

**MULTICRITERIA INVERSE KINEMATICS
FOR GENERAL SERIAL
ROBOTS**

by

RICHARD NELSON HOOPER, B.S.E.E., M.S.M.E.

DISSERTATION

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

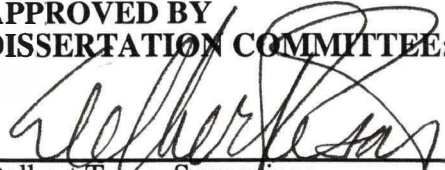
DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

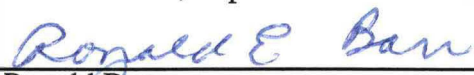
May, 1994

**MULTICRITERIA INVERSE KINEMATICS
FOR GENERAL SERIAL
ROBOTS**

**APPROVED BY
DISSERTATION COMMITTEE:**



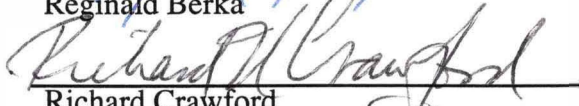
Delbert Tesar, Supervisor



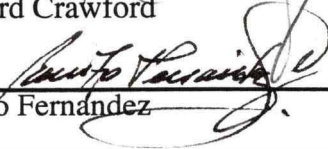
Ronald Barr



Reginald Berka



Richard Crawford



Benito Fernandez

Preface

Understanding the inverse kinematics problem is seductively easy, yet the problem is older than I am. Over the last five decades this problem has attracted the attention of a number of dedicated and gifted researchers. Kinematicians studying spatial mechanisms laid the foundation. Dimentberg in the 1950's and Freudenstein in the 1960's and 1970's were seminal authors. With the realization in the late 1960's that a serial robot could be modeled as a spatial mechanism, the disciplined and analytical theory of mechanisms was applied to the exciting new field of robotics. This work dominated inverse kinematics research during the 1970's as the search for a general closed-form solution became the “Mount Everest” of kinematics problems. Duffy, Pieper, and Roth were at the forefront of inverse kinematics research during this time. In the mid-1980's, the focus began to shift towards redundant robots and optimization methods. The robots of the 1990's, and into the future, will almost certainly have more redundancy, more sensors, and be controlled by ever more powerful computers. We must now solve the inverse kinematics problem while simultaneously making decisions in the face of multiple, possibly conflicting, performance measures derived from mathematical models and sensor feedback.

Rich Hooper

MULTICRITERIA INVERSE KINEMATICS FOR GENERAL SERIAL ROBOTS

Publication No. _____

Richard Nelson Hooper, Ph.D.
The University of Texas at Austin, 1994

Supervisor: Delbert Tesar

This work applies to all serial robots and allows for the incorporation of an unlimited number of performance criteria. This work includes:

- a formal statement of the multicriteria inverse kinematics problem,
- the development of a general multicriteria inverse kinematics method,
- experimental and simulation results.

Essentially, the inverse kinematics problem for a serial robot is to find the displacements of the robot's joints given position and orientation constraints on the placement of the robot's terminal link. Current industrial robots typically have six independent joints since six constraints completely specify the position and orientation of a rigid body in space. This gives current robots spatial tool placement capabilities.

Modern research robots often have more than six joints. For these robots the inverse kinematics problem is also a type of optimization problem. Current inverse kinematics efforts for these robots focus on solutions that minimize joint speeds, system kinetic energy, joint accelerations, or joint torques. The main problem with these formulations is that they only optimize a single performance criterion at a time. This is not enough to address the complexity of a modern robot.

The development of a multicriteria inverse kinematics method is one of the principal contributions of this work. This method uses a series of joint-level perturbations to generate a set feasible options for the robot's motion. A sophisticated decision making strategy evaluates these options based on multiple performance criteria and chooses one option as the next motion command for the robot's controller. This inverse kinematics method:

- applies to all serial robots,
- may incorporate an unlimited number of performance criteria,
- is effective throughout the robot's work space.

Experimentation and simulation validated the effectiveness of this inverse kinematics method for a number of different robots. Each of the robots in these examples has especially interesting or difficult geometries. The robots cover a spectrum, from an industrial robot with six joints to a conceptual robot with twenty-one joints. The results include the computational speed and accuracy of the solution for each of the example robots.

Table of Contents

Glossary	xi
Chapter 1 Introduction	1
Inverse Kinematics.....	3
Overview	6
Chapter 2 Literature Review	11
Mechanisms	17
Dimentberg's Contributions.....	17
Sequential Filters	18
Kinematic Influence Coefficients	19
Freudenstein's Contributions	20
Fully-Constrained Robots	23
Duffy's Contributions.....	24
Newton-Raphson Numerical Method	26
Predictor-Corrector Numerical Method.....	28
Modified Newton-Raphson and Modified Predictor Corrector	29
Table Look-up.....	30
Redundant Robots.....	31
Whitney's Contributions	31
Pseudoinverse	33
Extended Jacobian	36
Neural Nets	39
Genetic Algorithms.....	41
Performance Criteria.....	43
Chapter 3 Inverse Kinematics by Local Optimization.....	58
Analytical Solutions	60
Position-level Optimization	61
Minimizing Joint Speeds.....	62

Minimizing Kinetic Energy	64
Minimizing Joint Accelerations	66
Minimizing Joint Torques.....	67
Direct Search.....	69
Transformation of End-effector Constraints.....	71
Exploration Patterns.....	74
Decision Making.....	77
Optimality	81
Relative Computational Complexity.....	87
Recommended Direct Search Applications	88
Chapter 4 Multicriteria Inverse Kinematics.....	91
Problem Definition.....	93
Direct Search.....	101
Non-Dominated Solutions	105
Composite Performance Index.....	109
Sequential Filters	121
Factorial Design Pattern.....	128
Simulated Annealing.....	133
Conservative Paths.....	140
Comparison of Decision Making Strategies	144
Chapter 5 Automatic Plant Description	149
Inputs and Outputs	150
Specifications	153
Subsystem Identification.....	158
Calculation Subsystem.....	162
Database Subsystem.....	169
Model Formulation Subsystem.....	174
Implementation	182
Example	184

Chapter 6 Experimental and Simulation Results	188
Robotics Research kb2107.....	189
Lobster's Arm.....	202
Puma	206
Alpha.....	209
Titan 7F.....	213
Wafer Handling.....	214
Dexterous Robot with Nine Degrees of Freedom.....	216
Fault Tolerant Robot with Twenty-one Degrees of Freedom.....	220
Chapter 7 Road Map	228
Advanced Applications.....	228
Complex Assembly.....	230
Food Packaging.....	231
Micromanipulation.....	233
In-Space Assembly and Construction	234
Satellite Retrieval and Repair	236
EVA Site Preparation.....	237
Semiconductor Wafer Handling	238
Long Reach Arm.....	240
Inspection at Nuclear Facilities.....	242
Decontamination and Dismantlement.....	243
Supporting Technologies	246
Computer Hardware.....	247
Computer Architecture.....	255
Computer Software	257
Performance Criteria Development	260
Sensors	264
Data Communication	267
Control	269
Mechanical Architecture.....	271
Mechanical Design.....	273

Robot Actuators	274
Prioritization of the Component Technologies	276
Chapter 8 Conclusions	279
References.....	288
Vita	

Glossary

acceleration-level Mathematical formulations working with the change in joint speeds with respect to time. Integrating accelerations twice provides displacements. See position-level and velocity-level.

analytical methods Purely mathematical methods that do not require iteration.

closed-form - A problem formulation that does not require iteration for its solution.

conservative motion - A path where both the end-effector and the joints repeatedly follow their same respective trajectories.

degrees of freedom The number of independent variables in the system. Each joint in a serial robot represents a degree of freedom.

dexterity - A measure of the robot's ability to follow complex paths.

direct search A method of solving problems numerically using sets of trial solutions to guide a search. The search is direct because it does not explicitly evaluate derivatives.

dynamic model - A mathematical model describing the motions of the robot and the forces that cause them.

end-effector space - A fixed coordinate system referenced to the base of the robot.

end-effector The robot's last link. The robot uses the end-effector to accomplish a task. The end-effector may be holding a tool, or the end-effector itself may be a tool. The end-effector is loosely comparable to a human's hand.

equality constraint A restriction that requires the displacement or motion of the robot to equal a specified value. Equality constraints specify the position and orientation of the robot's end-effector.

error function The error function assigns a single value that represents the difference between the desired and actual values of one or several dependent variables.

fully constrained robot A robot with as many independent joints as there are equality constraints on the placement of the end-effector.

inequality constraint A restriction that limits the value of a dependent or independent variable. Inequality constraints limit the robot's joint travels (joint limits), joint speeds (speed limits), and torques, (torque limits).

inverse kinematics The inverse kinematics problem is to find the robot's joint displacements given position and orientation constraints on the robot's end-effector.

iteration - Repeatedly applying a series of operations to progressively advance towards a solution.

Jacobian - The matrix of first-order partial derivatives. For robots, the Jacobian relates the end-effector velocity the joint speeds.

joint space - A coordinate system used to describe the state of the robot in terms of it's joint states. Inverse kinematics may also be thought of as a mapping from end-effector space to joint space.

kinematic influence coefficients - These coefficients describe the total influence the N input joints have on the motion of the robot and allow a direct statement of the complex and coupled nonlinear differential equations controlling the response of the system.

LaGrange multipliers A mathematical technique for transforming equality constraints into performance criteria, thus expressing a constrained problem as an unconstrained problem.

linearly dependent A correspondence between quantities or functions that can be described by simply adding, subtracting, or multiplying a scalar.

normalize - Scaling a number of factors so that they will be of similar magnitudes.

numerical methods Iterative methods of solving problems on a computer. Numerical methods may have an analytical basis or they may involve heuristics.

optimization Calculating the independent variables in a function so as to generate the best function value for a given set of conditions. Optimization usually involves maximizing or minimizing a function.

performance criterion Measures based on kinematic and dynamic models of the robot useful for evaluating the state of the robot.

plant description A kinematic and dynamic model of the robot.

position-level Mathematical formulations working with the joint displacements. See acceleration-level and velocity-level.

pseudoinverse - A simple method of inverting a matrix that is not square. As commonly applied to redundant robots, the pseudoinverse minimizes the two-norm of the joint speeds.

redundancy More independent variables than constraints.

resolved-rate - An extremely simple inverse kinematics method at the velocity-level.

scale - Changing magnitude by linear operation, i.e. multiplying by a scalar.

self-motion The robot's ability to move its intermediate links while holding the placement of the end-effector constant.

serial robot A serial robot is a single chain of joints connected by links.

singularity A position in the robot's workspace where one or more joints no longer represent independent controlling variables. Commonly used to indicate a position where a particular mathematical formulation fails.

two-norm The square root of the sum of the squares. The magnitude of a vector.

velocity-level Mathematical formulations working with the joint speeds. Integrating the joint speeds once provides the displacements. See acceleration-level and position-level.

Chapter 1: Introduction

The goal when this work began was the development of an **inverse kinematics*** method for modular robots. Modular robots may assume an almost limitless number of different geometric configurations. Thus, generality with respect to the robot's geometry was requisite. There was also a growing belief that advanced robots would incorporate a wide variety of **performance criteria** during their operation. This led to the specification that the inverse kinematics method should consider performance criteria and include them in a general fashion as well. The insistence on generality led this work into becoming much more than an inverse kinematics method for modular robots. This dissertation details an inverse kinematics method applying to all **serial robots**.

Figure 1.1 shows a serial robot. A serial robot is a single chain of joints connected by links. When the joint displacements change, the links move relative to one another. A general serial robot has no a priori constraints on the size or shape of the links. The inverse kinematics problem for a serial robot is: given the position and orientation of the last link relative to the first link, find the joint displacements. There is almost always more than one set of joint displacements satisfying the position and orientation specifications. The robot's controller must choose a single solution based on one or more performance criteria.

* The glossary defines words appearing in bold.

The robot's last link is its **end-effector** and its first link is its base. The motion of the end-effector relative to the base specifies the robot's task. The robot moves its joints to move the end-effector and accomplish its task. Inverse

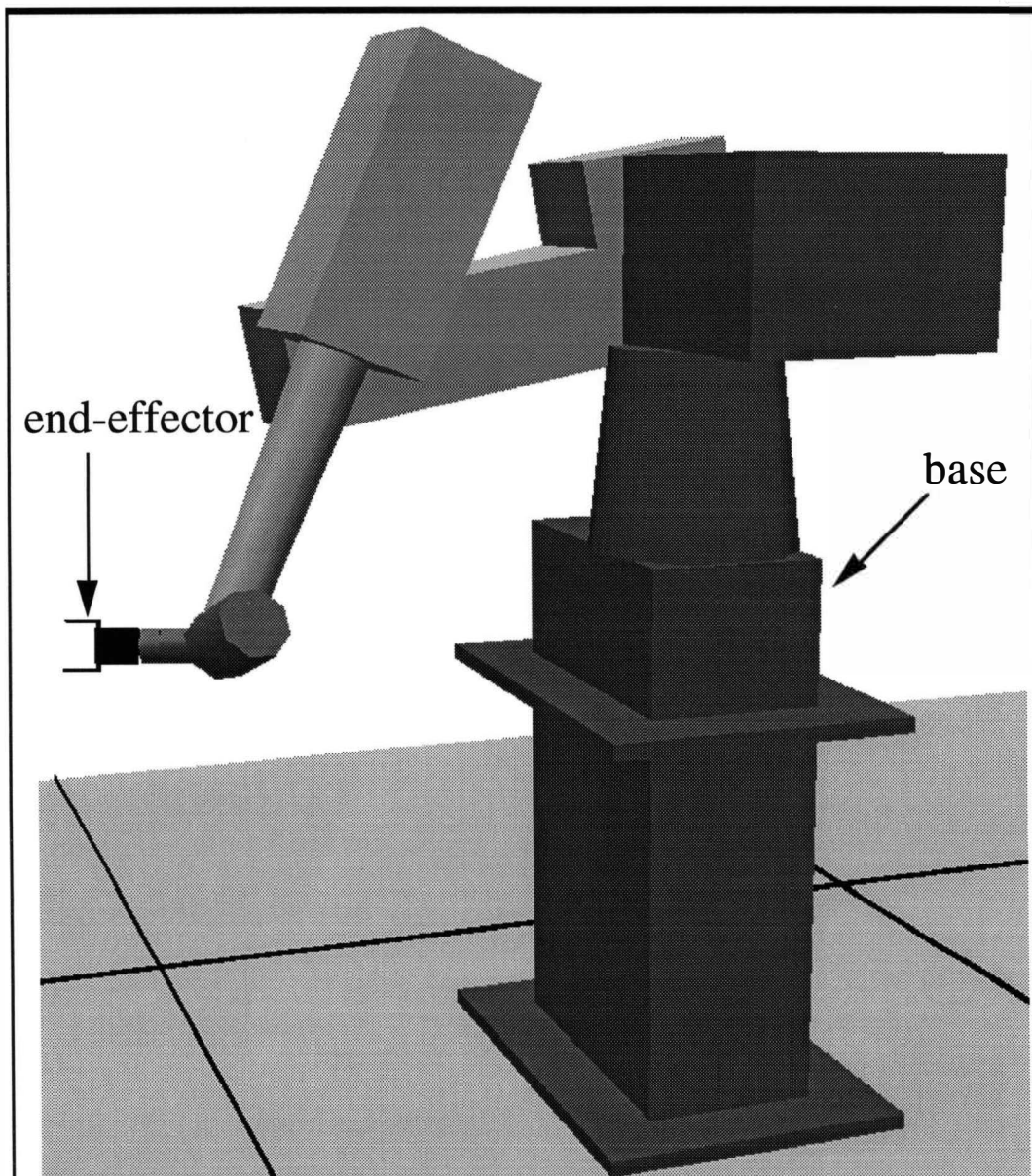


Figure 1.1 A serial robot.

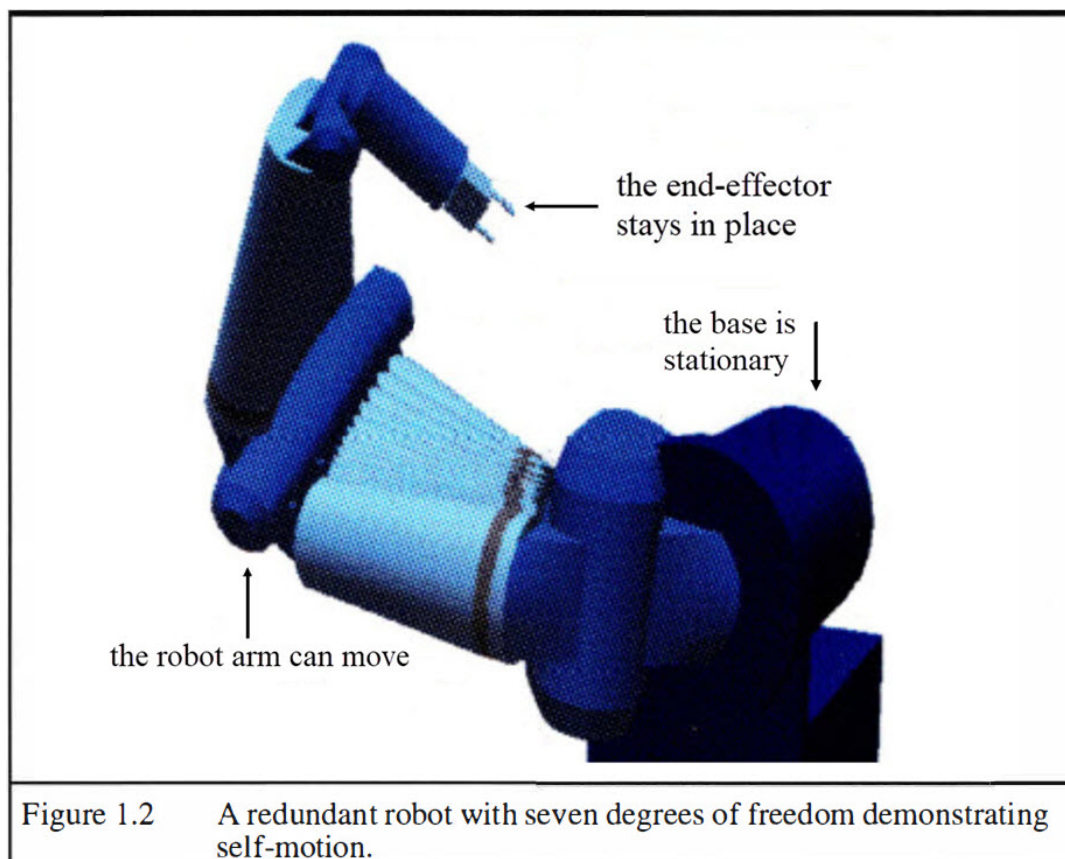
kinematics translates the task specification from end-effector motion to joint motion. A robot cannot do useful work without some form of inverse kinematics.

1.1 INVERSE KINEMATICS

Position and orientation constraints specify the placement of the end-effector relative to the base. The inverse kinematics problem is to find the set of joint displacements satisfying these constraints. There are two different types of robot joints. A revolute joint turns about an axis. An angle specifies the displacement of a revolute joint. A prismatic joint slides along an axis. A length specifies the displacement of a prismatic joint. All other types of robot joints may be described by various combinations of these two joints. Every joint gives the robot a degree of freedom. The different joints and the size and shape of the links determine the robot's geometry. Some inverse kinematics techniques solve for the change in joint displacements with respect to time. These **velocity-level** solutions find the joint speeds and then integrate them once to get the joint displacements. Other inverse kinematics techniques solve for the change in joint speeds with respect to time and then integrate twice to get the joint displacements. These are **acceleration-level** solutions. **Position-level** solutions find the joint displacements directly.

A **fully constrained robot** has as many **degrees of freedom** as there are constraints on the placement of the end-effector. There are a finite number of solutions to the inverse kinematics problem for a fully constrained robot. A robot with more degrees of freedom than constraints is redundant. There can be at most six constraints, three on the position and three on the orientation, on the placement

of the end-effector so any robot with more than six degrees of freedom is redundant. There is typically an infinite number of solutions to the inverse kinematics problem for a redundant robot. Thus, a redundant robot can hold the placement of the end-effector constant while still moving the intermediate links. This is called **self-motion**. Figure 1.2 shows a redundant robot with seven degrees of freedom exhibiting self-motion.



For redundant robots, the inverse kinematics problem is also a type of **optimization** problem. The inverse kinematics algorithm uses performance criteria to choose the best solution. Performance criteria are based on kinematic

and dynamic models of the robot. Often criteria measure the performance of each joint individually. For the whole robot, these criteria are vector quantities. A norm expresses these vector quantities as scalars. The **two-norm** is the square root of the sum of the squares, which is the magnitude of the vector.

Singularities are extremely important in inverse kinematics. A **singularity** occurs when two or more of the robot's joint motions are **linearly dependent**. In this case the normal method of mathematical inverse fails due to a reduction in structure; i. e., a row in the matrix describing the system's input to output relationships disappears. This is true any time two revolute or two prismatic joint axes are collinear. The joints involved in the singularity are no longer independent degrees of freedom. Most inverse kinematics solutions are undefined at singularities. The **direct search** method developed in this dissertation does not suffer from this problem. A fully constrained robot loses its ability to move the end-effector in at least one direction at singularities. Redundant robots lose at least one degree of **redundancy** at singularities, but because they have extra degrees of freedom, they still may be able to move the end-effector in all directions.

Modular robots are made from self-contained joints with one, two, or three degrees of freedom and generic links (Tesar and Butler, 1989a). Each joint or link is a module. The modules may be assembled into a large number of different robot configurations. Tesar and Butler describe a wide variety of benefits associated with modular robots. Because modular robots may assume a large number of different configurations, they demand a general inverse kinematics approach.

Generalized inverse kinematics addresses the inverse kinematics problem in the most general fashion. A generalized formulation of the inverse kinematics problem must make no a priori constraints on the robot's geometry or on the number or type of performance criteria treated by the solution. This dissertation formulates direct search as a solution to the generalized inverse kinematics problem.

1.2 OVERVIEW

The introduction presents a definition of the inverse kinematics problem and discusses its fundamental importance to robotics. The definition and discussion introduce the terminology used throughout this dissertation.

Chapter 2 is a critical literature review. This review describes, and discusses the merits of, important work presented in the literature. By tracing the development of inverse kinematics from its origins in mechanism theory to its present application with redundant robots and optimization, this chapter also provides some historical perspective. This chapter covers both **analytical** and **numerical methods**.

Chapter 3 begins by deriving several analytical solutions to the inverse kinematics problem using optimization methods. Each of these derivations employs **LaGrange multipliers** to express the constrained problem as an unconstrained problem. The derivations use calculus for the actual minimization. The chapter then develops an analogous numerical optimization method for solving the inverse kinematics problem. The method expresses the inverse kinematics problem as an unconstrained problem using an **error function** to

transform the position and orientation constraints on the end-effector. Direct search minimizes the error function. By minimizing the error to zero, the direct search finds a solution to the inverse kinematics problem. This method is general with respect to the robot's geometry and stable at singularities. Development of the direct search method and its successful application to the inverse kinematics problem is the primary contribution of this work.

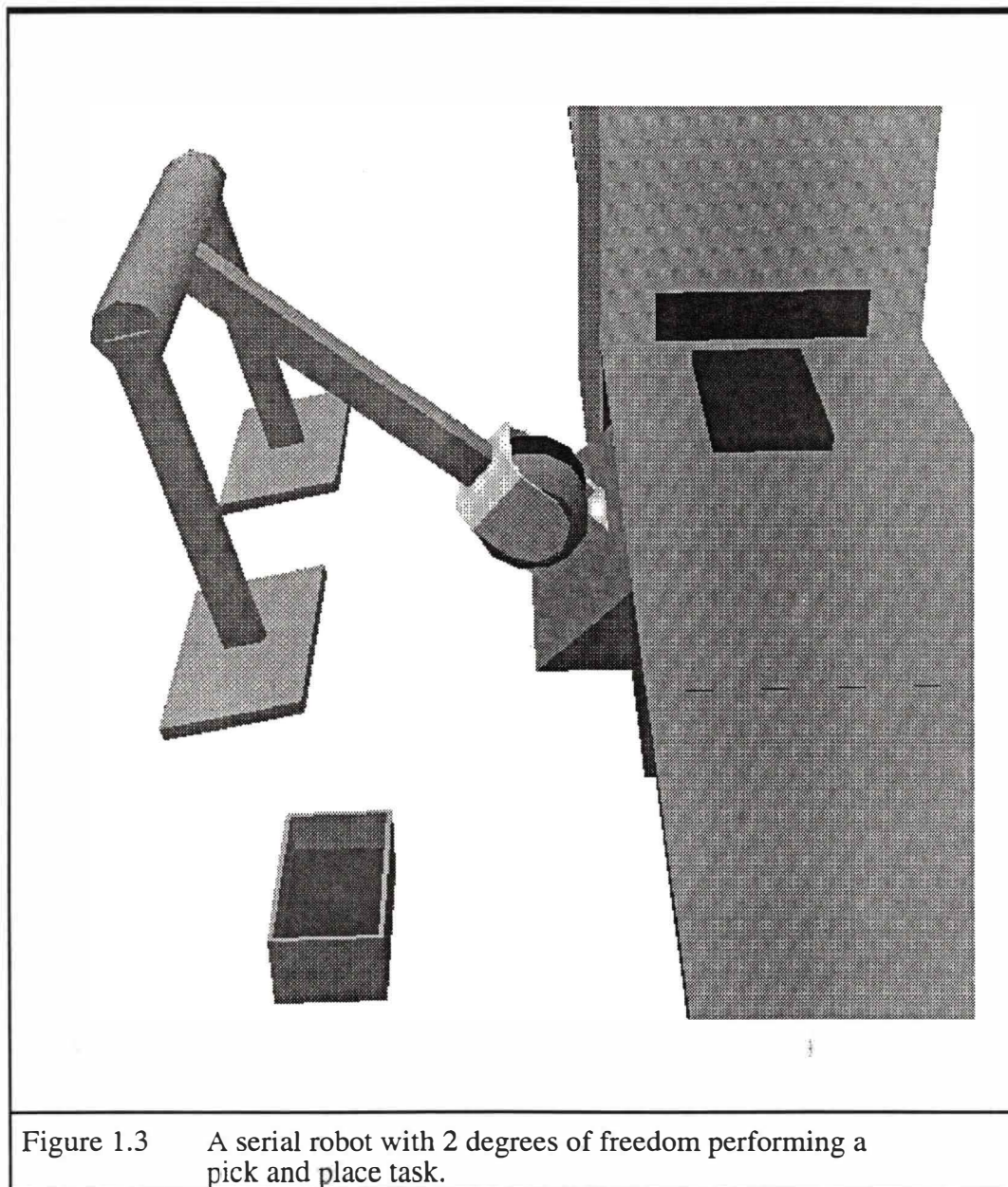
Chapter 4 extends the direct search method to address the multicriteria inverse kinematics problem. The chapter begins by defining the multicriteria inverse kinematics problem as an optimization problem with an unlimited number of performance criteria, **inequality constraints**, and **equality constraints**. The definition is general and concise. Specifically, it does not constrain the definition of optimality. This chapter discusses two different types of optima and develops direct search methods of finding both. The direct search method can explicitly calculate and consider any number and variety of performance criteria during the solution. The only constraint is that the performance criteria must be single-valued functions of the joint displacements and/or their derivatives with respect to time.

Chapter 5 details the systematic design of an automatic **plant description** system for robots. The plant description is essentially a dynamic model of the robot. The design process loosely follows the method of Pahl and Beitz (1988). This design method leaves a structured record of the goals, constraints, and specifications that influenced the design. In its current form, this plant description system calculates performance criteria values, joint forces or torques, and deflections of the robot's end-effector. The system is automatic because it

produces its outputs given its inputs with no further user intervention. A generalized inverse kinematics implementation must include a plant description system. Generalized inverse kinematics loses many of its associated benefits if the plant description system is not general and automatic. The automatic plant description system is also useful for: designing robots, choosing monolithic robots, configuring modular robots, and enhancing robot control.

Chapter 6 presents experimental and simulation results for several applications of the direct search inverse kinematics method. The robots in these examples have especially difficult or interesting geometries. They cover a spectrum, from an industrial robot with six degrees of freedom to a conceptual robot with twenty-one degrees of freedom. Most of the examples include a graph showing the accuracy of the inverse kinematics solution and timing results for the solution speed.

Chapter 7 maps the future of generalized inverse kinematics. This chapter is a road map for a road that does not yet exist. Mapping roads into the future is a risky activity. This map is based on ten advanced robot applications with the hope that applications will encourage relevance into the future. Each application possesses a degree of difficulty that likely requires generalized inverse kinematics. Applying generalized inverse kinematics, in turn, requires an array of supporting technology. This chapter discusses ten supporting technologies as they relate to generalized inverse kinematics and the application of advanced robots.



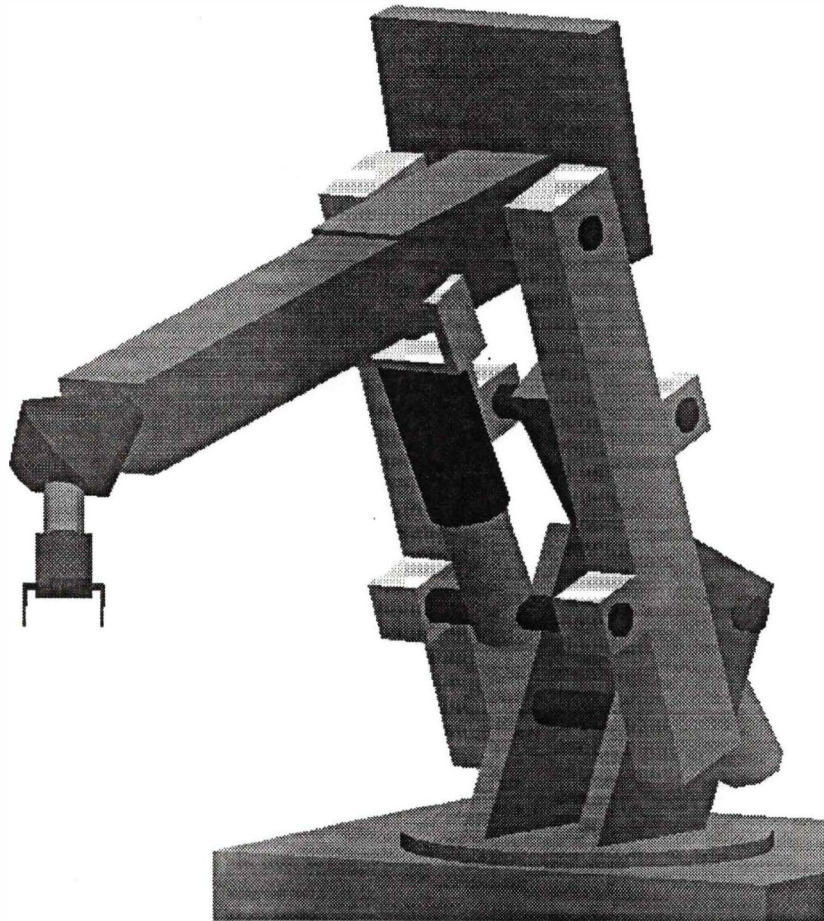


Figure 1.4 A common industrial robot often used for welding.

Chapter 2: Literature Review

Inverse kinematics is fundamental to robotics and has been extensively studied. Kinematicians whose mechanism theory provided elegant **closed-form*** solutions influenced the early years. Digital computers made the numerical solution of these complex nonlinear kinematic equations possible as well. During the 1970's, the search for an entirely general closed-form solution dominated inverse kinematics research. During the 1980's, the widespread availability of redundant research robots and powerful computers focused attention on optimization and **pseudoinverse** techniques. The robots of the 1990's will have more sensors, more redundancy, and be controlled by computers more powerful than ever before. The computers made by the Silicon Graphics company, for example, have shown a price-to-performance improvement of two hundred times in only five years. The inverse kinematics problem must now incorporate multiple performance criteria so that the robot can be in the best possible kinematic configuration while performing its task.

The synthesis of path-generating mechanisms is a close relative of the inverse kinematics problem. In both cases, kinematic displacements – link lengths for mechanism synthesis and joint displacements for inverse kinematics – are derived such that an end-effector will follow a prescribed path. By modeling a robot with six degrees of freedom as a mechanism with a mobility of one,

* The glossary defines words appearing in bold.

kinematicians applied mechanism analysis techniques developed during the 1950's to the inverse kinematics problem. Freudenstein and Roth are two researchers from mechanism analysis instrumental in the development of closed-form inverse kinematics. Though Freudenstein once declared the closed-form solution for a general robot with six degrees of freedom the "Mount Everest" of kinematic problems (Freudenstein, 1973), both Freudenstein and Roth published articles describing numerical solutions to the mechanism analysis problem; Freudenstein as early as 1959 (Freudenstein and Sandor, 1959)(Roth and Freudenstein, 1963). Other techniques from mechanism analysis have found application in robotics. **Kinematic influence coefficients** developed by Benedict and Tesar (1978) for the analysis of multi-degree of freedom planar mechanisms were later formulated for serial spatial robots by Thomas and Tesar (1982). The Kinematic influence coefficients are entirely general, applying to robots with any number of degrees of freedom. A physical plant description for the **dynamic model** is readily formulated in terms of Kinematic influence coefficients, which recommends their use for generalized inverse kinematics. A generalized inverse kinematics strategy may also exploit the sequential-filters method developed by Eschenbach and Tesar (1969) for mechanism synthesis.

A typical industrial robot has six degrees of freedom. This is because there are six spatial parameters necessary to specify the position and orientation of a rigid body in space. With the appropriate geometric relations the six independent variables, the joint displacements, are found in terms of the six parameters constraining the end-effector's displacement. Roth and other researchers have shown that there are a finite number of solutions to the inverse

kinematics problem for a fully constrained robot (Roth et al., 1973). Much thoughtful research has been devoted to finding these solutions in closed form. This research generated books full of closed-form inverse kinematic solutions for a vast array of specific robots (Roth, 1979) (Duffy, 1980) (Angeles, 1982). Since closed-form solutions are computationally efficient, most industrial robots are designed to have known closed-form solutions. Modern applications of closed-form inverse kinematics include a robot designed for the Space Station Remote Manipulator System (Crane and Duffy, 1991).

Table 2.1 Several influential researchers, their interests, and some of their contributions to inverse kinematics.		
Researcher	Interest	Contributions to Inverse Kinematics
Dimantberg 1940-1950	analysis of spatial mechanisms	dual number quaternions screw algebra framework for spatial mechanism analysis
Freudenstein 1950-1970	mechanism synthesis	loop equations vector analysis the “Mount Everest” declaration
Duffy 1970-present	6 DOF robots	general approach systematic and unified work closed solutions for most 6 DOF geometries
Whitney 1960-1980	human prostheses	resolved rate application of pseudoinverse

Numerical methods have long been applied to the inverse kinematics problem. The Newton-Raphson technique used by Uicker, Denavit and Hartenberg is generally cited as the first time numerical methods solved the

inverse kinematics problem and is still in wide use. The predictor-corrector technique, also used for solving differential equations, can solve the inverse kinematics problem. Standard optimization software, such as Minpack 1, and standard numerical techniques, such as Powell's method, can also solve this problem.

Most researchers discuss singularities when reporting on inverse kinematics methods. Physically, a singularity occurs when the robot loses one or more inputs as independent controlling parameters. A common singularity arises when two revolute axis are collinear. There are also singularities at workspace boundaries. A robot with six degrees of freedom loses at least one degree of end-effector **dexterity** at singularities. All robots can exhibit singularities and it is, in general, impossible to avoid them by simply restricting the workspace (Maciejewski and Klein, 1988). Thus, a generalized inverse kinematics solution should be robust in the presence of singularities.

During the 1980's, attention began to shift towards kinematically redundant robots and the nature of the research changed dramatically. Redundant robots are typically able to achieve a given end-effector displacement with an infinite combination of joint displacements. This gives rise to the possibility of incorporating performance criteria in order to improve the robot's operation while satisfying the displacement constraints on the end-effector. Virtually all of the inverse kinematics methods for redundant robots are extensions of Whitney's "**resolved-rate**" work (Whitney, 1969). Rather than solving the position-level equations, the resolved rate method uses the **Jacobian** matrix while solving for the joint speeds. The Jacobian matrix relates the end-effector velocity to the joint

speeds. The Jacobian matrix for redundant robots is not square because there are more joints than end-effector constraints. Integrating the joint speed gives the displacements. The Jacobian-based techniques typically fall into either the pseudoinverse or the extended Jacobian inverse category.

The pseudoinverse minimizes the two-norm of the vector of joint speeds. Though the pseudoinverse is very commonly used for redundant robots, there are some serious drawbacks. The simple-mindedness of only considering the joint speeds leads pseudoinverse solutions into a variety of traps. A common problem cited with the pseudoinverse is that it is not conservative. That is, when the end-effector follows a path that returns to its starting place, the joint displacements may not return to their starting place. This will cause the robot to drift in **joint space**, though still satisfying the end-effector constraints. The drift can often lead to a singularity. The drift is due to the loss of position-level information during the pseudoinverse of the Jacobian. It is entirely possible that the same two-norm of joint speeds can be achieved with the robot in an infinite number of different positions. This difficulty with position-level criteria has led to published suggestions for abandoning the pseudoinverse altogether if position-level criteria are to be incorporated in the solution (Carnigan, 1991) (Egeland et al., 1991). This is a very serious problem. Position-level geometry is fundamental to understanding the state of the robot. The very practical limits on joint displacements are at the position level as well. The pseudoinverse breaks down theoretically at singularities. Due to the finite numerical resolution of digital computers, the pseudoinverse also fails near singularities. Self-motion is the most common way researchers have attempted to alleviate the problems with the

pseudoinverse. By choosing the self-motion, researchers have incorporated other performance criteria into the solution. Self-motion has been applied to singularity avoidance (Bedrossian and Flueckiger, 1991), kinetic energy minimization (Dubey and Lu, 1988), dexterity optimization (Martin and Satterlee, 1987), joint limit and obstacle avoidance (Nenchev, 1989), and the incorporation of task-independent criteria (Cleary and Tesar, 1990). Researchers have also used self-motion to alleviate the problems associated with an ill-conditioned Jacobian near a singularity (Maciejewski and Klein, 1988).

The extended Jacobian method is another solution to the inverse kinematics problem for redundant robots. Rather than using the pseudoinverse to a non-square Jacobian, the extended Jacobian method makes the Jacobian square by adding constraints. This method has been used for avoiding obstacles and joint limits (Baillieul, 1986). The extended Jacobian method also breaks down theoretically exactly at singularities and fails numerically near singularities. The additional constraints exacerbate the problem by adding algorithmic singularities to the robot's geometric singularities.

Criteria for performance evaluation are fundamental to the operation of advanced robots. Combined with sensory input, these criteria can be used for enhancing the performance of the robot. Cleary and Tesar (1990) have proposed a set of criteria that are based on the physical parameters of the robot and are independent of the task. These criteria do not need to be reformulated each time the robot performs a different task. Sensory information and performance criteria must be ranked, **scaled**, **normalized**, and fused for intelligent use by a generalized inverse kinematics algorithm.

2.1 MECHANISMS

For hundreds of years, mechanisms have transmitted cyclic forces and motions. Mechanisms are found in a variety of mechanical systems, including: sewing machines, typewriters, automobiles, computer drives, and advanced semiconductor process tools. The complex geometry of mechanisms and their practical applications have long fascinated engineers and mathematicians. Steady research over the course of several decades extended the theory of mechanisms from the planar case to the spherical case, until finally advancing to the spatial case. Ultimately, the theory of spatial mechanisms provided the analytical foundation upon which inverse kinematics for serial robots was built.

Mechanism theory is applied to inverse kinematics by modeling the robot as a spatial mechanism. Imagine a robot turning the doorknob of a closed door. The doorknob represents a revolute joint. The base of the robot is connected to the ground, and so is the door. As it is turning the doorknob, the serial robot actually becomes a closed-loop spatial mechanism. A robot with six degrees of freedom has just enough dexterity to put a pure rotation on the doorknob. Thus, a robot with six degrees of freedom corresponds to a spatial mechanism with mobility one.

2.1.1 Dimentberg's Contributions

Dimentberg's application of dual-number quaternions and screw algebra to the analysis of spatial mechanisms during the late 1940's is seminal in the field of inverse kinematics. This work laid the mathematical foundation for much of the mechanism analysis that was later applied to the inverse kinematics of serial

robots. The interested reader is directed to the original works of Dimentberg and others (Dimentberg, 1948) (Yang and Freudenstein, 1964).

A quaternion is a set of four real numbers: d , a , b , and c . These numbers are ordered and associated with four units: $+1$, i , j , and k . The unit $+1$ has the same properties as the real number 1. The properties of i , j , and k are:

$$i^2 = j^2 = k^2 = -1 \quad (2.1)$$

and

$$ij = k, ji = -k \quad (2.2)$$

The quaternion q is written:

$$q = d + ai + bj + ck \quad (2.3)$$

The i , j , and k units are essentially the base vectors of a Cartesian frame.

A screw, L , is simply an axis in space associated with a pitch magnitude. The screw coordinates are a set of six numbers defining the axis and the pitch.

$$L = (L_1, L_2, L_3, L_4, L_5, L_6) \quad (2.4)$$

A screw system is closed under addition and multiplication. That is, the sum or product of any sets of screws in the system will remain in the system.

Dual-number quaternions and screw algebra provided a framework for analyzing mechanisms. This framework facilitated the extension of mechanism analysis from the plane to the sphere and ultimately from the spherical to the spatial case.

2.1.2 Sequential Filters

Eschenbach and Tesar (1969) proposed sequential filters for reducing the design space of the coplanar mechanism synthesis problem. More recently,

Cleary and Tesar (1990) incorporated multiple performance criteria in a solution of the inverse kinematics problem for redundant robots using sequential filters.

