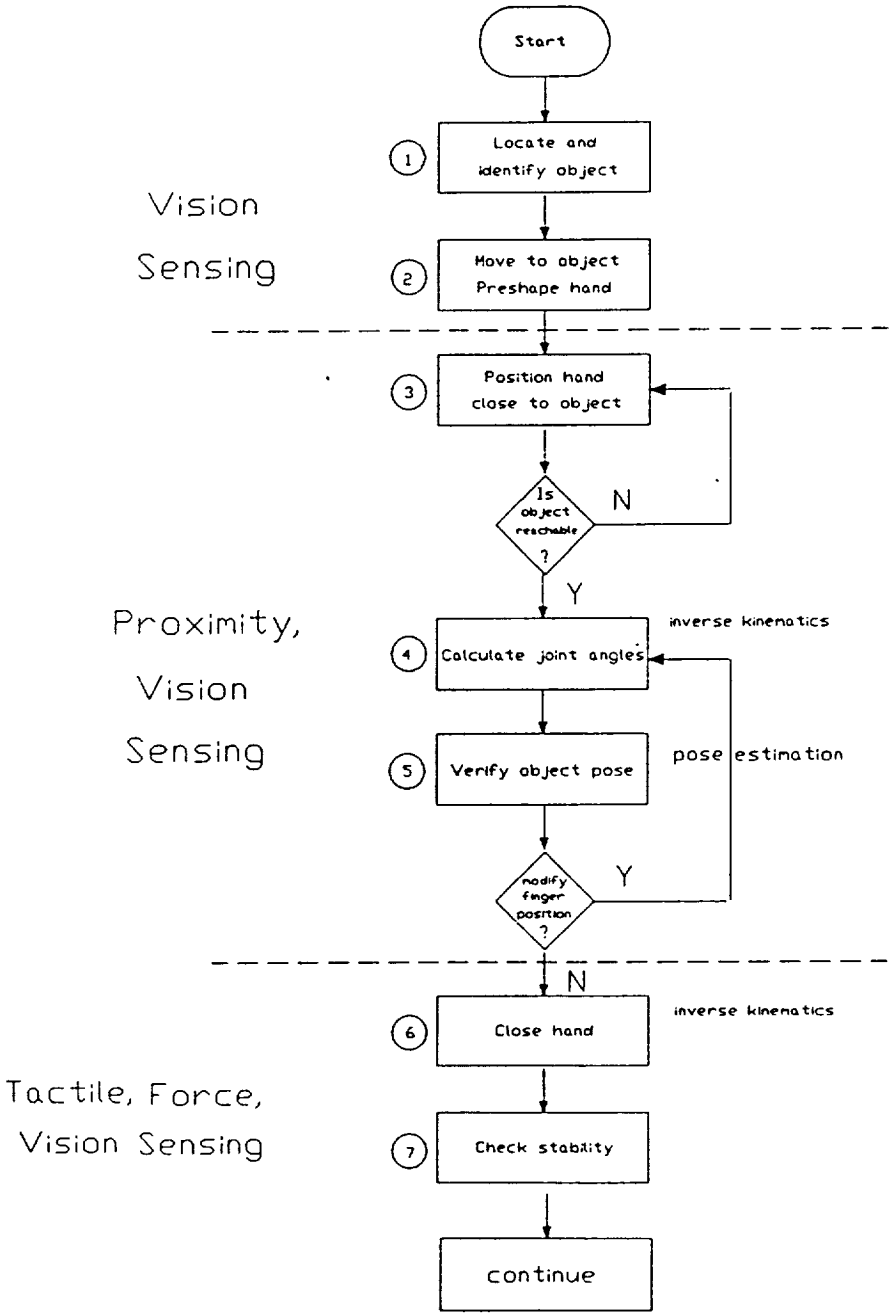


Figure 23. Grasping scheme.

1. Increase the sensor model complexity of the pose estimation algorithm to match the characteristics of actual sensors.
2. Use actual finger dimensions and other physical constraints to increase usefulness of the inverse kinematics algorithm.
3. Implement the *grasp algorithm* in conjunction with a robot hand controller and driver to adequately close the hand; i.e.; provide correct timing, force, and form of the hand on the object.
4. Use multisensory data together with the *grasp algorithm* to successfully manipulate an object.

CONCLUSIONS

A numerical scheme was developed to solve the inverse kinematics for a user-defined manipulator. The scheme was based on a nonlinear least-squares technique which determines the joint variables by minimizing the difference between the target end effector pose and the actual end effector pose. The scheme was adopted to a dexterous hand in which the joints are either prismatic or revolute and the fingers are considered open kinematic chains. Feasible solutions were obtained using a three-fingered dexterous hand.

An algorithm to estimate the position and orientation of a pre-grasped object was also developed. The algorithm was based on triangulation using an ideal sensor and a spherical object model. By choosing the object to be a sphere, only the position of the object frame was important. Based on these simplifications, a minimum of three sensors are needed to find the position of a sphere. A two dimensional example to determine the position of a circle coordinate frame using a two-fingered dexterous hand was presented.

REFERENCES

1 D. McFalls and E. Franke, "A Robotic Assistant for Space Station Freedom", *Robotics Today*, vol. 2, no. 2, pp. 1-6, Second Quarter, 1989.

2 R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, Cambridge, MA, The MIT Press, chaps. 1-3, 1981.

3 T. Koehler and M. Donath, "Inverse Kinematics for a Multifingered Hand", *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 234-239, 1988.

4 J.K. Poon and P.D. Lawrence, "Manipulator Inverse Kinematics Based on Joint Functions", *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 669-674, 1988.

5 A. A. Goldenberg, B. Benhabib, and R. G. Fenton, "A Complete Generalized Solution to the Inverse Kinematics of Robots", *IEEE Journal of Robotics and Automation*, vol. RA-1, no. 1, pp. 14-19, March, 1985.

6 R.J. Schilling, *Fundamentals of Robotics*, Englewood Cliffs, NJ, Prentice–Hall, chaps. 3 and 5, 1990.

7 A. Guez and A. Ziauddin, "Solution to the Inverse Kinematics Problem in Robotics by Neural Networks", *Neural Networks*, vol. 1, no. 1, p. 337, 1988.

8 H. Kobayashi, "Grasping and Manipulation of Objects by Articulated Hands", *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1514-1519, 1986.

9 G. B. Dunn and J. Segen, "Automatic Discovery of Robotic Grasp Configurations", *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 396-401, 1988.

10 K. Rao, G. Medioni, H. Liu, and G. A. Bekey, "Robot Hand–Eye Coordination: Shape Description and Grasping", *IEEE International Conference on Robotics and Automation*, vol. 1, pp. 407-411, 1988.

11 S. A. Stansfield, "Robotic Grasping of Unknown Objects: A Knowledge–Based Approach", *Sandia Report SAND–1087·UC–32*, Sandia National Laboratories, Albuquerque, NM, June, 1989.

12 B. Espiau, "Use of Optical Reflectance Sensors", *Recent Advances in Robotics*, G. Beni and S. Hackwood, Editors, Santa Barbara, CA, John Wiley & Sons, pp. 313-357, 1985.

13 E. Balaure, "Optical Reflectance Sensors and Their Applications in Automatic Grasping", *Recent Trends in Robotics: Modeling, Control, and Education*, M. Jamshidi, L.Y.S. Luh, and M. Shahinpoor, Editors, New York, Elsevier Science Publishing, pp. 523-528, 1986.

14 D.J. Balek and R.B. Kelley, "Using Gripper Mounted Infrared Proximity Sensors for Robot Feedback Control", *IEEE International Conference on Robotics and Automation*, 1985.

15 M. Furhman and T. Kanade, "Optical Proximity Sensor Using Multiple Cones of Light for Measuring Surface Shape", *Optical Engineering*, vol. 23, no. 5, pp. 546-553, September/October, 1984.

16 A. Romiti and T. Raparelli, "Dynamic Six Component Measurement of Robot Precision", *Proceedings of the 2nd International Conference on Robotics and Factories of the Future*, New York, Springer-Verlag, pp. 497-502, 1987.

17 B. Espiau, "An Overview of Local Environment Sensing in Robotics Applications", *Sensors and Sensory Systems for Advanced Robots*, P. Dario, Editor, New York, Springer-Verlag, pp. 125-151, 1988.

18 R. Tomovic, G.A. Bekey, and W.J. Karplus, "A Strategy for Grasp Synthesis with Multifingered Robot Hands", *IEEE International Conference on Robotics and Automation*, vol. 3, pp.83-89, 1987.

19 C.Z. Chammas and J.K. Salisbury, "Hands: An Automatic Grasping Approach", *MIT Technical Report*, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1989.

Supplementary Sources Consulted

1 G.B. Thomas and R.L. Finney, *Calculus and Analytic Geometry, 6th Edition*, Reading, MA, Addison-Wesley, pp. 717-718, 1984.

2 P. Coiffet, *Robot Technology Volume 2: Interaction with the Environment*, London, Kogan Page Ltd., p. 100, 1983.

APPENDIX A

MAIN PROGRAM FOR INVERSE KINEMATICS

```
************************************************************************
************************************************************************
*        PROGRAM MULTI.FOR   (LAST REVISION: 9-8-90)                   *
*                                                                      *
* BY:  VICTOR H. PINTO                                                 *
*      TEXAS A&M UNIVERSITY, DEPT OF MECHANICAL ENGINEERING            *
*      COLLEGE STATION, TEXAS 77843                                    *
************************************************************************

* PROGRAM DESCRIPTION:
* THIS PROGRAM CALCULATES THE NECESSARY JOINT ANGLES OF EACH FINGER
* OF A MULTIFINGERED HAND GIVEN THE KINEMATIC PARAMETERS OF EACH
* FINGER AND THE DESIRED POSITION OF THE CONTACT-POINT FRAME.
* THIS TASK IS ACCOMPLISHED USING A LEAST-SQUARES MINIMIZATION
* PROCESS.

* HAND MODELED: UNIV.. OF MINN. HAND

* PARAMETERS:
*     NOL = NUMBER OF LINKS PER FINGER
*     NOF = NUMBER OF FINGERS
*     LDFJAC,M,N = PARAMETERS USED IN THE LEAST-SQUARES ROUTINE, WHERE
*                  M IS THE NUMBER OF FUNCTIONS AND N IS THE  NUMBER
*                  OF UNKNOWNS.
* INPUT VARIABLES:
*     AL,AA,DD = MATRICES WHICH CONTAIN THE REQUIRED D-H KINEMATIC
*                PARAMETERS OF EACH FINGER.
*     C1,C2,C3 = MATRICES WHICH DEFINE THE POSITION OF THE
*                CONTACT-POINT WITH RESPECT TO THE OBJECT FRAME.
*     F1,F2,F3 = MATRICES WHICH DEFINE THE TRANSFORMATION FROM
*                THE PALM FRAME TO A FINGER-BASE FRAME.
*     BAR = TRANSFORMATION MATRIX FROM THE PALM FRAME TO THE
*           OBJECT FRAME (IN THIS CASE A RECTANGULAR BAR).
* OUTPUT VARIABLES:
*     X = VECTOR OF LENGTH N WHICH CONTAINS THE JOINT ANGLES FOR A
*         PARTICULAR FINGER (RADIANS)
*     XN,YN,ZN = POSITION COOR OF THE FINGERTIP FRAME (INCHES)
*     PHID,THETAD,PSID = EULER ANGLES OF THE FINGERTIP FRAME (DEGREES)

* ***NOTE*** NOF AND NOL MUST BE ENTERED INTO THE PARAMETER LIST OF
* SUBROUTINE FORWARD.

* ***NOTE*** THE JOINT VARIABLE CONSTRAINTS MUST BE SET IN SUBROUTINE
* CONSTRAINT.

* ASSUMPTIONS: 3 FINGERS, 3 LINKS
```

```
      INTEGER LDFJAC,M,N,I,J,NOL,NOF,COUNT,IPARAM(6),FLAG,FLAG2
      PARAMETER (LDFJAC=3,M=3,N=3)
      PARAMETER (NOF=3,NOL=3)
      DOUBLE PRECISION FJAC(LDFJAC,N),FSCALE(M),FVEC(M),RPARAM(7),
     & X(N),XGUESS(N),XSCALE(N),DEG,POS(3),XN,YN,ZN,PHID,THETAD,
     & PSID,PHI,THETA,PSI,AL(NOF,NOL),AA(NOF,NOL),DD(NOF,NOL),
     & F1(4,4),F2(4,4),F3(4,4),C1(4,4),C2(4,4),C3(4,4),BAR(4,4),
     & T(4,4)
      COMMON /CONTACT/ C1,C2,C3
      COMMON /DH_PARAMETERS/ AA,AL,DD
      COMMON /FING_BASE/ F1,F2,F3
      COMMON /BAR/ BAR
      COMMON COUNT,POS,T,XN,YN,ZN

      CHARACTER*23 OUTPUT
      EXTERNAL DUNLSF,FORWARD

C *********calculate the joint angles for each finger***********

      DO 10 COUNT=1,NOF

C open data files for results:

      WRITE (6,*) 'ENTER OUTPUT FILE NAME FOR FINGER ',COUNT
      READ (5,20) OUTPUT
20    FORMAT (A)
      OPEN (UNIT=15,FILE=OUTPUT,STATUS='NEW')

C call subroutine POSITION  to compute the position
C of the contact-point frame:

      CALL POSITION (COUNT,POS)

C call IMSL routine to compute least-squares minimization:

      DATA XGUESS /0.20,0.20,0.20/
      DATA XSCALE /N*1.0/, FSCALE /M*1.0/

      FLAG2=0
5     CONTINUE
      IPARAM(1)=0
      FLAG=0
      CALL DUNLSF (FORWARD,M,N,XGUESS,XSCALE,FSCALE,IPARAM
     +,RPARAM,X,FVEC,FJAC,LDFJAC)

C check joint variable constraints:

      FLAG2=FLAG2+1
      IF (FLAG2.GT.3) THEN
         WRITE (6,*) 'NO JOINT SOLUTION AVAILABLE FOR
     +FINGER ',COUNT
         GOTO 10
      ELSE
```

```
         WRITE (6,*) ' WORKING ...'
      ENDIF

      CALL CONSTRAINT (COUNT,FLAG,X,N)

C if FLAG is >= 1, then a joint limit has been exceeded.

      IF (FLAG.GT.0) THEN
      DO 6 I=1,6
         XGUESS(I)=X(I)
6     CONTINUE
      GOTO 5
      ENDIF

C compute the Euler angles of the contact-point frame:

      IF ((T(1,3).EQ.0.0).AND.(T(2,3).EQ.0.0)) THEN
         PHI=0.0
      ELSEIF ((T(1,3).LT.0.0).AND.(T(2,3).LT.0.0)) THEN
         PHI=DATAN2(DABS(T(2,3)),DABS(T(1,3)))+3.14159D0
      ELSE
         PHI=DATAN2(T(2,3),T(1,3))
      END IF

      THETA=DATAN2(DCOS(PHI)*T(1,3)+DSIN(PHI)*T(2,3),T(3,3))
      PSI=DATAN2(-DSIN(PHI)*T(1,1)+DCOS(PHI)*T(2,1),-DSIN(PHI)*T(1,2)
     ++DCOS(PHI)*T(2,2))

      PHID = PHI * 180.0D0/3.14159D0
      THETAD = THETA * 180.0D0/3.14159D0
      PSID = PSI * 180.0D0/3.14159D0

C write results to data file:

      WRITE (15,25) COUNT
25    FORMAT (' ',5X,'INVERSE SOLUTION FOR FINGER',1X,I2)
      DO 35 J = 1,N
         DEG = X(J) * 180.0D0/3.14159D0
         WRITE (15,30) COUNT,J,DEG,FVEC(J)
30       FORMAT (' ',2X,'THETA',1X,I2,I2,1X,'=',1X,F17.11,4X,
     +'TOLERANCE =',1X,F15.11)
35    CONTINUE
      WRITE (15,40) COUNT
40    FORMAT (' ',2X,'POSITION AND ORIENTATION OF CONTACT FRAME',1X,
     +I2)
      WRITE (15,45) 'X-DIRECTION =',XN,'Y-DIRECTION =',YN,
     +'Z-DIRECTION =',ZN
45    FORMAT (' ',2X,A,2X,F17.11)
      WRITE (15,45) 'PHI =',PHID,'THETA =',THETAD,'PSI =',PSID
10    CONTINUE

      CLOSE(15)
      CALL EXIT
      END
```