Eschenbach and Tesar's application of sequential filters to mechanism synthesis specified an initial criterion, all linkages satisfying the motion of a coupler point through four point locations, to generate as many as 60,000 different linkages. This large design space was sorted by means of zones and given a design criterion number to reduce the solution space and provide the designer with a manageable range of solutions.

The concept of sequential filters is general and may be applied to the inverse kinematics problem for redundant robots. Eschenbach and Tesar's manuscript classified the design zones as necessary or desirable. In the context of the inverse kinematics problem, meeting the end-effector constraints is necessary, while optimizing performance criteria is desirable. The necessary conditions reduce the range of solutions. The final solution is chosen according to the desirability. Chapter 4 of this dissertation includes a more thorough discussion of the application of sequential filters to the generalized inverse kinematics problem.

2.1.3 Kinematic Influence Coefficients

A generalized inverse kinematics implementation must include criteria based on joint torques and forces. Calculating these torques and forces requires dynamic modeling and analysis. Kinematic influence coefficients are a concise vehicle for formulating both the dynamic model and the necessary computational sequence. The controlling equations explicitly maintain the system parameter and input/output dynamics.

Benedict and Tesar (1978) developed kinematic influence coefficients as a generalized means of modeling an N degree of freedom planar mechanism. Their model formulation quantitized the mechanism geometry in terms of Assur subgroups. As formulated, their model includes the analysis of complex mechanisms with multiple inputs. The kinematic influence coefficients are the heart of the formulations. These coefficients describe the total influence the N inputs have on the motion of the mechanism and allow a direct statement of the complex and coupled nonlinear differential equations controlling the response of the system. The complete system model includes externally applied loads, system mass, compliance and damping terms. Later, a similar modeling and computational technique was applied to serial robots.

Thomas and Tesar (1982) formulated a dynamic rigid-link model for serial robots using kinematic influence coefficients. The model formulation is entirely general and based solely on the geometry and physical properties of the robot. The model expresses the robot dynamics referenced to the joints of the robot.

2.1.4 Freudenstein's Contributions

Ferdinand Freudenstein greatly influenced the course of inverse kinematics research. A brief accounting of his work on the displacement analysis of mechanisms during the 1950's and 1960's, and its transition to inverse kinematics during the early 1970's, provides a unique insight into the foundations of the inverse kinematics.

During the 1950's Freudenstein was working on the synthesis of path-generating mechanisms for motion conversion and force transmission. Several of

the ideas he published continue to permeate inverse kinematics research. In particular, the insistence on finding all existing solutions to a given problem is still pervasive; so much so that one respected introductory robotics text will not even admit that an algorithm has solved the inverse kinematics problem unless it is guaranteed to find all of the possible solutions (Craig, 1989). During the latter part of the fifties, Freudenstein worked on the synthesis of path-generating mechanisms using vector analysis. The equations were formulated so as to be solved by computer (Freudenstein and Sandor, 1959). Although a computer was used, the equations were still solved in closed form without **iteration**. During this time, Freudenstein espoused the idea of developing kinematic tools for use by a designer of mechanisms. The idea of using a set of tools for inverse kinematics analysis was also common during the 1970's and 1980's.

The geared five-bar was a mechanism that attracted Freudenstein's interest during the 1960's. Freudenstein's attempts at analyzing this mechanism produced equations with "a degree of complexity which renders closed-form algebraic solutions unobtainable" (Freudenstein and Roth, 1963). This forced Freudenstein to resort to an iterative Newton-Raphson numerical solution procedure. The Newton-Raphson procedure remains to this day a common method of solving the inverse kinematics problem. In 1964 Yang and Freudenstein applied dual-number quaternion algebra to the analysis of spatial mechanisms. The dual-number quaternion formulation allowed the derivation of explicit closed-form algebraic expressions solvable without iteration. At this time, Freudenstein expressed his opinion that, though complex, dual-number quaternion algebra could eventually become a tool for the displacement analysis of spatial mechanisms.

In the early 1970's, Yuan, Freudenstein, and Woo applied screw coordinates to the analysis of spatial mechanisms. They believed that screw algebra could solve for the displacements, velocities, accelerations, and forces of a general 7 link spatial mechanism (Yuan et al., 1971). The 7 link spatial mechanism represents the transition where mechanism analysis also includes serial robots. For the general seven-link spatial case, Freudenstein was forced to resort to numerical integration to solve for the displacements.

It was in 1972 that Freudenstein delivered an address entitled “Kinematics: Past, Present and Future” at the 12th Mechanisms Conference of the American Society of Mechanical Engineers. This address (and its subsequent publication) influences inverse kinematics research to this day. In it, Freudenstein declared the “closed form, algebraic solution of the input/output displacement equation of the general, single-loop, single-degree-of-freedom spatial mechanism consisting of seven links and seven turning pairs of arbitrary orientation in space” the “Mount Everest” of kinematics problems. Since then, researchers (to be discussed) have devoted their careers to finding this solution in closed form. Freudenstein also called for the development of a modern kinematics textbook similar in spirit to the seminal “Strength of Materials” by Timoshenko. In the following decade, at least four such textbooks appeared (Roth, 1979) (Duffy, 1980) (Angeles, 1982) (Barton, 1984).

In closing, as Freudenstein concentrated on mechanisms of increasing complexity: from four-bar, to five-bar, to the spatial 7-bar, so too did the complexity of his mathematical tools increase. Ultimately these mathematical tools became too complex for the design engineer. In Freudenstein's address at

the mechanisms conference he noted that “computational capabilities of an order of magnitude and speed not heretofore attainable” have made complex design calculations possible. The computational capabilities available to the design engineer have increased at least 10,000-fold since then.

2.2 FULLY-CONSTRAINED ROBOTS

Inverse kinematics methods for fully-constrained robots may be loosely classified as either closed-form or numerical. The closed-form methods are computationally very fast and execute in a repeatable and fixed amount of time. These qualities facilitate the application of closed-form methods in real-time control schemes. Numerical methods are slower, but they are much more general. Generality is recommended when a large number of options must be considered, such as is the case during robot design or in the reconfiguration process following a fault in one of the robot’s components.

Table 2.2 Some positive and negative qualities of several inverse kinematics methods for fully-constrained robots.		
Method	Positive Qualities	Negative Qualities
closed-form	finds all solutions very fast consistent speed	must be reformulated for each geometry singularities
Newton-Raphson	relatively general with respect to geometry	singularities non-convergent cycles
predictor-corrector	relatively general with respect to geometry fast for a numerical method	singularities difficulties with highly non linear equations

table look-up	very fast reliable	must be retabulated for each geometry vast amounts of memory requirements

2.2.1 Duffy's Contributions

When Freudenstein declared the closed-form solution of the inverse kinematics problem the “Mount Everest” of kinematics problems, Joseph Duffy was listening. This was a beautifully defined non-linear problem that Duffy intended to solve. In that same address Freudenstein called for a modern kinematics textbook “similar in spirit to the great works in the strength of materials.” Duffy also took these words to heart and in 1980 published a book detailing his systematic development of “a unified theory for the kinematic analysis of mechanisms and manipulators (robotic arms)” (Duffy, 1980).

By all accounts, Duffy's work is exemplary. It is systematic and always intends to be general. Duffy's methods find all existing solutions to a given problem. Algebraic solutions, as opposed to iterative solutions, provide insight into various aspects of linkages, such as: overclosure, rotatability, type, and transmission characteristics. Finally, analytical solutions, when they are available, are typically much faster to solve on a computer than their iterative counterparts.

Unfortunately, the inverse kinematics problem turned out to be more difficult than Duffy imagined (Duffy, 1980). Even the nomenclature is demanding and somewhat confusing. Most researchers in the field of robotics hoped that Duffy would ultimately provide analysis tools for solving the general

inverse kinematics problem. Unfortunately, the tools became so complex that it takes a kinematics expert just to use them. The algebraic equations are so complicated that even basic aspects, such as singularities, are difficult to discern. And finally, modern robots are beginning to exhibit serial kinematic redundancy. The inverse kinematics problem has changed considerably. It is no longer desirable to determine all existing solutions, but rather, the problem is to intelligently choose one. This is certainly not to say that Duffy's work is no longer relevant. Duffy and his students continue to develop their inverse kinematics methods and show that these methods apply to modern robots.

Crane, Duffy and Carnahan (1991) published a kinematic analysis of the space station remote manipulator system including inverse kinematics for the arm. This work serves as an excellent example of the applicability of Duffy's methods to modern redundant robots. Their approach chooses a desirable angle for one of the joints (either the 1st, 2nd or 7th), locks that joint, and then solves for the six remaining joint angles using Duffy's method. While there is some merit to this approach, especially given the limited computational capacity allotted for any one activity on the space station, it also illustrates how resolute Duffy and his associates are about solving the inverse kinematics problem in closed-form. Their approach could have easily been extended to allow numerical iteration on one of the joints, instead of locking, so as to optimize a performance criterion while still solving the remainder of the inverse kinematics problem in closed-form. This is probably an untenable solution to them. Fortunately, this type of approach is available to other researchers and will likely be exploited in the future.

Duffy's work is inarguably of the highest quality. The analytic closed-form solution to the general serial robot with six degrees of freedom continues to be of theoretical interest. It is also of practical importance to the application of both fully-constrained industrial robots and redundant robots.

2.2.2 Newton-Raphson Numerical Method

The Newton-Raphson technique as applied by Uicker, Denavit and Hartenberg (1964) is generally cited as the first time numerical methods were used for solving the inverse kinematics problem for a robot with six degrees of freedom. This method is still in wide use and papers describing minor modifications to the algorithm continue to be published (Gupta and Kazerounian, 1985). The popularity of the algorithm is due to its relatively simple formulation and quadratic rate of convergence. A formulation of the Newton-Raphson method and a description of several of its failure modes follows.

The Newton-Raphson method is essentially a root-finding scheme. It is applied by rewriting the inverse kinematics problem as

$$f(\Phi) - \mathbf{x} = \mathbf{0} \quad (2.5)$$

Because it is easiest to visualize the method in one dimension, the remainder of this discussion will focus on the one-dimensional case. The extension to multiple dimensions is straight-forward. Geometrically the Newton-Raphson method extends the tangent at an initial estimate of the solution, x_i , until the extension crosses zero. The point of the zero crossing is chosen as the next estimate of the solution, x_{i+1} , and the tangent at this point is again extended until it crosses zero. This process is repeated until the zero point is found to the desired level of

accuracy. The Newton-Raphson algorithm is formulated via the Taylor series expansion of the function about the current iterative point

$$f(\Phi + \Delta\Phi) = f(\Phi) + f'(\Phi)\Delta\Phi + f''(\Phi)\frac{\Delta\Phi^2}{2} + \text{HOTS} \quad (2.6)$$

where HOTS represents higher order terms. Rewriting Equation (2.6) omitting the terms of order two and higher

$$f(\Phi + \Delta\Phi) = f(\Phi) + f'(\Phi)\Delta\Phi \quad (2.7)$$

Since a zero-crossing is being sought

$$f(\Phi + \Delta\Phi) = 0 \quad (2.8)$$

and

$$\Delta\Phi = -\frac{f(\Phi)}{f'(\Phi)} \quad (2.9)$$

The most obvious failure of this method occurs when $f'(\Phi) = 0$, which is precisely the case at a singularity of the robot. Because of limited accuracy in the computer representation of numbers, the method fails near mathematical singularities as well. Other failure modes of the Newton-Raphson method are less obvious but well-known and are found in most books on numerical methods.

One important failure mode occurs when a local extremum is between the current solution and the actual solution. Another failure mode occurs when the method degenerates into a non-convergent cycle.

The Newton-Raphson method enjoys wide application to the inverse kinematics problem in spite of these failure modes. As will be shown, the only failure mode of the direct-search method proposed in this dissertation occurs when a local extremum is encountered before an actual solution is reached. This

is but one of the failure modes of the Newton-Raphson method. The direct-search method does not share either of the other failure modes: instability at singularities or non-convergent cycling.

2.2.3 Predictor-Corrector Numerical Method

The predictor-corrector method is another common numerical scheme for solving the inverse kinematics problem for robots with six degrees of freedom. The predictor-corrector method appears in numerous textbooks dating back at least thirty years. As might be inferred from the name, the predictor-corrector method has two parts: the predictor and the corrector. The predictor combines a polynomial curve fitting routine with several equally-spaced past values of the function to extrapolate, or predict, the next point. Information, including derivative information, at this new point corrects the prediction until it is within the desired tolerance. A smooth function will result in a good prediction and good overall performance. Unfortunately, if the function is rapidly changing, the prediction will likely be far removed from the actual value and the predictor-corrector method could diverge from the actual solution. There are also problems with starting and stopping the method. Because the corrector portion of the method uses information about the function's derivatives, the predictor-corrector method is unable to converge at singularities.

Since the predictor portion of the predictor-corrector method requires several previous values, another method of solving the inverse kinematics problem must start the algorithm. If another reliable method of solving the inverse kinematics problem must be available for starting the predictor-corrector

algorithm, why bother using the predictor-corrector method at all? The response to this question is that the predictor-corrector method is 5-15 times faster than the Newton-Raphson method once it is started (Gupta and Kazerounian, 1985). Whether this increase in performance is worth the added complexity of dealing with two distinct inverse kinematics methods is certainly debatable. There is also a problem with stopping the predictor-corrector algorithm. Since the predictor relies on equally spaced previous steps, a step will not necessarily fall directly upon the desired termination point. Some other method that does not rely on equally spaced step sizes must terminate the predictor-corrector algorithm.

As with any numerical scheme that relies on derivative information, the predictor-corrector method theoretically breaks down at singularities. The predictor-corrector method also produces inaccurate results near singularities. Gupta and Kazerounian (1985) report errors at the joint-level from five to twenty-five degrees.

2.2.4 Modified Newton-Raphson and Modified Predictor-Corrector

Both the Newton-Raphson and predictor-corrector methods theoretically break down at singular configurations of the robot. Because of practical considerations, both of these methods also fail near mathematical singularities. Because of this, researchers applying Newton-Raphson and/or predictor-corrector methods to the inverse kinematics problem developed modifications to the original methods that improve their performance near singularities. The modifications, however, introduce an additional failure mode.

The addition of a strict-descent criterion improves the performance of the Newton-Raphson and predictor-corrector methods near singularities. This criterion is composed of two error measures, one translational and rotational. The modified algorithms only allow intermediate solutions which decrease the errors.

Although the modified Newton-Raphson and predictor-corrector algorithms perform significantly better than their unmodified counterparts near singularities, the strict descent criterion introduces another failure mode. This mode occurs if the intermediate solutions encounter a position-level extremum. This failure mode is not typically mentioned, although it will almost certainly occur if these algorithms are applied to an actual robot. These modifications only improve the performance near singularities, and neither the modified Newton-Raphson nor the modified predictor-corrector will converge at exact singularities.

2.2.5 Table Look-up

Table look-up is another method of solving the inverse kinematics problem (Everett and McCarroll, 1986). This method is appealing in the sense that it is very fast and has no problems with singularities. The concept is quite simple, though it requires either another inverse kinematics solution method or some way of measuring the end-effector position of the actual robot. Simply cycle the robot, or the simulated robot, throughout its workspace and record the set of joint angles corresponding to each position of the end-effector.

Though undeniably fast, the table look-up method has a major problem with memory requirements. Given a six-revolute robot with joint travels of 180 degrees, the memory requirement for a resolution of 1 degree is 180^6 (3.4×10^{13})

bytes. This is at least four orders of magnitude greater than the storage capacity of current magnetic media, though not inconceivable for optical media. Most industrial robots have a joint level resolution of at least .001 degrees which increases the storage requirements to at least 6.8×10^{31} bytes. This is far beyond the capacity of any current memory storage device. Combining an interpolation scheme with a lower resolution table reduces the memory requirements but cannot guarantee accuracy.

2.3 REDUNDANT ROBOTS

The introduction of robots with serial kinematic redundancy fundamentally changed the nature of inverse kinematics research. For redundant robots, the inverse kinematics problem becomes very much an optimization problem. The desired placement of the end-effector represents equality constraints for the optimization.

2.3.1 Whitney's Contributions

Daniel Whitney introduced resolved motion rate control in 1969. His approach serves as the basis for virtually all the inverse kinematics methods developed for redundant robots since that time. Whitney was working on human prosthetic arms for forequarter (no arm stump at all) amputees. There was no arm left from which to measure a physical displacement as a controlling signal for the prosthesis. Only EMG signals (low-level electrical signals generated by active muscles) were available which Whitney suggested could specify a rate of change for the prosthetic arm's displacement. This represents the "rate control" portion of Whitney's "resolved motion rate control". The "resolved motion" comes from his

speculation that it would be easier for the amputee to control the prosthesis in world, rather than arm, coordinates. Hence, in Whitney's description, “the motions of the various motors are combined and resolved into separately controllable hand motions along world coordinates” (Whitney, 1969). This is a statement of the inverse kinematics problem at the velocity level.

Whitney's formulation of the problem is extremely simple. The world coordinates, represented by the vector $\mathbf{x}$, are related to the joint displacements, represented by the vector Φ , by some function

$$\mathbf{x} = f(\Phi) \quad (2.10)$$

As Whitney was interested in the rate of change of $\mathbf{x}$, he differentiated this equation to obtain

$$\dot{\mathbf{x}} = J(\Phi)\dot{\Phi} \quad (2.11)$$

where $J(\Phi)$ represents the Jacobian (the matrix of first derivatives of $f(\Phi)$).

After that, he simply solves for $\dot{\Phi}$ given $\dot{\mathbf{x}}$

$$\dot{\Phi} = J^{-1}(\Phi)\dot{\mathbf{x}} \quad (2.12)$$

This simple three line formulation served as the basis of virtually all redundant inverse kinematics for twenty-five years. Whitney even formulated the problem for the redundant case via a pseudoinverse.

While Whitney's formulation is concise and seductive in its simplicity, it is not without major limitations; the most obvious being when the Jacobian is rank-deficient. Another glaring deficiency is the neglect of position-level information by only considering the Jacobian. Position-level considerations are indispensable for actual implementations, which must consider joint travel constraints, obstacles in the environment, and **conservative motion** to name but a few.

These criticisms are not intended to diminish Whitney's contributions. The simple resolved motion rate control was perfectly reasonable given the computational power available during the 1960's. Also, since Whitney was working with forequarter amputees, he had a physical motivation for using rates as the controlling inputs. These criticisms should, however, be noted by the hundreds of researchers that have jumped on the resolved-rate bandwagon during the 1980's and 1990's. There is a richness and wealth of information at the position-level that should not be ignored.

2.3.2 Pseudoinverse

The pseudoinverse is by far the most common method of solving the inverse kinematics problem for redundant robots. Hundreds of papers detailing this method have been published, beginning when Whitney discussed the pseudoinverse in his original resolved rate paper. This section includes a definition of the pseudoinverse, a method for calculating the pseudoinverse, and a method for applying it to the inverse kinematics problem. The myriad of problems which plague pseudoinverse techniques are also discussed.

A $n \times m$ pseudoinverse, $\mathbf{A}^+$, of an $n \times m$ matrix, $\mathbf{A}$, satisfies the four Penrose conditions:

$$\begin{aligned} \mathbf{A}\mathbf{A}^+\mathbf{A} &= \mathbf{A} \\ \mathbf{A}^+\mathbf{A}\mathbf{A}^+ &= \mathbf{A}^+ \\ (\mathbf{A}\mathbf{A}^+)^T &= \mathbf{A}\mathbf{A}^+ \\ (\mathbf{A}^+\mathbf{A})^T &= \mathbf{A}^+\mathbf{A} \end{aligned} \tag{2.13}$$

If $n > m$, as is the case with redundant robots, then the pseudoinverse is calculated simply as

$$\mathbf{A}^+ = \mathbf{A}^T (\mathbf{A} \mathbf{A}^T)^{-1} \quad (2.14)$$

The application of the pseudoinverse to the inverse kinematics problem is via Whitney's resolved rate technique. Beginning with the resolved rate formulation

$$\dot{\mathbf{x}} = \mathbf{J}(\Phi) \dot{\Phi} \quad (2.11)$$

where $\mathbf{J}(\Phi)$ represents the Jacobian of $f(\Phi)$. Dividing both sides of Equation (2.11) by $\mathbf{J}(\Phi)$ gives the joint speeds.

$$\dot{\Phi} = \mathbf{J}^{-1}(\Phi) \dot{\mathbf{x}} \quad (2.12)$$

Since the Jacobian for a redundant robot is not square, the pseudoinverse, $\mathbf{J}^+(\Phi)$, is used instead of $\mathbf{J}^{-1}(\Phi)$

$$\dot{\Phi} = \mathbf{J}^+(\Phi) \dot{\mathbf{x}} \quad (2.15)$$

The simplicity of this formulation indicates that there might be some problems in applying it to something as complex as a redundant robot. The pseudoinverse as shown minimizes the two-norm of the joint speeds. This will be derived in the Chapter 3. Unfortunately, the minimum two-norm of joint velocities does not necessarily correspond to improved or even acceptable performance. If minimizing the two-norm of the joint velocities is the only criterion, better performance would likely be attained with a fully-constrained robot.

Since the pseudoinverse minimizes the two-norm of joint velocities, researchers originally thought that it would naturally avoid singularities. This is because singularities are often accompanied by very high demands on the joint velocities. As has since been realized, the pseudoinverse often leads the robot into singular configurations (Carnigan, 1991). This is a problem since the pseudoinverse fails at singularities.

One of the many problems with using the pseudoinverse for an actual robot is that physical constraints are not addressed. These constraints are at the position, velocity and acceleration levels. Anyone using an actual robot experiences the position-level constraint on joint travels (joint limits). The pseudoinverse does not address this very practical constraint. The top speed of the actuators is a constraint at the velocity-level. The pseudoinverse comes closest to addressing this constraint, though many researchers have noted that the pseudoinverse often makes unrealistic demands on the speed capabilities of the actuators. Finally, acceleration constraints relate to the finite ability of the actuators to produce forces or torques. The pseudoinverse does not address acceleration constraints.

The most common problem cited with the pseudoinverse is that it does not produce conservative motion. That is, following a closed path in **end-effector space** will not necessarily result in a closed path in joint space. In effect, the robot drifts in joint space. Numerous researchers studying this problem found that it is unpredictable and often leads to instabilities. It is easy to understand the origins of the non-conservative drift. The drift occurs at the position level, while the pseudoinverse is applied at the velocity level. It is altogether possible that the same two-norm of joint speeds produce different position-level results. A non-conservative path is unpredictable at the joint-level. This adversely affects repeatability in manufacturing operations.

A final problem with the pseudoinverse relates to unit invariance. That is; the pseudoinverse will give different results depending upon the units representing the dimensions of the robot and the task. For instance, measuring

lengths in feet instead of inches changes the results of the pseudoinverse. The physical meaning of the problem does not change, yet changes in parameterization yield different solutions (Lipkin, 1990). This is a fundamental consideration in inverse kinematics. It occurs in the very formulation of the problem: both translational and rotational coordinates describe the placement of the robot's end-effector. There are essentially three ways of handling this problem. The first is to ignore it and simple-mindedly mix units. This is what happens in the pseudoinverse. Careful algebraic manipulation may maintain separation between rotational and translational terms (Duffy, 1990). This method, however, severely limits the number and type of performance criteria that may be considered in the solution. Finally, systematic scaling procedures will provide a rational basis for comparing rotational and translational properties (Bevill and Tesar, 1990).

Considering the myriad of problems associated with the pseudoinverse as a practical solution to the inverse kinematics problem, it is difficult to understand why research on this method continues.

2.3.3 Extended Jacobian

The extended Jacobian method for solving the inverse kinematics problem for redundant robots receives considerable attention in the literature. The method is also applied to actual robot arms at the Jet Propulsion Lab in Pasadena and is the current choice for the redundant Ranger arm at the University of Maryland. Conceptually the extended Jacobian method is quite simple. For a redundant robot with n joints and m end-effector constraints, n is greater than m and the

Jacobian has more columns than rows. The extended Jacobian method makes the Jacobian square by adding an additional n minus m constraints. Inverting the extended Jacobian gives the joint speeds. Unfortunately, the addition of constraints adds algorithmic singularities that may prohibit inversion.

Mathematically the extended Jacobian method is not difficult to formulate. Simply define $n - m$ constraints

$$G_1(\Phi) = 0, G_2(\Phi) = 0, \dots, G_{n-m}(\Phi) = 0 \quad (2.16)$$

and add them to the forward kinematic relations of the robot

$$\begin{bmatrix} f(\Phi) \\ G_1(\Phi) \\ \vdots \\ G_{n-m}(\Phi) \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (2.17)$$

From here the method proceeds in the usual resolved rate fashion. Differentiating both sides of Equation (2.17) gives

$$\begin{bmatrix} \frac{\partial f}{\partial \Phi} \\ \vdots \\ \frac{\partial G}{\partial \Phi} \end{bmatrix} \dot{\Phi} = \begin{bmatrix} \dot{\mathbf{x}} \\ \vdots \\ \mathbf{0} \end{bmatrix} \quad (2.18)$$

The square matrix, $\mathbf{J}_e$,

$$\mathbf{J}_e = \begin{bmatrix} \frac{\partial f}{\partial \Phi} \\ \vdots \\ \frac{\partial G}{\partial \Phi} \end{bmatrix} \quad (2.19)$$

is called the extended Jacobian.

Dividing both sides of Equation (2.18) by the extended Jacobian gives the joint speeds.

$$\dot{\Phi} = J_e^{-1} \begin{bmatrix} \dot{x} \\ 0 \end{bmatrix} \quad (2.20)$$

The extended Jacobian method is mathematically simple and conceptually appealing in the sense that constraints may be chosen to optimize various performance criteria. There is, however, a serious singularity problem with the method. This is because adding constraints adds algorithmic singularities to the geometric singularities of the robot. Typically, researchers use the self-motion of the robot and a singularity detection scheme to avoid the singularities. This begs one to question the advantage of using a redundant robot that is devoting its redundancy to avoiding singularities that were artificially introduced by the inverse kinematics algorithm. Unfortunately, the more performance criteria that are added, the more complex the singularity problem becomes. This essentially eliminates the extended Jacobian method as a solution to the generalized inverse kinematics problem.

Since the extended Jacobian method is a velocity-level scheme, it also exhibits non-conservative motion and the associated joint-level drift. In spite of its singularity problems, the extended Jacobian method has been used for enhancing performance via various criteria, such as manipulability and obstacle avoidance (Baillieul, 1986). This effort does not provide an adequate test since Baillieul only demonstrated obstacle avoidance for a planar robot with three degrees of freedom performing a very simple positioning task .

2.3.4 Neural Nets

Applying artificial neural nets to the inverse kinematics problem seems natural. Humans and animals solve the inverse kinematics problem with ease. A snake, for instance, is a creature with a very small brain, yet it is able to solve the inverse kinematics problem in real-time while incorporating obstacle avoidance and other high-level criteria. Unfortunately, applications of artificial neural nets to the inverse kinematics problem have had limited success, even with simple robot geometries. Nonetheless, there are several advantages that a successful artificial neural inverse kinematics method would possess. Before being applied to a problem, artificial neural nets must go through an initial training phase.

Much as a baby learns hand-eye coordination, a neural net must be trained to solve the inverse kinematics problem. There are several ways of training the artificial neural net. If the actual robot is available, it can provide the artificial neural net with the joint angles that correspond to the desired end-effector position. If the actual robot is not available, another inverse kinematics algorithm could provide the artificial neural net with the actual joint angles that correspond to the desired end-effector position. In both of these cases the artificial neural net is trained with the correct set of joint angles. If no means of generating the proper solutions are available, then the artificial neural net can only be trained in a trial and error fashion. In this case the training period will be much longer than if the correct solutions were available.

Artificial neural nets as applied to the inverse kinematics problem typically use error back propagation in the training phase (Pourboghra, 1989).

Through error back propagation, multilayer artificial neural nets learn the inverse kinematics mapping experientially. For each training case the synaptic weights and thresholds are adjusted to minimize the error between the trial and actual solution. The process of generating trial solutions, computing the error, and adjusting the synaptic weights and thresholds continues until all the inverse kinematics solutions from the training set are learned within an acceptable error.

There are several advantages that an artificial neural net solution to the generalized inverse kinematics problem would possess. The main advantage is that an artificial neural net is non-algorithmic and robot independent. That is, it may be possible that the same artificial neural net could be trained the inverse kinematics relationship for any robot. This is certainly attractive, especially if the training phase could be accomplished in a reasonable amount of time. Another advantage is that the artificial neural nets can optimize performance criteria. In artificial neural net jargon the redundant inverse kinematics problem is known as a “one-to-many” mapping problem.

The main disadvantage with applying artificial neural nets to the inverse kinematics problem is their inability to guarantee the accuracy of the solution. Limited accuracy is also apparent in natural neural nets. Imagine a person trying to grind piece of steel to within a .001 inch tolerance using a hand-held grinder. It's just not possible. Researchers attempting to alleviate the problem with accuracy have combined the artificial neural net with iterative numerical methods, such as Newton-Raphson (Egeland et al., 1991). The artificial neural net generates an initial solution, while the Newton-Raphson method refines it to the desired level of accuracy.

A possible problem is that artificial neural nets are almost invariably demonstrated only for the planar case. Presumably a much larger net would be required for the vastly more complex spatial case. This large artificial neural net would require a large amount of machine memory and might be too slow for practical application. However, specialized artificial neural net electronic hardware may alleviate these problems.

2.3.5 Genetic Algorithms

Genetic algorithms are search procedures based on the principles of genetics and natural selection. These algorithms possess the two qualities required of a solution to the inverse kinematics problem: they can be applied to any robot geometry because they treat all problems as a “black box” and they allow the consideration of an unlimited number of performance criteria. These criteria are assembled as a single performance index called “fitness” in genetic algorithm jargon. The main components of a genetic algorithm are: population, reproduction, crossover, and mutation (Parker and Goldberg, 1989).

As applied to the inverse kinematics problem, genetic algorithms maintain a number of joint angle sets called the population. Binary numbers represent the joint angles. Each bit in the binary number depicts a bit of genetic information. The joint angle sets are evaluated according to their fitness for solving the inverse kinematics problem. For instance, the Cartesian error is a measure of fitness. The fittest (least error) joint angle sets are given the highest probability of selection for mating and its associated genetic action.

The genetic action takes place during crossover. The fittest sets of joint angles have respective bits within their binary encoded numbers swapped. This results in new sets of joint angles. In essence, these sets are the “offspring” of the previous sets.

Finally, mutation is the random changing of bits from zero to one or from one to zero. Mutation is typically used sparingly and is of secondary importance to reproduction and crossover.

The process of choosing joint angle sets, swapping bits, and then evaluating fitness continues until, presumably, a solution is found. This is where the problem occurs. The population may stabilize before reaching a solution. Typical errors are on the order of .25 inches for “industrial size” robots. This positioning error is altogether unacceptable and currently precludes genetic algorithms from being a practical method of solving the inverse kinematics problem. There may, however, be technological advances (specialized hardware, etc.) which, combined with the inherent generality of genetic algorithms, would warrant their future development as a solution to the generalized inverse kinematics problem.

Table 2.3 Some positive and negative qualities of several inverse kinematics methods for redundant robots.		
Method	Positive Qualities	Negative Qualities
pseudoinverse	simple formulation	not conservative singularities unreliable

extended Jacobian	can incorporate criteria	not conservative geometric singularities unreliable algorithmic singularities
neural nets	seem to work for humans general with respect to geometry general with respect to performance criteria	limited accuracy
genetic algorithms	general with respect to geometry general with respect to performance criteria	limited accuracy

2.3.6 Performance Criteria

Of all the research supporting generalized inverse kinematics, none is more important than developing of performance criteria. Performance criteria are based on kinematic and dynamic models of the robot. The generalized inverse kinematics strategy uses these criteria to evaluate the performance of the robot and make intelligent decisions as to which kinematic configuration will be most desirable for executing the specified task. Many different performance criteria are presented in the literature. While some of these performance criteria are valuable, their development is not thorough enough to support generalized inverse kinematics. Fortunately, the University of Texas Robotics Research Group has, through two doctoral dissertations and several masters theses, systematically and rigorously developed a mathematical formulation of at least thirty performance criteria. These performance criteria are categorized according to the order of the

geometry and energy domain represented. Normalization procedures have been developed which will enable a generalized inverse kinematics strategy to compare and consider many different performance criteria which may originally be of widely varying magnitude.

Of the performance criteria presented in the literature, the joint range availability is the criterion of choice when controlling an actual robot. The Robotics Research Corporation patented their inverse kinematics method incorporating joint range availability. The joint range availability is used with real robots because real robots have real joint travel limits that cannot be ignored. As it is typically formulated, the joint range availability, JRA, is a simple criterion easily calculated.

$$JRA = \sum_{i=1}^n \left[\frac{(\Phi_i - \tilde{\Phi}_i)^2}{\Phi_{i\max}^2} \right] \quad (2.21)$$

In this formulation, Φ_i is the joint displacement, $\tilde{\Phi}_i$ is the mid-range displacement and $\Phi_{i\max}$ is the displacement at the joint limit. Minimizing this criterion keeps the joints as near as possible to the midpoints of their travel. While observing joint limit constraints is mandatory, seeking to keep the joints near their center positions is not necessarily associated with enhanced performance. For a given task, it is only requisite that the joint limit constraints not be violated. Other performance criteria, such as those associated with forces and deflections, might indicate that superior performance is available if one or more joints are near, but not exceeding their limits.

The proximity to a singularity is a common performance criteria discussed in the literature (Burdick, 1991). The Jacobian matrix is rank deficient at singularities. As most inverse kinematics techniques for redundant robots invert the Jacobian (one way or another), it is obvious why singularity avoidance is of paramount importance to these algorithms. Several criteria for measuring the proximity of a singularity have been identified. The transmissibility of the Jacobian, ω , is a common one.

$$\omega = \sqrt{|\mathbf{J}\mathbf{J}^T|} \quad (2.22)$$

Another performance criteria for detecting singularities is the condition number of the Jacobian, $\text{cond}(\mathbf{J})$. As shown in Equation 2.23, the condition number is the ratio of maximum singular value, $\sigma_{\max}$, to the minimum singular value, $\sigma_{\min}$.

$$\text{cond}(\mathbf{J}) = \frac{\sigma_{\max}}{\sigma_{\min}} \quad (2.23)$$

Researchers suggest that the best and most direct measure of a singularity is the minimum singular value of the Jacobian. This raises a fundamental question. Are singularity avoidance criteria measures of the robot's performance or are they measures of the inverse kinematics algorithm's performance? Even for robots with six degrees of freedom, which lose end-effector dexterity at singularities, there are still feasible directions for moving the end-effector. For robots with one degree of redundancy, some singular positions represent areas of unacceptably high demands on joint speeds. Other singular positions do not diminish the capabilities of the robot. As the degree of redundancy increases, the relationship

between singularities and robot performance becomes even less clear. Consider the hyper-redundant robot shown in Figure 6.20. Each time one of the pivot joints moves through the center of its travel, the axes of the roll joints on either side of the pivot are collinear and the Jacobian is rank-deficient. These singularities, however, represent no loss in performance capabilities for the robot. For this robot, avoiding singularities results in extremely diminished performance capabilities. All things considered, the relationship between singularity avoidance performance criteria and the actual performance of the robot are dubious at best.

Obstacle avoidance is clearly of paramount importance with robots. Obstacle avoidance refers to moving the robot while avoiding collisions with objects in the environment. The objects include the robot itself, fixed objects in the environment, moving objects in the environment, and also other robots operating in the same workspace. The cost of an industrial robot colliding with an obstacle in the environment could be very high, both in terms of repair to the robot and its environment as well as the cost of lost productivity due to downtime. The consequences of a collision in a more sensitive environment – such as in space, nuclear, or military applications – could be enormous.

Conceptually, the obstacle avoidance problem is easy to understand. The mathematical formulation of the problem is, however, much more difficult. For instance, actual physical objects have edges and corners which are difficult to describe with continuous mathematical functions. There are numerous other complicating factors too extensive to be covered here. The following discussion merely describes several obstacle avoidance formulations used in conjunction with solving the inverse kinematics problem.

A problem similar to obstacle avoidance is target acquisition. This is basically the inverse of the obstacle avoidance. Rather than repelling the robot, as would an obstacle, the target attracts the robot. This formulation may be used to draw the robot's end-effector towards its desired placement.

Wellman and Tesar (1991) developed an obstacle avoidance technique using intersecting volumes. These volumes were placed around the robot and any obstacles in a known world model. Intersections of the volumes were used to calculate forces that would be fed back to the robot's operator so that the operator would not drive the robot or its payload into a collision. The main strength of this method is that its computational complexity matched current computational capabilities.

Maciejewski and Klein (1988) formulated the obstacle avoidance problem in terms of an obstacle avoidance point. They maximized the distance between this point and the obstacle using the pseudoinverse. A problem with the pseudoinverse, as applied to obstacle avoidance, concerns conservative motion. Since a robot employing the pseudoinverse typically drifts in joint space, it is not enough to monitor the robot the first time it performs a task among obstacles. Each time the robot performs the task it will likely move through different paths in joint space. One of these paths may lead to a singularity or collision.

Das, Slotine and Sheridan (1988) formulated the obstacle avoidance problem using what they named the Jacobian Transpose method. This method employs the transpose of the Jacobian matrix rather than the pseudoinverse. The method is quite interesting. Physically, it can be thought of as the application of a force that directs the tip of the robot towards the desired end-effector solution.

The problem is then formulated as a control loop driving the end-effector towards the desired displacement. The resulting joint displacements represent the solution to the inverse kinematics problem. For a redundant robot, other forces can be added to direct the robot away from obstacles. Das, Slotine and Sheridan simulated this algorithm for a planar robot with ten rotational degrees of freedom and the results seem to be very good. There is, however, quite an increase in complexity when moving from the planar to the spatial case. It will be interesting to follow this work and see if it can be successfully extended to spatial robots

The previously discussed performance criteria are dispersed in the literature and in no way represent a systematic development. Typically, these criteria are formulated for ease of computation or for convenience as related to their associated inverse kinematics methods.

In contrast, the University of Texas Robotics Research Group put forth a concerted and systematic effort developing a set of performance criteria physically relevant to the operation of robots. Cleary and Tesar (1990) arranged these criteria into the following categories: geometric, inertial, kinetic energy distribution, and system compliance. Bevill and Tesar (1990) developed a procedure for normalizing these criteria.

The geometric performance criteria are further subdivided into first and second order groups. The first order criteria are based on the Jacobian matrix while the second order criteria are based on the Hessian array. These criteria are task independent and based only on the geometry of the robot. Therefore, they need not be reformulated for different tasks.

The first five entries in Table 2.4 list the first order geometric criteria. The singularity detection criterion is simply the minimum singular value of the Jacobian and measures how close the Jacobian is to rank deficiency. As has been discussed, this criterion is of limited value for robots with a high degree of redundancy.

Table 2.4 Geometric performance criteria developed at U. T. Austin.	
Criterion	Symbol
singularity detection	η_{σ}
dexterity	η_{δ}
velocity transmission	$\eta_{v\chi}$
force and torque transmission	$\eta_{\tau\chi}$
Jacobian Frobenius Norm	η_{Σ}
rate of change of singularity detection criterion	$\eta_{\Delta\sigma}$
rate of change of dexterity criterion	$\eta_{\Delta\delta}$
end-effector velocity-induced joint acceleration	$\eta_{\phi\alpha}$
joint velocity-induced end-effector acceleration	$\eta_{v\alpha}$

The dexterity criterion, η_{δ} in Equation (2.24), is the ratio of the minimum and the maximum singular values of the Jacobian and measures the robot's dexterity at any given set of joint displacements.

$$\eta_{\delta} = \frac{\sigma_{\max}}{\sigma_{\min}} \quad (2.24)$$

This is an important criterion for robots with a limited degree of redundancy but, as with the singularity detection criterion, becomes less useful as the degree of redundancy of robot increases.

The velocity transmission criteria, η_{vx} in Equation (2.25), measures the robot's ability to move in a desired direction (the x direction in this case).

$$\eta_{vx} = \left\{ \sum_{i=1}^M \left[\frac{1}{\sigma_i} (\dot{\mathbf{u}}^e)^T \mathbf{h}_i \right]^2 \right\}^{-\frac{1}{2}} \quad (2.25)$$

In this equation, M is the number of degrees of freedom at the end-effector, σ_i is the i -th singular value of the Jacobian, $\dot{\mathbf{u}}^e$ is the desired end-effector motion, and $\mathbf{h}_i$ is the i -th end-effector space singular vector. This criterion applies equally well to redundant and hyper-redundant robots. Similar to the velocity transmission criteria, the force and torque transmission criteria, η_{tx} in Equation (2.26), measures the robot's ability to produce a force or torque in a desired direction (the x direction in this case).

$$\eta_{tx} = \left\{ \sum_{i=1}^M \left[\sigma_i (L^e)^T \mathbf{h}_i \right]^2 \right\}^{-\frac{1}{2}} \quad (2.26)$$

The terms in this equation are the same as those in (2.25) with the exception that L^e is the direction of the force at the end-effector. This criterion seems to be especially well-suited for redundant robots which can use self-motion so as to best apply forces and torques in a desired direction. This capability is of great value during insertion or forming tasks.

The Jacobian matrix Frobenius norm, η_x in Equation (2.27), also measures velocity, force, and torque transmission capabilities.

$$\eta_x = \sqrt{\sum_{i=1}^M \sigma_i^2} \quad (2.27)$$

As noted by Van Doren and Tesar (1992), this criterion is less accurate and does not exhibit as much variation over the workspace as the previously described transmission criteria. This diminishes the value of the Jacobian matrix Frobenius norm in a decision making framework.

The final four listings in Table 2.4 are the second order geometric performance criteria. These criteria are derived primarily from the Hessian array.

The rate of change of the singularity detection criterion, $\eta_{\Delta\sigma}$ in Equation (2.28), indicates whether the robot is moving towards or away from a singularity.

$$\eta_{\Delta\sigma} = \sqrt{\sum_{i=1}^n \left(\frac{\partial \sigma_{\min}}{\partial \Phi_i} \right)^2} \quad (2.28)$$

The summation is performed over n , which is the number of the robot's degrees of freedom. The rate of change of the dexterity criterion, $\eta_{\Delta\delta}$ in Equation (2.29), indicates how rapidly the robot's dexterity is increasing or decreasing.

$$\eta_{\Delta\delta} = \sum_{i=1}^n \left(\frac{\partial \eta_{\delta}}{\partial \Phi_i} \right)^2 \quad (2.29)$$

As with their first order counterparts, the usefulness of the second order singularity and dexterity criteria decreases as the redundancy of the robot increases.

Of the final two second order geometric criteria, the end-effector velocity-induced joint acceleration, $\eta_{\Phi\alpha}$ in Equation (2.30), seems to be the most important.

$$\eta_{\Phi\alpha} = \frac{\sum_{i=1}^n \lambda_{i \max}}{n} \quad (2.30)$$

In this equation, $\lambda_{i \max}$ is the maximum eigenvalue for the Hessian array. This criterion relates the joint accelerations to a constant end-effector velocity. It is important because many robotic applications, such as spray painting or welding, require constant velocities. It is also important because robot actuators have a finite ability to produce joint-level accelerations. Van Doren and Tesar (1992) show that for an example task this criterion varies considerably, which in turn indicates that it will be useful in a decision making framework.

Table 2.5 lists the force/torque performance criteria. These criteria are based on dynamic models of forces and torques within the robot and are essential to the intelligent design and application of robots. The first four criteria deal with actuator torques. Actuator torque demands are a fundamental consideration, as all robot actuators have a finite ability to produce torques.

Table 2.5 Force/torque performance criteria developed at the U. T. Austin.

Criterion	Symbol
<u>dynamic coupling</u>	η_{ζ}
<u>actuator torque</u>	
<u>equivalent</u> end-effector forces	$\eta_{v\tau}$
end-effector <u>space</u> actuator <u>torque</u>	$\eta_{v\phi\tau}$
rate of <u>change</u> of actuator <u>torque</u> criterion	
rate of <u>change</u> of <u>equivalent</u> end-effector forces criterion	$\eta_{\Delta v\tau}$
rate of <u>change</u> end-effector <u>space</u> actuator <u>torque</u> criterion	
<u>velocity-induced</u> actuator <u>torque</u>	
<u>velocity-induced</u> <u>equivalent</u> end-effector forces	$\eta_{vv\tau}$
end-effector <u>space</u> <u>velocity-induced</u> <u>equivalent</u> actuator <u>torque</u>	$\eta_{vv\phi\tau}$
GH norm	η_{γ}

The next three criteria deal with the rate of change of torques. The rate of change of the actuator torque criterion, $\eta_{\Delta\tau}$ in Equation (2.31), measures how fast the robot can respond to torque and force demands.

$$\eta_{\Delta\tau} = \sqrt{\sum_{i=1}^n \left(\frac{\partial \lambda_{\max}}{\partial \Phi_i} \right)^2} \quad (2.31)$$

This is an especially important criterion because larger actuators or higher gear ratios can supply more torque, but both will likely slow the overall response of the robot to external disturbances. Consideration of the basic torque demands as well as its rate of change could allow the intelligent allocation of the robot's resources for truly enhanced operation.

Bevill and Tesar (1990) developed a performance measure based on the product of the Jacobian matrix and Hessian array that they named the GH norm. The name reflects their physical interpretation of the Jacobian matrix as a collection of gain elements and the Hessian array as a composition of higher order geometric properties. Van Doren and Tesar (1992) then formulated this performance measure as a single criterion. It is shown as η_γ in Equation (2.32).

$$\eta_\gamma = \frac{\sum_{i=1}^n \lambda_{i \text{ GH max}}}{n} \quad (2.32)$$

In this equation, $\lambda_{i \text{ GH max}}$ is the maximum eigenvalue of the symmetric portion of each plane in the Jacobian and Hessian product. This criterion is a mathematical description of torque shock and the relative demand for responsiveness from the associated actuators based on higher order geometric properties.

Table 2.6 lists the kinetic energy performance criteria. Minimizing kinetic energy is important for high-speed precision operation. The kinetic energy criterion addresses high-level issues represented in relatively simply formulations.

Table 2.6 Kinetic energy performance criteria developed at U. T. Austin.	
Performance Criterion	Symbol
joint space kinetic energy	η_K
end-effector space kinetic energy	η_{vK}
rate of change of joint space kinetic energy	$\eta_{\Delta K}$
rate of change of end-effector space kinetic energy	$\eta_{\Delta vK}$
kinetic energy partition values	$\eta_{K\pi}$

The kinetic energy, KE, of a serial robot may be formulated in a straightforward manner.

$$KE = \frac{1}{2} \dot{\Phi}^T [I_{\Phi\Phi}^*] \dot{\Phi} \quad (2.33)$$

$I_{\Phi\Phi}^*$ is the effective inertia matrix. This equation represents the entire kinetic energy content of the robot. In this formulation, the kinetic energy value is dependent upon the joint speeds and is therefore dependent upon the task being performed. Van Doren and Tesar (1992) formulated a task-independent kinetic energy criterion. It is shown as η_K in Equation (2.34).

$$\eta_K = \sqrt{\sum_{i=1}^n \lambda_i^2} \quad (2.34)$$

The λ_i 's are the eigenvalues of the effective inertia matrix.

The rate of change of the kinetic energy criterion is also important. This criterion, $\eta_{\Delta K}$, is formulated as shown in Equation (2.35).

$$\eta_{\Delta K} = \sqrt{\sum_{i=1}^n \left(\frac{\partial \eta_K}{\partial \Phi_i} \right)^2} \quad (2.35)$$

Large changes in kinetic energy correspond to very large demands on actuator power. Very rapid changes in the kinetic energy represent shocks to the robot.

Table 2.7 lists the compliance performance criteria. The compliance criteria describe the robot's ability to perform precision operations under load. They also correspond to the vibratory modes of the robot. As pointed out by Ambrose and Tesar (1992), the compliance of a robot's joints is typically one order of magnitude greater than the compliance of its links.

Table 2.7 Compliance criteria developed at U. T. Austin.	
Criterion	Symbol
oscillation	η_{ω}
system stiffness	η_{ϕ}
system potential energy	η_{ρ}
potential energy partition values	$\eta_{\rho\pi}$

Of the compliance criteria, the potential energy partition values, are particularly important. The potential energy partition values measure the distribution of compliance energy and how it changes as the robot moves. An unusually high compliance energy content in any part of the robot indicates a problem with the robot's design. Rapid changes in compliance energy indicate large local forces, which correspond to large actuator demands and decreased precision.

Using these criteria to make intelligent decisions and enhance the robot's performance is the goal of generalized inverse kinematics. An intelligent decision making process must compare the contributions of each performance criteria and consider the overall performance of the system. The decision making process must use scaling factors to make fair comparisons leading to intelligent decisions.

Bevill and Tesar (1990) suggest using average values calculated over the robot's entire workspace as scaling factors. This approach gives a global measure of the performance criteria's magnitude, though it is computationally intensive. For instance, calculating criteria values for a robot with 7 joints, each having a range of 180 degrees, at a resolution of 10 degrees requires 18^7 or approximately 600×10^6 criteria calculations. Although this represents a significant computational burden, the calculations must only be performed once per robot. Van Doren and Tesar (1992) propose another normalization and scaling procedure that is less of a computational burden. This scaling approach is based on velocity and torque limitations at the joint level.

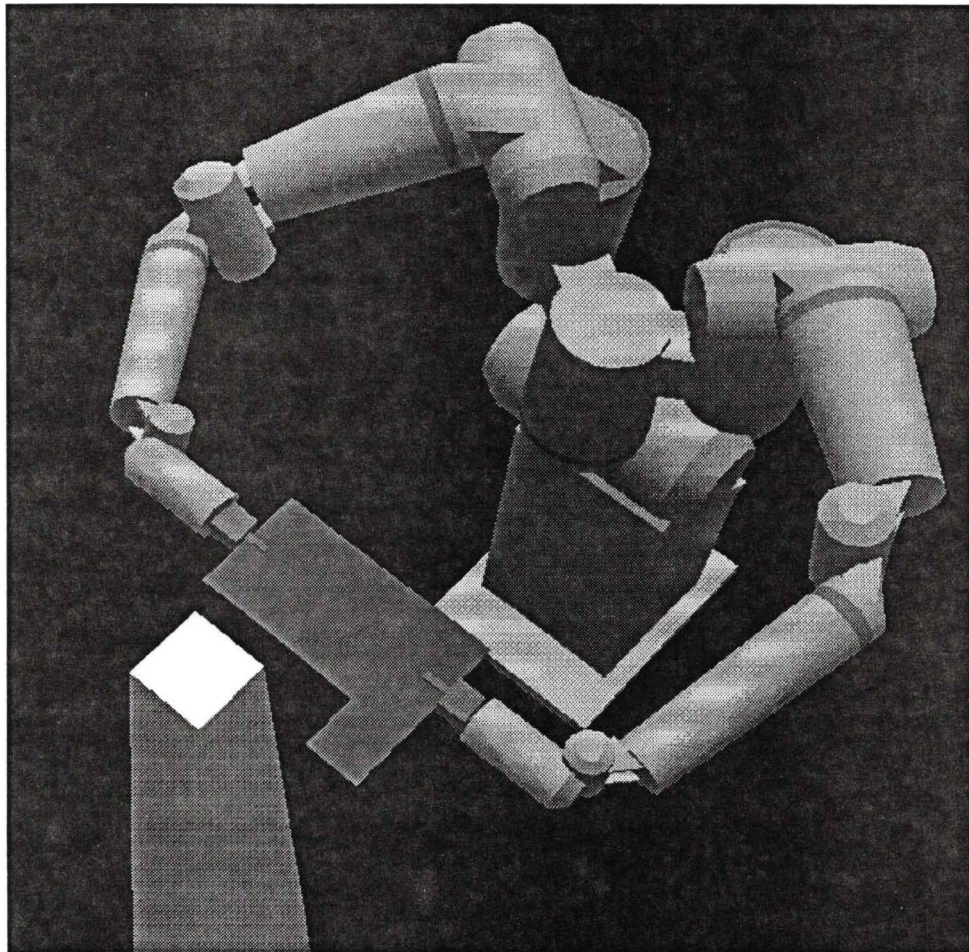


Figure 2.1 An advanced dual-arm robot with 17 total degrees of freedom
handling a thin material.

Chapter 3: Inverse Kinematics by Local Optimization

This chapter discusses several methods of solving the inverse kinematics problem using local optimization. The methods fall into two categories: analytical and direct search. The analytical methods optimize a single performance criterion while simultaneously solving the inverse kinematics problem for a redundant robot. The direct search method of solving the inverse kinematics problem is one of the major developments in this dissertation. This chapter develops the direct search method for robots with six degrees of freedom. The next chapter extends the method to include redundant robots.

Analytical methods can solve the inverse kinematics problem at the position, velocity, and acceleration levels. At the position level, the problem is highly non-linear and closed-form solutions only exist for special cases. Due to the instantaneous nature of the relationships, the problem is linear at the velocity and acceleration levels. This chapter derives analytical solutions minimizing joint speeds and kinetic energy at the velocity level. At the acceleration level, solutions minimizing joint accelerations and joint torques are derived.

In the most basic sense, direct search describes a method of solving problems numerically using sets of trial solutions to guide a search. The search is direct because it does not explicitly evaluate derivatives, as do gradient and higher-order numerical methods. Because the direct search does not use derivatives, it is stable at singularities. Direct search's name may give the impression that it is somehow less analytical than other numerical methods. This

is not true. For all but the simplest of problems, all numerical methods are, in essence, searches. This is the nature of numerical methods. Most of them, including Newton-Raphson and predictor-corrector, have an analytical basis for linear or quadratic functions. For functions of degree higher than two, the analytical basis disappears. As the non-linearity increases, the relationship between a function's derivatives and its solution becomes less and less clear. Nothing exemplifies this more than a robot at a singularity. The derivative in at least one direction is undefined, yet there is still a solution. A sixteenth degree equation describes the inverse kinematic relationship for a general robot with six degrees of freedom (Duffy, 1980). For this degree of non-linearity, all numerical methods are searches.

Most direct search strategies share an amount of commonalty. The search begins with an estimate of the solution, which serves as an initial base point. A pattern of local explorations around the base point generates information about the locality. A decision making strategy chooses a new base point based on this information and then the search proceeds with another series of explorations. This process continues until the search, at least a successful one, finds a solution. This chapter describes three different exploration patterns and three different decision making strategies. These exploration patterns and decision making strategies combine to form a number of different direct methods. Some of these methods guarantee optimality and some do not. This chapter discusses the reliability of the various methods. Finally, the chapter concludes with a discussion as to each of the method's relative computational complexity and their recommended applications.

3.1 ANALYTICAL SOLUTIONS

An analytical solution to the inverse kinematics problem that simultaneously optimizes a performance criterion can be formulated at the position, velocity and acceleration levels. At the position level, the non-linear mapping between joint space and end-effector space is

$$\mathbf{x} = \mathbf{f}(\Phi) \quad (3.1)$$

where the vector $\mathbf{x}$ represents the position and orientation of the end-effector, the vector Φ represents the joint displacements, and $\mathbf{f}(\Phi)$ is a description of the robot's input-output positional relationships.

Differentiating Equation (3.1) relates the joint speeds to the end-effector velocity.

$$\dot{\mathbf{x}} = \mathbf{J}(\Phi)\dot{\Phi} \quad (3.2)$$

Differentiating Equation (3.2) relates the end-effector acceleration to the Jacobian, $\mathbf{J}(\Phi)$, the time rate of change of the Jacobian, $\dot{\mathbf{J}}(\Phi)$, the joint speeds, $\dot{\Phi}$, and the joint accelerations, $\ddot{\Phi}$.

$$\ddot{\mathbf{x}} = \mathbf{J}(\Phi)\ddot{\Phi} + \dot{\mathbf{J}}(\Phi)\dot{\Phi} \quad (3.3)$$

Because Equations (3.2) and (3.3) provide an instantaneous relationship between joint states and end-effector states, they are inherently linear in terms of the higher order properties. This linearity provides a unique relationship for the joint speeds or joint accelerations much simpler than solutions for the joint displacements. Integrating the joint speeds once, or the accelerations twice, gives the joint displacements that represent solutions to the inverse kinematics problem.

3.1.1 Position-level Optimization

At the position level, the inverse kinematics problem for a redundant robot is very complex. It may be posed as an optimization problem with very coupled and non-linear constraint equations. Only extremely simple sub problems, such as the planar case, have closed-form solutions. A general formulation for the optimization of a single position-level criterion is obtained using the classical LaGrange multiplier technique.

To optimize a position level criterion of the form

$$c(\Phi) \tag{3.4}$$

begin by rewriting Equation (3.1) as an equality constraint.

$$\mathbf{x} - \mathbf{f}(\Phi) = \mathbf{0} \tag{3.5}$$

Adjoining the constraint set, Equation (3.5), to the performance criterion, Equation (3.4), via a LaGrange multiplier vector, λ , forms the Hamiltonian

$$H(\Phi, \lambda) = c(\Phi) + \lambda^T [\mathbf{x} - \mathbf{f}(\Phi)] \tag{3.6}$$

The conditions for optimality are

$$\frac{\partial H}{\partial \lambda} = \mathbf{0} \tag{3.7}$$

and

$$\frac{\partial H}{\partial \Phi} = \mathbf{0} \tag{3.8}$$

The exact form of the equations described by (3.7) and (3.8) depends upon the robot geometry as well as the performance criterion. Not only must Φ be found, as in the fully-constrained case, but the LaGrange multiplier, λ , must be found as well.

3.1.2 Minimizing Joint Speeds

The inverse kinematics problem with optimization is linear at the velocity level. It is linear because of the instantaneous nature of the velocity-level equations. The LaGrange multiplier technique solves the problem in a straightforward fashion. A general solution minimizing the sum of the squares of the joint speeds is developed below.

A measure of the magnitude of the joint speeds is written as

$$c(\dot{\Phi}) = \dot{\Phi}^T \dot{\Phi} \quad (3.9)$$

A solution which minimizes this performance criterion while simultaneously finding the required joint speeds is readily formulated. Equation (3.2) is rewritten as an equality constraint

$$\dot{x} - J\dot{\Phi} = 0 \quad (3.10)$$

Adjoining the constraint set, Equation (3.9), to the performance criterion via a LaGrange multiplier forms the Hamiltonian.

$$H(\dot{\Phi}, \lambda) = \dot{\Phi}^T \dot{\Phi} + \lambda^T [\dot{x} - J\dot{\Phi}] \quad (3.11)$$

The conditions for optimality are

$$\frac{\partial H}{\partial \lambda} = 0 \quad (3.7)$$

and

$$\frac{\partial H}{\partial \dot{\Phi}} = 0 \quad (3.12)$$

Rewriting Equation (3.11) by taking the partial derivative with respect to $\dot{\Phi}$ gives

$$2\dot{\Phi}^T - \lambda^T J = 0 \quad (3.13)$$

Solving Equation (3.13) for $\dot{\Phi}$ gives

$$\dot{\Phi} = \frac{1}{2} \mathbf{J}^T \lambda \quad (3.14)$$

Now, rewriting Equation (3.11) by taking the partial derivative with respect to λ gives

$$\dot{\mathbf{x}} - \mathbf{J}\dot{\Phi} = 0 \quad (3.15)$$

and substituting for $\dot{\Phi}$ as given by Equation (3.14) gives

$$\dot{\mathbf{x}} - \frac{1}{2} \mathbf{J}\mathbf{J}^T \lambda = 0 \quad (3.16)$$

It is now possible to solve for the LaGrange multipliers as

$$\lambda = 2[\mathbf{J}\mathbf{J}^T]^{-1} \dot{\mathbf{x}} \quad (3.17)$$

This step requires the crucial assumption that the Jacobian is full-rank. Given that the Jacobian is full-rank, $\mathbf{J}\mathbf{J}^T$ is a full-rank and square matrix which can be uniquely inverted as indicated in Equation (3.17). The Jacobian is not full-rank at singularities, where this method fails. Assuming, however, that the Jacobian is full-rank and substituting for λ in Equation (3.16) gives

$$\dot{\Phi} = \mathbf{J}^T [\mathbf{J}\mathbf{J}^T]^{-1} \dot{\mathbf{x}} \quad (3.18)$$

The general form of this equation is well-known and seductively simple. Several comments are in order. First, Equation (3.18) only optimizes a single criterion. That criterion is extremely simple and only loosely associated with any improvement in robot performance. Second, this solution fails at singularities. Finally, the solution does not consider position-level constraints, such as joint limits. Taken together, these characteristics make Equation (3.18) unsuitable for practical application. The derivation is, however, correct mathematically. Thus,

Equation (3.18) is a general solution to the inverse kinematics problem that minimizes the sum of the squares of the joint speeds.

3.1.2 Minimizing Kinetic Energy

The kinetic energy content of the robot is a performance criterion with a more subtle physical interpretation. Minimizing this criterion may give increased precision and higher operating speeds. The kinetic energy criterion is

$$c(\dot{\Phi}) = \dot{\Phi}^T [\mathbf{I}^*] \dot{\Phi} \quad (3.19)$$

where $\mathbf{I}^*$ is the inertia tensor for the robot. A solution minimizing this performance criterion while simultaneously solving for the required joint speeds may be derived. Equation (3.2) is rewritten as an equality constraint

$$\dot{\mathbf{x}} - \mathbf{J}\dot{\Phi} = \mathbf{0} \quad (3.20)$$

Adjoining the constraint set, Equation (3.20), to the performance criterion via a LaGrange multiplier vector forms the Hamiltonian.

$$H(\dot{\Phi}, \lambda) = \dot{\Phi}^T [\mathbf{I}^*] \dot{\Phi} + \lambda^T [\dot{\mathbf{x}} - \mathbf{J}\dot{\Phi}] \quad (3.21)$$

The conditions for optimality are

$$\frac{\partial H}{\partial \lambda} = \mathbf{0} \quad (3.7)$$

and

$$\frac{\partial H}{\partial \dot{\Phi}} = \mathbf{0} \quad (3.12)$$

Rewriting Equation (3.21) by taking the partial derivative with respect to $\dot{\Phi}$ gives

$$2\dot{\Phi}^T \mathbf{I}^* - \lambda^T [\mathbf{J}] = 0 \quad (3.22)$$

and solving Equation (3.22) for $\dot{\Phi}$ gives

$$\dot{\Phi} = \frac{1}{2} [\mathbf{I}^*]^{-1} \mathbf{J}^T \lambda \quad (3.23)$$

Now, rewriting Equation (3.21) by taking the partial derivative with respect to λ gives

$$\dot{\mathbf{x}} - \mathbf{J} \dot{\Phi} = 0 \quad (3.24)$$

Substituting for $\dot{\Phi}$ as given by Equation (3.23) gives

$$\dot{\mathbf{x}} - \frac{1}{2} \mathbf{J} [\mathbf{I}^*]^{-1} \mathbf{J}^T \lambda = 0 \quad (3.25)$$

The LaGrange multipliers are now found as

$$\lambda = 2 [\mathbf{J} [\mathbf{I}^*]^{-1} \mathbf{J}^T]^{-1} \dot{\mathbf{x}} \quad (3.26)$$

Again this step requires a full-rank Jacobian. It also requires that $\mathbf{I}^*$ is definite (which it is). Given these assumptions, $\mathbf{J} [\mathbf{I}^*]^{-1} \mathbf{J}^T$ is a full-rank square matrix and can be uniquely inverted. Substituting for λ in Equation (3.23) gives

$$\dot{\Phi} = [\mathbf{I}^*]^{-1} \mathbf{J}^T [\mathbf{J} [\mathbf{I}^*]^{-1} \mathbf{J}^T]^{-1} \dot{\mathbf{x}} \quad (3.27)$$

This equation provides the joint speeds .

Again, this derivation is only good when the Jacobian is full-rank. By minimizing system kinetic energy, Equation (3.27) may enhance performance under certain conditions. Equation (3.27) does not, however, consider position-level constraints which may make it unsuitable for practical applications. Nonetheless, this solution is general in the sense that it finds the minimum kinetic energy solution for any serial robot geometry.

3.1.2 Minimizing Joint Accelerations

It is also possible to formulate the inverse kinematics problem with optimization at the acceleration level. As with the velocity-level solutions, these relationships are linear and are thus much easier to solve than the position-level equations. General solutions for minimizing the joint accelerations and the joint torques may be derived.

A performance criterion which measures the magnitude of the change in joint speeds with respect to time is

$$c(\ddot{\Phi}) = \ddot{\Phi}^T \ddot{\Phi} \quad (3.28)$$

The solution which minimizes this criterion while simultaneously solving for the required joint accelerations may be found using the method of LaGrange multipliers. Equation (3.3) is rewritten as an equality constraint

$$\ddot{\mathbf{x}} - \mathbf{J}\ddot{\Phi} - \dot{\mathbf{J}}\dot{\Phi} = \mathbf{0} \quad (3.29)$$

Adjoining the constraint set, Equation (3.29), to the performance criterion, Equation (3.28), via a LaGrange multiplier forms the Hamiltonian.

$$H(\ddot{\Phi}, \lambda) = \ddot{\Phi}^T \ddot{\Phi} + \lambda^T (\ddot{\mathbf{x}} - \mathbf{J}\ddot{\Phi} - \dot{\mathbf{J}}\dot{\Phi}) \quad (3.30)$$

The conditions for optimality are

$$\frac{\partial H}{\partial \lambda} = \mathbf{0} \quad (3.7)$$

and

$$\frac{\partial H}{\partial \ddot{\Phi}} = \mathbf{0} \quad (3.31)$$

Rewriting Equation (3.30) by taking the partial derivative with respect to $\ddot{\Phi}$ gives

$$2\ddot{\Phi}^T - \lambda^T \mathbf{J} = 0 \quad (3.32)$$

Solving Equation (3.32) for $\ddot{\Phi}$ gives

$$\ddot{\Phi} = \frac{1}{2} \mathbf{J}^T \lambda \quad (3.33)$$

Now, rewriting Equation (3.30) by taking the partial derivative with respect to λ gives

$$\ddot{\mathbf{x}} - \mathbf{J}\ddot{\Phi} - \dot{\mathbf{J}}\dot{\Phi} = 0 \quad (3.34)$$

and substituting for $\ddot{\Phi}$ as given by Equation (3.33) gives

$$\ddot{\mathbf{x}} - \frac{1}{2} \mathbf{J}\mathbf{J}^T \lambda - \dot{\mathbf{J}}\dot{\Phi} = 0 \quad (3.35)$$

The LaGrange multipliers may now be found as

$$\lambda = 2[\mathbf{J}\mathbf{J}^T]^{-1}[\ddot{\mathbf{x}} - \dot{\mathbf{J}}\dot{\Phi}] \quad (3.36)$$

Which assumes the Jacobian is full-rank. Substituting for λ in Equation (3.35) gives

$$\ddot{\Phi} = \mathbf{J}^T[\mathbf{J}\mathbf{J}^T]^{-1}[\ddot{\mathbf{x}} - \dot{\mathbf{J}}\dot{\Phi}] \quad (3.37)$$

Equation (3.37) represents a solution to the inverse kinematics problem which simultaneously minimizes the sum of the squares of the joint accelerations.

As expected, Equation (3.37) is only valid if the Jacobian is full-rank.

3.1.1 Minimization of Joint Torques

It is also possible to solve for the joint torques which, neglecting modeling errors, result in the desired end-effector motion. The sum of the squares of the joint torques is the performance criterion.

$$c(\boldsymbol{\tau}) = \frac{1}{2} \boldsymbol{\tau}^T \boldsymbol{\tau} \quad (3.38)$$

The vector $\boldsymbol{\tau}$ represents the joint torques. The joint torques are given by the relationship

$$\boldsymbol{\tau} = \mathbf{I}^* \ddot{\boldsymbol{\Phi}} + \mathbf{c} + \mathbf{g} \quad (3.39)$$

where $\mathbf{I}^*$ represents the inertia tensor, $\mathbf{c}$ represents the centripetal and Coriolis acceleration terms, and $\mathbf{g}$ the effect of gravity. Writing the joint accelerations in terms of the joint torques gives

$$\ddot{\boldsymbol{\Phi}} = [\mathbf{I}^*]^{-1} (\boldsymbol{\tau} - \mathbf{c} - \mathbf{g}) \quad (3.40)$$

Substituting the joint accelerations into the original acceleration-level constraints on the end-effector motion, Equation (3.3), gives an equality constraint in terms of the joint torques

$$\ddot{\mathbf{x}} - \mathbf{J}[\mathbf{I}^*]^{-1} (\boldsymbol{\tau} - \mathbf{c} - \mathbf{g}) - \dot{\mathbf{J}}\dot{\boldsymbol{\Phi}} = \mathbf{0} \quad (3.41)$$

Adjoining the constraint set, Equation (3.41), to the performance criterion, Equation (3.38), forms the Hamiltonian.

$$H(\boldsymbol{\tau}, \boldsymbol{\lambda}) = \frac{1}{2} \boldsymbol{\tau}^T \boldsymbol{\tau} + \boldsymbol{\lambda}^T [\ddot{\mathbf{x}} - \mathbf{J}[\mathbf{I}^*]^{-1} (\boldsymbol{\tau} - \mathbf{c} - \mathbf{g}) - \dot{\mathbf{J}}\dot{\boldsymbol{\Phi}}] \quad (3.42)$$

The conditions for optimality are

$$\frac{\partial H}{\partial \boldsymbol{\lambda}} = \mathbf{0} \quad (3.7)$$

and

$$\frac{\partial H}{\partial \boldsymbol{\tau}} = \mathbf{0} \quad (3.43)$$

Taking the partial derivatives as indicated by Equation (3.7) gives

$$\ddot{\mathbf{x}} - \mathbf{J}[\mathbf{I}^*]^{-1}(\boldsymbol{\tau} - \mathbf{c} - \mathbf{g}) - \dot{\mathbf{J}}\dot{\boldsymbol{\Phi}} = \mathbf{0} \quad (3.44)$$

Taking the partial derivatives as indicated by Equation (3.43) gives

$$\boldsymbol{\tau} - [\mathbf{I}^*]^{-1} \mathbf{J}^T \boldsymbol{\lambda} = \mathbf{0} \quad (3.45)$$

Solving Equations (3.44) and (3.45) for $\boldsymbol{\tau}$ and equating the result gives the LaGrange multiplier, $\boldsymbol{\lambda}$. Substituting the LaGrange multiplier into (3.45) gives

$$\boldsymbol{\tau} = [\mathbf{I}^*]^{-T} \mathbf{J}^T [\mathbf{J}[\mathbf{I}^*]^{-1} [\mathbf{I}^*]^{-T} \mathbf{J}^T]^{-1} [\ddot{\mathbf{x}} - \mathbf{J}[\mathbf{I}^*]^{-1}(\mathbf{c} + \mathbf{g}) - \dot{\mathbf{J}}\dot{\boldsymbol{\Phi}}] \quad (3.46)$$

Equation (3.46) minimizes the sum of the squares of the joint torques while simultaneously producing the desired end-effector motion. Other researchers have also derived this solution (Carnigan, 1991). In a practical application of Equation (3.46), the accuracy of the end-effector motion is determined by the accuracy of the dynamic model represented by Equation (3.39). Since no model is perfect, some sort of feedback to correct end-effector errors would have to be included if this method were applied to an actual robot.

3.2 DIRECT SEARCH

Each of the preceding examples expressed the inverse kinematics problem as an unconstrained minimization problem using LaGrange multipliers. Calculus was then used for the actual minimization. The numerical solution of the inverse kinematics problem may employ an analogous technique. The displacement constraints on the end-effector are expressed as an unconstrained error function. For instance, the Cartesian error between the current and the desired displacement of the end-effector is one possible unconstrained error function. The error function is minimized using numerical minimization techniques. A solution to the

inverse kinematics problem is found if the error is minimized to zero. Direct search is one numerical minimization technique applied to the inverse kinematics problem in this manner.

At its most basic level, direct search is a method of solving problems numerically using sets of trial solutions to guide a search. The search is successful if an actual solution is found. Direct search methods use only function values and are distinguished from gradient and higher-order methods requiring values for the function's derivatives. Because a direct search does not require a function's derivatives, an inverse kinematics implementation based on direct search may converge at singularities.

Most direct search strategies share a degree of commonality. The search begins at an estimate of the solution. This estimate serves as an initial base point for the search. A pattern of explorations about the base point generates a set of trial solutions. The trial solutions are evaluated and a decision making process selects one trial solution as the next base point. Another pattern of local exploration is then performed about this new base point. This process of local exploration and decision making continues until the search, at least a successful one, finds a solution. Different exploration patterns and different decision making processes may be combined to form a number of direct search strategies. The rest of this chapter develops direct search inverse kinematics techniques for robots with six degrees of freedom. Chapter 4 extends direct search to the multicriteria inverse kinematics problem for redundant robots.

3.2.1 Transformation of End-effector Constraints

The inverse kinematics problem may be expressed as an unconstrained optimization problem by transforming the end-effector constraints via an error function. The error represents the difference between the desired and the current end-effector displacements. In a direct search, the error is calculated for each of the trial solutions. Much of the generality of direct search as an inverse kinematics method is associated with expressing the end-effector constraints as a transformed error function. The error is found using the forward transformation equations. The forward transformation equations determine the position and orientation of the end-effector for a given set of joint displacements. The Denavit and Hartenberg kinematic parameters for the robot determine the forward transformation equations in an entirely general fashion.

Cartesian coordinates (in translation and rotation) are a natural choice for expressing the error function. There are several ways of writing the error in terms of Cartesian coordinates. One of them is as the sum of the errors in position and orientation.

$$E = E_p + cE_o \quad (3.47)$$

In Equation (3.47), E represents the total error, E_p the error in position, E_o the error in orientation, and c a scaling parameter to balance the units and magnitudes between position and orientation. The error in position is calculated as

$$E_p = \sqrt{(x_T - x_D)^2 + (y_T - y_D)^2 + (z_T - z_D)^2} \quad (3.48)$$

where x , y , and z represent the translations. The subscript T indicates trial solution and the subscript D indicates desired solution. The error in orientation is

calculated in a similar manner using the three end-effector orientations rather than the translations. XYZ fixed angles are used here, but there are many ways of describing the end-effector orientation and any of them may be used in a similar fashion. XYZ fixed angles are three successive rotations about the X , Y , and Z axes of a fixed reference frame.

$$E_O = \sqrt{(R_{xT} - R_{xD})^2 + (R_{yT} - R_{yD})^2 + (R_{zT} - R_{zD})^2} \quad (3.49)$$

In this equation, R represents the magnitude of the rotation. The subscripts x , y , and z represent the axes of rotation. The subscript T indicates trial solution, and the subscript D indicates desired solution.

The scaling parameter in Equation (3.47), c , may be chosen in a variety of ways. Chapter 4 discusses procedures for calculating scaling parameters in detail. Providing comparable magnitudes of error in position and orientation is the basic goal when calculating the scaling parameter.

Calculating the error in the manner of Equations (3.48) and (3.49) requires calculating the end-effector placement corresponding to a given trial solution. Calculating the end-effector placement involves the forward transformation equations. The forward transformation equations may be written for a general robot geometry in terms of Denavit and Hartenberg parameters. These parameters assign a local reference frame to each of the robot's joint axes.

For a given robot, a local reference frame, $\{i\}$, is related to the previous local frame, $\{i - 1\}$, by two screw operators

$${}^{i-1}\mathbf{T}_i = \text{Screw}_x(a_{i-1}, \alpha_{i-1}) \text{Screw}_z(d_i, \Phi_i) \quad (3.50)$$

where ${}^{i-1}\mathbf{T}$ denotes the transform relating the two frames. $\text{Screw}_x(a_{i-1}, \alpha_{i-1})$ denotes a translation along the x-axis by the link length a_{i-1} and a rotation about the x-axis by the twist angle α_{i-1} . Similarly, $\text{Screw}_z(d_i, \Phi_i)$ denotes a translation along the z-axis by the offset d_i and a rotation about the z-axis by the joint variable Φ_i . These two screw operations completely define the transform relating frame $\{i\}$ to frame $\{i - 1\}$.

$${}^{i-1}\mathbf{T} = \begin{bmatrix} \cos\Phi_i & -\sin\Phi_i & 0 & a_{i-1} \\ \sin\Phi_i\cos\alpha_{i-1} & \cos\Phi_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\Phi_i\sin\alpha_{i-1} & \cos\Phi_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.51)$$

This equation represents a homogeneous transform because it contains both rotational and translational elements. In a general fashion, concatenating these transforms produces a single transform which relates the final frame $\{n\}$ to the base frame $\{0\}$.

$${}^0\mathbf{T} = {}^0\mathbf{T}_1 {}^1\mathbf{T}_2 {}^2\mathbf{T}_3 \dots {}^{n-1}\mathbf{T}_n \quad (3.52)$$

The homogeneous transform ${}^0\mathbf{T}$ represents the kinematic displacement equations in joint space and completely defines the end-effector placement for a given trial solution.

Extracting the end-effector placement from the homogeneous transform ${}^0\mathbf{T}$ forms an expression for the Cartesian error as given in Equation (3.47). The translational portion appears explicitly as the first three numbers in the last column of ${}^0\mathbf{T}$ and thus presents no difficulty. There is, however, a singularity associated with extracting the rotational portion. This singularity depends on the

particular angle set convention describing the orientation of the end-effector. In the case of XYZ fixed rotation angles, there is a singularity when the rotation about the Y axis is ninety degrees. In this case, the X and Z axes of rotation are collinear and only their sum may be computed. Logic must be included to detect and resolve the singularity. Including only the sum of the X and Z rotations when formulating the rotational portion of the Cartesian error function at this singularity is a simple resolution.

$$E_O = \sqrt{\left[(R_{xT} + R_{zT}) - (R_{xD} + R_{zD}) \right]^2 + (R_{yT} - R_{yD})^2} \quad (3.53)$$

This singularity is associated with the angle set convention and is not dependent upon the particular robot geometry. Thus, the procedure for formulating the error function remains general with respect to the robot's geometry.

Using the rotational portion of the homogeneous transform ${}^0_n\mathbf{T}$ without extracting any particular angle set generates the trial solutions and associated errors without introducing a singularity. Simply formulate the error function using the nine rotational elements in ${}^0_n\mathbf{T}$ as shown in Equation (3.54).

$$E_O = \sqrt{\sum_{i=1}^3 \sum_{j=1}^3 (r_{ijT} - r_{ijD})^2} \quad (3.54)$$

The scaling may also be simplified since the individual elements, r_{ij} , in ${}^0_n\mathbf{T}$ always vary between 1 and -1.

3.2.2 Exploration Patterns

An exploration strategy generates options during the direct search. Each option is called a trial solution. Many different exploration strategies can be formulated. They may be systematic, random, or both. There are also local and

global strategies. The direct search developed here is based on local explorations using the concept of a joint-level perturbation. The perturbations are performed in a pattern about the current base point of the search. Perturbation at the joint level simply means temporarily changing one or more of the joint displacements in either the positive or negative direction. The perturbations are only performed on the forward kinematic equations, and not on the actual robot. This chapter presents three different systematic exploration strategies. Chapter 4 discusses a strategy with a degree of randomness, called simulated annealing.

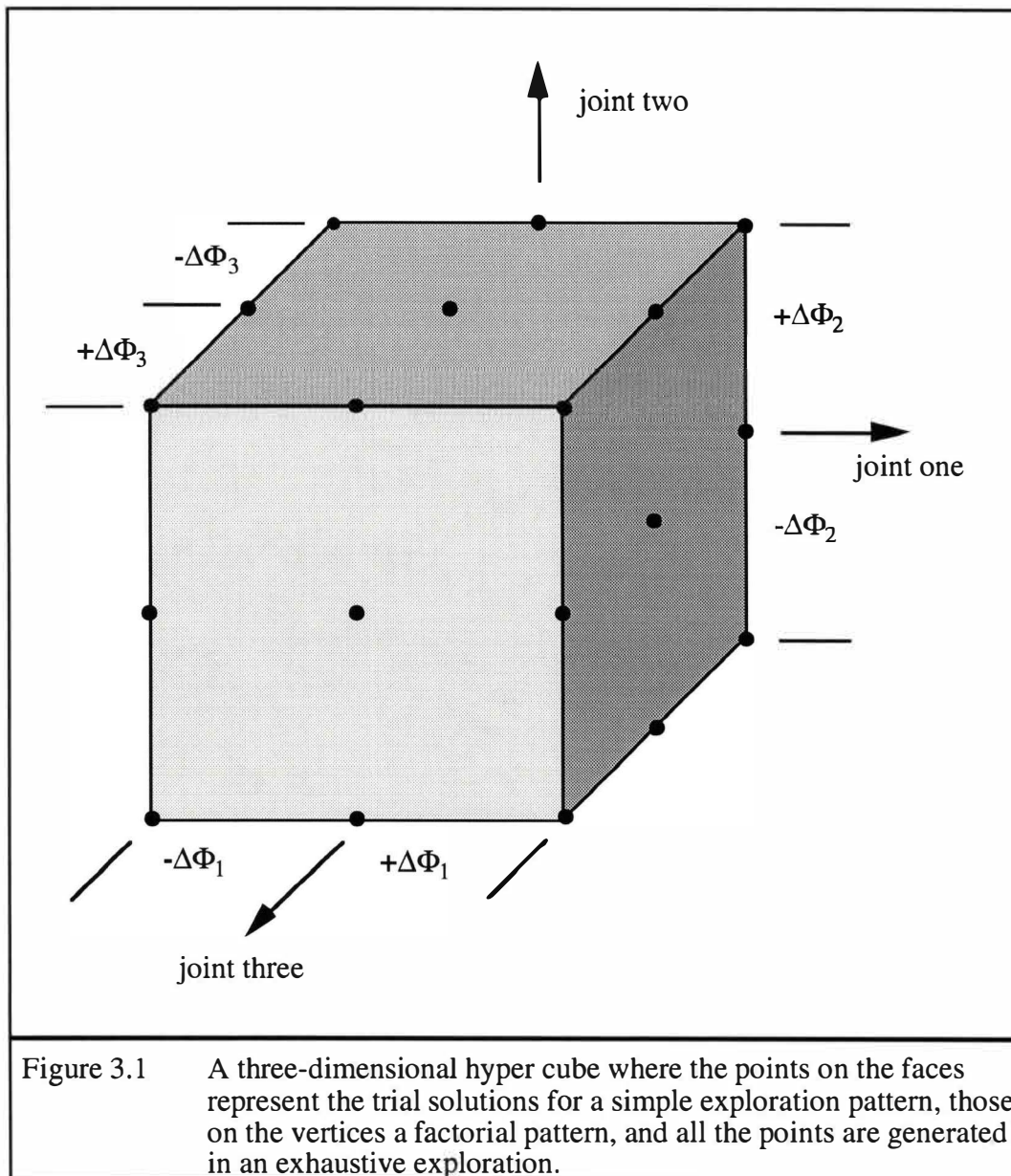
Simple exploration describes a strategy perturbing only one joint displacement at a time. This exploration pattern finds the individual influence of each joint on the end-effector motion at any given base point during the search. The simple pattern is fast computationally, but only explores a limited portion of the locality around the current base point.

Table 3.1 Three local exploration patterns and some of their attributes.	
Exploration Pattern	Attributes
simple	fast finds individual influence of each joint explores only a limited portion of the locality
factorial	based on factorial design principles finds combinational effects somewhat slower than simple exploration much faster than exhaustive exploration
exhaustive	guaranteed to locate a local minimum explores every possible direction computationally intensive

Another perturbation pattern is based on 2^n factorial design. Factorial design is a mature and interesting research area in its own right. It includes far more than just 2^n exploration patterns. However, for the purposes of this discussion, the pattern based on 2^n factorial design is called factorial exploration. The factorial exploration pattern consists of all combinations of perturbed joint displacements. It does not generate combinations including unperturbed joint displacements. The factorial pattern explores more of the locality than the simple pattern, but less than the exhaustive pattern described below.