APPENDIX B

SUBROUTINE TO MINIMIZE RESIDUAL FUNCTIONS

```
      SUBROUTINE FORWARD (M,N,X,F)

C THIS SUBROUTINE CALCULATES THE NON-LINEAR FUNCTIONS FOR USE
C IN THE IMSL ROUTINE DUNLSF.

      INTEGER M,N,I,J,COUNT,NOF,NOL

      PARAMETER (NOF=3,NOL=3)

      DOUBLE PRECISION X(N),F(M),R(4,4),TO(4,4), T(4,4),TEMP(4,4),
     & F1(4,4),F2(4,4),F3(4,4),AL(NOF,NOL),AA(NOF,NOL),DD(NOF,NOL),
     & TH(NOF,NOL),POS(3),XN,YN,ZN

      COMMON /DH_PARAMETERS/ AA,AL,DD
      COMMON /FING_BASE/ F1,F2,F3
      COMMON COUNT,POS,T,XN,YN,ZN

C initialize TO matrix to identity matrix:

      DATA TO /1.,0.,0.,0.,0.,1.,0.,0.,0.,0.,1.,0.,0.,0.,0.,1./

C initialize variables:

      TH(COUNT,1)=X(1)
      TH(COUNT,2)=X(2)
      TH(COUNT,3)=X(3)

C initialize T matrix to identity matrix:

      DO 20 I=1,4
      DO 30 J=1,4
         T(I,J)=TO(I,J)
30    CONTINUE
20    CONTINUE

C compute the transformation matrix T for the finger:

      DO 40 I=1,NOL
         CALL TRANSFORM (AL(COUNT,I),AA(COUNT,I),DD(COUNT,I),
     +                   TH(COUNT,I),R)
         CALL MATMULA (T,R)
40    CONTINUE

C compute the overall transformation matrix with respect to the
C hand base frame
```

```
      IF (COUNT.EQ.1) THEN
         DO 50 I=1,4
         DO 60 J=1,4
            TEMP(I,J)=F1(I,J)
60       CONTINUE
50       CONTINUE
      ELSE IF (COUNT.EQ.2) THEN
         DO 70 I=1,4
         DO 80 J=1,4
            TEMP(I,J)=F2(I,J)
80       CONTINUE
70       CONTINUE
      ELSE IF (COUNT.EQ.3) THEN
         DO 90 I=1,4
         DO 100 J=1,4
            TEMP(I,J)=F3(I,J)
100      CONTINUE
90       CONTINUE
      END IF
         CALL MATMULA (TEMP,T)

      DO 110 I=1,4
      DO 120 J=1,4
         T(I,J)=TEMP(I,J)
120   CONTINUE
110   CONTINUE

C compute position:

      XN=T(1,4)
      YN=T(2,4)
      ZN=T(3,4)

C calculate functions:

      F(1)=(POS(1))-(XN)
      F(2)=(POS(2))-(YN)
      F(3)=(POS(3))-(ZN)

C calculate rms values for position errors:

      PRMS=DSQRT(F(1)**2+F(2)**2+F(3)**2)

C print rms values to screen:

C      PRINT *, PRMS

      RETURN
      END
```

APPENDIX C

SUBROUTINE TO CHECK JOINT CONSTRAINTS

```
      SUBROUTINE CONSTRAINT (COUNT,FLAG,X,N)

C SUBROUTINE TO VERIFY AND CORRECT THE JOINT VARIABLES
C IN ORDER TO MAINTAIN MECHANICAL LIMITS.

      INTEGER N, FLAG, COUNT
      DOUBLE PRECISION X(N), PI, PI2, PI3, PI6

      PI=3.14159265D0
      PI2=PI/2
      PI3=PI/3
      PI6=PI/6

      IF ((COUNT.EQ.1).OR.(COUNT.EQ.2)) THEN
      IF (X(1).LT.-PI6) THEN
         X(1)=-PI6
         FLAG=FLAG+1
      ENDIF
      IF (X(1).GT.PI6) THEN
         X(1)=PI6
         FLAG=FLAG+1
      ENDIF
      IF (X(2).LT.-PI3) THEN
         X(2)=-PI3
         FLAG=FLAG+1
      ENDIF
      IF (X(2).GT.0.0D0) THEN
         X(2)=0.0D0
         FLAG=FLAG+1
      ENDIF
      IF (X(3).LT.-PI2) THEN
         X(3)=-PI2
         FLAG=FLAG+1
      ENDIF
      IF (X(3).GT.0.0D0) THEN
         X(3)=0.0D0
         FLAG=FLAG+1
      ENDIF
      ELSEIF (COUNT.EQ.3) THEN
      IF (X(1).LT.-PI6) THEN
         X(1)=-PI6
         FLAG=FLAG+1
      ENDIF
      IF (X(1).GT.PI6) THEN
         X(1)=PI6
         FLAG=FLAG+1
```

```
ENDIF
IF (X(2).GT.PI3) THEN
   X(2)=PI3
   FLAG=FLAG+1
ENDIF
IF (X(2).LT.0.0D0) THEN
   X(2)=0.0D0
   FLAG=FLAG+1
ENDIF
IF (X(3).GT.PI2) THEN
   X(3)=PI2
   FLAG=FLAG+1
ENDIF
IF (X(3).LT.0.0D0) THEN
   X(3)=0.0D0
   FLAG=FLAG+1
ENDIF
ENDIF

RETURN
END
```

APPENDIX D

SUBROUTINE TO CALCULATE

CONTACT-POINT LOCATION

```
      SUBROUTINE POSITION (POS)

C THIS SUBROUTINE CALCULATES THE REQUIRED POSITION OF
C THE CONTACT-POINT WITH RESPECT TO THE PALM FRAME.

      INTEGER I,J

      DOUBLE PRECISION X(4,4),POS(3),CP(3),C(4,4),BAR(4,4)

      COMMON /CONTACT/ CP
      COMMON /BAR/ BAR

C initialize matrices:

      DATA C /1.,0.,0.,0.,0.,1.,0.,0.,0.,0.,1.,0.,0.,0.,0.,1./
      DO 100 I=1,3
100      C(I,4)=CP(I)

      DO 110 I=1,4
      DO 110 J=1,4
110      X(I,J)=BAR(I,J)

      CALL MATMULA (X,C)

C retrieve positions:

      POS(1)=X(1,4)
      POS(2)=X(2,4)
      POS(3)=X(3,4)

      RETURN
      END
```

APPENDIX E

SUBROUTINE TO CALCULATE THE A MATRIX

```
      SUBROUTINE TRANSFORM (AL,AA,DD,TH,TT)

C SUBROUTINE TO COMPUTE THE INDIVIDUAL T MATRIX

      DOUBLE PRECISION AL,AA,DD,TH,TT(4,4)

      TT(1,1)=DCOS(TH)
      TT(1,2)=-DSIN(TH)*DCOS(AL)
      TT(1,3)=DSIN(TH)*DSIN(AL)
      TT(1,4)=DCOS(TH)*AA
      TT(2,1)=DSIN(TH)
      TT(2,2)=DCOS(TH)*DCOS(AL)
      TT(2,3)=-DCOS(TH)*DSIN(AL)
      TT(2,4)=DSIN(TH)*AA
      TT(3,1)=0.0
      TT(3,2)=DSIN(AL)
      TT(3,3)=DCOS(AL)
      TT(3,4)=DD
      TT(4,1)=0.0
      TT(4,2)=0.0
      TT(4,3)=0.0
      TT(4,4)=1.0

      RETURN
      END
```

APPENDIX F

SUBROUTINE TO POST-MULTIPLY MATRICES

```
      SUBROUTINE MATMULA (A,B)

C SUBROUTINE TO POST-MULTIPLY MATRICES

      INTEGER I,J
      DOUBLE PRECISION A(4,4), B(4,4), C(4,4)

      DO 10 I=1,4
      DO 20 J=1,4
        C(I,J)= A(I,1)*B(1,J)+A(I,2)*B(2,J)+A(I,3)*B(3,J)
     ++A(I,4)*B(4,J)
20    CONTINUE
10    CONTINUE

      DO 30 I=1,4
      DO 40 J=1,4
         A(I,J)=C(I,J)
40    CONTINUE
30    CONTINUE

      RETURN
      END
```