The most thorough exploration strategy is called the exhaustive pattern. The exhaustive exploration pattern generates all combinations of perturbed and unperturbed joint displacements. It explores in every possible direction from the current base point. Of the three exploration strategies just discussed, the exhaustive pattern is the most thorough, but also the most computationally demanding. Section 3.2.4 shows that the exhaustive exploration strategy is the only one that guarantees the finding of a local minimum.

These three exploration patterns correspond to points on the faces, edges, and vertices of a hypercube with the current base point at its center. Each point represents a trial solution. The dimension of the hypercube is equal to the robot's number of degrees of freedom. Figure 3.1 shows an example for a robot with three degrees of freedom. The points on the faces of the cube correspond to the simple pattern and the points at the vertices of the cube correspond to the factorial pattern. The exhaustive pattern includes all of the points.



3.2.3 Decision Making

A decision making strategy chooses one of the trial solutions as the next base point for the search. The decision is based on information calculated for

each of the trial solutions. A fully-constrained robot only requires calculating the Cartesian error. For a redundant robot, the decisions may be based on performance criteria as well. Three different decision making strategies are discussed below. These strategies may be applied to both redundant and fully-constrained robots. Chapter 4 discusses their application to redundant robots.

The steepest descent decision making strategy calculates the error for each of the trial solutions. This strategy is called steepest descent because it calculates the error for each of the trial solutions before choosing the minimum as the next base point.

An opportunistic decision making strategy often considerably increases the speed with which direct search finds the solution. In the opportunistic strategy, as soon as a trial solution decreasing the error is found, that solution is immediately chosen as the next base point and the search continues from there. This decreases the computational overhead by evaluating fewer trial solutions at each base point.

Table 3.2 Three decision making strategies and some of their attributes.	
Strategy	Attributes
steepest descent	always chooses the minimum from a given set of trial solutions most demanding computationally
opportunistic	chooses a better solution as soon as one is found considerably increases speed of solution
pattern-move	incorporates the history of the search as part of the algorithm considerably increases the speed of solution effectiveness decreases as the non-linearity increases

The most famous of the direct search decision making strategies is the pattern-move due to Hooke and Jeeves (1961). The pattern-move strategy

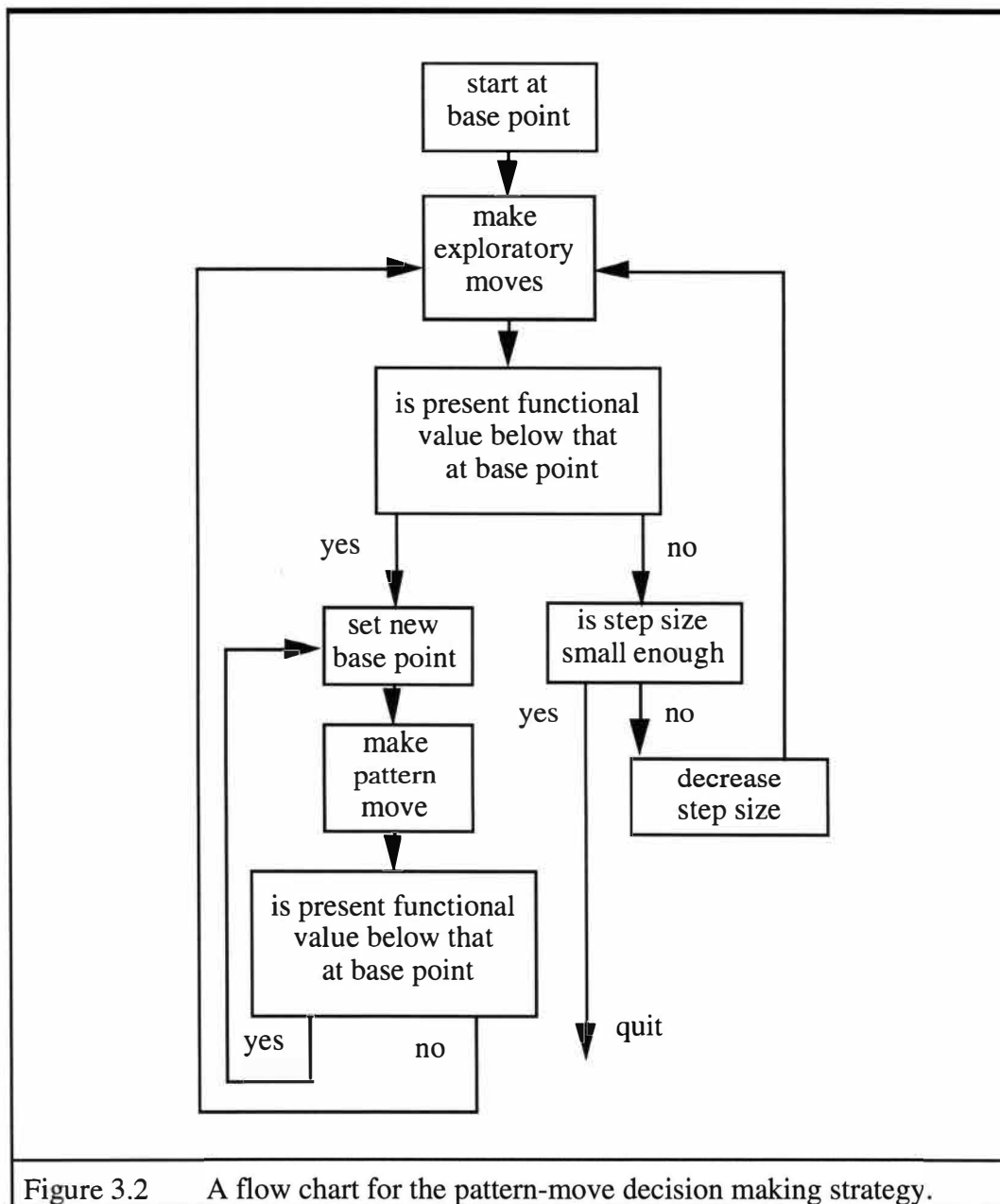


Figure 3.2 A flow chart for the pattern-move decision making strategy.

predicts which search direction likely leads towards a solution using a recent history of the search. Essentially, whenever a move is successful, that move is tried again. The movement towards the solution is the pattern-move. Figure 3.2 presents a flow chart of the algorithm, but the seminal manuscript of Hooke and Jeeves provides much more insight.

These three strategies make decisions based only on the first order (position level) information provided by the transformed end-effector constraints. The use of higher order geometric properties would certainly increase the convergence speed of the direct search. Thomas and Tesar (1982) formulated the first and second order geometric properties for a general serial robot in a concise fashion that explicitly shows the effect of each joint's motion on the resulting motion of the system. Using this information seems natural for an inverse kinematics strategy for general serial robots. The problem occurs at mathematical singularities. At these singularities the information provided by the higher order geometric properties becomes very difficult to interpret. Including the higher order properties in the basis for decision making would result in an algorithm that behaved differently at singularities. One of the goals for this inverse kinematics work was a method that would be equally effective throughout the robot's workspace.

Any search strategy must ultimately decide to terminate the search. Each of the direct search strategies discussed here repeatedly calculate an error between the trial and the actual solutions. Zero error (within an acceptable tolerance) is a reasonable condition for terminating the search. For an industrial robot, this could

require very small step sizes at the joint level. Beginning the search at a coarse resolution and then reducing the step size after bounding a solution is often advantageous. This suggests another condition for terminating the search that has physical relevance to robots under servo control at the joints. Simply terminate the search when the step size is smaller than the servo resolution.

Unfortunately, terminating the search without finding a solution is sometimes necessary. This may be because the search failed to find a solution, though one exists, or it may be because there is no solution. In either case, the search reaches an impasse from which it cannot find any better base point from which to continue. Chapter 4 develops several methods of adjusting scale factors to force the convergence of direct search algorithms. To prevent the algorithm from degenerating into an infinite loop, the scale factors may only be adjusted a finite number of times. After adjusting the scale factors several times, deciding to terminate the search without finding a solution may be the most reasonable choice.

3.2.4 Optimality

This chapter presents several direct search strategies. In each of them, a Cartesian error is formulated as an unconstrained objective function. Minimizing the function until its value is zero is the goal of the search. It is certainly reasonable to question which, if any, of the strategies guarantee minimization of an unconstrained objective function and exactly what sort of minimum is found. Remember that direct search is a discrete method which only finds minima to within the resolution of the search. Also remember that the direct search

strategies developed in this chapter employ local exploration patterns. Therefore, only local minimization will be discussed.

This section first defines the terms local minimum and isolated local minimum. The conditions such that a given set of joint displacements represents a minimum are then derived. Finally, it is shown that direct search with an exhaustive exploration pattern always finds a set of joint displacements which satisfies the conditions for a local minimum.

Definition 1: *A local minimum is a point such that all other points in the locality represent function values that are greater than or equal to the function value at the local minimum. Symbolically, this may be written as*

$$\Delta f(\Phi) = f(\Phi) - f(\bar{\Phi}) \geq 0 \quad (3.55)$$

where $\Delta f(\Phi)$ represents the change in the function value, $f(\bar{\Phi})$ represents the function value at the local minimum and $f(\Phi)$ represents the function value at any other point in the locality.

Definition 2: *An isolated local minimum is a point such that all other points in the locality represent function values that are strictly greater than the function value at the isolated local minimum. Symbolically, this may be written as*

$$\Delta f(\Phi) = f(\Phi) - f(\bar{\Phi}) > 0 \quad (3.56)$$

where $\Delta f(\Phi)$, $f(\bar{\Phi})$, and $f(\Phi)$ are as defined above.

The decision making strategies discussed in this chapter always choose a trial solution which decreases the value of the objective function if there is one. No provision was made for choosing trial solutions which represent equivalent function values and the search may terminate in their presence. Therefore, these

direct search strategies cannot possibly guarantee an isolated local minimum. The conditions under which the joint angle set $\bar{\Phi}$ meets the definition of a local minimum are now derived. The derivation begins with the familiar Taylor series expansion for a function of multiple variables.

$$f(\Phi) = f(\bar{\Phi}) + \nabla f(\bar{\Phi})^T \Delta\Phi + \frac{1}{2} \Delta\Phi^T \nabla^2 f(\bar{\Phi}) \Delta\Phi + \text{HOTS} \quad (3.57)$$

In this equation, $f(\Phi)$ represents the objective function, $\bar{\Phi}$ the expansion point, $\nabla f(\bar{\Phi})$ and $\nabla^2 f(\bar{\Phi})$ the first and second derivatives evaluated at the expansion point, and HOTS the higher order terms. Dropping the higher order terms and solving for changes in the objective function yields

$$\Delta f(\Phi) = \nabla f(\bar{\Phi})^T \Delta\Phi + \frac{1}{2} \Delta\Phi^T \nabla^2 f(\bar{\Phi}) \Delta\Phi \quad (3.58)$$

The first requisite for an extremum of a function is the stationary condition

$$\nabla f(\bar{\Phi}) = \mathbf{0} \quad (3.59)$$

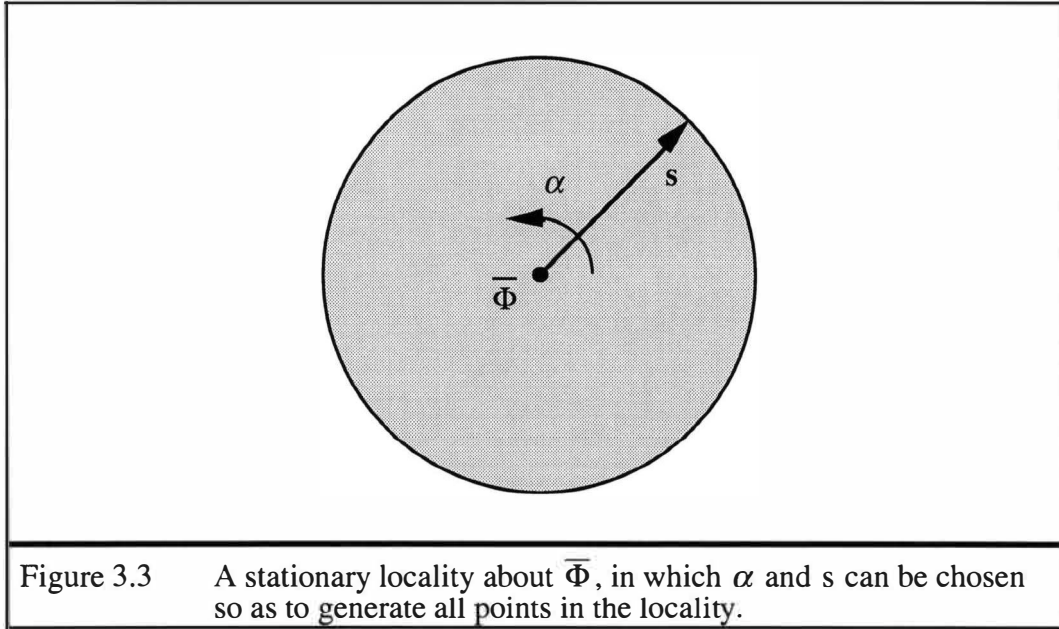
Equation (3.59) is often called the necessary condition for a minimum. With the stationary condition, Equation (3.58) now becomes

$$\Delta f(\Phi) = \frac{1}{2} \Delta\Phi^T \nabla^2 f(\bar{\Phi}) \Delta\Phi \quad (3.60)$$

The sign of $\Delta f(\Phi)$ is very important and is determined by the right hand side of this equation. If it can be shown that $\Delta f(\Phi)$ is always greater than or equal to zero, then by Definition 1, the current expansion point is a local minimum. Notice that the right hand side of Equation (3.60) is in quadratic form. If the quadratic form is positive semi-definite, then by the definition of positive semi-definiteness, $\Delta f(\Phi)$ is greater than or equal to zero for all $\Delta\Phi$ and the current expansion point represents a local minimum.

The condition that $\nabla f(\bar{\Phi})$ must equal zero is treated first. Recall that the direct search inverse kinematics strategies test feasible directions by varying each of the joint displacements in Φ . The strategy is to move in the direction of decreasing function values, $f(\Phi)$, until there is no direction which further decreases $f(\Phi)$. Once the search is started, it continues in directions decreasing $f(\Phi)$ until finding no direction that further decreases $f(\Phi)$.

The term $\nabla f(\bar{\Phi})$ is the vector of first partial derivatives with respect to each of the joint displacements. Addressing this term requires searching the principle directions. The principle directions relate to changing each joint displacement individually while holding the others constant. Changing each joint displacement individually and calculating the function value generates a discrete analog of one element of the vector $\nabla f(\bar{\Phi})$. Both the simple and the exhaustive exploration patterns search the principle directions. That a search direction will not further decrease $f(\Phi)$ indicates that the function value experiences either a curvature reversal or a loss of curvature in that direction. Recall that curvature is defined at a point as the value of the derivative of the polar angle with respect to the arc length. Given that $f(\Phi)$ is continuous in the first derivatives, either a curvature reversal or a loss of curvature must be accompanied by the first derivative passing through zero.



To show that direct search, at least with an exhaustive exploration pattern, finds a point where $\nabla^2 f(\bar{\Phi})$ is positive semi-definite consider the two dimensional case shown in Figure 3.3. Directions radiating from stationary point $\bar{\Phi}$ are generated by

$$\tilde{\Phi} = \bar{\Phi} + \alpha s(\bar{\Phi}) \quad (3.61)$$

where α and s can be chosen so as to generate all points in the locality of $\bar{\Phi}$. Substituting Equation (3.61) into (3.60) gives

$$\Delta f(\Phi) = \frac{\alpha^2}{2} s^T \nabla^2 f(\bar{\Phi}) s \quad (3.62)$$

which is in the quadratic form

$$Q(\mathbf{z}) = \mathbf{z}^T \mathbf{A} \mathbf{z} \quad (3.63)$$

The quadratic form of Equation (3.62) is positive semi-definite if, for all Φ , the value of $f(\Phi)$ is greater than or equal to the value of $f(\bar{\Phi})$. Thus, to guarantee a

local minimum, all directions must be searched in order to generate all $f(\Phi)$ values in the locality. Only the exhaustive perturbation strategy searches all feasible directions. The direct search terminates when there are no feasible directions decreasing $f(\Phi)$, though there may be directions of equivalent values. This corresponds to the positive semi-definite case.

Thus it is shown that direct search with an exhaustive exploration strategy guarantees finding a local minimum of the objective function, $f(\Phi)$. An argument based on common sense is perhaps more convincing. If the search tests every feasible direction, and none decrease the function value, then the current point must be a local minimum. Without testing every feasible direction, direct search cannot guarantee that there isn't at least one direction that further decreases the function value.

Showing that direct search finds local minima also shows the condition under which direct search fails to find a solution to the inverse kinematics problem. Direct search may fail if there is a minimum directly between the current and the desired end-effector displacements. The physical interpretation of this situation is that the robot cannot move directly from where it is, to where it is commanded to go. This situation always occurs at workspace boundaries as the robot physically cannot move in the desired direction. The failure mode of the direct search is rather benign under these circumstances. The search terminates and is ready to find a new solution if given an attainable end-effector placement.

Unfortunately, there may be bounded local minima between the current and the desired end-effector displacements, even though both are attainable. This may result in the search terminating prior to finding a solution. Note that a

bounded local extremum is but one of the failure modes of the Newton-Raphson numerical solution method. A direct search typically deals with bounded local minima by changing scaling factors in order to change the shape of the objective function, $f(\Phi)$, and force convergence. Chapter 4 develops several methods for initially choosing scaling factors and for dynamically changing their values as the search progresses.

3.2.5 Relative Computational Complexity

The computational burden and the complexity of the computer code are reasonable considerations with any method of solving a problem on a computer. Unfortunately, it is difficult to assess the computational burden of most direct search strategies. This is because direct search often includes heuristics, such as the pattern move or the opportunistic decision making strategy, which may accelerate the solution speed. The effectiveness of these acceleration strategies influences the solution speed.

Without any heuristics, the relative computational burdens for the three perturbation methods (simple, factorial and exhaustive) are respectively: n , 2^n , and 3^n ; where n represents the robot's number of degrees of freedom. However, even without the acceleration heuristics, it is possible that a more thorough exploration strategy will generate trial solutions leading more directly to a solution. For this reason, even without acceleration heuristics, the relative solution speed can only be estimated.

A direct search algorithm which solves the inverse kinematics for a general serial robot can be translated into computer code in a simple and concise

fashion. The program which generated the results presented in this dissertation is less than 300 lines of C computer code. This program includes the simple, factorial, and exhaustive exploration patterns as well as the steepest descent, opportunistic, and pattern-move decision making strategies. The code is general and requires only the appropriate Denavit and Hartenberg parameters to solve the inverse kinematics problem for any specific serial robot. The simplicity of the computer code for these direct search methods results from the repeated application of simple logic. The simplicity of the formulation suggests the possibility of implementing direct search inverse kinematics with specialized high-speed electronic hardware. In particular, hardware for performing the forward transformations that generate the error function would increase the speed of a direct search algorithm considerably.

3.2.6 Recommended Direct Search Applications

Since there are many different inverse kinematics methods, considering the conditions under which direct search techniques would be recommended is very important. These conditions may be discussed in terms of the generality of direct search inverse kinematics and its ability to converge at exactly singular positions of the robot.

This dissertation develops direct search techniques general in the sense that they will solve the inverse kinematics problem for a robot of any serial geometry. The exploration patterns and decision making strategies are formulated without geometric simplification or algebraic manipulation particular to any given

robot. This makes direct search inverse kinematics attractive when considering many different robot geometries, as is typically the case during robot design.

Table 3.3 Some positive and negative qualities of the direct search inverse kinematics method.

Positive Qualities	Negative Qualities
General with respect to geometry	slower than closed-form solutions
General with respect to objective function	slower than numerical methods which incorporate derivative information (except at, or near, singularities)
can be automatically formulated	
converges at singularities	
straightforward implementation	

A general approach to the inverse kinematics problem is essential to the application of a modular robot architecture. The generality of direct search inverse kinematics also allows its application with robots having complex geometries. This generality gives direct search inverse kinematics a distinct advantage over closed-form techniques in these application areas. However, other numerical methods of solving the inverse kinematics problem may also be formulated in a general fashion. For instance, an iterative method based on inverting the Jacobian is described in the literature as part of a reconfigurable and modular manipulator system (Kelmar and Khosla, 1990).

Direct search's ability to converge at exact singularities is unique among inverse kinematics methods, closed-form or numerical. Because all robots have singularities, this is an important recommendation for the continued development of direct search inverse kinematics. Direct search converges at singularities

because generating trial solutions and making decisions does not require any scalar or matrix inversion. The Newton-Raphson, predictor-corrector, or any method requiring derivatives, not only can't converge exactly to singularities, but also become ill-behaved near singularities. The difficulties associated with singularities is also the reason higher order geometric properties, such as those formulated by Thomas and Tesar (1982) for general serial robots, were not included as a basis for decision making in the strategies this chapter develops.

Chapter 4: Multicriteria Inverse Kinematics

This chapter defines multicriteria inverse kinematics as a decision making problem with an unlimited number of performance criteria, inequality constraints, and equality constraints. The chapter discusses two different approaches for making decisions and develops direct search as a method for solving the multicriteria inverse kinematics problem. Sequential filters and factorial design are techniques for improving the performance of the direct search methods developed in Chapter 3 and extended here to include multiple performance criteria. Another direct search method, called simulated annealing, may find globally, rather than locally, optimal solutions. This chapter applies simulated annealing to the multicriteria inverse kinematics problem. Finally, the chapter concludes with a discussion of conservative paths.

Direct search is one way of solving the multicriteria inverse kinematics problem. There are two main factors recommending direct search for this application. The first is that direct search is completely stable in the presence of singularities. The second is that the decision making process can explicitly calculate and consider any number and variety of performance criteria when choosing from among the trial solutions.

The definition of the multicriteria inverse kinematics problem presented in the next section is both general and concise. Specifically, the definition does not constrain the definition of optimality. The definition of optimality is one of the fundamental considerations when working with multiple performance criteria.

How are the multiple criteria fused to choose just one optimum? This chapter discusses two definitions of optimality. The first defines an optimum as a non-dominated solution. This definition keeps the criteria separate from one another, thus essentially eliminating scaling difficulties. A minimum of a composite performance index is the other definition of optimality this chapter considers. A composite performance index is essentially a weighted sum of the criteria. The composite index may include any number of different performance criteria; but there are myriad scaling issues involved. For instance, a criterion relatively large in magnitude may artificially dominate the other criteria in the index. Properly choosing the scaling factors provides performance criteria of comparable magnitude. Also, the multiple criteria may create extraneous minima in the index. These extraneous minima do not represent solutions to the inverse kinematics problem, yet the direct search may converge to them rather than to true solutions. Judiciously manipulating the scaling factors will guide the search to proper solutions.

Sequential filters and factorial design are options available for improving the performance of the direct search. Sequential filters improve the solution speed and alleviate the problems associated with extraneous minima generated by the performance criteria. Sequential filters are compatible with both definitions of optimality. Factorial design is a mature and interesting field in its own right. Factorial design techniques can generate a multivariable polynomial approximating the combined effects of the performance criteria. Using the approximating polynomial during the search rather than the actual criteria greatly

increases the solution speed. Factorial design techniques are also compatible with both definitions of optimality.

Simulated annealing is another direct search method. As opposed to the systematic direct search methods discussed so far, simulated annealing derives much of its strength from randomness. The main benefit is that simulated annealing may find global, rather than local, minima. The main drawback is that simulated annealing is extremely demanding computationally. Nonetheless, simulated annealing shares most of the attributes making the systematic direct search approaches attractive. This chapter develops a simulated annealing method of solving the multicriteria inverse kinematics problem and discusses a hybrid algorithm combining simulated annealing with systematic direct search. This hybrid algorithm has a number of attractive attributes.

Finally, the chapter concludes with a discussion of conservative paths. A conservative path is closed in both joint and end-effector space. That is, both the end-effector and the joints repeatedly follow their same respective trajectories. Though not a part of the multicriteria inverse kinematics problem as defined in the next section, a conservative path is important in repetitive manufacturing and assembly applications.

4.1 PROBLEM DEFINITION

This is a general and concise definition of the multicriteria inverse kinematics problem:

find the optimal set of joint displacements

$$\Phi$$

given the performance criteria

$$P_j \left(\Phi, \frac{d\Phi}{dt}, \frac{d^2\Phi}{dt^2}, \dots, \frac{d^n\Phi}{dt^n} \right) \quad j = 1, 2, \dots, J \quad (4.1)$$

subject to the constraints

$$Q_k \left(\Phi, \frac{d\Phi}{dt}, \frac{d^2\Phi}{dt^2}, \dots, \frac{d^n\Phi}{dt^n} \right) = 0 \quad k = 1, 2, \dots, K \quad (4.2)$$

$$R_l \left(\Phi, \frac{d\Phi}{dt}, \frac{d^2\Phi}{dt^2}, \dots, \frac{d^n\Phi}{dt^n} \right) \geq 0 \quad l = 1, 2, \dots, L \quad (4.3)$$

This definition does not imply the method for fusing the multiple criteria to determine one set of optimal joint displacements. This chapter presents two methods for considering multiple performance criteria. One method defines an optimum as a non-dominated solution amongst the individual criteria. The other defines an optimum as a minimum of a single composite performance index, or a series of distinct performance indices as powerful indicators to the robot's operator. The problem definition also does not imply whether the optimum is of the local, global, or whole-path variety (though it may be further subdivided into these categories). This is a general problem statement and does not in any sense guarantee that a solution to a particular formulation exists.

Notice that the criteria and constraint equations must be functions of the joint displacements and/or their time derivatives only. This is because the joint displacements are the only independent variables manipulated during the solution of the inverse kinematics problem. There may be other useful measures of the robot's performance, such as minimum time or minimum energy, but the inverse

kinematics method cannot address them directly. These types of performance measures may be incorporated into the solution by including criteria that have a physical interpretation relating to speed or energy. It is essential to understand the distinction between the performance criteria and their physical interpretation. The operator of the robot uses the physical interpretation to choose performance criteria important for a given task. The inverse kinematics method uses the performance criteria themselves during the course of the solution. Including performance criteria that are functions of the time derivatives of the joint displacements only makes sense when solving the inverse kinematics problem for a path, rather than for a single point.

The performance criteria are employed when choosing an optimum from among the set of feasible solutions. As discussed, the only requirement is that the performance criteria must be single-valued functions of the joint displacements and their derivatives with respect to time. There are, however, many desirable properties for meaningful criteria. Ranking the performance criteria according to priority is also possible.

Table 4.1 lists some desirable properties for performance criteria. The criteria should be independent mathematically. Dependent criteria lead to “double counting” of their shared effects. Independent performance criteria may give conflicting information as to the optimum solution. This is simply a reality of the multicriteria inverse kinematics problem. The criteria should exhibit sensitivity to changes in the joint displacements. In other words, the criteria should vary over the robot’s workspace. Criteria with numerical values that do not change significantly over the robot’s workspace are of little use in decision making. The

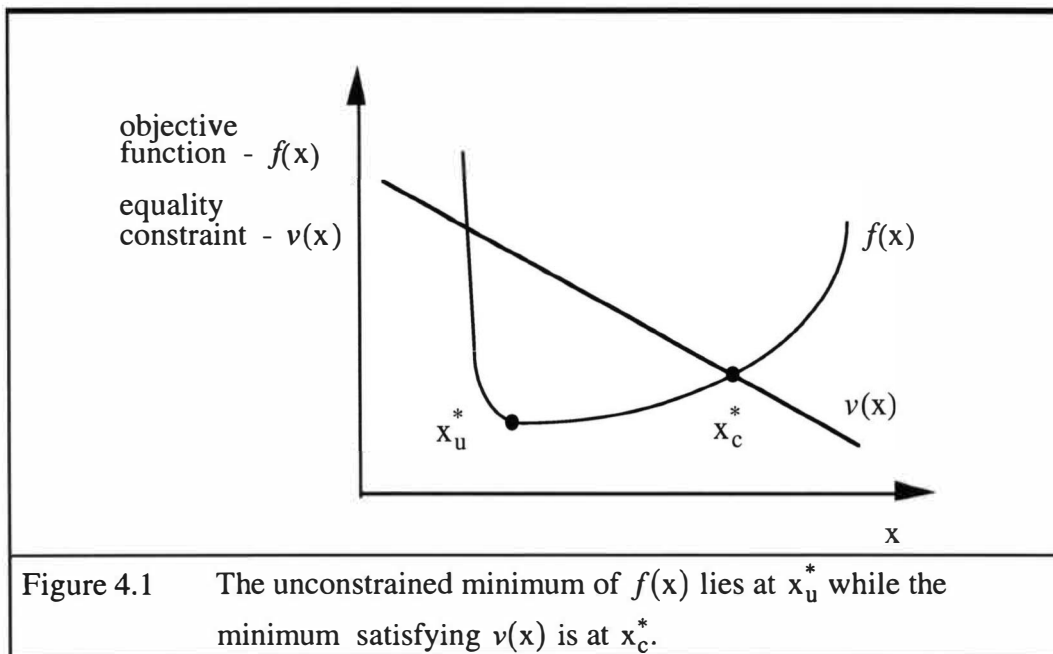
criteria must be single valued functions. This allows a deterministic solution of the decision making problem. A deterministic process will always choose the same solution when presented with the same options. Also, the criteria must be single valued so that the decision making process can rank the possible solutions. The criteria should be continuous functions to prevent a criteria from artificially dominating the solution in the area of the discontinuity. The criteria should be formulated in as simple a fashion as possible. This decreases the computational demand associated with calculating the criteria values. The criteria should have multiple physical meanings. This allows a single criterion to be a powerful indicator of the robot's performance. The criteria should be bounded in magnitude to allow scaling and thus provide a balanced basis for decision making. Finally, the criteria should be task independent. Task independent criteria do not need to be reformulated or scaled for each individual task the robot performs.

Table 4.1 Some desirable properties for the performance criteria and the rationale behind their desirability.

Desirable Properties	Rationale
mathematical independence	prevents "double counting" of shared effects
vary over workspace	allows the criteria's use as a basis for decisions
single valued	allows deterministic solutions
continuous	prevents the criteria from dominating the solution in the area of the discontinuity
computationally efficient	increases the computational speed
multiple physical meanings	a single criterion will be a powerful basis for decisions
bounded in magnitude	so the criteria may be scaled
task independent	so the criteria need only be formulated once

The equality constraints represent the heart of the inverse kinematics problem. The fundamental equality constraints are the three position and the three orientation constraints on the placement of the end-effector. Solving for the set of joint displacements given these six end-effector constraints is the inverse kinematics problem. If these position and orientation constraints on the end-effector are not met, be assured that a solution has not been found. There may also be equality constraints on the higher order properties at the end-effector. An equality constraint on the velocity of the end-effector is important in many painting and welding applications. Other equality constraints might be placed on the forces at the end-effector. As with the performance criteria, the equality constraints must be single-valued functions of the joint displacements and/or the derivatives of the joint displacements with respect to time. It is certainly possible to over-constrain a problem and thus preclude the existence of a solution. For instance, specifying three position and three orientation constraints on a robot with only five degrees of freedom will very likely preclude the existence of a solution.

As an example of the difference between the constrained and the unconstrained optimization problem, consider Figure 4.1 which shows a simple objective function $f(x)$ and a simple equality constraint $v(x)$. The values of $f(x)$ and $v(x)$ are plotted on the vertical axis versus the value of x on the horizontal axis. In the unconstrained case, the minimum of the objective function is at x_u^* . The equality constraint, however, requires that the solution lie on $v(x)$.



The constrained minimum is therefore at x_c^* . The multicriteria inverse kinematics problem always includes equality constraints. The position and orientation equality constraints on the end-effector are the fundamental part of the problem. A solution to the multicriteria inverse kinematics problem must therefore lie on an intersection of these equality constraints. If there is no set of joint displacements where all of the equality constraints intersect, then there is no feasible solution to this particular formulation of the problem. There is also no guarantee, indeed it is quite unlikely, that the intersection of the equality constraints coincides with an optimum of the performance criteria.

Inequality constraints typically become important during the operation of an actual robot. Though these constraints are also criteria in the sense that they relate to the performance of the robot, they are fundamentally different than the

performance criteria discussed in the literature review. The performance criteria are used to enhance the performance of the robot. The constraint-based criteria define the range of acceptable robot performance. The physical travel limits on the joint displacements (joint limits) are very real inequality constraints. Attempting to violate the joint limits will certainly cause a deviation from the desired path and could possibly damage the robot. All robot actuators have a finite maximum speed which represents another inequality constraint. The commanded joint speed must be less than or equal to the maximum speed of the actuator. Actuator acceleration, torque, and force limits are all examples of inequality constraints. Obstacle avoidance issues may also be formulated as inequality constraints. Basically, the constraint is formulated so as not to allow the manipulator or its payload to enter a volume which contains an obstacle. As with the equality constraints, the inequality constraints must be single-valued functions of the joint displacements and/or the derivatives of the joint displacements with respect to time. Table 4.2 summarizes these constraint criteria.

Table 4.2 Some inequality constraints and their physical basis.	
Inequality Constraint	Physical Basis
joint travel limits	finite joint displacement capabilities due to mechanical design considerations
joint speed limits	finite speed capabilities of the robot's actuators
joint torque/force limits	finite torque/force capabilities of the robot's actuators
joint acceleration limits	limited frequency response of the robot's actuators

The constraint criteria may also be formulated as simple performance criteria. These criteria do not actually enforce the constraints, but instead encourage the inverse kinematics solution to meet the constraints. For instance, the two-norm of joint displacements is related to the joint limit constraints. The two-norm of joint speeds is related to the maximum and minimum joint speed constraints. These criteria have extremely simple physical interpretations, yet they are also very important performance criteria. Table 4.3 lists some of these simple criteria and the constraints from which they were derived.

Table 4.3 Simple performance criteria and their corresponding constraints.	
Simple Criteria	Corresponding Constraints
two-norm of joint displacements	joint travel limits
two-norm of joint speeds	joint speed limits
two-norm of joint torques/forces	joint torque/force limits
two-norm of time rate of change of joint speeds	joint acceleration limits

The multicriteria inverse kinematics problem is now defined in a general fashion which allows for the inclusion of an unlimited number of performance criteria, equality constraints, and inequality constraints. The problem definition does not imply any specific method for fusing the performance criteria and choosing an optimum from among the feasible solutions defined by the constraint set. The only constraint on the performance criteria, equality constraints, and inequality constraints is that they must be single-valued functions of the joint displacements and/or the derivatives of the joint displacements with respect to time.

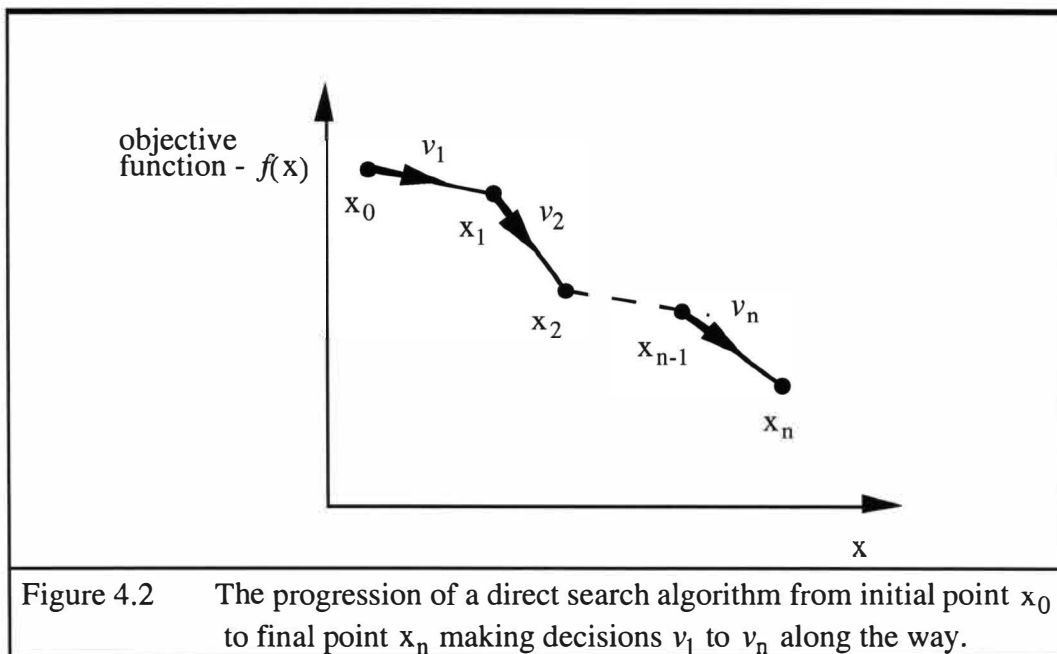
4.2 DIRECT SEARCH

The direct search method described in Chapter 3 for solving the inverse kinematics problem for fully-constrained robots may also be applied to the multicriteria inverse kinematics problem. The direct search is a local, as opposed to a global, method of solving the multicriteria inverse kinematics problem. Repeatedly applying the direct search method generates a whole-path solution, however there is no guarantee that the whole-path solution is optimal. Nonetheless, there are many factors recommending direct search for solving the multicriteria inverse kinematics problem. The most pervasive of the factors is that each trial solution completely determines the joint-level state of the robot. This allows all performance criteria, equality constraints, and inequality constraints to be explicitly calculated and used as a basis for decision making.

Recall that the direct search inverse kinematics method begins at an estimate of the solution. This estimate serves as an initial base point for the search. From this base point, the search conducts a series of exploratory moves by perturbing the joint variables. Each set of perturbations represents a direction for the search. The exhaustive exploration pattern generates all of the possible directions and was shown to be the only optimal strategy. Each set of directions represents one possible choice for the decision making part of the direct search. The performance criteria, along with the constraint set, are used as a basis for choosing from among the possible directions. The elegance of the direct search method, and the characteristic allowing its application to the generalized inverse kinematics problem, is that for each set of joint perturbations all of the

performance criteria and constraints can be calculated directly and explicitly. There is no need for any matrix inversion, algebraic manipulation, or geometric simplification. Once a choice is made as to the optimum direction, a move is made and a new base point set. From this new base point, the exploration and decision making process repeats and the search, when it is successful, proceeds to a solution.

Each of the direct search strategies discussed thus far are deterministic. That is, the current base point and the decision making strategy completely determine the next base point. This is why the performance criteria and the constraints must be single-valued functions. As an example of a direct search, consider Figure 4.2.



The beginning base point is labeled x_0 and the final base point is labeled x_n . The first decision is labeled v_1 and the final decision is labeled v_n . Thus, the search begins at point x_0 and after decision v_1 the new base point is x_1 . This procedure continues until reaching base point x_n after making decision v_n from base point x_{n-1} . In this example the search terminates at x_n . It is possible that different decisions might have resulted in the same final base point. It is also possible that a different initial base point and the same decision making strategy might reach the same final point,. In any case, each new base point is a function of the previous base point and the decision making strategy. Thus, in general, the sequence of base points may be written as

$$\begin{aligned} x_1 &= f(x_0, v_1) \\ x_2 &= f(x_1, v_2) \\ x_3 &= f(x_2, v_3) \\ &\vdots \\ x_n &= f(x_{n-1}, v_n) \end{aligned} \tag{4.4}$$

Inspection of these equations may seem to indicate that the final base point is a function of the initial base point, x_0 , all of the intermediate base points - $x_1, \dots, x_{n-1}$ - and all of the decisions - $v_1, v_2, \dots, v_n$. However, each of the intermediate base points is a function of the previous base point and decision. Therefore, the final point, x_n , is a function only of the initial base point, x_0 , and the decisions, $v_1, v_2, \dots, v_n$.

$$x_n = f(x_0, v_1, v_2, \dots, v_n) \tag{4.5}$$

Equation (4.5) illustrates that x_0 is the only base point chosen independently of the search and suggests that finding a good starting position may be well worth the effort. One method of finding a starting point, called simulated

annealing, is discussed in detail later in this chapter. Another method, known as the “shotgun approach” simply starts the search from several different places and uses whichever result is best.

Certain inequality constraints may explicitly eliminate trial solutions during the decision making process. For instance, including joint limit constraints in the search is particularly straight-forward. Each perturbation of a joint represents a particular joint displacement. If the joint displacement violates a joint limit, then that trial solution is removed from the possible solution set. The condition for discarding the trial solution is simply

$$\Phi_{\min} > \Phi_{\text{trial}} > \Phi_{\max} \quad (4.6)$$

Essentially, there is one less choice available to the decision making strategy. Joint speed limits may be included in a similar fashion. As the direct search proceeds, the joints are moved away from their position at the initial base point. If the change in displacement becomes too great, a joint speed limit is violated and the trial solution withdrawn from the choices available to the decision making strategy. The condition for a joint speed violation is

$$\Delta\Phi_{\min} > \Delta\Phi_{\text{trial}} > \Delta\Phi_{\max} \quad (4.7)$$

Other inequality constraints, such as those at the acceleration level, are not typically satisfied at the initial base point. Therefore these constraints must be included in the decision making strategy.

Certain fault-tolerance issues may also be addressed while generating trial solutions. Specifically, a fault causing a joint to be locked at a known displacement may be handled at this level. No trial solutions corresponding to

perturbations of the locked joint are generated or added to the alternatives available as the next base point. Locking joints may decrease the dexterity of the robot to the extent that the direct search will no longer find a solution.

The direct search method of solving the multicriteria inverse kinematics problem is particularly well-suited for addressing the obstacle avoidance issue. This is because the direct search is based on the forward kinematic equations for the robot. A natural by-product of using the forward kinematic equations is that the position of all local reference frames along the manipulator are calculated and are available at no additional computational burden. If the direct search is generating a path for the manipulator, the velocities and accelerations of the local frames may also be calculated at very little computational burden. This information may then be formulated as a performance criterion and used in the decision making process.