APPENDIX G

DATA FOR IKP EXAMPLE USING UNIV. OF MINN. HAND

```
      BLOCK DATA
      DOUBLE PRECISION AL(3,3),AA(3,3),DD(3,3)
      DOUBLE PRECISION F1(4,4),F2(4,4),F3(4,4)
      DOUBLE PRECISION C1(4,4),C2(4,4),C3(4,4)
      DOUBLE PRECISION BAR(4,4)
      COMMON /DH_PARAMETERS/ AA,AL,DD
      COMMON /FING_BASE/ F1,F2,F3
      COMMON /CONTACT/ C1,C2,C3
      COMMON /BAR/ BAR
      DATA AL /1.5708D0,1.5708D0,1.5708D0,
     +         0.0,0.0,0.0,
     +         0.0,0.0,0.0/
      DATA AA /1.0D0,1.0D0,1.0D0,
     +         1.0D0,1.0D0,1.0D0,
     +         1.0D0,1.0D0,1.0D0/
      DATA DD /9*0.0/
      DATA F1 /1.0,0.0,0.0,0.0,
     +         0.0,1.0,0.0,0.0,
     +         0.0,0.0,1.0,0.0,
     +         1.0D0,-1.0D0,4.0D0,1.0/
      DATA F2 /1.0,0.0,0.0,0.0,
     +         0.0,1.0,0.0,0.0,
     +         0.0,0.0,1.0,0.0,
     +         1.0D0,1.0D0,4.0D0,1.0/
      DATA F3 /0.70711D0,0.0,-0.70711D0,0.0,
     +         0.0,      1.0, 0.0,      0.0,
     +         0.70711D0,0.0, 0.70711D0,0.0,
     +         0.5D0,     0.0D0,1.0D0,  1.0/
      DATA C1 /1.0,0.0,0.0,0.0,
     +         0.0,1.0,0.0,0.0,
     +         0.0,0.0,1.0,0.0,
     +         0.0D0,-1.0D0,0.5D0,1.0/
      DATA C2 /1.0,0.0,0.0,0.0,
     +         0.0,1.0,0.0,0.0,
     +         0.0,0.0,1.0,0.0,
     +         0.0D0,1.0D0,0.5D0,1.0/
      DATA C3 /1.0,0.0,0.0,0.0,
     +         0.0,1.0,0.0,0.0,
     +         0.0,0.0,1.0,0.0,
     +         0.0D0,0.0D0,-0.5D0,1.0/
      DATA BAR /1.0,0.0,0.0,0.0,
     +          0.0,1.0,0.0,0.0,
     +          0.0,0.0,1.0,0.0,
     +          2.0D0,0.0D0,2.0D0,1.0/
      END
```

APPENDIX H

MAIN PROGRAM FOR POSE ESTIMATION

```
************************************************************************
************************************************************************
*          PROGRAM POSTI.FOR    (LAST REVISION: 10-9-90)                 *
*                                                                        *
* BY:  VICTOR H. PINTO                                                   *
*      TEXAS A&M UNIVERSITY, DEPT OF MECHANICAL ENGINEERING              *
*      COLLEGE STATION, TEXAS 77843                                      *
************************************************************************

* PROGRAM DESCRIPTION:
* THIS PROGRAM FINDS THE POSITION OF A SPHERE GIVEN THE SENSOR
* DISTANCES (INVERSE SOLUTION).  THIS IS ACCOMPLISHED USING A LEAST-
* SQUARES MINIMIZATION PROCESS.

* HAND MODELED:  UNIV. OF MINN. HAND

* PARAMETERS:
*     NOL = NUMBER OF LINKS PER FINGER
*     NOF = NUMBER OF FINGERS
*     NOS = NUMBER OF SENSORS PER LINK
*     SSEN = LINK NUMBER OF THE REFERENCE SENSOR
*     LDFJAC,M,N = PARAMETERS USED IN THE LEAST-SQUARES ROUTINE.
*                  WHERE M IS THE NUMBER OF FUNCTIONS, AND N IS
*                  THE NUMBER OF UNKNOWNS.
* INPUT VARIABLES:
*     AL,AA,DD,TH = MATRICES WHICH CONTAIN THE REQUIRED D-H KINEMATIC
*                   PARAMETERS OF EACH FINGER LINK
*     F1,F2,F3 = TRANSFORMATION MATRICES BETWEEN THE PALM FRAME
*                TO THE FINGER BASE FRAME
*     ZETA = (NOF,2) MATRIX WHICH CONTAINS THE NECESSARY SENSOR FRAME
*            ROTATIONS
*     RS = (NOF,NOL) MATRIX OF SENSOR VALUES
*     XGUESS = VECTOR OF LENGTH N CONTAINING THE INITIAL GUESS FOR
*              THE LEAST-SQUARES ROUTINE.
*     XSCALE = VECTOR OF LENGTH N CONTAINING THE DIAGONAL SCALING
*              MATRIX FOR THE VARIABLES (USED IN LSQ ROUTINE).  ALL
*              ENTRIES HAVE BEEN SET TO 1.0.
*     SGAMMA,GAMMAI,GAMMAJ,SBETA,BETAI,BETAJ = SENSOR PARAMETERS
*          THAT ARE CALCULATED PRIOR TO FINDING THE OBJECT POSITION
* OUTPUT VARIABLES:
*     XO,YO,ZO = POSITION COORDINATES OF OBJECT FRAME

* ***NOTE*** RS(NOF,NOL) AND ZETA(NOF,2) MUST BE DIMENSIONED
* IN SUBROUTINE FORWARD.  ALSO, NOF AND NOL MUST BE ENTERED
* INTO THE PARAMETER LIST OF SUBROUTINE FORWARD.
```

```
*     ASSUMPTIONS: 3 FINGERS, 3 SENSORS PER FINGER, 1 SENSOR PER LINK
*                  SSEN = 1

      INTEGER LDFJAC,M,N,NOL,NOF,NOS,FING,LINK,SENS,IPARAM(6),
     & SSEN
      PARAMETER (LDFJAC=6,M=6,N=6)
      PARAMETER (NOF=3,NOL=3,NOS=1)
      DOUBLE PRECISION FJAC(LDFJAC,N),FSCALE(M),FVEC(M),RPARAM(7),
     & X(N),XGUESS(N),XSCALE(N),XO,YO,ZO,RADIUS,R,RS(NOF,NOL),
     & ZETA(NOF,2),SGAMMA,GAMMAI,GAMMAJ,SBETA,BETAI,BETAJ,
     & AL(NOF,NOL),AA(NOF,NOL),DD(NOF,NOL),TH(NOF,NOL),F1(4,4),
     & F2(4,4),F3(4,4)
      COMMON /DH_PARAMETERS/ AL,AA,DD
      COMMON /JOINT_ANGLES/ TH
      COMMON /FING_BASE/ F1,F2,F3
      COMMON /SENSOR_VALUES/ RS
      COMMON /OPARAM/ RADIUS
      COMMON /FINGER/ FING
      COMMON /LINK/ LINK
      COMMON /REF_SENSOR/ SSEN
      COMMON /SENSOR_ORIENT/ ZETA

      CHARACTER*23 OUTPUT
      EXTERNAL DUNLSF,FORWARD

C initialize data:

      SSEN = 1

C open data files for results:

      WRITE (6,*) 'ENTER OUTPUT FILE NAME '
      READ (5,120) OUTPUT
120   FORMAT (A)
      OPEN (UNIT=15,FILE=OUTPUT,STATUS='NEW')

C determine parameters of the object:

         WRITE(6,*) 'ENTER RADIUS OF SPHERE'
         READ(5,*) RADIUS
         WRITE (15,*) '     OBJECT TYPE = SPHERE'
         WRITE (15,*) '     RADIUS (INCHES)= ',RADIUS,' INCHES'

C enter sensor values:

      OPEN (UNIT=16,FILE='SEN_VAL.DAT',STATUS='OLD')
      DO 5 I=1,NOF
        WRITE (15,*)
        WRITE (15,*) ' FING',I,' SENSOR VALUES'
      DO 6 J=1,NOL
        READ (16,*) RS(I,J)
6       WRITE (15,*) 'FING(',I,') LINK(',J,')= ', RS(I,J)
5     CONTINUE
      CLOSE (16)
```

```
C enter joint angles:

      DO 1 I=1,NOF
        WRITE (6,*) 'ENTER FING',I,' JOINT ANGLES'
        WRITE (15,*)
        WRITE (15,*) 'FING',I,' JOINT ANGLES'
        DO 2 J=1,NOL
          WRITE (6,*) 'FING(',I,') LINK(',J,')'
          READ (5,*) TH(I,J)
          WRITE (15,*) 'FING(',I,') LINK(',J,')= ',TH(I,J)
2       CONTINUE
1     CONTINUE

C ***************  calculate object position ********************

      DO 10 FING=1,NOF
      DO 20 LINK=SSEN+1,NOL

C call least-squares  IMSL routine:

      IPARAM(1)=0
C                  the guesses are arbitrary
      DATA XGUESS /0.2,0.2,0.2,0.2,0.2,0.2/
      DATA XSCALE /N*1.0/, FSCALE /M*1.0/

C ***NOTE*** RS(NOF,NOL) AND ZETA(NOF,2) MUST BE DIMENSIONED
C IN SUBROUTINE FORWARD.  ALSO, NOF AND NOL MUST BE ENTERED
C INTO THE PARAMETER LIST OF SUBROUTINE FORWARD.

      CALL DUNLSF (FORWARD,M,N,XGUESS,XSCALE,FSCALE,IPARAM
     +,RPARAM,X,FVEC,FJAC,LDFJAC)

C calculate the sensor parameters based on constraint eqn's:

       R=RS(FING,LINK)
       CALL TRUE (R,X(3),X(4),GAMMAI,BETAI)

       WRITE(15,*)
       WRITE(15,*) 'FINGER:',FING,' LINK:',LINK
       WRITE(15,*) 'R =',R, 'INCHES'
       WRITE(15,*) 'GAMMA =',GAMMAI*180.0D0/3.14159D0,' DEGREES'
       WRITE(15,*) 'BETA =',BETAI*180.0D0/3.14159D0,' DEGREES'

20    CONTINUE

       R=RS(FING,SSEN)
       CALL TRUE (R,X(1),X(2),SGAMMA,SBETA)

       WRITE(15,*)
       WRITE(15,*) 'FINGER:',FING,' LINK:',SSEN
       WRITE(15,*) 'R =',R, 'INCHES'
       WRITE(15,*) 'GAMMA =',SGAMMA*180.0D0/3.14159D0,' DEGREES'
       WRITE(15,*) 'BETA =',SBETA*180.0D0/3.14159D0,' DEGREES'
```

```
      CALL OBJ (NOF,NOL,RS,ZETA,FING,SSEN,SGAMMA,
     +          SBETA,XO,YO,ZO)

      WRITE(15,*)
      WRITE(15,*) 'POSE OF OBJECT (WITH RESPECT TO THE BASE FRAME)'
      WRITE(15,*) ' X-COORDINATE',XO,' INCHES'
      WRITE(15,*) ' Y-COORDINATE',YO,' INCHES'
      WRITE(15,*) ' Z-COORDINATE',ZO,' INCHES'

10    CONTINUE
      CLOSE(15)
      CALL EXIT
      END
```