The obstacle avoidance issue may also be addressed while generating the trial solutions in a manner similar to the joint limit constraints. As each trial solution is generated, the positions of each of the local frames along the manipulator are checked to see if this trial solution results in a collision. If it does, the trial solution is rejected. This method essentially reduces the obstacle avoidance problem to a collision detection problem.

4.3 NON-DOMINATED SOLUTIONS

The process of perturbing the joint variables and generating trial solutions presents the decision making process with a set of choices. Of the entire set of trial solutions, one must be chosen as the next base point so that the direct search

may proceed. The difficulty lies in choosing the next base point when confronted with multiple, and possibly conflicting, performance criteria. The concept of a non-dominated solution is one way of making this choice. Essentially, a non-dominated solution makes at least one criterion better while making no other criteria worse. The usefulness of the non-dominated solution is that scaling difficulties are essentially eliminated. There is no explicit summing of effects, as there is in a composite performance index, and there is no mixing of units. The search for a non-dominated solution allows for the ranking of performance criteria according to their relative importance. The operator of the robot will rank the criteria according to the task at hand. This ranking is employed when choosing from among a group of non-dominated solutions or as a recourse when there are no non-dominated trial solutions. Though the non-dominated solution does keep the robot's operator "in the loop" by allowing the ranking of the criteria, the actual decision making process is based on a very simple rule. Thus, the non-dominated solution is a rather weak method of criteria based decision making.

For the multicriteria inverse kinematics problem, a formal definition of a non-dominated solution is:

given the solution sets

$$\Phi_1 \text{ and } \Phi_2$$

and the performance criteria

$$P_j \left(\Phi, \frac{d\Phi}{dt}, \frac{d^2\Phi}{dt^2}, \dots, \frac{d^n\Phi}{dt^n} \right)$$

solution set Φ_1 is said to dominate Φ_2 iff

$$P_j\left(\Phi_1, \frac{d\Phi_1}{dt}, \frac{d^2\Phi_1}{dt^2}, \dots, \frac{d^n\Phi_1}{dt^n}\right) \leq P_j\left(\Phi_2, \frac{d\Phi_2}{dt}, \frac{d^2\Phi_2}{dt^2}, \dots, \frac{d^n\Phi_2}{dt^n}\right) \quad (4.8)$$

for $j = 1, 2, \dots, J$ and with the strict inequality holding for at least one P_j .

The alleviation of difficulties associated with scaling different criteria is the primary motivation for using dominance to define optimality. In the search for a non-dominated solution, different criteria are always kept independent from one another. Using dominance to define the optimum also provides unit invariance, which some researchers feel is very important. Unit invariance means that the solution won't change just because the unit of measure for a particular criterion, for instance feet versus meters, changes. A non-dominated solution remains non-dominated regardless of the units or scale of any particular criterion. This is quite easy to show using Equation (4.8).

$$P_j\left(\Phi_1, \frac{d\Phi_1}{dt}, \frac{d^2\Phi_1}{dt^2}, \dots, \frac{d^n\Phi_1}{dt^n}\right) \leq P_j\left(\Phi_2, \frac{d\Phi_2}{dt}, \frac{d^2\Phi_2}{dt^2}, \dots, \frac{d^n\Phi_2}{dt^n}\right) \quad (4.8)$$

If each performance criterion is multiplied by it's own scaling factor, s_j , this equation becomes:

$$s_j P_j\left(\Phi_1, \frac{d\Phi_1}{dt}, \frac{d^2\Phi_1}{dt^2}, \dots, \frac{d^n\Phi_1}{dt^n}\right) \leq s_j P_j\left(\Phi_2, \frac{d\Phi_2}{dt}, \frac{d^2\Phi_2}{dt^2}, \dots, \frac{d^n\Phi_2}{dt^n}\right) \quad (4.9)$$

The scaling factor, s_j , has the same effect on both sides of Equation (4.9). The inequality is therefore unaffected.

Using a non-dominated solution as the definition of optimality allows the ranking of the performance criteria according to their relative importance. The ranking is employed when choosing from among non-dominated solutions (if there is more than one non-dominated solution). The non-dominated solution

improving the highest ranking criterion the most is chosen. The criteria generated by transforming the position and orientation constraints on the end-effector naturally receive the highest rank, since failing to meet these constraints is failing to solve the inverse kinematics problem.

The perturbation process generates a set of trial solutions from which a non-dominated solution must be found. The mechanics of identifying the non-dominated solutions are not difficult. For each trial solution, Φ_{trial} , the corresponding value for each performance criteria, P_j , is calculated. The trial solutions are then searched to find the nondominated solution that improves the highest ranking performance criterion the most. That solution is then chosen as the next base point for the direct search. There is some computational burden associated with searching the trial solutions after calculating the criteria values. This sort of list-searching is, however, very common in computer science and efficient algorithms are available (Wiederhold, 1983).

An important situation arises when no non-dominated trial solutions are generated during the perturbation process. This is very likely if many performance criteria are considered. If no non-dominated solution is found, the lowest ranked criterion is removed from consideration and the trial solutions are again searched for a non-dominated solution. If none is found, the next lowest ranked criterion is removed from consideration and the process repeated. Eventually, either a non-dominated solution is found, or all but the equality constraints are removed from consideration. If no trial solution is non-dominated among the equality constraints, the search terminates. A successful search terminates at a solution placing the end-effector in the desired position and

orientation. Whether or not the search was successful is easily checked using the final solution and the forward kinematic equations for the manipulator. Fortunately, there are no singularities or numerical instabilities associated with an unsuccessful search. Unfortunately, it is difficult to tell why a particular search is unsuccessful. The only information when the search terminates is that there are no non-dominated solutions among the set of trial solutions. Either the specified end-effector placement is not realizable by this robot, or the algorithm failed to find a solution, though one existed.

4.4 COMPOSITE PERFORMANCE INDEX

Basing decisions on a composite performance index is one option when confronted with multiple performance criteria. The composite performance index is essentially a weighted sum of the criteria. Using PI to denote the composite performance index, P the performance criteria, and s the weightings (or scale factors) the composite index is written as

$$PI = s_1P_1 + s_2P_2 + \dots + s_iP_i \quad i = 1, 2, \dots, I \quad (4.10)$$

Essentially, the composite performance index measures the average effects of the performance criteria. Clearly, questions of scale are fundamental when formulating a composite performance index. A criterion much larger than the others will artificially dominate the index. The performance criteria, equality constraints, and inequality constraints might all appear in the composite performance index.

Before being expressed as part of the composite performance index, constraints must be transformed into criteria. In a direct search solution, the

equality constraints on the displacement of the end-effector are transformed into criteria via a Cartesian error function. Section 3.2.1 developed a method for transforming equality constraints via an error function. Direct search also addresses some of the inequality constraints, namely joint travel and speed limits, while generating the trial solutions. These inequality constraints will not appear in the composite performance index. Other inequality constraints, such as joint acceleration and torque or force limits, must be transformed and addressed at the decision making stage. The procedure for transforming an inequality constraint is very similar to that for transforming an equality constraint. Note the problem definition expresses inequality constraints as

$$R_l\left(\Phi, \frac{d\Phi}{dt}, \frac{d^2\Phi}{dt^2}, \dots, \frac{d^n\Phi}{dt^n}\right) \geq 0 \quad l = 1, 2, \dots, L \quad (4.3)$$

The actual physical constraint, maximum joint-level acceleration for example, is written as

$$\ddot{\Phi}_{\text{actual}} \leq \ddot{\Phi}_{\text{max}} \quad (4.11)$$

which may be easily rewritten as

$$\ddot{\Phi}_{\text{max}} - \ddot{\Phi}_{\text{actual}} \geq 0 \quad (4.12)$$

A bracket operator is employed to transform the inequality constraint and express it as a criteria. The bracket operator is defined as

$$\langle \alpha \rangle = \begin{cases} \alpha & \text{if } \alpha \leq 0 \\ 0 & \text{if } \alpha > 0 \end{cases} \quad (4.13)$$

Using the bracket operator, the inequality constraint is expressed as a performance criterion.

$$P = |\langle R \rangle| \quad (4.14)$$

The form of this equation ensures that solutions not violating the inequality constraint are not penalized. Solutions violating the constraint are assigned a criterion value equal to the magnitude of the violation. There are also other methods for transforming inequality constraints. Reklaitis et al. (1983) conduct a thorough discussion.

The motivation for transforming the constrained problem into an unconstrained problem has both classical and practical roots. In the classical sense, LaGrange multipliers are another tool for forming an unconstrained problem from a constrained problem. In the practical sense, Section 3.2.4 showed that the direct search method will find a local minimum of an unconstrained objective function. The direct search method will therefore find a local minimum of the composite performance index. For this local minimum to be meaningful, the performance criteria in the index must be properly scaled.

In general, the performance criteria are of different dimension and magnitude. For example, summing the effect of a performance criterion measured in meters with that of a criterion measured in radians requires a scaling factor. The scaling factor prevents any criterion from artificially dominating the composite index and masking the effects of the other criteria. Without question, the choice of scaling factors affects the final solution to the multicriteria inverse kinematics problem. A systematic approach to choosing these factors makes the composite performance index a useful basis for decisions.

The literature review described a global method of scaling the Jacobian matrix. Essentially, the method cycles the simulated robot through its workspace and calculates a moment arm at many regularly spaced discrete points. A single moment arm is derived from these calculations and used as a basis for comparing rotational and translational quantities in the Jacobian matrix. While discussing this technique, Bevill and Tesar (1990) develop two approaches for deciding upon the moment arm. The first is based upon the extended length of the robot. This corresponds to the maximum moment arm found in the workspace. The second is based on a moment arm averaged over the workspace. Bevill and Tesar note that choosing a scaling factor based on the maximum moment arm artificially masks the effects of the translational outputs relative to the rotational outputs. They thus concluded that an average value represented a superior basis for scaling.

This global method can also be used for scaling many of the performance criteria; specifically the task independent criteria based solely on the geometry of the manipulator. A simulated robot is cycled through its workspace and the values corresponding to each performance criterion are calculated at many discrete points. Following the same line of reasoning as Bevill and Tesar, the average values of the performance criteria then form the basis for the scaling factors. Specifically, the inverse of the average values.

$$s_i = P_{i \text{ ave}}^{-1} \quad i = 1, 2, \dots, I \quad (4.15)$$

The ave subscript indicates the average over the workspace. Equation (4.15) evokes a possible problem with this method of scaling, which occurs if the average value of the criterion is zero. If this occurs, Equation (4.15) is undefined. This condition can only occur in two situations. The first is that the performance

criterion value is always zero. The second is if the criterion assumes positive and negative values and the average happens to be zero. To allay this problem, it is best if the performance criteria values are always greater than or equal to zero. This is easily accomplished as part of the scaling process. While cycling the manipulator through the workspace, simply maintain the minimum criteria values. The performance criteria are then defined as

$$P_i = P_i - \langle P_{i \min} \rangle \quad i = 1, 2, \dots, I \quad (4.16)$$

The subscript min describes the minimum over the workspace and the bracket operator is described by Equation (4.13). $\langle P_{i \min} \rangle$ represents a constant offset ensuring that the criteria values are always greater than or equal to zero. After these procedures, the scaling factors are

$$s_i = \begin{cases} P_i^{-1} & \text{if } P_i > 0 \\ 0 & \text{if } P_i = 0 \end{cases} \quad i = 1, 2, \dots, I \quad (4.17)$$

Equations (4.16) and (4.17) guarantee positive finite values for the scaling factors.

This global scaling method has several advantages. It only needs to be performed once for any given robot and criteria set, and this can be done before the inverse kinematics algorithm begins. The global basis provides scaling factors ensuring the contributions from each of the performance criteria are of similar magnitude over the entire workspace. Finally, an average value does not mask the effects of criteria with relatively high peak values.

Unfortunately, the method of cycling the robot through the workspace and calculating criteria values at independent and discrete points is not valid for all of the performance criteria. Only criteria which are functions of the joint displacements, but not functions of the time derivatives of the joint displacements

may be scaled in this manner. Cleary and Tesar (1990) call these criteria task-independent and they are invaluable for evaluating the performance of a manipulator. The multicriteria inverse kinematics problem as defined in Equations (4.1) to (4.3), however, demands that task-dependent criteria be addressed as well.

The direct search method provides an opportunity to locally scale performance criteria which cannot be scaled globally. At each new base point in the search, a set of trial solutions is generated via a series of perturbations. The next base point must be chosen from this set of trial solutions. The criteria values associated with the trial solutions are very much like those generated by cycling the robot through its workspace in the global scaling method. The difference is that the trial solutions may be referenced to the original base point of the search. Because the trial solutions may be referenced to the state of the robot at a point, criteria values which are functions of the time derivatives of the joint displacements may be calculated as well. In this search-based method, either a maximum or an average value can be used as a basis for calculating a scaling factor.

The local scaling method provides an additional motivation for preferring a scaling factor based on an average, rather than maximum, value. A scaling factor based on a maximum may mask the effects of rapidly changing criteria. However, the rapidly changing performance criteria are of the most use for differentiating between the trial solutions.

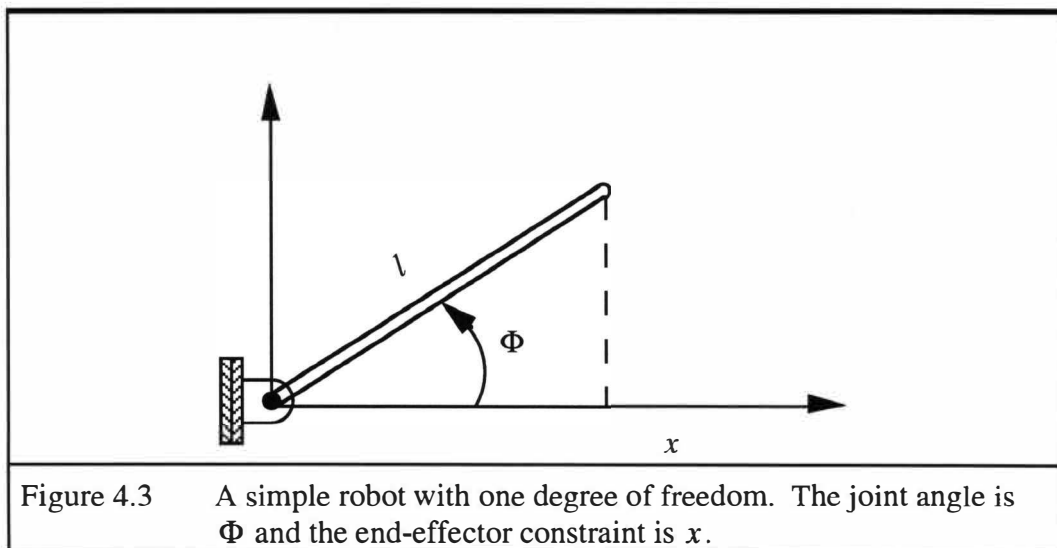
Calculating a scaling factor based on a local average represents very little computational overhead. As the trial solutions are generated and associated

criteria values calculated, running sums are maintained for each criterion. Once the criteria values are all calculated, each sum is simply divided by the total number of trial solutions. This provides an average criterion value from which to derive the scale factor. Generating the trial solutions and calculating the criteria values are operations that must be performed as part of the direct search no matter what the scaling method. Maintaining a running sum of the values represents very little additional overhead.

A systematic method for scaling any performance criterion that is a function of the joint displacements and/or the time derivatives of the joint displacements has now been developed. This scaling method assures no performance criterion artificially dominates a composite performance index written in the form of Equation (4.10). After scaling, direct search will find a local minimum of the composite performance index.

After finding a local minimum of the composite performance index, it still must be determined if this local minimum represents a solution to the multicriteria inverse kinematics problem. Evaluating the constraints determines whether or not a solution was found. Specifically, the end-effector must be in the proper position and orientation, as determined using the forward kinematic transformation equations. If all the constraints are met, the successful search terminates. It is entirely possible, however, that the local minimum of the composite performance index does not represent a solution to the problem. The scaling method was developed to ensure that all of the performance criteria were given similar representation in the composite performance index.

Satisfying the constraint equations is of primary importance. If a search terminates without satisfying the constraints, it has been unsuccessful. After an unsuccessful search, the scale factors of unmet constraints may be increased and the search continued or repeated. For solving other optimization problems with transformed constraint equations, some researchers recommend increasing the scale factors of unmet constraints by powers of ten after each unsuccessful search (Reklaitis et al., 1983). Experience with direct search shows this to be a reasonable strategy.



This example problem, Example 4.1, gives a physical impression of the effects of varying the scale factors. Essentially, varying the scale factors changes the shape of the composite performance index. Figure 4.3 shows a simple serial robot with one revolute joint and one link. The displacement along the x axis is

the constraint and the joint angle Φ is the independent variable. Using simple geometry, an equality constraint is written as

$$l \cos \Phi - x = 0 \quad (4.18)$$

Transforming this constraint via an error function yields a performance criterion of the form

$$P_1 = |l \cos \Phi - x| \quad (4.19)$$

Though Figure 4.3 does not represent a redundant robot, another performance criterion is defined for the sake of example. This performance criterion is based on the goal of minimizing the joint displacement and may be written as

$$P_2 = |\Phi| \quad (4.20)$$

Writing a composite performance index in the form of Equation (4.10) yields

$$PI = s_1 P_1 + s_2 P_2 \quad (4.21)$$

Examining these three equations reveals that there are more performance criteria than independent variables. The statement of the multicriteria inverse kinematics problem given in Equations (4.1) to (4.3) does not preclude this situation and it must be addressed. Numerical values allow the solution of this example problem using the methods developed in this chapter. Let:

$$\begin{aligned} l &= 100 \text{ meters} \\ x &= 25 \text{ meters} \\ \Phi_{\min} &= -\pi / 2 \text{ radians} \\ \Phi_{\max} &= \pi / 2 \text{ radians} \end{aligned}$$

The lengths, l and x , are large numbers relative to the joint displacement criterion. This will show the effects of the scale factors, s_1 and s_2 , which may be

found by the global scaling method. Using $\Phi_{\min}$ and $\Phi_{\max}$ to define the workspace, the global scaling method gives

$$P_{1 \text{ ave}} = 42.66 \text{ meters}$$

$$P_{2 \text{ ave}} = .8225 \text{ radians}$$

and

$$s_1 = P_{1 \text{ ave}}^{-1} = .02344 \text{ meters}^{-1}$$

$$s_2 = P_{2 \text{ ave}}^{-1} = 1.216 \text{ radians}^{-1}$$

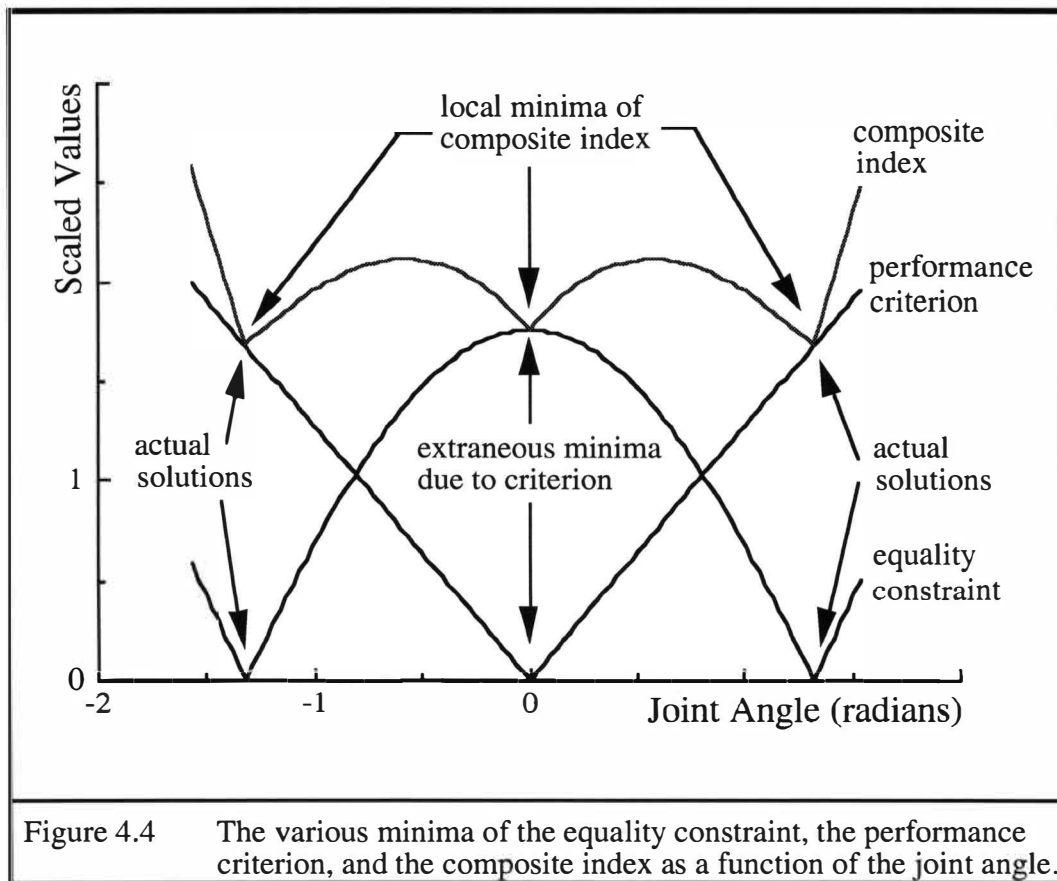
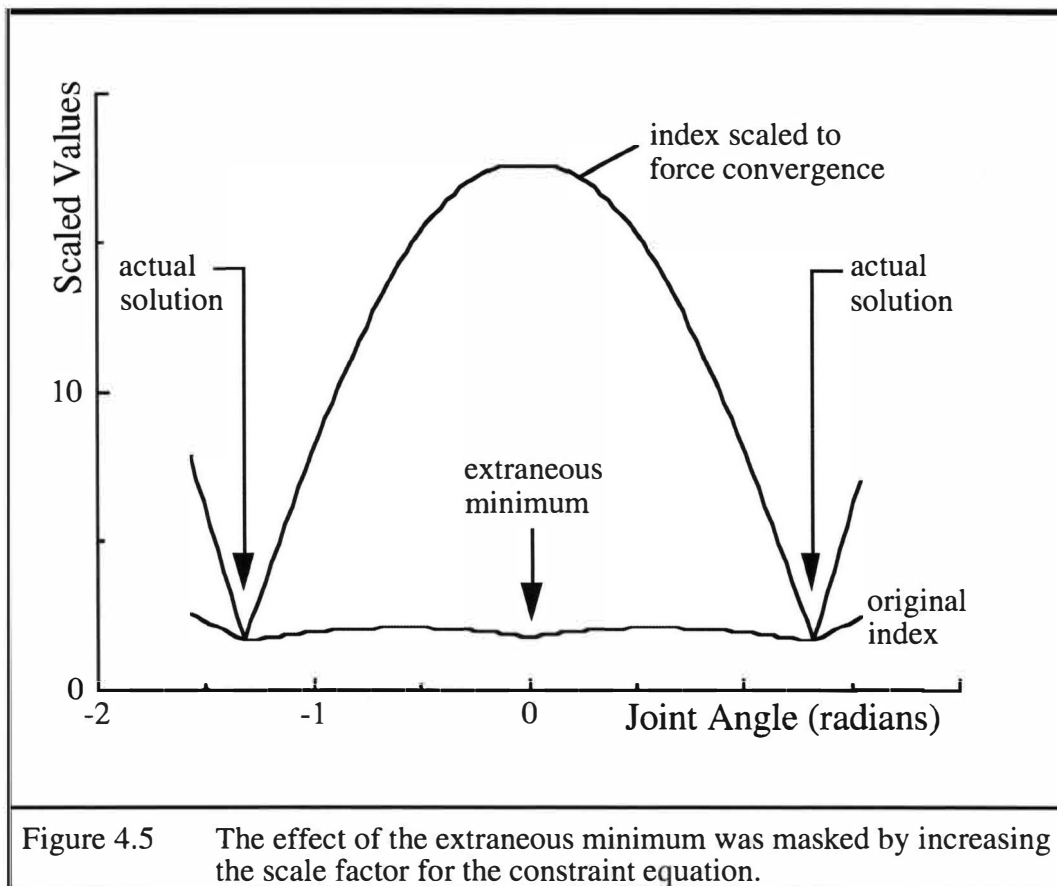


Figure 4.4 shows the values of the two scaled performance criteria and the composite performance index graphed versus the joint displacement, Φ , over the

workspace of this simple manipulator. This figure shows the scaling procedure was successful in producing performance criteria of a similar magnitude. The graph of $s_1 P_1$ shows minima at $\pm \cos^{-1} \frac{x}{l}$ (± 1.318 radians), which represent the solutions to the original problem. The composite performance index, however, has three local minima. The extra minimum is due to the effect of the joint displacement criterion. Whether or not the direct search finds one of the minima corresponding to the correct solution depends entirely on the starting place for the search. If the search starts between the two local maxima of the performance index ($\pm \sin^{-1} \left(\frac{s_2}{s_1 l} \right)$, which may be found by differentiating the composite index with respect to Φ and setting the result equal to zero), it will converge to the minimum at Φ equals zero. This minimum does not represent a solution to the problem.



As discussed, the procedure for escaping from this type of local minimum increases the scale factor, s_1 , for the transformed constraint equation by a power of ten. Multiplying s_1 by ten changes the shape of the composite performance index as shown in Figure 4.5. Note the change in scale did not change the location of the minima. The change in shape has, however, made it far more likely the direct search will converge to one of the two minima representing actual solutions to the problem. Increasing s_1 by another power of ten would essentially eliminate the effects of the joint-displacement criterion and its extraneous minimum.

This example problem showed the difficulties associated with haphazardly adding performance criteria to a composite performance index. Adding performance criteria when the manipulator is not redundant only complicates the solution of the problem. In this example, only by drastically changing the scaling factors would a direct search have a high likelihood of converging to a solution. There is, however, a method of giving priority to the primary goal of finding a solution, while considering secondary performance criteria whenever possible. This is the method of sequential filters, which will now be discussed.

4.5 SEQUENTIAL FILTERS

As originally applied to the mechanism synthesis problem, the method of sequential filters employed a primary criterion, all linkages satisfying the motion through four points, for generating as many as 60,000 different linkages (Eschenbach and Tesar, 1969). This large solution space was then sorted into zones and given a design criterion number to reduce the solution space and thereby provide the designer with a more manageable number of ranked choices. The design zones were classified as either necessary or desirable.

The sequential filters method is a powerful and inherently interactive design and optimization tool. The method begins with the operator selecting and ranking criteria important to a given problem. The operator should have as many criteria as possible to choose from. As discussed, these criteria may include constraints and performance criteria. At this point the operator may also impose acceptable bounds for the performance criteria. This process of choosing acceptable bounds is based on the idea that in a complex design process it is more

important to be sure that many criteria are within acceptable bounds than it is to optimize a single criterion. Once the criteria are selected and ranked, a synthesis process generates a large set of possible solutions. In Eschenbach and Tesar's application (1969), the synthesis process eliminated up to fifty percent of the free design parameters and reduced the solution space from ∞^2 to approximately 60,000 possible solutions. Though not actually a part of the sequential filters method, this synthesis step should precede the application of sequential filters whenever possible.

After generating the solution space, the filtering sequence begins. The filters are based on the constraints and the criteria bounds. These filters will typically decrease the solution space by a very large factor. Eschenbach and Tesar (1969) report reducing the solution space from 60,000 to approximately fifty with constraints and criteria bounds. These filters are applied sequentially according to their computational complexity, with the least computationally demanding filters applied first. Thus, the relatively complex filters are applied to a smaller and smaller solution space. The remaining solutions are ranked according to their criteria values and presented to the operator for the final selection of one solution. If no solutions remain after the filtering process, the operator is informed of the results and asked to relax some of the constraints and/or criteria bounds. The steps involved in the sequential filters process may be summarized as:

1. Present the operator with as many criteria as possible
2. The operator chooses, ranks, and imposes acceptable bounds on each of the criteria

3. Synthesis (or its equivalent) is used to generate a set of possible solutions
4. Filters based on physical constraints and criteria bounds (as set by the operator) are applied sequentially with the least computationally demanding filters applied first. The filtering process reduces the solution space.
5. Rank the remaining solutions according to their criteria values
6. Present the operator with results.

In the original application of sequential filters to the coplanar synthesis problem, the design zones were classified as either necessary or desirable. (Eschenbach and Tesar, 1969). An analogous situation exists with the multicriteria inverse kinematics problem. Satisfying the constraint equations is necessary, while optimizing the effects of the multiple performance criteria is desirable. In the mechanism synthesis problem, the designer must choose one solution from among the many options. In the direct search method, the decision making strategy must choose the next base point from among the set of trial solutions. Combined with direct search, the sequential filters method is compatible with both the non-dominated solution and the composite performance index. In both cases, the sequential filters method increases the solution speed, resolves premature termination of the algorithms, and guarantees the primary goal of satisfying the constraint equations is given priority during the course of the search.

Recall that a non-dominated solution improves at least one criterion while making no others worse. Rather than calculating all criteria values for all of the

trial solutions, a filter is employed for discarding some solutions when it becomes apparent they are not non-dominated. For instance, a reasonable first filter might discard all trial solutions which are not non-dominated among the constraint equations. This saves the time involved with calculating high-level performance criteria for many of the trial solutions.

Sequential filters may also resolve the premature termination of an algorithm searching for non-dominated solutions. For each base point in the search, there is a set of trial solutions from which the next base point must be chosen. The highest ranking non-dominated solution is clearly the choice for the next base point. A problem arises if there is no non-dominated solution among the set of trial solutions. At this point it should be apparent that using non-dominated solutions is actually a method of filtering. All but the non-dominated solutions are removed by the filter. Ranking the criteria creates a set of sequential filters. If no non-dominated solutions remain after filtering, the sequential filters must be relaxed. One method of relaxing the filters withdraws the performance criteria, one at a time. The process begins with the performance criteria of the least importance and proceeds towards the criteria of the most importance until finding a non-dominated solution. This procedure gives the most important performance criteria the highest priority. As discussed, the operator ranks the criteria according to their importance for a given problem.

Successively removing from consideration performance criteria of lesser importance works until reaching performance criteria relating to transformed constraint equations. Here, the current base point is optimal in the sense that it dominates, with respect to the constraint equations, all other possible solutions in

the locality. Hopefully, this local optimum represents a solution to the inverse kinematics problem, because there is very little recourse from here. An attempt to escape from the local minimum might be made, but it is not possible to adjust scale factors to force the solution towards unmet constraints. This is both the beauty and the bane of using non-dominated solutions to define an optimum. Because there is no scaling involved, an optimum is invariant with respect to units. Also because no scaling is involved, there is limited recourse when “stuck” in an extraneous local optimum.

The sequential filters method is also compatible with a composite performance index. Sequential filters increase the reliability and speed of the direct search inverse kinematics method and give the primary constraint equations priority during the course of the search

Dividing the composite performance index into two parts before applying the sequential filters method significantly increases the unmodified direct search method’s solution speed and reliability. This is a dual-class sequential filter. The dual-class sequential filter is based on the fundamental difference between performance criteria and physical constraints previously discussed. The transformed constraint equations comprise the first part of the composite index. This part is the constraint index. The second part is composed of the performance criteria and is called the performance index. An initial filter removes trial solutions that do not improve the constraint index. The criteria index is only calculated for trial solutions passing the first filter. This saves the time involved with calculating criteria values for solutions that do not pass the first filter. Additional performance increase is achieved because the search typically

proceeds more directly towards satisfying the constraint equations without being excessively detoured by the performance criteria. The cost of the speed increase is that the performance criteria are allowed less influence over the final solution point.

Besides increasing the solution speed, the dual-class sequential filters method guarantees finding a local optimum with respect to the transformed constraint equations. Essentially, the filter rejects all choices except those improving the composite index. Thus, the search can never proceed in a direction that would increase the composite index, regardless of the performance criteria values. Ultimately, no solutions pass the filter. This indicates the perturbations only generated trial solutions with composite index values greater than or equal to the value at the current base point. When using the exhaustive perturbation method, this satisfies the conditions for a local minimum.

Applying dual-class sequential filters changes the nature of the direct search strategy. Without the filters, the path of the search is determined by the entire performance index. With the filters, only solutions improving the constraint index influence the search. This strategy completely eliminates the problems caused by the extraneous performance criterion in Example 4.1. The benefits associated with formulating a constraint index and using it as an initial filter are persuasive. The major drawback is that the performance criteria are allowed less influence during the course of the direct search. A different filtering strategy increases the influence of the performance criteria, though it also increases the computational overhead.

The multi-zone sequential filters method does not make a fundamental distinction between the transformed constraint equations and the performance criteria. The operator ranks the performance criteria in order of importance and the operator is allowed to give the transformed constraint equations priority if the operator feels that is the best strategy. The direct search method then minimizes the composite performance index, which includes the constraint index and the performance criteria. After finding this minimum, the constraint equations are checked. If the constraints are met, the algorithm terminates with the assurance that a true solution was found. Satisfying the constraint equations guarantees a minimum, both global and local, of the constraint index. This is because the constraint index is formulated by transforming the constraint equations via an error function based on the absolute value of the constraint violations. Satisfying the constraint equations confirms the error is zero. There can be no solution, global or local, which makes the error less than zero. The sequential filters strategy is employed when the minimum of the composite performance index does not satisfy the constraint equations. In this case, the filter removes the lowest-ranking performance criterion from the composite index and the direct search is reapplied. After finding a minimum of the modified performance index, the constraint equations are again checked. If the constraints are not satisfied, the filter removes the next lowest-ranking performance criterion and the direct search is again applied. This procedure repeats until the only remaining criterion in the composite performance index is that given the highest priority by the operator. The next application of direct search (if it employs the exhaustive exploration strategy) is guaranteed to locally minimize this performance criterion. This will

only guarantee a minimum of the constraint index if the operator felt that the transformed constraint equations were the most important criteria.

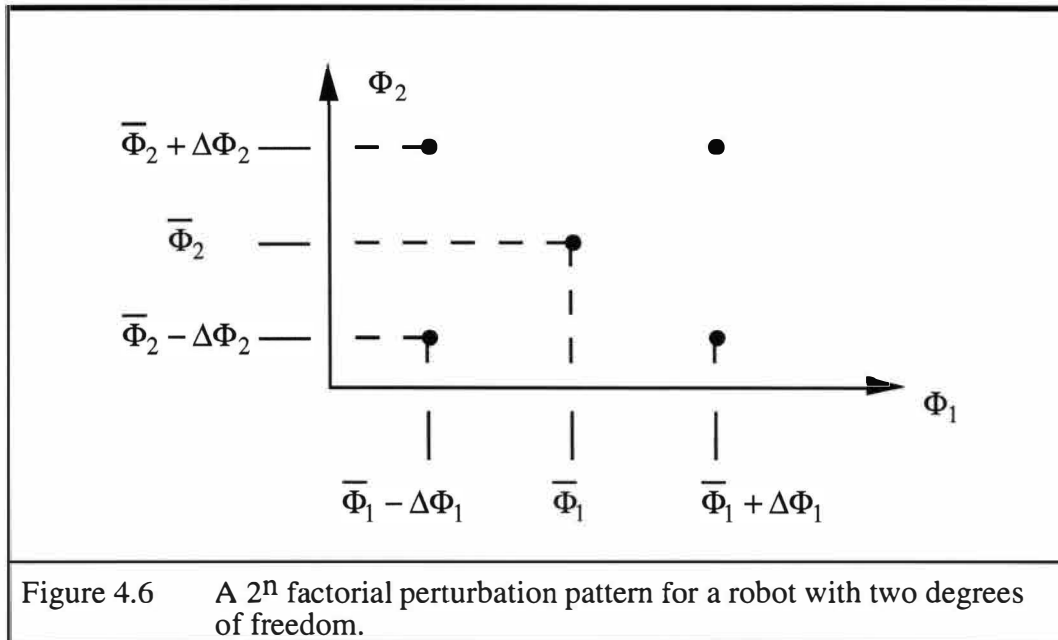
It may not be necessary to calculate all of the performance criteria at every step in the direct search process. The operator may also choose different sampling rates for different rankings of the criteria. For instance, the essential criteria will be included at every step. Important criteria, but not essential, may only be calculated and included in the optimization every tenth step. Somewhat important and infrequently important criteria may only need to be calculated every hundredth or thousandth step, respectively. This strategy will also increase the computational speed of the direct search inverse kinematics algorithm.

4.6 FACTORIAL DESIGN PATTERN

This chapter develops several methods of incorporating multiple performance criteria in a direct search strategy. All of these methods calculate criteria values at each of the trial solutions. Since many of the criteria involve complex computations, singular value decomposition for example, calculating them repeatedly represents a significant computational overhead. A more simple representation of the combined effects of the criteria would significantly improve the solution speed. A multivariable polynomial derived using factorial design techniques approximates these combined effects. This multivariable polynomial may be calculated during the course of the direct search using only the addition, subtraction, and averaging of the criteria values from the first set of trial solutions. Since the search calculates the criteria values anyway, deriving the multivariable

polynomial represents very little computational overhead. Using the polynomial approximation saves computation time during the remainder of the search.

Chapter 3 discusses the general method for applying a 2^n factorial design strategy as one of the perturbation strategies. The 2^n factorial design technique calculates the main effects of each joint variable on the performance criteria values as well as interaction effects between the joint variables. These effects are combined to form a multivariable polynomial approximating the actual performance index. As an example, Figure 4.6 shows a factorial perturbation pattern for a robot with two degrees of freedom about the base point: $\bar{\Phi}_1$, $\bar{\Phi}_2$.



Each of the four points about $\bar{\Phi}_1$ and $\bar{\Phi}_2$ represent a trial solution and its associated performance index value. These values are defined relative to the direction of the perturbation. For instance, at the point $\bar{\Phi}_1 + \Delta\Phi_1$ and $\bar{\Phi}_2 - \Delta\Phi_2$,

the trial solution value is $T_{+\Phi_1-\Phi_2}$. The 2^n factorial design technique calculates the main and interaction effects using the faces and the diagonals of the square formed by the four trial solutions.

The main effects are calculated using the trial solutions in which the variable of interest changes while the other variables stay the same. In Figure 4.6 there are two cases where Φ_1 changes while Φ_2 does not. These two cases are at $\bar{\Phi}_2 + \Delta\Phi_2$ and $\bar{\Phi}_2 - \Delta\Phi_2$. The main effect of changing Φ_1 is the average of these two effects.

$$E_{\Phi_1} = \frac{(T_{+\Phi_1+\Phi_2} - T_{-\Phi_1+\Phi_2}) + (T_{+\Phi_1-\Phi_2} - T_{-\Phi_1-\Phi_2})}{2} \quad (4.22)$$

In this equation, E_{Φ_1} represents the main effect of changing the joint displacement, Φ_1 . In general, the main effect of changing a joint displacement is the sum of the trial solution values with positive perturbations minus the sum of those with negative perturbations; all divided by half the total number of trial solutions.

Whereas calculating the main effects uses the faces of the square in Figure 4.6, calculating the interaction effects uses the diagonals.

$$E_{\Phi_1\Phi_2} = \frac{(T_{+\Phi_1+\Phi_2} + T_{-\Phi_1-\Phi_2}) - (T_{+\Phi_1-\Phi_2} + T_{-\Phi_1+\Phi_2})}{2} \quad (4.23)$$

which corresponds to the diagonals in Figure 4.6. Yates (1937) described a procedure for calculating the main and interaction effects for any number of independent variables using only simple arithmetic operations.

Combining the main and interactive effects forms a multivariable polynomial. The polynomial approximates the summed values of the performance criteria. The approximation polynomial for the two degree of freedom case represented in Figure 4.6 is

$$f(\Phi_1, \Phi_2) = A + E_{\Phi_1} \Phi_1 + E_{\Phi_2} \Phi_2 + E_{\Phi_1 \Phi_2} \Phi_1 \Phi_2 \quad (4.24)$$

This polynomial is defined about the base point where the perturbations were performed and is most accurate in this region. The coefficient, A , is simply the average of the trial solutions about the base point. The values for the joint variables, Φ_1 and Φ_2 must be normalized with respect to the base point and according to the perturbation size. Recalling that the effects were calculated over two perturbation steps, from $\bar{\Phi}_i + \Delta\Phi_i$ to $\bar{\Phi}_i - \Delta\Phi_i$, the equation for normalizing joint variable, Φ_i , is

$$\Phi_{i \text{ normalized}} = \frac{\Phi_i - \Phi_{i @ \text{basepoint}}}{2\Delta\Phi_i} \quad (4.25)$$

where $\Delta\Phi_i$ is the perturbation size. The availability of a polynomial approximating the performance criteria values is simply an option. There is no requirement for calculating or using it during the direct search at all. In situations where the solution speed is not a consideration, such as off-line programming, using a polynomial approximation is not recommended. If, however, performance criteria calculation is creating a bottleneck, using a multivariable polynomial approximation, such as Equation (4.24) might be warranted.