APPENDIX I

SUBROUTINE TO CALCULATE SENSOR PARAMETERS

```
      SUBROUTINE FORWARD (M,N,X,F)

C THIS SUBROUTINE CALCULATES THE NON-LINEAR FUNCTIONS FOR USE
C IN THE IMSL ROUTINE DUNLSF.

      INTEGER M,N,FING,LINK,NOF,NOL,SSEN
      PARAMETER (NOF=3,NOL=3)
      DOUBLE PRECISION X(N),F(M),RS(3,3)
      DOUBLE PRECISION SSMAT(4,4),LMATI(4,4),LMATJ(4,4)
      DOUBLE PRECISION RADIUS,ZETA(3,2)
      COMMON /OPARAM/ RADIUS
      COMMON /FINGER/ FING
      COMMON /LINK/ LINK
      COMMON /SENSOR_VALUES/ RS
      COMMON /SENSOR_ORIENT/ ZETA
      COMMON /REF_SENSOR/ SSEN

C calculate sensor to object transformations:

      CALL PRODUCT (NOF,NOL,RS,ZETA,N,X,SSMAT,LMATI,LMATJ)

C calculate functions:

      F(1) = SSMAT(1,4) - LMATI(1,4)
      F(2) = SSMAT(2,4) - LMATI(2,4)
      F(3) = SSMAT(3,4) - LMATI(3,4)
      F(4) = SSMAT(1,4) - LMATJ(1,4)
      F(5) = SSMAT(2,4) - LMATJ(2,4)
      F(6) = SSMAT(3,4) - LMATJ(3,4)

C calculate rms values for position functions:

C      PRMS=DSQRT(F(1)**2+F(2)**2+F(3)**2+F(4)**2+
C     +     F(5)**2+F(6)**2)

C print rms values to screen:

C      PRINT *, PRMS

      RETURN
      END
```