The availability of a simple polynomial approximation raises an interesting question. Could the polynomial approximate the entire composite performance index, including both performance criteria and transformed

constraint equations? The direct search could then simply minimize the approximation polynomial. Unfortunately, the approximation polynomial cannot be minimized because it has no extrema. This is easily shown. As discussed in Chapter 3, the condition for a local minimum is that the Hessian matrix is positive semidefinite. If the Hessian is indefinite, the point is neither a minima nor a maxima. Showing that the Hessian matrix is always indefinite shows that the approximation polynomial has no extrema. Consider the multivariable approximation for a robot with two degrees of freedom.

$$f(\Phi_1, \Phi_2) = A + E_{\Phi_1} \Phi_1 + E_{\Phi_2} \Phi_2 + E_{\Phi_1 \Phi_2} \Phi_1 \Phi_2 \quad (4.24)$$

The first partial derivatives are

$$\begin{aligned} \frac{\partial f(\Phi_1, \Phi_2)}{\partial \Phi_1} &= E_{\Phi_1} + E_{\Phi_1 \Phi_2} \Phi_2 \\ \frac{\partial f(\Phi_1, \Phi_2)}{\partial \Phi_2} &= E_{\Phi_2} + E_{\Phi_1 \Phi_2} \Phi_1 \end{aligned} \quad (4.26)$$

The second partial derivatives are

$$\begin{aligned} \frac{\partial^2 f(\Phi_1, \Phi_2)}{\partial \Phi_1^2} &= 0 \\ \frac{\partial^2 f(\Phi_1, \Phi_2)}{\partial \Phi_1 \partial \Phi_2} &= E_{\Phi_1 \Phi_2} \\ \frac{\partial^2 f(\Phi_1, \Phi_2)}{\partial \Phi_2^2} &= 0 \\ \frac{\partial^2 f(\Phi_1, \Phi_2)}{\partial \Phi_2 \partial \Phi_1} &= E_{\Phi_1 \Phi_2} \end{aligned} \quad (4.27)$$

which may be formed into the Hessian matrix

$$\mathbf{H}(\Phi_1, \Phi_2) = \begin{bmatrix} 0 & E_{\Phi_1 \Phi_2} \\ E_{\Phi_1 \Phi_2} & 0 \end{bmatrix} \quad (4.28)$$

Notice the Hessian matrix has all zeros in the diagonal. This will always be true because the polynomial approximation has no variables of degree higher than one. A matrix with all zero diagonal elements is indefinite. Thus, it has been shown that the multivariable polynomial approximation has no extrema. This underscores the fact that the polynomial calculated using these factorial design techniques is an approximation only relevant in the locality from which it was derived.

4.7 SIMULATED ANNEALING

Several factors recommend applying direct search to the generalized inverse kinematics problem. For instance, the direct search can be automatically generated for any serial robot geometry. Also, the search can consider an essentially unlimited number of performance criteria. Perhaps most importantly, the direct search is completely stable in the presence of robot singularities. This feature is unique among inverse kinematics algorithms, closed-form or iterative. All of these strengths are derived from using trial solutions to guide the search. Another numerical method based on trial solutions would presumably share these strengths. Simulated annealing is another such method. Annealing is used in the heat treatment of materials. Simulated annealing, as the name suggests, simulates the process on a computer. The main advantage is that simulated annealing may find global minima whereas direct search is only guaranteed to find local minima. This section more thoroughly discusses simulated annealing. The discussion

includes a method for applying simulated annealing to the multicriteria inverse kinematics problem, the associated benefits, and the difficulties.

Annealing describes a process of heating a material to an elevated temperature and then cooling it very slowly. The slow cooling allows the material to reach a low energy state in which it is relatively ductile. Somehow, with no intelligence or systematic strategy, a material minimizes its own energy state during the slow cooling. Simulated annealing is an approximation of this natural process carried out on a computer and is based on the Boltzmann probability distribution.

$$\text{Prob}(E) \approx \exp\left(-\frac{E}{kT}\right) \quad (4.29)$$

In this equation, E is the energy of the system, k is Boltzmann's constant, and T is the temperature. Essentially, Equation (4.30) states that a system's energy is probabilistically distributed depending upon the temperature. As the temperature increases, the probability of the system assuming a higher energy state increases. As the temperature is lowered, the odds of the system leaving a lower energy state decrease. Because simulated annealing algorithms sometimes leave lower energy states for higher ones, they can escape from local minima. Simulated annealing algorithms typically include a method of generating random changes in the system's configuration. The random changes represent trial configurations which are evaluated using Equation (4.29). If so indicated, the system assumes the trial configuration; otherwise it is discarded. This procedure of randomly generating trial solutions and evaluating them with Equation (4.29) repeats over and over in the simulated annealing process.

The method of simulated annealing may be applied to the multicriteria inverse kinematics problem. The procedure for calculating the criteria and constraint values for a given trial solution is the same as in the direct search approach. Generating the trial solutions, however, proceeds in a very different manner. Whereas a direct search generates a systematic pattern of trial solutions, in simulated annealing, they are generated randomly. The composite performance index corresponds to the system's energy and remains a basis for comparing one trial solution with another. As has been shown, a minimum of the composite performance index, even a global one, will not necessarily correspond to a solution of the multicriteria inverse kinematics problem.

This chapter discussed two methods of dealing with extraneous minima. One increases the scale factors of unmet constraints, while the other uses sequential filters. The sequential filters method is very systematic and fits nicely with the systematic direct search approach. Simulated annealing, on the other hand, derives much of its strength from randomness and it seems that the systematic use of sequential filters would dilute this strength. Increasing the scale factors for unmet constraints, however, fits nicely into a simulated annealing algorithm. Essentially, after each round of simulated annealing the solution is checked to see if the constraints have been met. If the constraints are satisfied, a solution was found and the algorithm terminates. If a solution was not found, the scale factors for the unmet constraints are increased and the simulated annealing is performed again. This process continues until it finds a solution.

Implementing a simulated annealing algorithm requires choosing several parameters. These parameters are: an initial temperature, a final temperature, an

initial set of scale factors, a schedule for decreasing the temperature, and a schedule for increasing the scale factors of unmet constraints. The literature presents several methods for choosing an initial temperature, a final temperature, and an annealing schedule. Press et al. (1991) develop a logical method. The method first calculates the energy for a series of exploratory trial solutions and chooses an initial temperature of ten times the typical energy value. It then decreases the temperature by ten percent after generating and evaluating $100 \cdot n$ (where n is the number of degrees of freedom) trial solutions. The method follows this schedule until the energy value decreases no further. The initial scale factors can be calculated by the methods developed in this chapter. After each round of simulated annealing, increasing the scale factors of unmet constraints is necessary. Previously this chapter suggested increasing the scale factors of unmet constraints by a power of ten. However, that was for the locally-based direct search method. A local method might require completely obliterating minima due to the performance criteria in favor of satisfying the constraint equations. This is why a sequential filters method which guarantees the constraints priority without completely overwhelming the effects of the performance criteria is so attractive. Because the simulated annealing method is globally-based, the scale factors of unmet constraints need not change so drastically. Allowing the performance criteria as much influence as possible is desirable. There is, however, a tradeoff involved. Increasing the scale factors of unmet constraints by only a small amount, say ten percent, may require many rounds of simulated annealing and this could be quite time consuming. Doubling the scale factors of unmet constraints after each round of simulated annealing is a reasonable compromise.

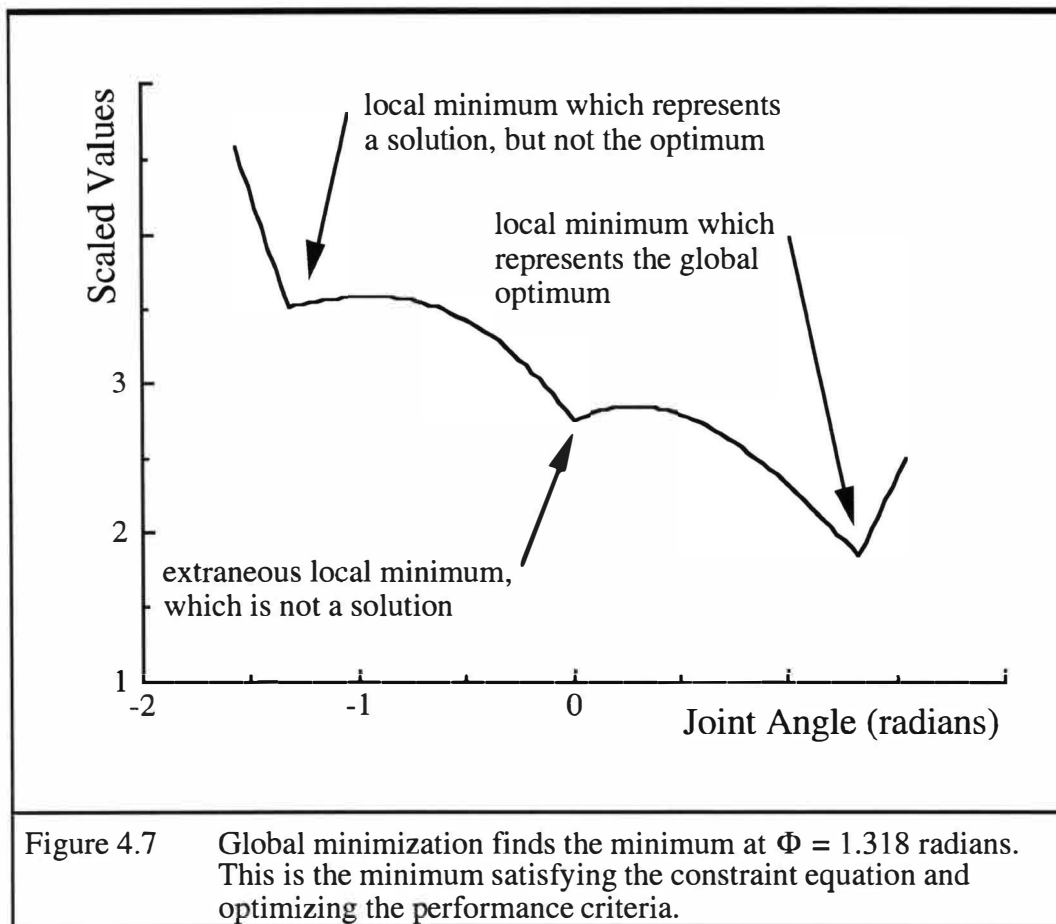
An additional performance criterion added to Example 4.1 illustrates the benefits of global, as opposed to local, minimization. This additional performance criterion makes the solution at positive Φ more attractive than the one at negative Φ .

$$P_3 = \left| \frac{\pi}{2} - \Phi \right| \quad (4.30)$$

At the positive boundary of the workspace, $\frac{\pi}{2}$, the value of this criterion is at its minimum, zero. At the negative boundary of the workspace this criterion is at its maximum, π . Forming a composite performance index

$$PI = s_1 P_1 + s_2 P_2 + s_3 P_3 \quad (4.31)$$

The global scaling method generated the scaling factor, s_3 . Figure 4.7 graphs the composite index as a function of the joint angle Φ . There are three local minima. The minimum at $\Phi = -1.318$ is a solution to the problem, but not the one minimizing the performance criteria. The minimum at $\Phi = 0$ is extraneous and not a solution to the problem. If the simulated annealing successfully locates the global minimum, it will find $\Phi = 1.318$ radians. This is the minimum satisfying the constraint equation and optimizing the performance criteria.



Locating global, rather than local, minima is simulated annealing's main strength. Natural annealing is a proven method of finding global minimum and, since no intelligence is involved, it likely can be simulated on a computer. Other factors recommend simulated annealing as well. It essentially solved the classic "traveling salesman" problem, as well as several interesting permutations of the problem (Press et al., 1991). Simulated annealing is attracting increased interest within the research community, which should result in improved algorithms. One such improvement is called fast simulated annealing. The name of this

improvement alludes to the main problem associated with applying simulated annealing to the multicriteria inverse kinematics problem.

Like its natural counterpart, simulated annealing takes some time. Simulated annealing has the most success with problems, like the traveling salesman problem, having a discrete and combinational solution space. Essentially, every combination represents a solution. The multicriteria inverse kinematics problem, on the other hand, has a continuous solution space and requires solutions with a very high degree of precision. Finding a solution at this level of precision using randomly generated sets of joint displacements will likely take a very long time. The amount of time varies depending upon the joint range and the manipulator geometry. An estimate is, however, possible. In dealing with other continuous nonlinear minimization problems, Spang (1962) derives Equation (4.32) which predicts the number of random reconfigurations required for a 90% certainty of finding a solution.

$$2.3 \prod_{i=1}^n \epsilon^{-1} \quad (4.32)$$

In this equation, ϵ represents the precision of the solution and n is the number of degrees of freedom. Evaluating Equation (4.32) for $\epsilon = .001$ and $n = 7$, gives 2.3×10^{21} randomly generated trial solutions. If evaluating each trial solution takes a microsecond, Equation (4.32) predicts finding an actual solution will take about 10^8 years. Clearly, this calculation is affected by the choice of units and is only an approximation. Nonetheless, it is unlikely that simulated annealing will become a real-time method of solving the multicriteria inverse kinematics problem in the near future. Using simulated annealing at a much lower precision,

say 1 degree, and then refining the solution with another method, such as direct search, is one answer to this dilemma. This combination results in a composite algorithm with a variety of very attractive attributes, including: a global basis, automatic generation, singularity robustness, and multicriteria capability.

4.8 CONSERVATIVE PATHS

Conservative paths, though not a part of the multicriteria inverse kinematics problem as defined by Equations (4.1) to (4.3), are almost invariably mentioned when discussing inverse kinematics methods for redundant robots. As commonly defined, a conservative path is closed in both joint and end-effector space. This section includes a discussion as to the adequacy of this definition and proposes a more useful one. Though the local nature of direct search cannot guarantee a conservative path, experience shows that direct search will typically find one after repeating several cycles of the same end-effector path. This section presents a conceptual explanation of this observation and shows that once a conservative path is found, the direct search remains on that conservative path.

A conservative path is only important when the robot repeatedly performs the same task, as is the case in assembly and manufacturing operations. Whether or not the path is conservative affects the accuracy and reliability of the robot in these applications.

Modern industrial robots typically boast a very fine, on the order of .001 inches, repeatability even though their absolute precision is typically limited to .01 inches. Manufacturing tolerances limit the precision. Repeatability cancels the effects of these tolerances because it is a difference measurement. Thus, once

an industrial robot performs a task, it typically repeats that task over and over with a very fine resolution. As industrial robots begin to incorporate kinematic redundancies, it becomes possible for the end-effector to repeatedly follow the same path while the joints follow different paths. Unless the joints follow their same respective paths each time the end-effector follows the same path, the repeatability is no better than the absolute precision.

Reliability is another issue driving the need for conservative paths during repetitive operations. If the path is not conservative, the robot may eventually encounter a problem configuration, such as a joint limit or singularity. For this reason, pseudoinverse, or other velocity-level, solutions cannot reliably repeat the same path.

A conservative path is commonly defined as closed in both end-effector and joint space. That is, when the end-effector follows a path which returns to its starting place, the joint displacements must also return to their starting place. This definition, however, allows for the possibility that the joints may finish at their initial displacements without repeating their same paths. In other words, the joints get back to where they started, but follow a different path each time. If this happens, the benefits to the repeatability and reliability are sacrificed. A more useful definition of a conservative path is: a path in which both the joints and the end-effector repeatedly follow their same respective trajectories.

This section discusses the importance of conservative paths to the repeatability and reliability of repetitive operations. Unfortunately, it also shows that direct search cannot guarantee a conservative path. The pseudoinverse and extended Jacobian methods of solving the inverse kinematics problem also cannot

guarantee a conservative path. On an abstract level, it is not difficult to understand why these methods cannot guarantee conservative paths. These methods are all local, while the conservative path specification is whole-path. If a guaranteed conservative path is absolutely essential, a whole-path method of solving the inverse kinematics problem, where the final joint displacements are specified as boundary conditions, is probably required. Why direct search cannot guarantee a conservative path may be explained in terms of the search strategy. Any given solution direct search finds is a function of the initial set of joint displacements and the decisions made while the search proceeds towards a solution. To facilitate this discussion, let $\mathbf{x}_1$ represent the first set of displacement constraints on the end-effector. The next set in the path is $\mathbf{x}_2$, the next $\mathbf{x}_3$, and so on until the final point in the path, $\mathbf{x}_f$. Because this is a closed path in end-effector space

$$\mathbf{x}_1 = \mathbf{x}_f \quad (4.33)$$

The set of joint displacements corresponding to the end-effector constraints are represented by Φ_0 through Φ_f and the set of decisions by $\mathbf{v}_1$ through $\mathbf{v}_f$. The final set of joint displacements is a function of the initial set of joint displacements and the intermediate decisions.

$$\Phi_f = f(\Phi_0, \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_f) \quad (4.34)$$

Even though Φ_f is a function of Φ_0 , the direct search makes each of the decisions, $\mathbf{v}_1$ through $\mathbf{v}_f$, on a local basis without any “global knowledge” of whether they lead to a solution where Φ_f equals Φ_0 . Each time the robot repeats the end-effector path, the locally-based decisions may very well lead to different

paths in joint space. Nonetheless, in almost every case, direct search typically finds a conservative path.

Experience shows that after cyclically repeating the same end-effector path, the direct search typically converges to a conservative path and then continues to repeat that path. This is true for many different robot geometries and performance criteria. Though this observation is difficult to explain analytically, an analogy with a natural phenomenon seems appropriate. The analogy is with many small creeks that flow into one major river. Even though the creeks only make decisions based on their local landscape, all of the creeks in the area eventually flow into the same river. In other words, the creeks follow their own local minima and the general trend in the area leads to the river. The first few times through the end-effector path, the direct search follows creeks. Eventually, it finds the river. Once in the river, the direct search does not escape and the path becomes conservative. Certainly, this is only an analogy and no guarantee that the direct search will find a conservative path. It can, however, be shown that after finding a conservative path, direct search will continue to follow it.

For a conservative path, Φ_0 equals Φ_f .

$$\Phi_0 = \Phi_f \quad (4.35)$$

Substituting Equation (4.35) into Equation (4.34) gives

$$\Phi_f = f(\Phi_f, \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_f) \quad (4.36)$$

Thus, as long as the search generates the trial solutions in a systematic fashion and the basis for making decisions remains constant, a conservative path will repeat. This is why the performance criteria must be single valued functions. Essentially,

the decision making process is presented with the same set of options and chooses from among those options based on the same set of performance. Conversely, a search based on random trial solutions, such as the simulated annealing method, cannot guarantee a conservative path.

4.9 COMPARISON OF DECISION MAKING STRATEGIES

This chapter posed generalized inverse kinematics as an optimization problem and developed a decision making process for solving the problem. As with all optimization, the definition of optimality is a paramount concern. This chapter discusses two different definitions of optimality. A non-dominated solution is the first definition of optimality. The second is a minimum of a composite performance index. There are strengths and weaknesses associated with both definitions. The chapter then develops a decision making process based on direct search for locating these optima. Three different varieties of this decision making process are formulated. The first is an unmodified direct search. The second is direct search with dual-class sequential filters. The final decision making strategy is direct search with multi-class sequential filters. These decision making strategies vary according to their level of sophistication, their capability for interaction with the robot's operator, and their reliability in finding an optimum.

A non-dominated solution is one possible definition of optimality. A solution is non-dominated with respect to other solutions if it makes at least one criterion better while making no other criteria worse. Section 4.3 presents a formal definition of a non-dominated solution. The usefulness of the non-

dominated solution is that scaling difficulties are eliminated, but this also means that there is limited recourse when a non-dominated solution does not correspond to an actual solution of the problem at hand. The robot's operator may choose and rank the criteria in order of their importance to the task; which provides a small amount of interactive control. Once the criteria are ranked, however, decisions are based on a very simple rule. Thus, the non-dominated solution is a weak method of criteria based decision making.

A minimum of a composite performance index is the other definition of optimality this chapter discusses. The composite performance index may include an unlimited number of performance criteria and transformed constraints. Scaling issues are the paramount concern. By systematically scaling the criteria, the composite performance index becomes a powerful basis for decision making. It is also possible to the index into a series of indices and address them sequentially. Table 4.4 summarizes the essential characteristics of the non-dominated solution and the composite performance index.

Table 4.4 Some characteristics of the two different definitions of optimality this chapter discusses.	
Definition of Optimality	Characteristics
non-dominated solution	eliminates scaling difficulties decisions based on a very simple rule, thus this is a weak criteria based decision making method limited recourse when the non-dominated solution is not an actual solution

minimum of composite performance index	may include an unlimited number of criteria scaling issues are paramount may be divided into a series of distinct indices as powerful indicators of the robot's performance

This chapter develops several decision making strategies for finding solutions to the multicriteria inverse kinematics problem. These strategies are all based on direct search. Without modification, direct search may include an unlimited number of performance criteria and these criteria ultimately determine the results the search. The unmodified direct search is susceptible to extraneous minimum that do not correspond to actual solutions. There is some opportunity for interactive operation because the robot's operator may choose and rank criteria appropriate for a given task. This strategy does not allow the robot's operator to specify acceptable bounds for the criteria, however, so it is only mildly interactive. This strategy is also the most computationally demanding since all criteria are calculated for every option at each decision making step in the search.

The addition of a dual-class sequential filter to the direct search decision making strategy essentially eliminates the problems associated with extraneous minima. The dual-class sequential filter acknowledges the fundamental difference between transformed constraints and true performance criteria. Only trial solutions that improve the transformed constraints are passed by the dual-class sequential filter. This initial filter is computationally very simple and eliminates the need to calculate criteria values for most of the options at each decision making step in the search. This strategy is therefore the least computationally demanding of the three decision making strategies this chapter discusses. The

dual-class sequential filter allows the robot's operator to choose, rank, and impose acceptable bounds on the criteria that will be considered during the search. Thus, this method is interactive. The criteria, however, are only allowed to influence the search for trial solutions that pass the initial filter. Thus, the transformed constraint equations dominate the search and the influence of the operator's decisions and choices are diminished.

The final decision making strategy this chapter discusses is direct search with multi-class sequential filters. This method shares with the dual-class sequential filters the ability to eliminate problems associated with extraneous minima. This method is also extremely interactive because the robot's operator may choose, rank, and impose acceptable bounds on the criteria for a given. Thus, the operator's choices ultimately determine the results of the search. This method is more computationally demanding than the dual-class sequential filter method, but increases in the computation speed may be obtained by sampling the criteria at different rates during the decision making process. Table 4.5 summarizes the characteristics of the three decision making strategies this chapter discusses.

Table 4.5 Some characteristics of the three decision making strategies this chapter discusses.

Strategy	Characteristics
unmodified direct search	susceptible to extraneous minima limited reliability mildly interactive most computationally demanding may consider unlimited performance criteria performance criteria influence the search
dual-class sequential filters	eliminates extraneous minima extremely reliable some operator interaction least computationally demanding transformed constraints dominate the search may consider unlimited performance criteria
multi-class sequential filters	extremely interactive operator's choices determine the results eliminates extraneous minima computationally demanding, but this can be minimized by sampling the criteria at appropriate intervals may consider unlimited performance criteria reliable

Chapter 5: Automatic Plant Description

This chapter details the systematic design and implementation of an automatic physical plant description system for robots. The plant description is essentially a dynamic model of the robot. In its current form, this plant description system calculates performance criteria values, the joint forces or torques, and the deflection of the robot's end-effector. The system is automatic because it produces its numerical outputs given its parametric inputs with no further user intervention. There are many uses for an automatic plant description system and many reasons for preferring a systematic design process.

Designing a robot requires evaluating many different design alternatives. The automatic plant description system can be an important part of the evaluation process. Choosing a monolithic robot for a given task also requires evaluating robots. Configuring a modular robot for a given task requires evaluating a large number of different configuration alternatives. The automatic plant description system will certainly play an important role in modular robot configuration. The automatic plant description system is very much tied to generalized inverse kinematics. A generalized inverse kinematics algorithm must incorporate performance criteria. Joint torques or forces and end-effector deflections are important performance criteria. The generalized inverse kinematics algorithm generates these criteria values using the plant description system. Many of the benefits associated with a general inverse kinematics algorithm will be lost if the plant description is not general and automatic. An automatic plant description

system could also be a part of a commercial software package. Finally, incorporating a plant description improves the performance of robot controllers.

This chapter details a systematic design process that loosely follows the method of Pahl and Beitz (1988). This design method leaves a structured record of the goals, constraints, and specifications that influenced the design. The design begins by identifying the system's inputs and outputs. These inputs and outputs combine with the intended uses of the automatic plant description system to determine a set of specifications. The design then divides the automatic plant description system into three subsystems to satisfy the specifications. A database subsystem stores and retrieves information associated with different robot modules. The model formulation subsystem accepts a description of the robot as its input. This subsystem combines the robot's description with information from the database to formulate a model of the robot. The final subsystem actually calculates the performance criteria values, joint torques or forces, and end-effector deflections. The model formulation subsystem communicates with the calculation subsystem via datafiles. This allows automatic downloading of the robot's model to any computer using standard file transfer protocols. The calculation subsystem may reside on a high-speed processor for real-time control applications.

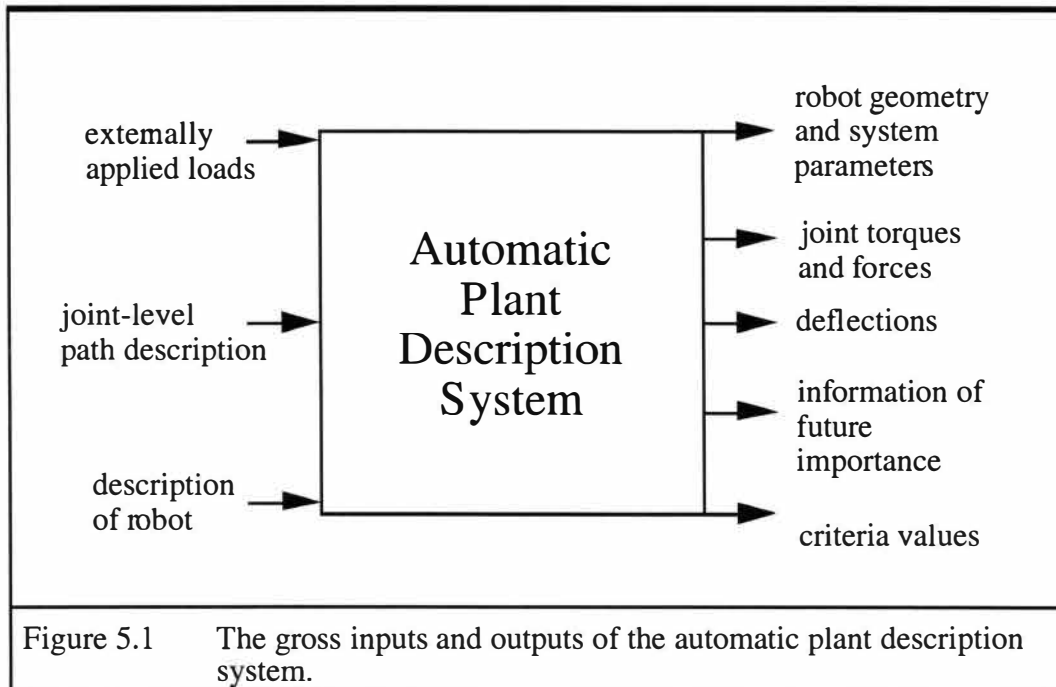
5.1 INPUTS AND OUTPUTS

Good systematic design demands the complete understanding and clear specification of all requirements at the outset of the design process. This facilitates an understanding of the task at hand and discourages unnecessary efforts that do not lead towards a solution. It is important that the specifications

do not imply any specific implementation as this might bias the outcome of the design and adversely affect the results. A clear specification sheet also provides a yardstick by which to judge the final design. An understanding of the automatic plant description system's inputs and outputs facilitates the development of a specification sheet. Fig. 5.1 shows these inputs and outputs graphically. This figure is a preliminary description of the system intended to facilitate the development of a set of specifications. As the design progresses, it will detail the structure of the data.

The robot's description is a fundamental input to the automatic plant description system. This description is also somewhat problematic since the robot may be modular or it may be monolithic. In the case of a modular robot, the automatic plant description system requires a list of the modules, the order in which the modules are arranged, and the geometry of the connections between the modules. For a monolithic robot, the system requires the robot's geometry as well as other physical data, such as mass and compliance parameter distribution. The description of the robot is specific to each particular robot, but independent of the task. The path description and the externally applied loads, however, are task-dependent inputs.

The path description is communicated to the automatic plant description system at the joint level. Describing the path at the joint level maintains a separation between the automatic plant description system and the inverse kinematics method. This separation allows for the system's use in applications not requiring generalized inverse kinematics. The separation also prevents the system from being tied to any one inverse kinematics method.



The externally applied loads are specific to the task and the environment. Gravity, for instance, is an externally applied load that is of little concern to a robot working in outer-space. Other externally applied loads include the various forces and torques at the robot's end-effector necessary to perform a given task.

Calculating the torques and forces at the joints of the robot is a primary function of the automatic plant description system. The torques and forces represent the load on the robot's actuators.

End-effector deflection refers to deviations between the desired and actual end-effector displacement due to deformations of the robot's structural and drive components. Both applied and inertial loads cause end-effector deflections. Gravitational forces, when applicable, typically cause significant deflections at the

end-effector. Vibrations, particularly if the arm is excited at a resonant frequency, may also cause significant deflections.

Performance criteria values are another output of the automatic plant description system. The performance criteria are mathematical measures with a physical interpretation. The literature review describes a number of criteria for evaluating the robot's performance.

The robot's geometry and other physical parameters are outputs of the system available for downloading onto a high-speed processor for real-time control. This output of the automatic plant description system must contain all the physical plant information required by the real-time controller.

The final output is information of future importance. The exact nature of this output is unknown, but undoubtedly in the future there will be requirements for the automatic plant description system that cannot be foreseen at this time. These requirements might concern thermal effects, reliability, or power demands. By including a provision for future requirements, the current capabilities of the system will not become embedded constraints and limit the usefulness of the system in the future.

5.2 SPECIFICATIONS

At this point in the design process, it is possible to develop a set of specifications for the system. Table 5.1 shows the specification sheet. The right column contains a brief description of the specification and the left column indicates whether the specification is a demand or a wish. The design is not satisfactory if it does not meet a demand. Wishes are incorporated if possible, but

the design will still fulfill its intended uses if that specification is not met. The intended applications of the automatic plant description system determined these specifications. This discussion will therefore focus on each specification as it affects the intended applications.

Table 5.1 A specification sheet for the automatic plant description system	
Demand or Wish	Specifications
D	formulate robot geometry and system parameters
D	calculate joint torques
D	calculate deflections at the end-effector
D	expandable to include new robot modules
D	calculate performance criteria values
D	portable among many different computing platforms
D	automatic (requires no intervention to produce outputs given inputs)
D	expandable to include new modeling and analysis capabilities
D	general with respect to the robot's geometry
W	use only University of Texas Robotics Research Group software

Generality with respect to the robot's geometry is important for several of the intended applications. Robot design involves a large number of design parameters that may vary independently. The definition of a general robot structure is that all the physical parameters may vary. A general automatic plant description system would also be useful when choosing from among the different available monolithic robots or configuring modular robots given a particular application. Finally, generality is important in an automatic plant description system embedded in a commercial software package. Essentially, a more general formulation has more market value.

Portable software is machine independent. Basically, the automatic plant description system should be compatible with as many computers as possible. Because computer technology is advancing at a rapid pace, tying the automatic plant description system to a specific computer platform would virtually guarantee obsolescence within just a few years. Portable software also has much more market value. Installed computer hardware varies from industry to industry and even a single customer might have several different computer platforms within the same factory. Software that runs on all of their computers and allows them to upgrade their computers without changing software will be very attractive to these customers. Maintaining software compatibility on different computer platforms is also more expensive and difficult than writing portable software from the outset.

Restricting this effort to software developed by the University of Texas Robotics Research Group's software is not a rigid requirement, but rather falls under the "wish" category. Doing so, however, ensures maintaining maximum flexibility as far as modifying and updating the software. This approach also acknowledges the Group's large inventory of available software and continuing work in software development.

Expandability to include new robot modules and new modeling techniques will allow the automatic plant description system to remain useful as the state of the art in modular robotics advances. New robot modules are being developed at the University of Texas and elsewhere. The automatic plant description system's structure should allow for the addition of these new modules. Work is also continuing in the area of generalized compliance modeling and dynamic

modeling. The system's design should allow for the inclusion of these new modeling techniques as well.

Several of the intended applications of the automatic plant description system may involve calculating forces and torques at the joints. During the robot's design, calculated forces and torques are useful when sizing the actuators. The robot's actuators must be powerful enough to achieve the payload and motion specifications. Distributing the torques or forces such that the actuators nearest the base of the robot support as much of the load as possible may be important for a given design. The required joint torques or forces are extremely important when evaluating a monolithic robot's ability to perform a given task. The most fundamental concern is that the robot must have actuators with a maximum capability greater than what is required at all points along the path. For a modular robot, the forces or torques at the joints are but one of the criteria used to assemble modules in response to a task specification. Finally, an inverse kinematics strategy for redundant robots can make use of calculated forces or torques to choose joint-level paths that optimize their distribution among the actuators.

Deflection calculation during the design of a robot measures the effects various design options have on compliance. A robot designed to withstand end-effector disturbances without significant deflections is essential for precision operation. Applied and inertial loads cause deflections that ultimately limit a robot's precision. With compliance modeling and deflection calculation, a monolithic robot capable of performing the desired task within the required precision may be intelligently chosen. When configuring modular robots,

compliance modeling and deflection calculation can form part of the module selection process. Properly choosing and arranging the modules will minimize the total deflection or, if the task requires, allow deflections in certain directions but not others. An inverse kinematics strategy for redundant robots can make use of deflection calculation to choose a joint-level path that minimizes undesirable deflections.

The performance criteria developed by the University of Texas Robotics Research Group (described in the literature review) are very high-level mathematical formulations for evaluating a robot as it performs a task. Robot design involves a large number of choices. Even a simple robot with six degrees of freedom has eighteen geometric, forty-two mass, thirty-six deformation, and eighteen actuator parameters, for a total of 114 parameters (Tesar and Butler, 1989a). The different combinations of these design parameters represent a huge number of options available to the designer. This large number of design parameters recommends the use of performance criteria to give a “big” picture of how the different design alternatives affect the capabilities of the robot. Choosing a monolithic robot for a given application might involve performance criteria calculation. This is not, however, a very good application of the high-level performance criteria. Most industrial robots are of a similar design. These performance criteria are more appropriate for choosing from among a larger number of options. On the other hand, configuring modular robots involves choosing from among a very large, yet finite, number of options. The performance criteria are ideally suited to this approach to design. Performance criteria are a fundamental part of generalized inverse kinematics. A generalized

inverse kinematics strategy must make intelligent choices to provide enhanced performance. Performance criteria form the fundamental basis for these decisions.

The specification that the plant description system be automatic requires that the system produce the desired operational outputs given the parametric inputs without further intervention by the user. Most of the intended uses of the automatic plant description system involve evaluating a large number of robots. An automatic system is much more convenient than one requiring user intervention for each different set of robot parameters. Also (assuming the algorithms are correct) an automatic system is much less likely to make errors. Finally, a Bachelors-level engineer, or even a trained salesperson, could operate an automatic plant description system. This gives an automatic system much more market value than one requiring intervention by a robotics expert.

5.3 SUBSYSTEM IDENTIFICATION

Several of the specifications call for specific calculations. These calculations are based on a model of the robotic system. Formulation of the model, in turn, requires physical parameters specific to each robot. Thus it can be seen that the automatic plant description system may be divided into three main subsystems: calculation, model formulation, and information storage (database).

The calculation subsystem is responsible for generating the numerical values for the joint torques or forces, the deflections, and the performance criteria. This subsystem could reside on the same computer as the other subsystems. This would likely be the case in a simulation environment. For real-time control, the

calculation subsystem would reside on a dedicated high-speed processor. Computer code for performing these calculations already exists and is available for use in the automatic plant description system. Unfortunately, the existing code does not have a unified format for the robot model. The structure for the model input has taken two distinct formats. The earliest and most general code accepts the Denavit and Hartenberg parameters as a description of the robot's geometry. The newer code uses a modular description of the robot's geometry (essentially the specific modules and the geometry of the connections). In both cases, the mass and compliance distributions are referenced to the local frame. There are positive and negative qualities associated with both modeling methods and the automatic plant description system should accommodate both approaches.

Table 5.2 The inputs and outputs of the three subsystems which comprise the automatic plant description system.		
Subsystem	Input	Output
calculation	system model in either a DH or a modular format task information, including the joint displacements, joint speeds, joint accelerations, and end-effector loads.	deflections at the end-effector joint torques performance criteria values
formulation	description of the robot in either a DH or modular format pertinent data from the database	system controlling model in either a DH or a modular format
database	robot or module of interest parameters entered into the database	requested operational parameters

The model formulation subsystem assembles the physical parameters of the robot into a format which can be understood by the calculation subsystem. The design of the model formulation subsystem is complicated by the two different model formats required by the calculation subsystem. If the robot of interest is modular and the calculation software requires the Denavit and Hartenberg parameters, a processing step will be necessary to extract these parameters from the modular robot description. Several researchers have proposed procedures for performing this extraction process. Unfortunately, their procedures do not allow for a general modular geometry. A process which is general with respect to the modular robot is developed as part of this design process.

The final subsystem performs the data storage and retrieval. This is a database where the physical parameters of each module, such as the geometry or mass, are stored. Database management is a common use for computers and there are many developed techniques available for this database application. This database is somewhat unusual because it will be updated infrequently – only when new modules are added – but accessed often. The information that needs to be associated with each module is determined by the needs of the calculation subsystem. The raw data in the database is assembled by the model formulation subsystem and then communicated to the calculation subsystem.

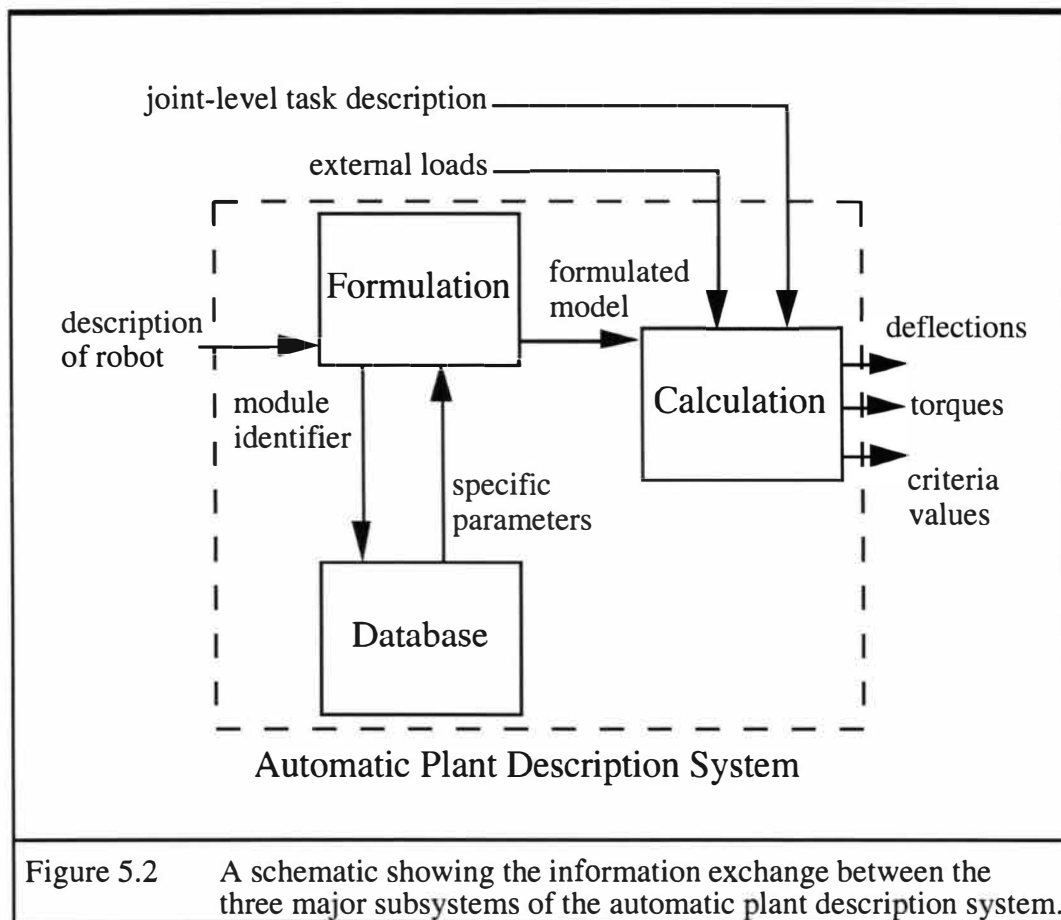


Figure 5.2 A schematic showing the information exchange between the three major subsystems of the automatic plant description system.

Figure 5.2 shows the interconnections between the subsystems. Each of the subsystems will now be discussed in detail and several options for fulfilling their functions will be identified and developed. As the subsystems are being discussed, the inputs and outputs of each subsystem will be explicitly formulated. By carefully defining the structure of the interface between the subsystems, functional independence of each subsystem is maintained. Thus it will be possible to upgrade an individual subsystem without requiring changes to the rest of the automatic plant description system. For instance, a calculation subsystem

developed for a specialized real-time controller will be able to access to the model formulation subsystem.

5.3.1 Calculation Subsystem

The calculation subsystem is where several of the outputs of the automatic plant description system are actually generated. The input to this system is a parametric description of the robot and the task that the robot is to perform. The outputs are the joint torques or forces, the end-effector deflections, and performance criteria values. This subsystem could reside on a general-purpose computer for simulation applications, or it could reside on a specialized computer for real-time control applications. The important issue is the identification and description of the subsystem's inputs. The proper inputs can then be communicated to the calculation subsystem wherever it resides. There are two general types of inputs. The robot-specific inputs describe physical parameters of the robot and the task-specific inputs describe the task.

The calculation subsystem essentially performs kinematic and dynamic analysis of robotic systems; a research area in which the University of Texas Robotics Research Group has made a significant investment. This research has built upon the kinematic influence coefficients first developed for the analysis of planar Assur groups by Benedict and Tesar (1978). Various researchers have extended the use of kinematic influence coefficients to include the analysis of: torque requirements for general serial manipulators (Thomas and Tesar, 1982), end-effector deflections of general serial manipulators (Fresonke et al., 1988), torque requirements for modular manipulators (Kang et al., 1992), and

performance criteria for general serial manipulators (Van Doren and Tesar, 1992). This effort has resulted in a large stock of computer software that will satisfy the requirements for the calculation subsystem.

Table 5.3 Some functions and attributes of robot analysis software available for use in the calculation subsystem.	
Software	Functions and Attributes
NDOF serial analysis software	entirely general Denavit and Hartenberg-style inputs calculates joint torques calculates deflections referenced to the end-effector
Module-Based Manipulator Analysis Package	calculates joint torques modular-style inputs seven different types of modules are addressed
Performance Criteria Code	calculates the values for twenty-nine different criteria general with respect to the robot's geometry Denavit and Hartenberg-style inputs

Work in the area of kinematic and dynamic analysis of robots is continuing at the University of Texas Robotics Research Group and elsewhere. The deflection and compliance analysis for modular manipulators is but one example of this continuing work. It is also likely that kinematic and dynamic analysis code for real-time control algorithms will be specialized for the controller hardware. Given that the calculation subsystem will continue to evolve, the input format of the analysis code that already exists will be identified and standards suggested so that future code can maintain compatibility with the existing software.

The foundation of the computational effort in kinematic and dynamic analysis is the NDOF serial modeling code (Wander and Tesar, 1987). This code was originally written for execution on a specialized array processor but was later rewritten in the portable “C” programming language. This code is entirely general in the sense that it will calculate the joint torques or forces for a serial robot with any number of degrees of freedom. The robot-specific inputs are the geometry and the mass distribution of the robotic system. The geometry is expressed in terms of the Denavit and Hartenberg parameters. The mass distribution is given by the individual masses, the location of the mass-centers and the inertia tensors; all referenced to the local coordinate system. The robot-specific inputs are communicated via datafiles. The task-specific inputs are the applied end-effector loads and the joint states. The applied loads are referenced to the base frame coordinates. The joint states are simply the displacements, speeds, and the changes in speeds with respect to time.

The NDOF compliance code calculates the deformations expressed at the end-effector due to applied and inertial loads acting upon structural and drive flexibilities (Hernandez and Tesar, 1989). Local deformations are transferred to the end-effector via kinematic influence coefficients. This code was also originally written for execution on an array processor and later rewritten in its portable form in the “C” programming language. The compliance code is general in the sense that it applies to any serial robot with any number of degrees of freedom. The robot specific inputs to this code are the compliance distribution, the mass distribution and the robot geometry. The robot geometry is described using Denavit and Hartenberg parameters. The mass distribution is given by the

individual mass values, the location of the mass centers, and the inertia tensors; all referenced to the local coordinate frames. The compliance is described in six general directions, three translations and three rotations, also referenced to the local frames. The robot specific parameters are communicated via datafiles. The task-specific inputs are the applied loads and the individual joint states. The applied loads are referenced to the base frame and the joint states are the displacements, speeds, and changes in speeds with respect to time.

The module-based manipulator analysis package written by Kang (1992) is significantly different from the NDOF code in the way that the robot-specific inputs are formulated. The modular analysis package is tailored towards modular robots, rather than robots described by Denavit and Hartenberg parameters. The module-based manipulator analysis package will analyze any modular robot composed of nine basic modules: constant link, prismatic joint, base joint, elbow joint, 2 DOF knuckle, 3 DOF wrist/shoulder, 1 DOF parallel elbow, 2 DOF five-bar, and 2 DOF seven bar. The dynamic analysis is based on the recursive Newton-Euler formulation. The robot specific inputs are: the robot modules, the connection geometry between the modules, the ordering of the modules, and the mass distribution within the modules. The robots modules are simply specified by their names. The connection geometry is specified via *XZX* Euler angles. The ordering of the modules is serial; meaning they are simply listed from first to last. The mass distribution is specified by the individual mass values, the location of the mass centers and the inertia tensors referenced to local frames. For the serial modules the mass distribution is specified just as in the NDOF code. A convention is developed to specify the geometry and mass distribution for the

parallel modules. For the larger modules, such as the five-bar and seven-bar, this represents a significant amount of information. There are over thirty parameters which describe these module's geometry and mass distribution. The module-based manipulator analysis package is written in the "C" programming language and the robot-specific inputs are communicated via datafiles. The task-specific inputs are the joint states and are also communicated via datafiles. The module-based manipulator analysis package, as it is currently written, does include end-effector loads.

A module-based manipulator compliance analysis effort is currently under way. Though still in development, several aspects of this system are known. The computer code is written in "C". The analysis is based upon a known set of modules connected at rigid interfaces and the information pertinent to each module is expressed in local coordinates. This module-based manipulator compliance analysis package is mentioned to emphasize that continuing work is expected in the area of module-based kinematic and dynamic analysis.

A generalized inverse kinematics implementation calculates performance criteria values using the automatic plant description system. The University of Texas Robotics Research Group has developed software that will calculate criteria values for at least twenty-nine different performance criteria (Van Doren and Tesar, 1992). This code is unique to the University of Texas Robotics Research Group and there are no alternatives available from outside sources. The code is written in the "C" programming language. The literature review in this dissertation contains a listing of the performance criteria values which may be calculated using this code. The code is general in the sense that it can calculate

the criteria values for any serial robot with any number of degrees of freedom. The robot-specific inputs are the mass distribution, the compliance distribution and the robot geometry. The mass distribution is given by the individual mass values, the location of the mass centers, and the inertia tensors; all referenced to the local coordinate frames. The compliance is described in six general directions, three translations and three rotations, also referenced to the local coordinate frames. The robot geometry is described using Denavit and Hartenberg parameters. The robot-specific parameters are communicated via datafiles. The task-specific parameters are the joint displacements, speeds, and changes in speeds with respect to time. The task-specific parameters are communicated to the code either through datafiles, or as arguments to a function call which also identifies the specific criterion of interest.

The main difference in all the previously mentioned software packages is the format for describing the robot's geometry. The NDOF code and the performance criteria code use Denavit and Hartenberg parameters, while the module-based manipulator analysis package and the (in-development) module-based compliance modeling package use a sequence of known modules. In both approaches, the mass and compliance distribution is described relative to the local frames. There are positive and negative qualities associated with both methods of specifying the robot's geometry.

Denavit and Hartenberg parameters are suited for monolithic robots. The Denavit and Hartenberg parameters are commonly used by most researchers in the robotics field and their use facilitates the importation of outside technology. Denavit and Hartenberg parameters do not, however, provide enough information

for some types of analysis, especially those associated with modular robots. For instance, the Denavit and Hartenberg parameters do not give information about where one module ends the next begins. It is not possible to extract which modules comprise the robot simply from the Denavit and Hartenberg parameters. This is because different modules could be arranged so as to have the same Denavit and Hartenberg parameters.

The modular approach to kinematic and dynamic analysis is much simpler with regard to the robot-specific inputs. A graphical user interface, such as that in the robo_cad software package, makes it quite easy to assemble the modules and generate a file describing the modules and their connectivity (Hooper and Tesar, 1990). Assigning Denavit and Hartenberg parameters, on the other hand, requires some skill in the art of robotics. The modular approach also allows parallel modules to form hybrid chains with both serial and parallel structures. The standard Denavit and Hartenberg parameters do not include hybrid chains. Finally, a modular description is rather clumsy for monolithic robots, though it is possible using very general modules.

The commonalties among the various software packages suggest standards so that future modeling and analysis efforts can maintain compatibility. The most important standard is communicating the robot-specific inputs with a datafile. All of the existing University of Texas software accepts the robot-specific inputs in this fashion and files provide a convenient interface that allows software modules to maintain functional independence. File transfer protocols exist for easy automatic downloading to high-speed processors for real-time control applications. Another standard is that the task specific inputs should include, as a

minimum: the joint displacements, speeds, and changes in speeds with respect to time. Dynamic analysis code should also include the loads at the end-effector referenced to the base frame coordinates. All of the kinematic and dynamic analysis software, with the exception of the module-based manipulator analysis package, adheres to this standard. The inability to include end-effector loads is a shortcoming of the module-based manipulator analysis package. A final standard is that the mass and compliance distribution should be referenced to the local frames. For purely serial structures the protocol of Wander, Hernandez and Tesar should be used, and for in-chain parallel structures the protocol of Kang and Freeman should be used. No clear standard as to whether the geometry of the robot should be represented by Denavit and Hartenberg parameters or as a sequence of modules has evolved. There are benefits to both approaches. The development of a method for extracting Denavit and Hartenberg parameters from a modular sequence will, to a large extent, alleviate difficulties that might arise from failing to set a standard in this area.

5.3.2 Database Subsystem

A database keeps records on a permanent storage medium that is accessed by various users at different times. The automatic plant description system's database keeps a record of the available modules and associates with each module a variety of information. The modeling subsystem accesses the database and gathers the information required by the calculation subsystem. Other processes may access the database as well. For instance, the automatic configuration of a modular robot in response to task requirements requires information about the

available modules. The database should be designed to facilitate the addition of new information.

As a first step, the design of this subsystem identifies the information that will be placed into the database. This information includes an identifier, such as a name, for each of the currently available modules. The information associated with each of the modules is determined primarily by the requirements of the calculation subsystem. All calculations require the geometry of the module. For torque and force calculations, the mass, location of the center of mass, and the inertia tensor must be specified relative to the local frames of each module. The calculation of deflections at the end-effector requires the compliance distribution as well. This information satisfies the current requirements of the calculation subsystem. There is also other information that may be important in the future. This information includes: gear ratios, speed limits, torque limits, thermal time constants, frequency response, and cost.

Even if there were hundreds of available modules and thousands of parameters associated with each module, this would not be a complex database application. For instance, the number of available modules will undoubtedly be much less than the number of names in the phone book. Because the amount of information in the database is relatively small, the data can reside in directly-addressable computer memory. This eliminates the need for repeatedly swapping data to and from the permanent storage device. Also, the information in the database will be updated infrequently. This prevents conflicts between processes accessing and updating the database. The resolution of these conflicts is a principle function of most database software. Because the demands are limited,

there are many options available for satisfying the database requirements. These options include: a simple database of lists, a hierarchical database, a relational database, and an object-oriented database.

A simple list is one example of a database. A list can be ordered, such as the residential pages of the phone book. A list can be indexed, such as the chapters in a book, or a list can be both ordered and indexed, such as the yellow pages in the phone book. The list-type of database is sufficient for the automatic plant description system application. The list associates parameters with each module. The classification of the parameters, such as geometric or inertial, indexes the list. The data for each module can simply be typed into a file. Since conflicts between processes updating and reading the database are not expected, the code for reading the database will be very straightforward. Finally, modules can be added to the database by creating new files and entering the appropriate data. New information can be associated with each module by simply adding the new information at the bottom of the list. It is important that new information be added to the end of the list, rather than in the middle, so as not to cause errors in previously written software.

A hierarchical database is slightly more complicated than a simple database of lists. The hierarchical database is particularly effective when the data can be arranged in a tree structure. Information can be manipulated at the highest level while the associated data beneath that level is handled automatically. This type of database could be used for the automatic plant description system application, though it is not particularly well-suited. This is because there is no hierarchy among the modules.

The most common database is the relational database. The relational database assigns references, dependencies, and commonalties to the information in the database. This type of database is useful when changes in one piece of information imply changes to information stored elsewhere. This database technique could be applied to a complete modular robot system. For instance, a given set of modules might all use the same actuator and therefore have the same torque capabilities. In a relational database, a dependency between the actuator and the torque capabilities could be established. Substituting a different actuator in the specification for the modules would automatically enact a change in the record for torque capabilities. Another example is that changes in material specifications might also relate to changes in stiffness. This level of sophistication is not, as yet, found in modular robots. Because the automatic plant description system is designed with the database as an independent subsystem, a relational database could be incorporated when warranted by sufficient advances in robotics technology.

Object-oriented databases are becoming increasingly popular. These databases adhere to the object-oriented programming paradigm (Zdonik and Maier, 1990). The data are considered objects and are categorized according to the class in which they belong. In object-oriented programming jargon, each datum object is called an instance of its class. Each class is defined by certain properties that apply to all of the data objects in the class. Combining old classes creates new classes. In the jargon, a new class created from old classes is said to inherit their properties. The object-oriented paradigm is useful for creating very flexible software modules that can be used in a variety of applications. The

flexibility is derived from a very careful definition of the classes and the inheritance structure. An object-oriented database is certainly capable of meeting the requirements of the automatic plant description system. At this time, however, the benefits do not warrant the additional complexity.

Table 5.4 Some factors associated with application of the four different database types as part of the automatic plant description system.		
Database	Positives	Negatives
list	software to create and manage is straightforward matches the modest demands of the automatic plant description system	changes in one part of the database don't automatically cause changes in other parts of the database
hierarchical	effective when data can be arranged in a tree structure data is manipulated at a high-level	database architecture doesn't match the modular robotics architecture
relational	changes in one part of the database automatically cause changes in other parts of the database may have future application in the automatic plant description system	software for implementation is complex
object-oriented	expansion of the database is easy	software for implementation is complex "overkill" for the automatic plant description system

Each of these database techniques is able to satisfy the database requirements of the automatic plant description system. This is because the automatic plant description system is a very simple database application with a relatively small amount of infrequently-updated information. Of these database

techniques, the relational database and the indexed list are most applicable. A relational database is somewhat complex and implementation will require either purchasing software or a significant development effort. An indexed list is a database technique that provides a good match with the automatic plant description system's database requirements. Data for the indexed list can be entered directly with any editor. Portable software to read the data can be written with minimal effort.

5.3.3 Model Formulation Subsystem

The model formulation subsystem is the heart of the automatic plant description system. This subsystem's input is a description of the robot. This description may be of either a modular or monolithic robot. Using the robot's description and information in the database, this subsystem assembles the parameters and formulates a model of the robot. The model of the robot is generated and transferred to the calculation subsystem for analysis. The model must be expressed in either a modular or Denavit and Hartenberg format depending upon the analysis software. There are four combinations of input and output formats: a modular robot with module-based analysis, a modular robot with Denavit and Hartenberg-based analysis, a monolithic robot with module-based analysis, and a monolithic robot with Denavit and Hartenberg-based analysis.

The input of the modeling subsystem is a description of either a monolithic or a modular robot. Denavit and Hartenberg parameters are the standard method of describing the geometry of a monolithic robot. The defacto

standard for specifying a modular robot configuration (within the University of Texas Robotics Research Group) is the robo_cad file (Hooper and Tesar, 1990). Robo_cad files specify the robot's modules, their sequence, and their connection geometry. These files are created interactively by choosing and scaling modules through the robo_cad graphical user interface. Robo_cad files have also been written by optimization routines configuring a limited number of modules in response to simple task priorities (Ambrose and Tesar, 1992). In any event, a description of the robot geometry is available via Denavit and Hartenberg parameters for monolithic robots or as robo_cad files for modular robots.

Table 5.5 The combinations of input and output formats possible in the modeling subsystem.	
Input	Output
Modular	Modular
Modular	Denavit and Hartenberg
Denavit and Hartenberg	Modular
Denavit and Hartenberg	Denavit and Hartenberg

The format for the output of the modeling subsystem is determined by the kinematic and dynamic analysis software. Again, two basic methods have been identified, the module-based approach and the Denavit and Hartenberg-based approach. For both cases, the available kinematic and dynamic analysis software packages require that the model be communicated via a datafile. Communication of the robot-specific inputs via datafiles has also been suggested as a standard during the course of this design. Communication via datafiles facilitates

functional independence and modularity of the software and is ideally suited for automatic downloading of system parameters via standard file transfer protocols.

A modular robot description is inherently compatible with a module-based analysis package. The input is a robo_cad file and the output file is written in the module-based manipulator analysis package format. The modeling subsystem reads the robo_cad file to identify the sequence of modules and the geometry of the connections. It then accesses the database to acquire the parameters associated with each module. Finally, each module's identifier along with the corresponding geometry and mass distribution is written sequentially into a datafile. The datafile forms the input to the module-based manipulator analysis package.

The most challenging combination of input and output for the modeling subsystem is the modular robot description with the Denavit and Hartenberg-based analysis software. Addressing this combination requires extracting the Denavit and Hartenberg parameters from the modular robot description. There are several motivations for developing this capability. It will allow the NDOF torque and compliance software to be used with a modular robot. The performance criteria software is also based on a Denavit and Hartenberg description of the robot. Finally, the bulk of manipulator theory uses a Denavit and Hartenberg-based description of the robot's geometry. Other research institutions have developed procedures for extracting Denavit and Hartenberg parameters from a modular robot description. The most notable of them are from Carnegie Mellon University by Kelmar and Khosla (1990), and the University of Toronto by Benhabib (1989). Though the procedures seem to work for simple

module geometries and constrained connection methods, the development is lacking in generality. A brief discussion of these extraction procedures will be followed by the derivation of a general procedure. This derivation is a specific development of this dissertation.

Carnegie Mellon University's procedure for extracting Denavit and Hartenberg parameters from a modular robot geometry relies on very simple module geometries with constrained connection methods. The links are designed as a single linear translation along the Z -axis. The rotational joints only allow rotation about the Z axis and translations along the Y and Z axes. A twist about the Z axis specifies the connections between modules.

Benhabib's procedure for extracting the Denavit and Hartenberg parameters is inefficient because it generates two sequential Denavit and Hartenberg transformations, rather than one, for each joint axis. Also, the extraction method is only valid for modular robots with strictly Denavit and Hartenberg geometries. That is, the modular geometry must be expressed as $\text{Screw}_x(a_{i-1}, \alpha_{i-1})$ then $\text{Screw}_z(d_i, \Phi_i)$.

The methodology of both of these extraction procedures is essentially the same. By restricting the module and the connection geometry, the geometric problem of extracting parameters from general homogeneous transformations is reduced to simply adding parameters together as modules are added to the robot chain. Though this procedure is valid for simple cases, there are reasons for preferring a more general approach. For instance, a metrology effort may find that a joint axis for a specific module is skewed by some amount from the nominal design. The above-mentioned extraction procedures are invalid in this

case. Also, the previous extraction procedures constrain the design of the modules as well as the configuration possibilities for the modular robot. This is contrary to the goal of designing the modules based on engineering principles and configuring them to best perform a given task. Clearly, a general extraction procedure is superior.

In the general case, each link module may be represented by a single homogeneous transform from its input to its output. Each joint may be represented by one homogeneous transform from its input to its axis of motion and by another homogeneous transform from its axis of motion to its output. Concatenating all the transforms between the two joint axes generates a single homogeneous transform from one axis to the next. Without loss of generality, two local reference frames, $\{i-1\}$ and $\{i\}$, are defined with their Z axes coinciding with the joint axes. The Denavit and Hartenberg parameters to be extracted from the homogeneous transform between these two successive joint axes, $\hat{Z}_{i-1}$ and $\hat{Z}_i$, are defined as follows

α_{i-1} = The angle between $\hat{Z}_{i-1}$ and $\hat{Z}_i$ measured in a right-hand sense about their mutual perpendicular, $\hat{x}_{i-1}$.

a_{i-1} = The distance along $\hat{x}_{i-1}$ between $\hat{Z}_{i-1}$ and $\hat{Z}_i$.

d_i = The distance along $\hat{Z}_i$ between $\hat{x}_{i-1}$ and $\hat{x}_i$.

Φ_i = The angle between $\hat{x}_{i-1}$ and $\hat{x}_i$ measured in a right-hand sense about $\hat{Z}_i$.

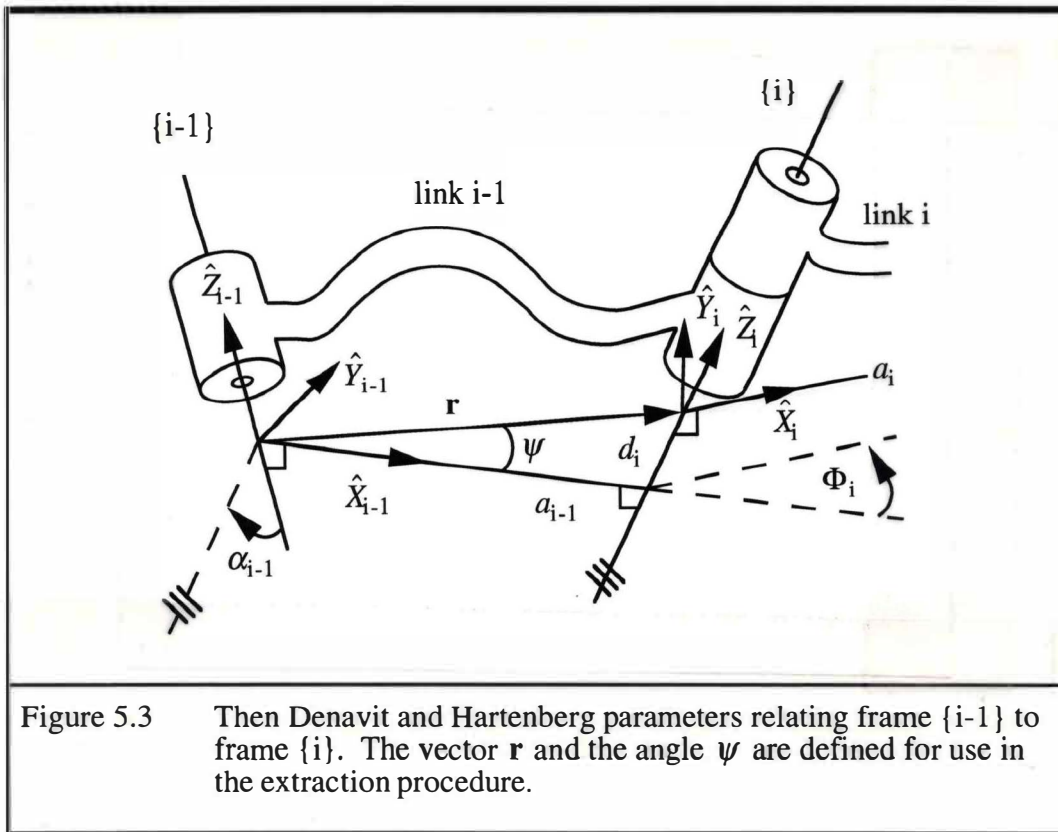


Figure 5.3 illustrates these quantities. The angle between $\hat{Z}_{i-1}$ and $\hat{Z}_i$ is α_{i-1} , which can be found using the inner product.

$$\cos \alpha_{i-1} = \hat{Z}_{i-1} \cdot \hat{Z}_i \quad (5.1)$$

$$\alpha_{i-1} = \cos^{-1}(\hat{Z}_{i-1} \cdot \hat{Z}_i) \quad (5.2)$$

Though the inner product gives the magnitude of α_{i-1} , there is an ambiguity concerning the direction. The ambiguity is because there are two solutions for Θ in the equation

$$\cos \Theta = b \quad (5.3)$$

which are

$$\Theta = \cos^{-1}(b) \quad (5.4)$$

and

$$\Theta = -\cos^{-1}(b) \quad (5.5)$$

The Atan2 function will resolve the ambiguity. Lipkin and Duffy (1985) suggest

$$\cos \alpha_{i-1} = \hat{Z}_{i-1} \bullet \hat{Z}_i \quad (5.6)$$

and

$$\sin \alpha_{i-1} = \hat{Z}_{i-1} \times \hat{Z}_i \bullet \hat{x}_{i-1} \quad (5.7)$$

then

$$\alpha_{i-1} = \text{A tan 2}(\sin \alpha_{i-1}, \cos \alpha_{i-1}) \quad (5.8)$$

There are some problems with the Atan2 function. For instance, the solution to $\text{A tan 2}(0,0)$ is ambiguous and may vary among computer platforms. Assuming a sign and then performing the rotation by the assumed α_{i-1} as a check will also resolve the ambiguity in Equation (5.2). If the $\hat{Z}_{i-1}$ and $\hat{Z}_i$ axes are in the same direction, the assumption was correct. If they are in opposite directions, the assumed α_{i-1} is multiplied by -1 to produce the correct direction. Given that a modern computer can check this assumption in a small fraction of a second, using the assume and check method may be preferable. In any case, the remainder of this derivation assumes that similar ambiguities in direction are resolved either by the A tan 2 function or by checking.

Figure 5.3 defines a vector $\mathbf{r}$ to find the two lengths, a_{i-1} and d_i . The vector $\mathbf{r}$ appears explicitly as the translational portion of the homogeneous transform between the frames $\{i-1\}$ and $\{i\}$. The angle, ψ , defines the angle

between $\hat{x}_{i-1}$ and $\mathbf{r}$. The unit vector, $\hat{x}_{i-1}$, defines the line perpendicular to both $\hat{Z}_{i-1}$ and $\hat{Z}_i$.

$$\hat{x}_{i-1} = \hat{Z}_{i-1} \times \hat{Z}_i \quad (5.9)$$

The angle ψ is then found as

$$\cos \psi = \frac{\mathbf{r} \bullet \hat{x}_{i-1}}{|\mathbf{r}|} \quad (5.10)$$

$$\psi = \cos^{-1} \left(\frac{\mathbf{r} \bullet \hat{x}_{i-1}}{|\mathbf{r}|} \right) \quad (5.11)$$

The lengths a_{i-1} and d_i are

$$a_{i-1} = |\mathbf{r}| \cos \psi \quad (5.12)$$

and

$$d_i = |\mathbf{r}| \sin \psi \quad (5.13)$$

The joint angle, Φ_i , (the angle between $\hat{x}_{i-1}$ and $\hat{x}_i$) is the final Denavit and Hartenberg parameter to be extracted. The angle Φ_i is found as

$$\cos \Phi_i = \hat{x}_{i-1} \bullet \hat{x}_i \quad (5.14)$$

$$\Phi_i = \cos^{-1} (\hat{x}_{i-1} \bullet \hat{x}_i) \quad (5.15)$$

This procedure for extracting the four Denavit and Hartenberg parameters from a homogeneous transform matrix is general and allows the NDOF analysis software and the performance criteria software to be used with a modular robot description.

Another combination of robot description and analysis software is the monolithic robot and the Denavit and Hartenberg-based analysis software. With this combination, the modeling subsystem simply queries the user for the pertinent information. Once the modeling subsystem has obtained the necessary data, the subsystem simply writes the information into a datafile. The datafile can

be saved in the database for future use. Eventually the database will include a library of monolithic robots.

The final combination is the monolithic robot and the module-based analysis software. This is, however, an unreasonable combination. The monolithic robot does not have interfaces that allow it to be decomposed into a set of joint and link modules. The automatic plant description system must use the Denavit and Hartenberg-based analysis software to simulate a monolithic robot.

5.4 IMPLEMENTATION

A working version of the automatic plant description system described in this chapter was written in the "C" programming language and implemented on a Unix computing platform, though it will run on most modern computers. As with any design, implementation of the automatic plant description system required choosing from among several options. Options were identified for each of the three main subsystems: calculation, model formulation, and information storage (database). This discussion focuses on the options that were chosen and the reasoning behind the choices.

The main result in the implementation of the modeling subsystem was the procedure for extracting the Denavit and Hartenberg parameters from a homogenous transformation matrix. The translation of this procedure into computer code was not difficult. The extraction process for a robot with seven degrees of freedom runs on a Silicon Graphics Indigo R4000 in less than one second. Other than this extraction procedure, the modeling subsystem simply must read and write datafiles.

Four types of databases were identified as options for use in the automatic plant description system (list, hierarchical, relational, and object-oriented). Of these options, the list structure was chosen for implementation. There were several reasons for making the choice. The simple list structure matches the very modest demands of the automatic plant description. There will be a relatively small amount of data which can easily be handled by a list structure. Also, because the database will be updated infrequently, managing conflicts between processes simultaneously attempting to read from and write to the database is not difficult. The list database can easily be updated to incorporate new information associated with the existing modules and also allows the addition of new modules. Finally, the software to manage the list database could be written entirely in-house and with only a modest effort.

Four different software packages were identified for use in the analysis subsystem of the automatic plant description system. These software packages are: the NDOF joint force and torque calculation software, the NDOF end-effector deflection calculation software, the performance criteria calculation software, and the module-based manipulator analysis package. The first three of these software packages use a Denavit and Hartenberg-based description of the robot geometry. The module-based manipulator analysis package generates the robot geometry from the sequence of known modules. All of these software packages were addressed in the implementation. The crucial factor is that the modeling subsystem communicates the robot-specific inputs to the calculation subsystem via datafiles. Communication of the robot-specific inputs via datafiles facilitates functional independence and modularity of the software. Data files are also

ideally suited for automatic downloading of system parameters to specialized and real-time processors via standard file transfer protocols.

5.5 EXAMPLE

This section presents an example of using the Automatic Plant Description System for a modular robot with seven degrees of freedom. This example shows the steps to the user for running the appropriate software and entering the necessary parameters. There are essentially three distinct steps:

1. Assemble the robot graphically using the **robo_cad** software,
2. Extract the D. H. parameters using the **extract** program,
3. Run the dynamic modeling software (**dyn** in this example).

This example also includes representative plots of the output data and run times for an Analogic AP500 array processor and an Indigo personal workstation.

Robo_cad is a copyrighted, interactive assembly program for modular and reconfigurable robots. Through a graphical user interface, the program allows the operator to assemble a modular robot on the computer screen and then immediately animate the model thus providing an impression of the simulated robot's dexterity. The program is based on a set of palettes, each containing a variety of one, two, and three degree of freedom joint modules and generic links. By selecting from among these modules and supplying the appropriate parameters, the operator assembles the modular robot.

After running the robo_cad program (simply type "robo_cad"), choose a base for the robot. This is accomplished by selecting "add base" from the "build" menu. The program will display the palette of base modules. This example uses

the slanted mounting pedestal (Figure 5.4). Choose the color for the module and then click in the “O.K.” box, and the slanted mounting pedestal appears on the computer screen.

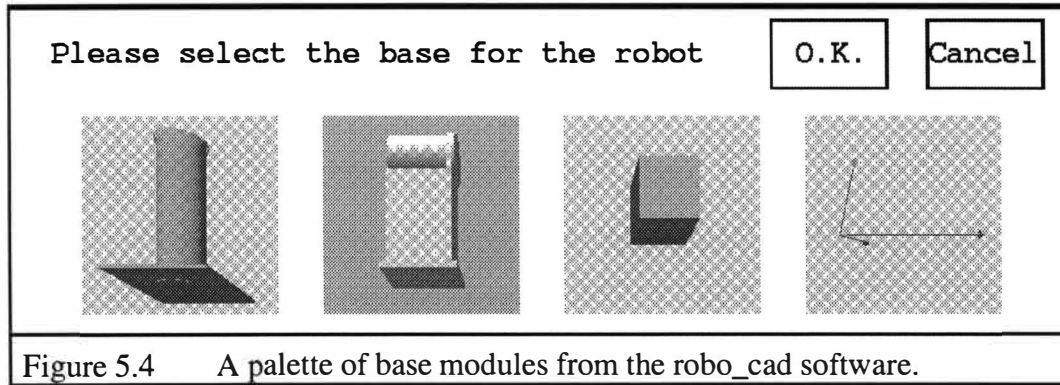


Figure 5.4 A palette of base modules from the robo_cad software.

The next step is to choose a joint for the robot. Select “add joint” from the build menu and the palette of joint modules appears on the screen. Choose a pivot module and click in the “O.K.” box. A dialogue box requiring several input parameters will appear on the screen (Figure 5.5). Specify a length for the pivot joint, a twist angle, and the color. Then click in the “O.K.” box and the pivot module appears on the screen connected to the base module.

Continue selecting and adding modules until the simulated robot is complete. There is essentially no limit as to the number of modules that may be added to the model, though this example gives timing results for a robot with seven degrees of freedom. After the robot is complete, save the model by choosing “save” from the “file” menu. The program will ask for a name for the model. Any name is fine. Simply type the name into the dialogue box.

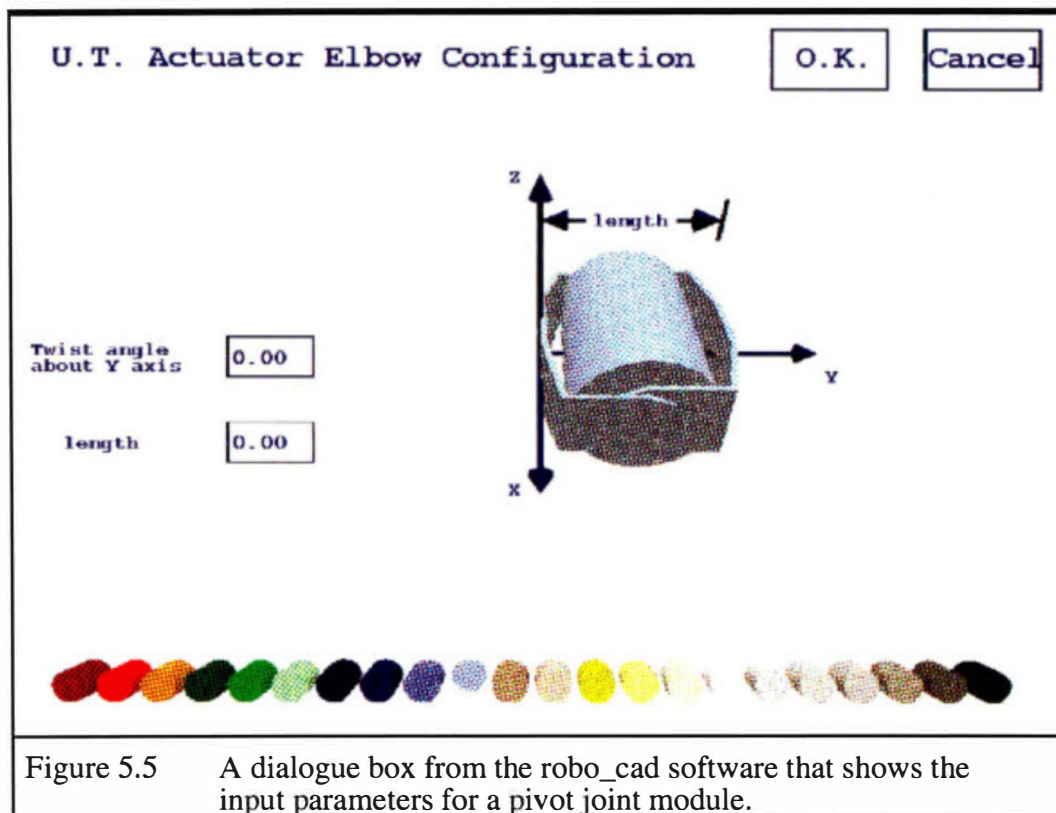


Figure 5.5 A dialogue box from the robo_cad software that shows the input parameters for a pivot joint module.

The next step in using the Automatic Plant Description System is to extract the Denavit and Hartenberg parameters using the **extract** program. This program is essentially the extraction procedure given in section 5.3.3 translated into computer code. It also includes a simple protocol to access a database for the physical parameters associated with each module. To run this program, type “extract”. The program will ask for the name of the robo_cad file and for the name of a file for the D. H. and physical parameters. After entering these file names, the program runs without any further intervention. For a robot with seven degrees of freedom, the extraction procedure takes less than one second on an Indigo computer.

Once the Denavit and Hartenberg parameters are available, the next step is to run the dynamic modeling software. Running this software requires one component that is not part of the Automatic Plant Description System, and that is a path for the robot. The path for the robot may come from an off-line programming environment, or it may be part of a real-time robot control system. For the off-line operation, the **dyn** program will generate the joint torques and end-effector deflections. Figure 5.6 shows the joint torques and the end-effector deflections calculated by **dyn** for a representative path. On an Indigo R4000 personal workstation, the joint torques and end-effector deflections together take 2.4 milliseconds to calculate for a robot with seven degrees of freedom. For real-time operation, the Analogic AP500 is currently the choice for a processor. This array processor will calculate the joint torques and end-effector deflections in 3.7 milliseconds for a robot with seven degrees of freedom.

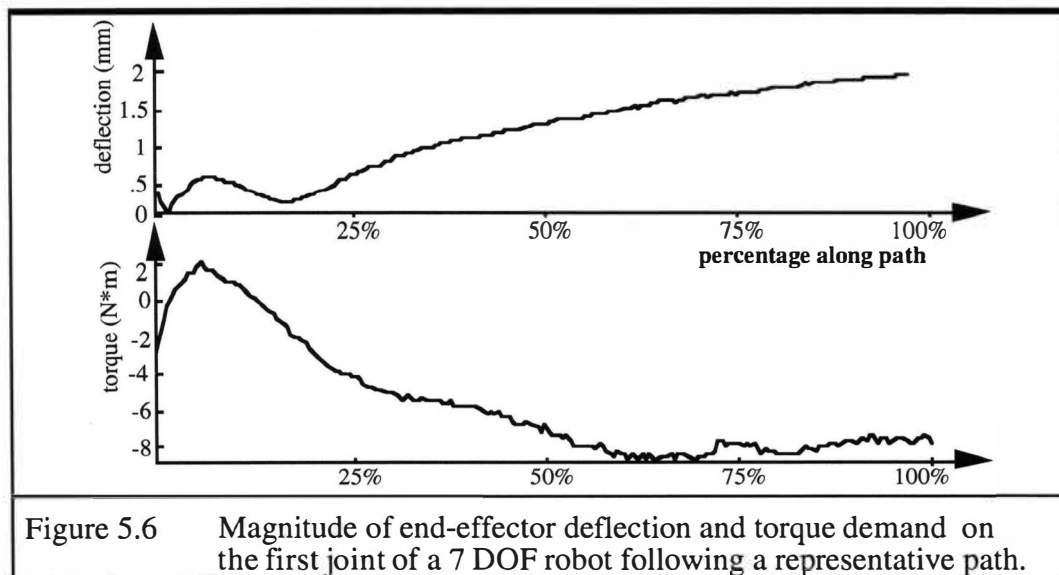


Figure 5.6 Magnitude of end-effector deflection and torque demand on the first joint of a 7 DOF robot following a representative path.

Chapter 6: Experimental and Simulation Results

Experimentation and simulation verified the effectiveness of the direct search inverse kinematics method for several different robots. The robots in these examples have especially difficult or interesting geometries. They cover a spectrum, from the industrial variety with six degrees of freedom to a conceptual hyper-redundant robot with twenty-one degrees of freedom. Most of the examples include a graph showing the solution's accuracy and timing results for the speed.

All numerical methods find solutions with a finite accuracy. For direct search, the step size for the perturbations determines the accuracy of the solution. A smaller step size results in a more accurate solution. The step size also influences the solution speed. A larger step size increases the solution speed. Clearly there is a trade-off between solution speed and accuracy. The resolution of an industrial robot determined the acceptable accuracy, and thus the step size, for the examples presented in this chapter. A good resolution for an industrial robot is .000254 meters (.001 inches) (Tesar and Butler, 1989a). The accuracy of the direct search solution corresponds to the resolution of the robot if there are no errors other than those introduced by the inverse kinematics method. An accuracy at least ten times better than the resolution of the robot ensures the direct search inverse kinematics solution has little effect on the overall resolution of the robot. An accuracy of .0000254 meters (.0001 inches) should therefore be sufficient.

Solution speed is a practical consideration for inverse kinematics algorithms. This chapter quotes speeds for algorithms executing on a Silicon Graphics R4000 personal workstation. This computer has a single reduced instruction set microprocessor operating at a frequency of fifty-million Hertz. The computer's numerical performance is sixteen MFLOPS (Gorey, 1992). All the inverse kinematics software discussed in this chapter was written in the "C" programming language and is completely portable.

Computer animation played an important part in the development of the direct search inverse kinematics method. This data visualization technique shows the complex kinematic coupling between the robot's joints in a particularly intuitive format.

6.1 ROBOTICS RESEARCH KB2107

This experiment tested the performance of a direct search inverse kinematics algorithm in both simulation and in application to an actual robot. Figure 6.1 shows a photograph of the robot, a Robotics Research kb2107. The experiment determined if application would expose any problems with the direct search technique that were not apparent in simulation. A direct search algorithm generated joint angles off-line as a simulated robot followed several different paths. The actual robot was then made to follow these paths by successively sending the joint angles to the robot's controller. The robot's controller measured and recorded the actual joint angles and joint torques as the robot followed the path. The difference between the desired and the actual joint angles shows how closely the robot was able to follow the path. Because there was no end-effector

sensing, this comparison is valid only at the joint level. The measured torques and several geometric performance criteria are compared along four different paths. These paths were chosen specifically to be difficult for the robot; with two of them being performed at its maximum payload.