APPENDIX J

SUBROUTINE TO CALCULATE OBJECT LOCATION

```
      SUBROUTINE OBJ (NOF,NOL,RS,ZETA,FING,SSEN,SGAMMA,
     &                SBETA,XO,YO,ZO)

C This subroutine calculates the matrix which specifies the
C transformation from the palm frame to the object frame.
C The data used is based on values obtained for the reference sensor.

      INTEGER FING,SSEN,NOF,NOL
      DOUBLE PRECISION RADIUS,XO,YO,ZO,SGAMMA,SBETA,ZETA(NOF,2),
     & OBJECT(4,4),R(4,4),RS(NOF,NOL),F1(4,4),F2(4,4),F3(4,4),
     & FS(4,4),AL(3,3),AA(3,3),DD(3,3),TH(3,3)
      COMMON /DH_PARAMETERS/ AL,AA,DD
      COMMON /JOINT_ANGLES/ TH
      COMMON /FING_BASE/ F1,F2,F3
      COMMON /OPARAM/ RADIUS

      CALL BASSEN (NOF,NOL,FING,SSEN,OBJECT)
      CALL SENOBJ (NOF,NOL,RS,FING,SSEN,ZETA(FING,1),
     &             ZETA(FING,2),SGAMMA,SBETA,R)

      CALL MATMULA (OBJECT,R)
      XO=OBJECT(1,4)
      YO=OBJECT(2,4)
      ZO=OBJECT(3,4)

      RETURN
      END
```

APPENDIX K

SUBROUTINE TO CALCULATE THE TRANSFORMATION FROM THE FINGER BASE FRAME TO THE OBJECT FRAME

```
      SUBROUTINE PRODUCT (NOF,NOL,RS,ZETA,N,X,SSMAT,
     +                    LMATI,LMATJ)

C This subroutine calculates the product of: (trans. from the finger
C base frame to the sensor frame) * (sensor frame orientation matrix)
C * (Sph(gamma,beta,r+R).

      INTEGER I,J,N,FING,LINK,SSEN,TEMP,NOF,NOL
      DOUBLE PRECISION X(N),ZETA(NOF,2),R(4,4)
      DOUBLE PRECISION RADIUS,RS(NOF,NOL)
      DOUBLE PRECISION SSMAT(4,4),LMATI(4,4),LMATJ(4,4)
      COMMON /OPARAM/ RADIUS
      COMMON /FINGER/ FING
      COMMON /LINK/ LINK
      COMMON /REF_SENSOR/ SSEN

      TEMP = FING
      CALL SENOBJ (NOF,NOL,RS,TEMP,SSEN,ZETA(TEMP,1),
     &             ZETA(TEMP,2),X(1),X(2),R)
      CALL BASSEN (NOF,NOL,TEMP,SSEN,SSMAT)
      CALL MATMULA (SSMAT,R)
      CALL SENOBJ (NOF,NOL,RS,TEMP,LINK,ZETA(TEMP,1),
     &             ZETA(TEMP,2),X(3),X(4),R)
      CALL BASSEN (NOF,NOL,TEMP,LINK,LMATI)
      CALL MATMULA (LMATI,R)

      TEMP = TEMP + 1
      IF (TEMP.GT.NOF) THEN
         TEMP = 1
      ENDIF

      CALL SENOBJ (NOF,NOL,RS,TEMP,LINK,ZETA(TEMP,1),
     &             ZETA(TEMP,2),X(5),X(6),R)
      CALL BASSEN (NOF,NOL,TEMP,LINK,LMATJ)
      CALL MATMULA (LMATJ,R)

      RETURN
      END
```

APPENDIX L

SUBROUTINE TO CALCULATE THE TRANSFORMATION FROM THE SENSOR FRAME TO THE OBJECT FRAME

```
      SUBROUTINE SENOBJ (NOF,NOL,RS,FING,LINK,THETA1,
     &                   THETA2,GAMMA,BETA,ROTX)

      INTEGER FING,LINK,NOF,NOL
      DOUBLE PRECISION ROTX(4,4),ROTZ(4,4),PROD3(4,4)
      DOUBLE PRECISION GAMMA,BETA,THETA1,THETA2
      DOUBLE PRECISION RS(NOF,NOL),RADIUS
      COMMON /OPARAM/ RADIUS

C calculate sensor frame orientation:

      ROTX(1,1)=1.0
      ROTX(1,2)=0.0
      ROTX(1,3)=0.0
      ROTX(1,4)=0.0
      ROTX(2,1)=0.0
      ROTX(2,2)=DCOS(THETA1)
      ROTX(2,3)=-DSIN(THETA1)
      ROTX(2,4)=0.0
      ROTX(3,1)=0.0
      ROTX(3,2)=DSIN(THETA1)
      ROTX(3,3)=DCOS(THETA1)
      ROTX(3,4)=0.0
      ROTX(4,1)=0.0
      ROTX(4,2)=0.0
      ROTX(4,3)=0.0
      ROTX(4,4)=1.0

      ROTZ(1,1)=DCOS(THETA2)
      ROTZ(1,2)=-DSIN(THETA2)
      ROTZ(1,3)=0.0
      ROTZ(1,4)=0.0
      ROTZ(2,1)=DSIN(THETA2)
      ROTZ(2,2)=DCOS(THETA2)
      ROTZ(2,3)=0.0
      ROTZ(2,4)=0.0
      ROTZ(3,1)=0.0
      ROTZ(3,2)=0.0
      ROTZ(3,3)=1.0
      ROTZ(3,4)=0.0
      ROTZ(4,1)=0.0
      ROTZ(4,2)=0.0
      ROTZ(4,3)=0.0
      ROTZ(4,4)=1.0
```

```
C calculate Sph(gamma,beta,r+R):

      PROD3(1,1)=DCOS(GAMMA)*DCOS(BETA)
      PROD3(1,2)=-DSIN(GAMMA)
      PROD3(1,3)=DCOS(GAMMA)*DSIN(BETA)
      PROD3(1,4)=(RS(FING,LINK)+RADIUS)*DCOS(GAMMA)*DSIN(BETA)
      PROD3(2,1)=DSIN(GAMMA)*DCOS(BETA)
      PROD3(2,2)=DCOS(GAMMA)
      PROD3(2,3)=DSIN(GAMMA)*DSIN(BETA)
      PROD3(2,4)=(RS(FING,LINK)+RADIUS)*DSIN(GAMMA)*DSIN(BETA)
      PROD3(3,1)=-DSIN(BETA)
      PROD3(3,2)=0.0
      PROD3(3,3)=DCOS(BETA)
      PROD3(3,4)=(RS(FING,LINK)+RADIUS)*DCOS(BETA)
      PROD3(4,1)=0.0
      PROD3(4,2)=0.0
      PROD3(4,3)=0.0
      PROD3(4,4)=1.0

      CALL MATMULA (ROTX,ROTZ)
      CALL MATMULA (ROTX,PROD3)

      RETURN
      END
```

APPENDIX M

SUBROUTINE TO CALCULATE THE TRANSFORMATION FROM THE FINGER BASE FRAME TO THE SENSOR FRAME

```
      SUBROUTINE BASSEN (NOF,NOL,FING,LINK,PROD)

C This subroutine computes the product of the finger base matrix F
C and the sensor transformation matrix.

      INTEGER FING,LINK,NOF,NOL
      DOUBLE PRECISION PROD(4,4),FLPROD(4,4),TEMP(4,4),
     & F1(4,4),F2(4,4),F3(4,4),FS(4,4),AL(3,3),AA(3,3),
     & DD(3,3),TH(3,3),R(4,4)
      COMMON /DH_PARAMETERS/ AL,AA,DD
      COMMON /JOINT_ANGLES/ TH
      COMMON /FING_BASE/ F1,F2,F3

      IF (FING.EQ.1) THEN
          DO 10 I=1,4
          DO 20 J=1,4
          PROD(I,J)=F1(I,J)
20        CONTINUE
10        CONTINUE
       ELSEIF (FING.EQ.2) THEN
          DO 30 I=1,4
          DO 40 J=1,4
          PROD(I,J)=F2(I,J)
40        CONTINUE
30        CONTINUE
       ELSEIF (FING.EQ.3) THEN
          DO 50 I=1,4
          DO 60 J=1,4
          PROD(I,J)=F3(I,J)
60        CONTINUE
50        CONTINUE
       ENDIF

C calculate transformation from the finger base frame to the link
C frame specified by FING and LINK:

      DATA TEMP /1.,0.,0.,0.,0.,1.,0.,0.,0.,0.,1.,0.,0.,0.,0.,1./
      DO 70 I=1,4
      DO 80 K=1,4
         FLPROD(I,K)=TEMP(I,K)
80    CONTINUE
70    CONTINUE

C compute link matrix:
```

```
      DO 90 J=1,LINK
      CALL TRANSFORM (AL(FING,J),AA(FING,J),DD(FING,J),
     +                TH(FING,J),R)
      CALL MATMULA (FLPROD,R)
90    CONTINUE

      CALL MATMULA (PROD,FLPROD)

C compute FS matrix:

       FS(1,1)=1.0
       FS(1,2)=0.0
       FS(1,3)=0.0
       FS(1,4)=-AA(FING,LINK)*0.5
       FS(2,1)=0.0
       FS(2,2)=1.0
       FS(2,3)=0.0
       FS(2,4)=0.0
       FS(3,1)=0.0
       FS(3,2)=0.0
       FS(3,3)=1.0
       FS(3,4)=0.0
       FS(4,1)=0.0
       FS(4,2)=0.0
       FS(4,3)=0.0
       FS(4,4)=1.0

C compute PROD matrix:

       CALL MATMULA (PROD,FS)

       RETURN
       END
```

APPENDIX N

SUBROUTINE TO CALCULATE THE TRANSFORMATION FROM THE SENSOR FRAME TO THE TRIGGER POINT

```
      SUBROUTINE TRUE (RS,GAMMA,BETA,GAMMAS,BETAS)

      DOUBLE PRECISION RS,GAMMA,BETA,FS(4,4),GAMMAS,BETAS

C compute the transformation from the sensor frame to the
C trigger point:

      FS(1,1)=1.0
      FS(1,2)=0.0
      FS(1,3)=0.0
      FS(1,4)=RS*DCOS(GAMMA)*DSIN(BETA)
      FS(2,1)=0.0
      FS(2,2)=1.0
      FS(2,3)=0.0
      FS(2,4)=RS*DSIN(GAMMA)*DSIN(BETA)
      FS(3,1)=0.0
      FS(3,2)=0.0
      FS(3,3)=1.0
      FS(3,4)=RS*DCOS(BETA)
      FS(4,1)=0.0
      FS(4,2)=0.0
      FS(4,3)=0.0
      FS(4,4)=1.0

      IF ((FS(1,4).EQ.0.0).AND.(FS(2,4).EQ.0.0)) THEN
         GAMMAS=0.0
      ELSEIF ((FS(1,4).LT.0.0).AND.(FS(2,4).LT.0.0)) THEN
         GAMMAS=DATAN2(DABS(FS(2,4)),DABS(FS(1,4)))+3.14159D0
      ELSE
         GAMMAS=DATAN2(FS(2,4),FS(1,4))
      ENDIF

      IF (GAMMAS.LT.0.0) THEN
         GAMMAS=GAMMAS+2D0*3.14159D0
      ENDIF

      BETAS=DATAN2(FS(1,4)*DCOS(GAMMAS)+FS(2,4)*DSIN(GAMMAS),FS(3,4))
      RS=DSIN(BETAS)*(FS(1,4)*DCOS(GAMMAS)+FS(2,4)*DSIN(GAMMAS))+
     +FS(3,4)*DCOS(BETAS)

      RETURN
      END
```

APPENDIX O

DATA FOR POSE ESTIMATION EXAMPLE

```
 BLOCK DATA
 DOUBLE PRECISION AL(3,3),AA(3,3),DD(3,3)
 DOUBLE PRECISION F1(4,4),F2(4,4),F3(4,4),ZETA(3,2)
 COMMON /DH_PARAMETERS/ AL,AA,DD
 COMMON /FING_BASE/ F1,F2,F3
 COMMON /SENSOR_ORIENT/ ZETA
 DATA AL /1.5708D0,1.5708D0,1.5708D0,
+         0.0,0.0,0.0,
+         0.0,0.0,0.0/
 DATA AA /1.0D0,1.0D0,1.0D0,
+         1.0D0,1.0D0,1.0D0,
+         1.0D0,1.0D0,1.0D0/
 DATA DD /9*0.0/
 DATA F1 /1.0,0.0,0.0,0.0,
+         0.0,1.0,0.0,0.0,
+         0.0,0.0,1.0,0.0,
+         1.0D0,-1.0D0,5.0D0,1.0/
 DATA F2 /1.0,0.0,0.0,0.0,
+         0.0,1.0,0.0,0.0,
+         0.0,0.0,1.0,0.0,
+         1.0D0,1.0D0,5.0D0,1.0/
 DATA F3 /0.70711D0,0.0,-0.70711D0,0.0,
+         0.0,      1.0, 0.0,      0.0,
+         0.70711D0,0.0, 0.70711D0,0.0,
+         1.0,      0.0, 1.0,      1.0/
 DATA ZETA /1.57080D0,1.57080D0,-1.57080D0,
+               0.0D0,    0.0D0, 3.14159D0/
 END
```