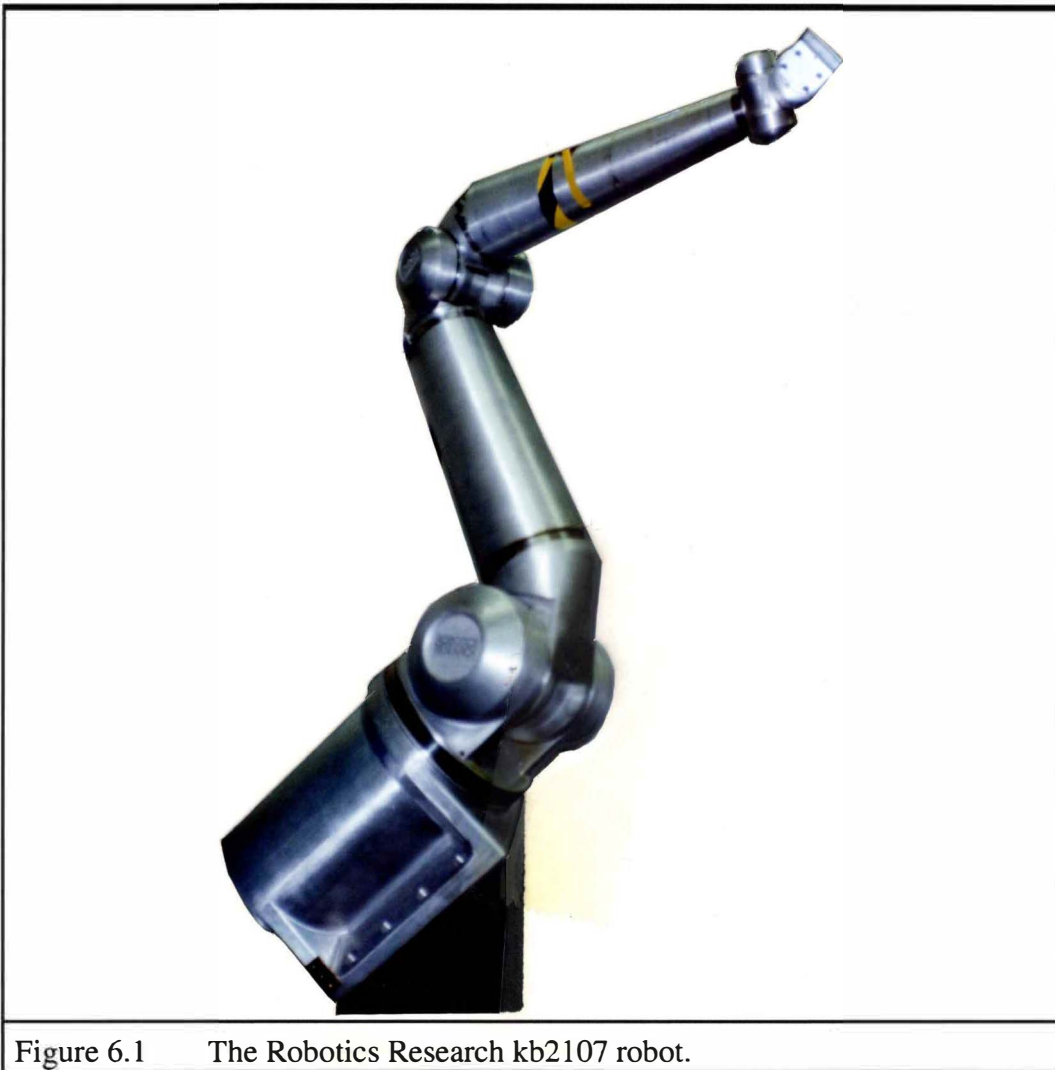


Figure 6.1 The Robotics Research kb2107 robot.

There were several reasons for performing this experiment. The first was to test the numerical performance of the algorithm as to the accuracy and speed of the solution. The second reason for performing the experiment was to bring to light any problems with the direct search solution which were not apparent during simulation. Frequency content that might cause an actual robot to resonate was considered to be a possible problem. Another possible problem concerned unrealistic demands being placed on the robot's controller and actuators. The final goal for this experiment was to measure and record the joint torques as the robot followed the paths. These joint torques were then to be compared with various performance criteria values along the paths.

The direct search algorithm employed an exhaustive exploration pattern to generate the trial solutions. A steepest descent decision making strategy chose a single solution from among the set of trial solutions. As has been discussed, this combination of exploration and decision making is guaranteed to find a local minimum of an unconstrained objective function. Two different objective functions were used, each of which contained a term generated by transforming the Cartesian error into a performance criterion. The difference was in the secondary performance criterion which were added to the objective function. In the first case, the two-norm of the joint speeds was combined with the Cartesian error to form the objective function. In the second case, the two-norm of the change in joint speeds with respect to time was combined with the Cartesian error to form the objective function. In both cases, the direct search was performed to a joint-level resolution of .001 degrees. All scaling factors were derived using the search-based scaling procedure described in Chapter 4. As just described, the

direct search algorithm's solution speed on a Silicon Graphics Indigo R4000 personal workstation was about .6 Hertz. This speed is about fifty times too slow for real-time application. Using one of the suboptimal exploration strategies would significantly increase the speed. For example, Section 6.7 describes results for a robot with nine degrees of freedom using the simple exploration strategy. The solution speed in this example is about ten Hertz. A faster computer would also increase the solution speed.

Table 6.1 Search parameters and results for the kb2107 experiment.	
exploration pattern -	exhaustive
decision making strategy -	steepest descent
objective function (two cases) -	Cartesian error and two-norm of joint speeds Cartesian error and two-norm of rate of change of joint speeds
search resolution -	.001 degrees
scaling method -	global
solution speed -	.6 Hertz
maximum translational error -	.000018 meters (.00071 inches)
maximum rotational error -	.000029 radians
degrees of freedom -	7

This experiment used a Robotics Research Corporation kb2107 robot. It is a research robot with seven rotational joints. The robot is rather long at 2.1 meters and has an extremely low maximum payload of five pounds. Nonetheless, the robot's redundancy, instrumentation, and versatile computer interface allow it to be used for testing multicriteria inverse kinematics algorithms. NASA/Johnson

Space Center was gracious enough to provide the robot and the facilities for this experiment.

A discrete path was described at the joint-level by generating multiple sets of joint angles off-line. Every five milliseconds, these angle sets were communicated to the robot's controller as displacement commands at the joint level. Two and one-half milliseconds after each angle set was sent, the joint angles and joint torques were measured and recorded. The joint angles are measured at the motor, rather than at the output of the joint. Unfortunately, this robot employs harmonic drives as speed reducers within the joints. Because harmonic drives are very compliant, and they are between the motor and the output of the joint, there can be significant deviation between the measured and the actual joint angles. This severely limits the usefulness of these measurements. Only an impression of how well the motor and controller were able to track the commanded joint displacements can be obtained. The joint torques, however, are measured with strain gauge bridges at the outputs of each joint. This provides a valid measurement of the torque at the joints.

Two different end-effector paths were used along with the two different objective functions described above to generate four different paths for the robot to follow. The first end-effector path was a figure-8 performed in a vertical plane. This path was chosen because it is kinematically challenging for the robot. The second path was a simple lifting motion performed directly against gravity. Though not challenging kinematically, a five pound weight (the maximum payload of the kb2107) was attached to the robot's end-effector while it followed this path.

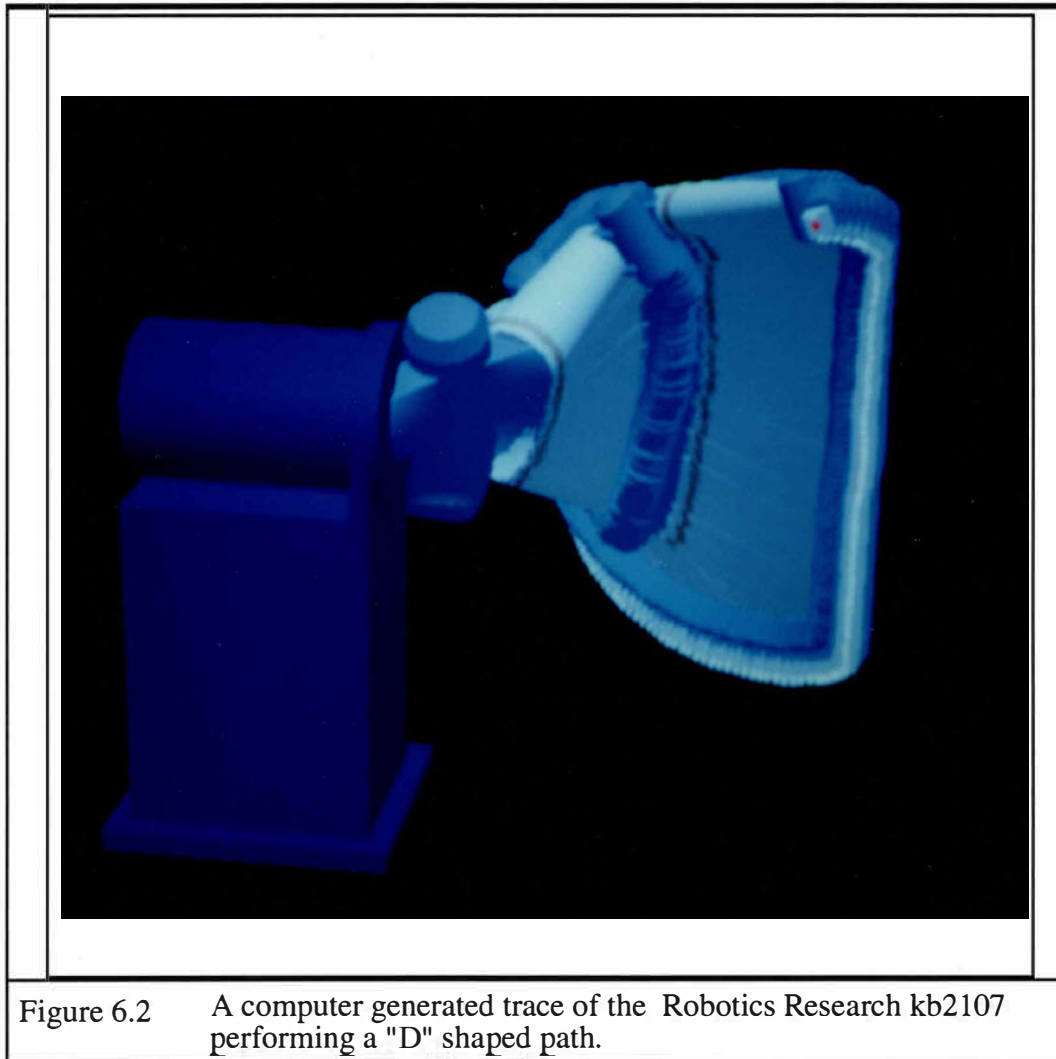
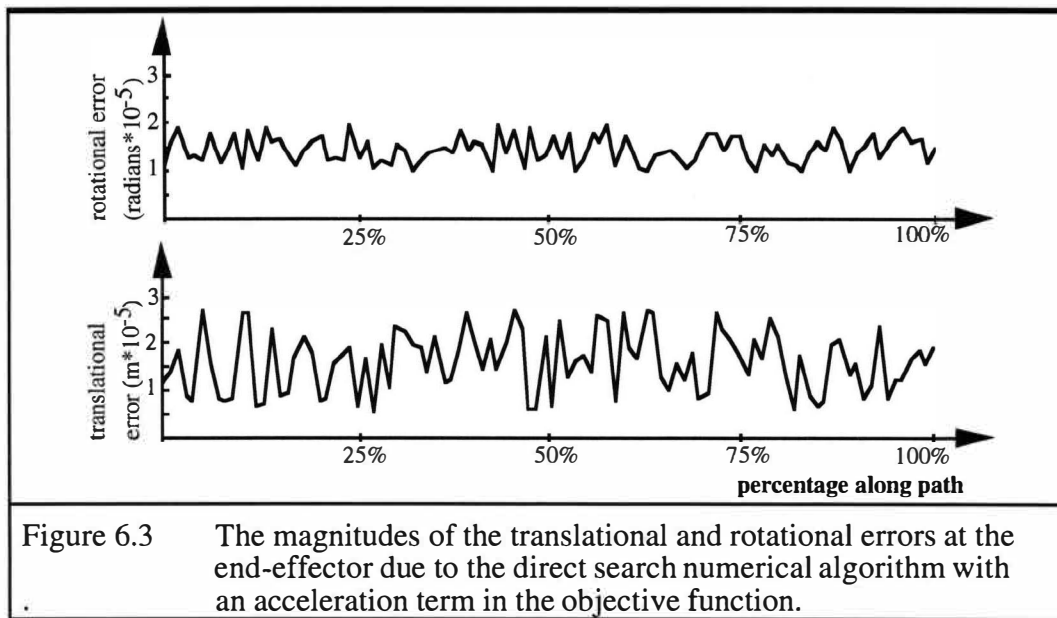
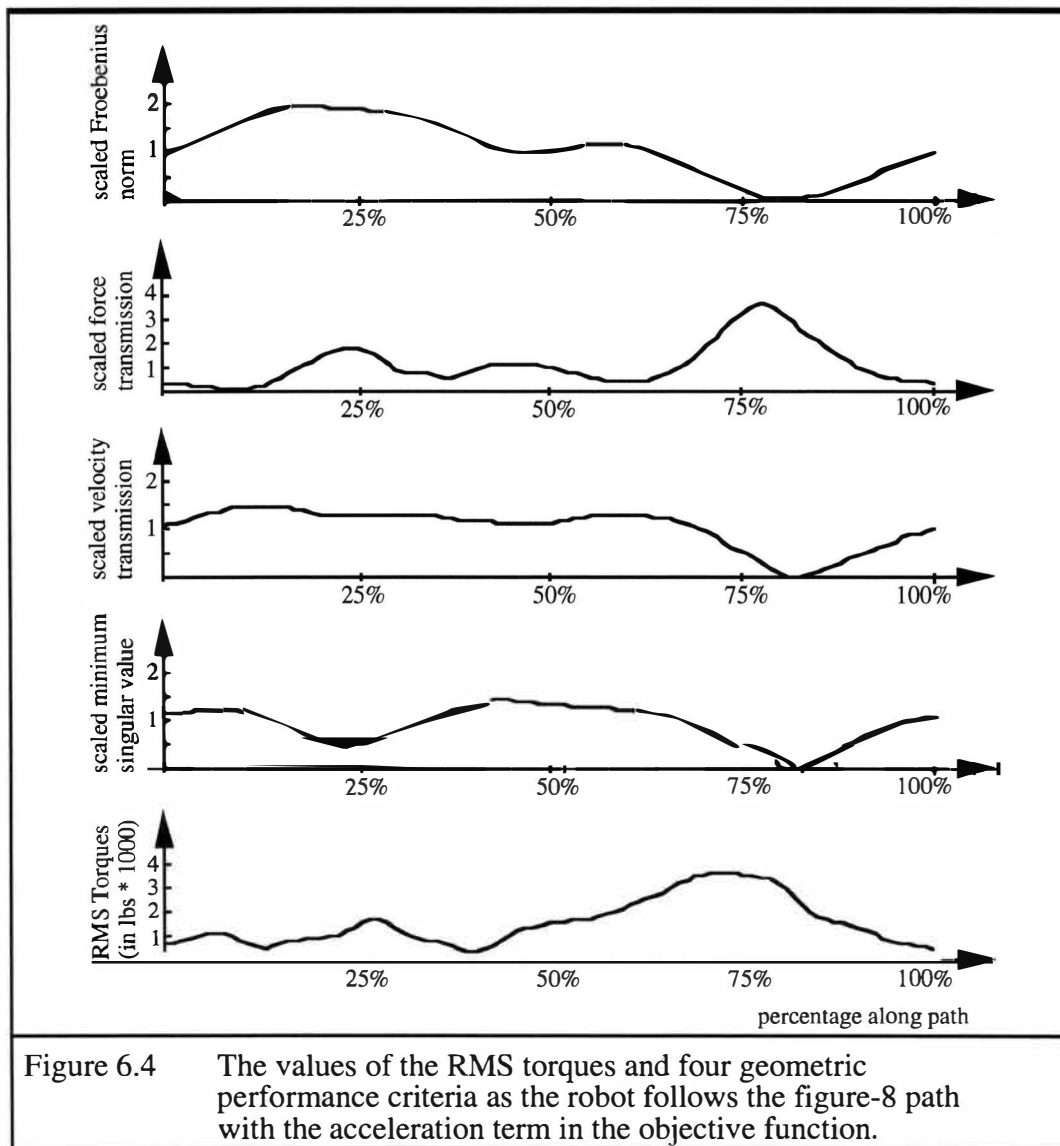


Figure 6.2 A computer generated trace of the Robotics Research kb2107 performing a "D" shaped path.

Figure 6.2 shows a computer generated trace of the robot as it performs a "D" shaped path with the acceleration term in the objective function. Figure 6.3 graphs the numerical accuracy of the solution as the robot follows the path. The maximum translational error is .000027 meters (.00011 inches) and the maximum rotational error is .000019 radians.



In this example, the resolution of the direct search algorithm is at least an order of magnitude better than the resolution of the robot. Thus, the error due to the direct search will have little effect on the overall resolution of the system. The average difference between the measured and the commanded joint angles was less than .8 percent. This shows that the control system was able to track the input but, as has been discussed, does not measure the actual displacements at the outputs of the joints. Measurement of the joint torques showed that no actuators reached saturation. Figure 6.4 graphs the two-norm of the joint torques along this path. The peak value is 3,900 inch-pounds and the average value is 2,173 inch-pounds. For comparison, four different performance criteria are graphed above the torques in the same figure.



The scaled minimum singular value of the Jacobian is a singularity detection criterion that decreases as the robot approaches a singularity. A larger value of the velocity transformation criterion indicates increased transformation of joint speeds to end-effector velocities. A larger value of the force transmission

criterion indicates increased transmission of joint torques to end-effector forces and torques. The scaled Frobenius norm measures the size of the Jacobian. A complete physical interpretation of these, and the rest of the performance criteria, is very complex and the subject of ongoing research. Though there does seem to be some relationship between the criteria and the torque values, it is not necessarily true that optimizing any of these criteria would reduce torque demands.

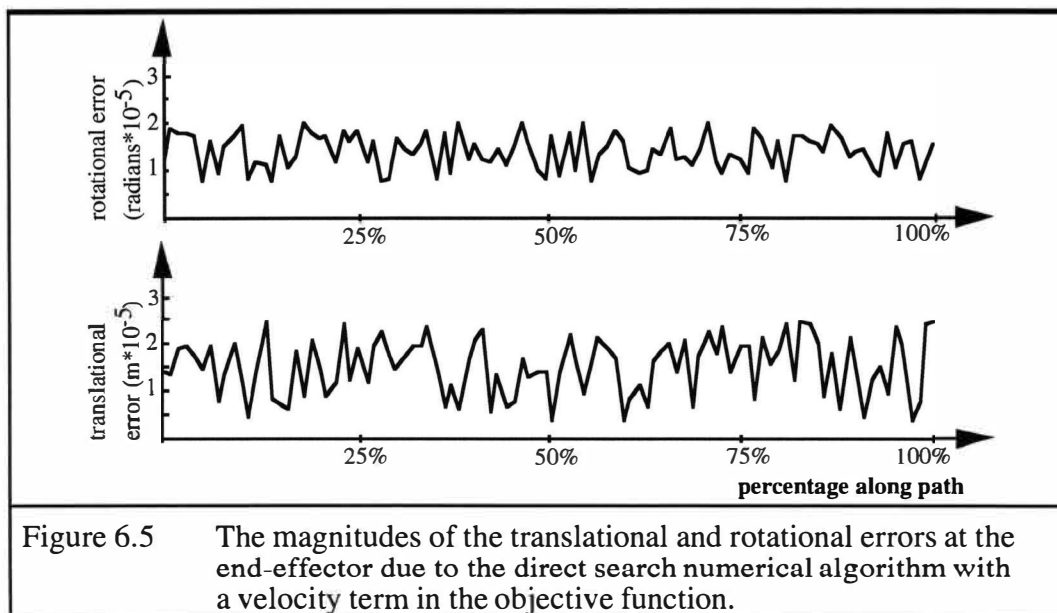
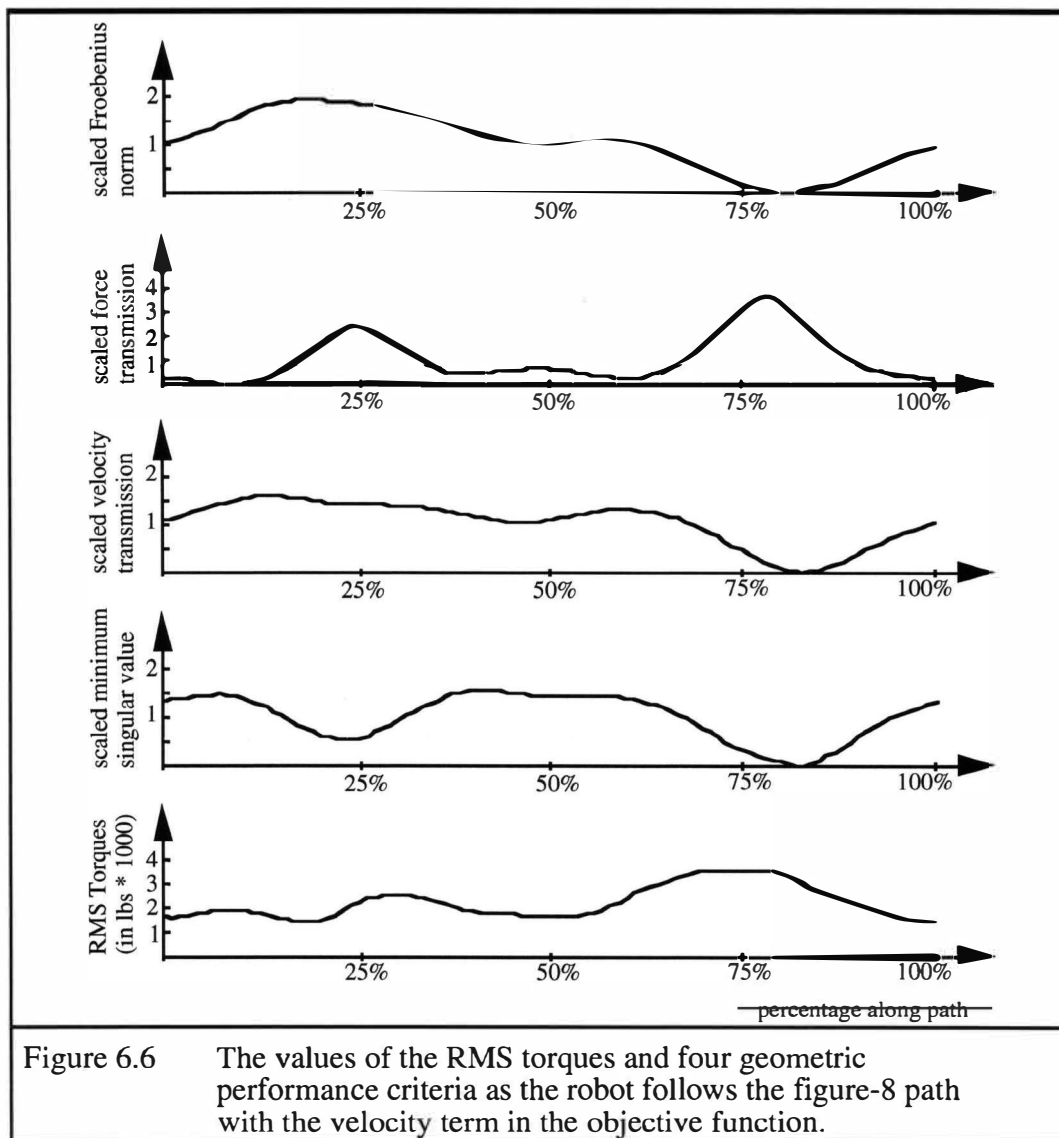
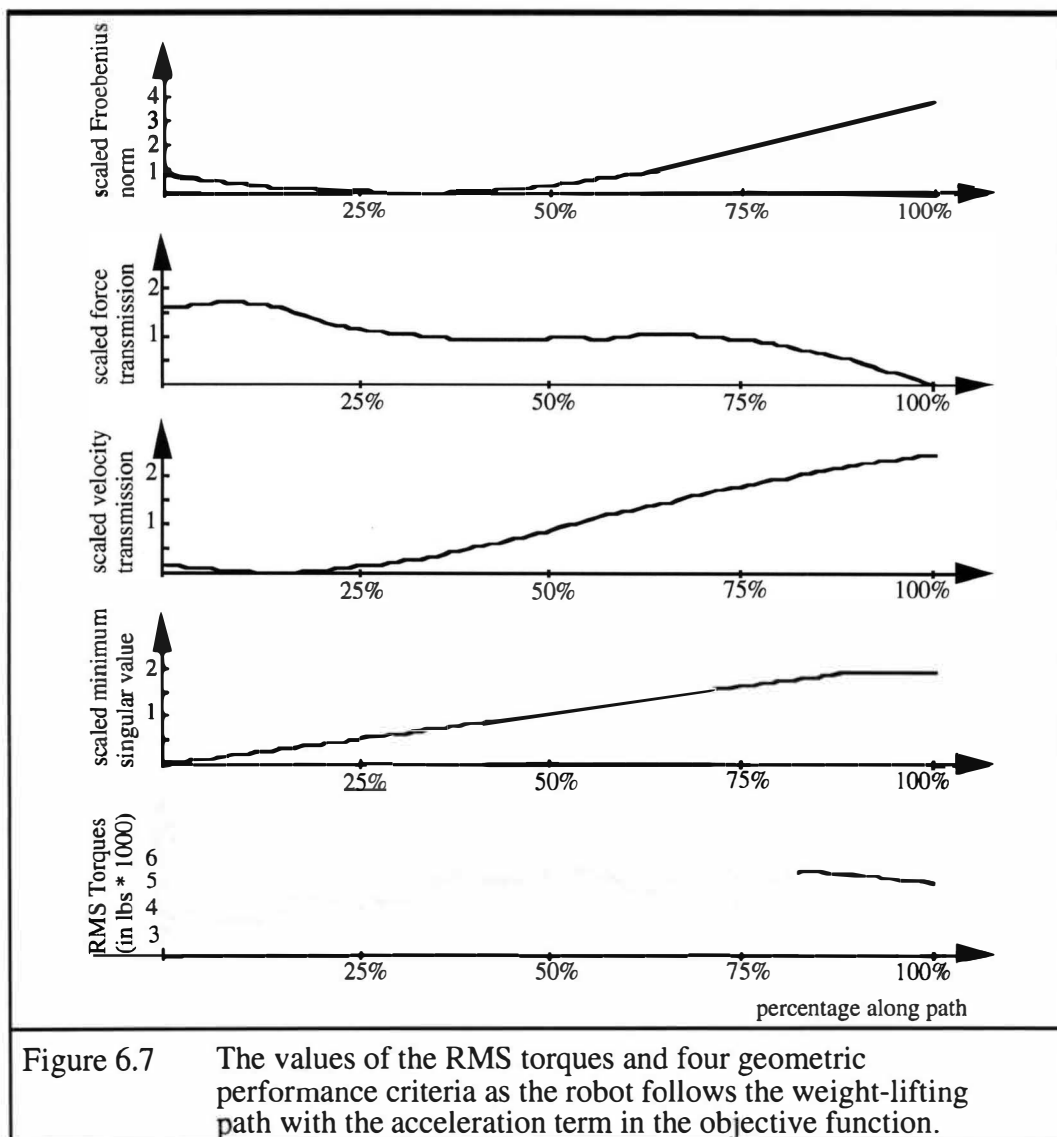


Figure 6.5 shows the error of the direct search as the robot follows the same end-effector path, but with a velocity term in the objective function. The maximum translational error is .000025 meters (.00098 inches) and the maximum rotational error is .000021 radians. Again, this numerical error is at least an order of magnitude lower than the resolution of the robot.



The average difference between the measured and the commanded joint angles was again less than .8 percent. The torque measurements showed that no actuator was forced into saturation while the robot followed the path. Figure 6.6 graphs the two-norm of the joint torques as the robot followed the path. The

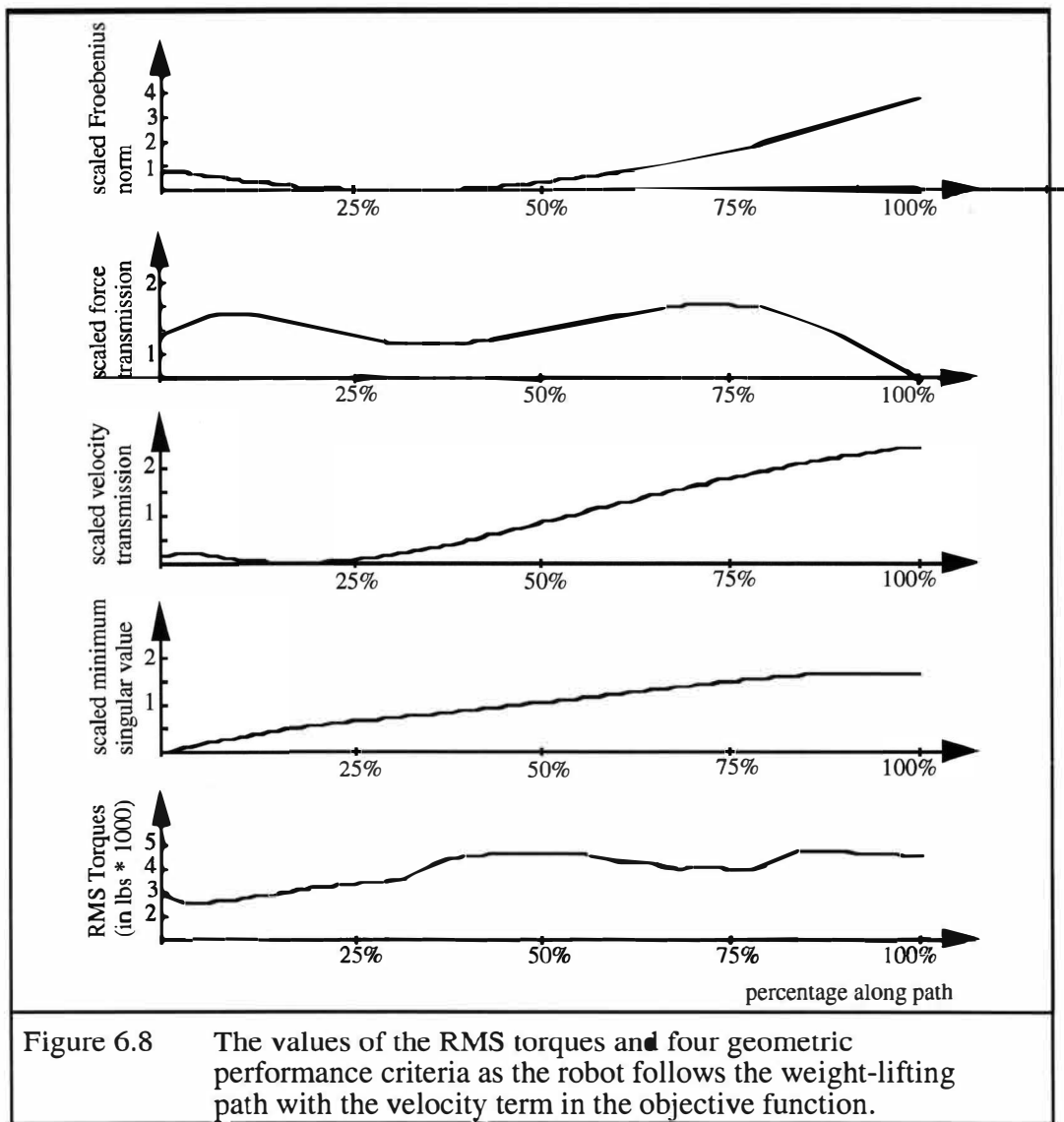
profile of the torques is very similar to that of the previous path, but the peak and average torques are somewhat lower at 3,500 and 2,160 inch-pounds respectively. The same four performance criteria are graphed above the torques in the same figure. The profiles of these performance criteria values are very similar to those of the figure-8 path with the acceleration term in the objective function.



The second end-effector path was a simple straight-line motion directly against gravity. Kinematically, it was much less challenging. However, the robot's maximum payload of 5 pounds was attached to the end-effector to increase the torque demands on the actuators. The first run was with the acceleration term in the objective function. Measurement of the joint torques showed that no actuators reached saturation. Figure 6.7 graphs the two-norm of the joint torques for this run. As before, the four performance criteria are graphed above the torques. For this weight-lifting path, there is much less correspondence between the performance criteria and the joint torques than there was for the figure-8 path.

The final test in this series of experiments is the weight-lifting path with the joint velocity term in the objective function. Measurement of the joint torques showed that no actuators reached saturation. Figure 6.8 shows the two-norm of the joint torques and the four performance criteria values along this path.

There are several results worth noting from this series of experiments. The first is that for each of these paths the direct search algorithm was able to solve the inverse kinematics problem to a resolution that is sufficient for modern robots. The second observation is that the control system was able to track the commanded inputs very closely without saturating the actuators. The third observation is that the paths with the velocity term in the objective function were somewhat better in terms of the smoothness and magnitudes of the joint torques. The final observation is that the criteria values correlated with the torque demands on the actuators much better for the figure-8 path than for the straight-line path.



The difference in correlation is likely because of the relative complexity of the two end-effector paths. These geometric criteria are less important in the case of the simple straight-line path. Clearly, a complete physical interpretation of the performance criteria will be necessary before they can be used to optimize the performance of an actual robot.

6.2 LOBSTER'S ARM

A robot with the geometry of a lobster's arm was used for this simulation. A lobster's arm has six degrees of freedom with consecutive pair axes intersecting and mutually perpendicular. It is essentially three Hooke (universal) joints connected in series, as shown in Figure 6.9.

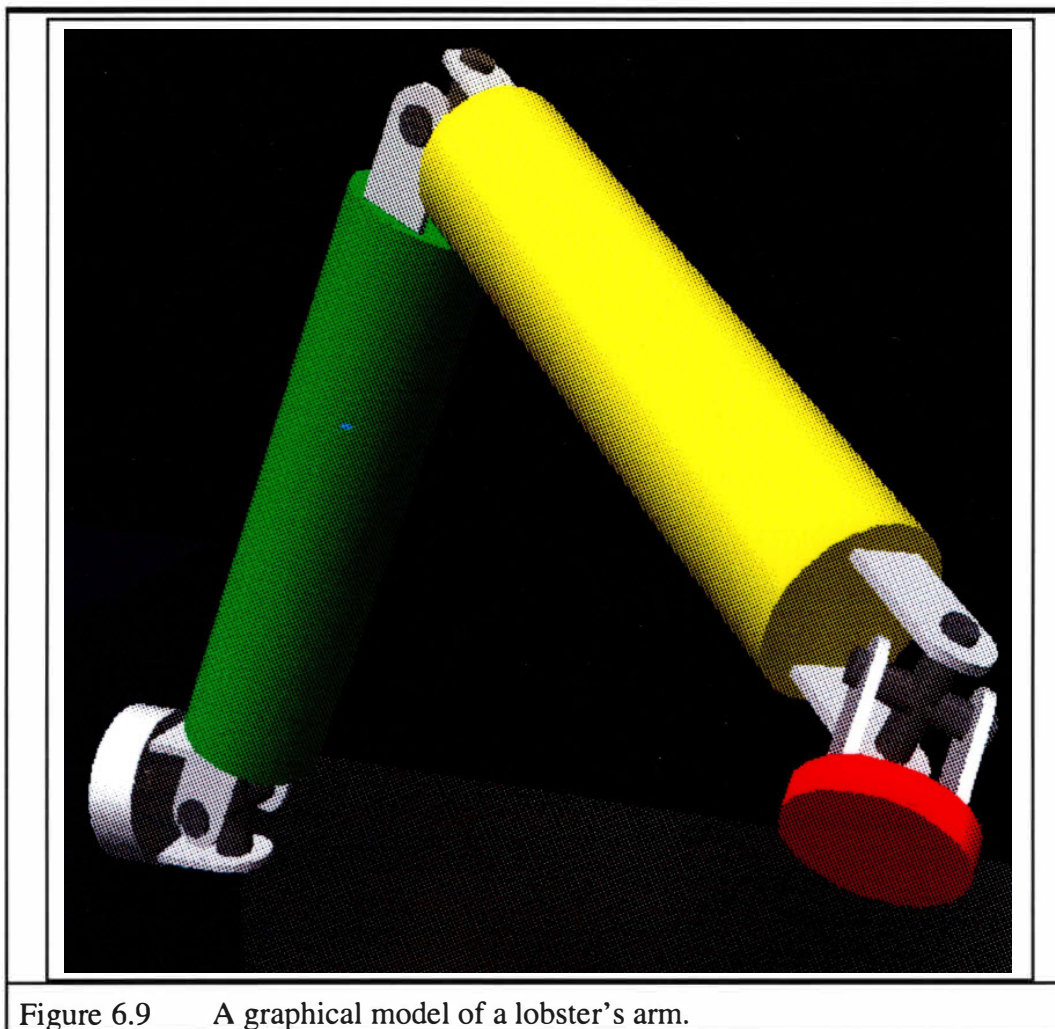


Figure 6.9 A graphical model of a lobster's arm.

Having been first described by Willis in 1841, the geometry of the lobster's arm is of historical interest (Duffy and Derby, 1979). It has also been the subject of current inverse kinematics research (Lin and Duffy, 1989). Removing the requirement that the intersecting axes must be mutually perpendicular forms a generalized lobster's arm. A direct search algorithm simulated this robot following a path somewhat like an actual lobster's arm might follow.

To give some historical perspective, the inverse kinematics problem for the generalized lobster's arm was first solved analytically in 1979 (Duffy and Derby). Ten years before this, Roth and Pieper (1969) had analyzed a variety of geometries with three consecutive axes intersecting at a point. Duffy and Derby believed the analysis of the generalized lobster's arm, with only two consecutive axes intersecting, was another step towards the solution of the inverse kinematics problem for a general six degree of freedom serial robot. In this context, the lobster's arm is indeed interesting. The lobster's arm is also interesting because it has an abundance of singularities scattered throughout the workspace. The singularities reflect the limited dexterity of this geometry, especially in its end-effector orientation capabilities.

An exhaustive exploration pattern and opportunistic decision making formed the direct search strategy for this lobster's arm example. This combination of exploration and decision making is guaranteed to find a local minimum of an unconstrained objective function. The Cartesian error formed the objective function and the direct search proceeded at a joint-level resolution of .001 degrees. A scaling factor for the rotational elements in the objective function was derived using the global scaling procedure described in Chapter 4. As it has just

been described, the solution speed on a Silicon Graphics Indigo R4000 personal workstation was about 2 Hertz.

Table 6.2 Search parameters and results for the Lobster's arm simulation.

exploration pattern -	exhaustive
decision making strategy -	opportunistic
objective function -	Cartesian error
search resolution -	.001 degrees
scaling method -	global
solution speed -	2 Hertz
maximum translational error -	.000022 meters (.00087 inches)
maximum rotational error -	.00012 radians
degrees of freedom -	6

The experiment first described in end-effector coordinates a path something like an actual lobster's arm might follow. Direct search inverse kinematics then found the corresponding joint angles. Observation of actual lobsters in a tank revealed that the geometry of a lobster's legs was essentially the same as that of its arms. For this reason, a path approximating a walking motion was devised. The path is a half circle with its two ends connected by a straight line. The straight line is where the lobster's leg would be in contact with the ground before advancing through the half circle towards another step. The direct search algorithm (described above) generated the joint angles corresponding to this end-effector path and computer animation displayed the results. The numerical error due to the direct search inverse kinematics algorithm was recorded as the simulated lobster's arm followed the path.

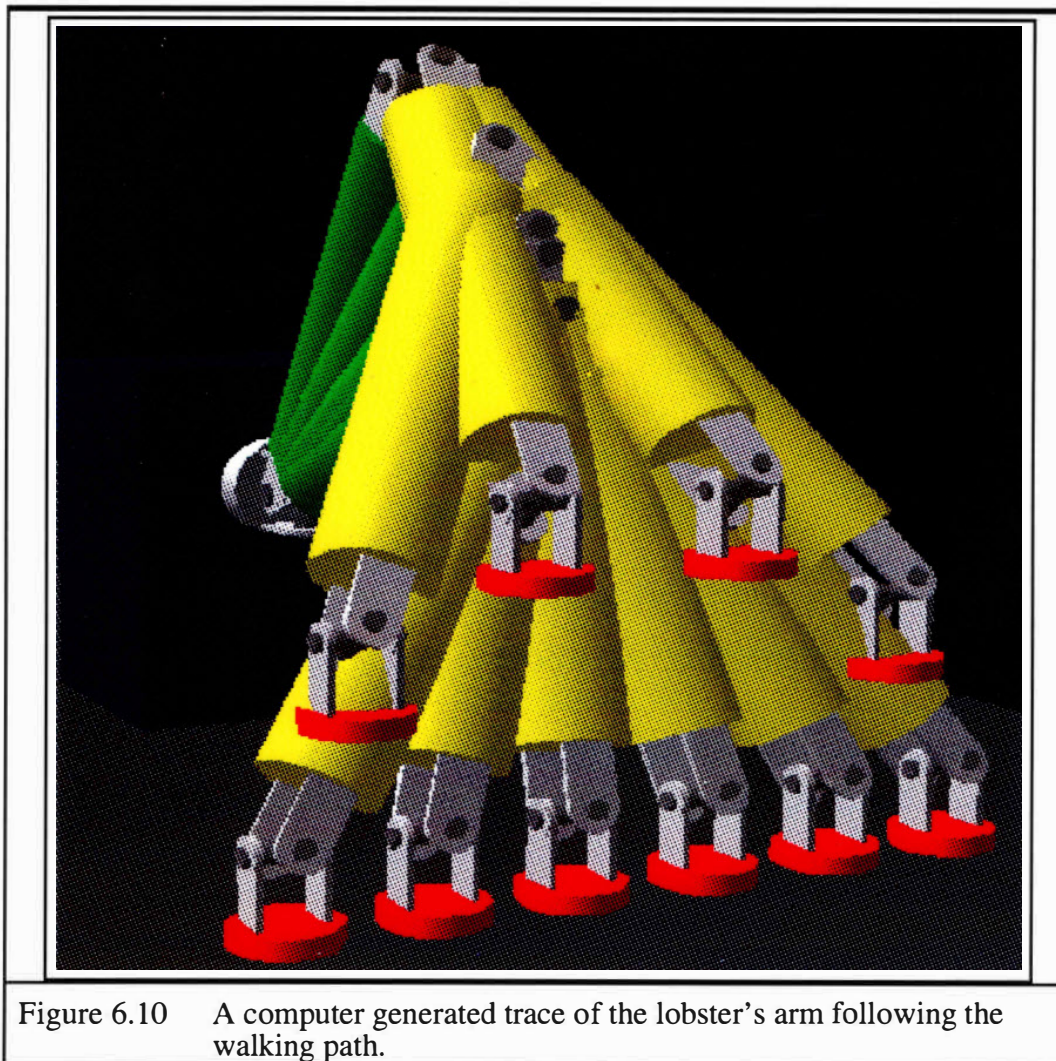


Figure 6.10 A computer generated trace of the lobster's arm following the walking path.

Figure 6.10 shows a computer generated trace of the lobster's arm following the path. The straight-line and semicircle parts are evident. More importantly, Figure 6.11 graphs the magnitudes of the translational and rotational components of the numerical error due to the direct search inverse kinematics algorithm.

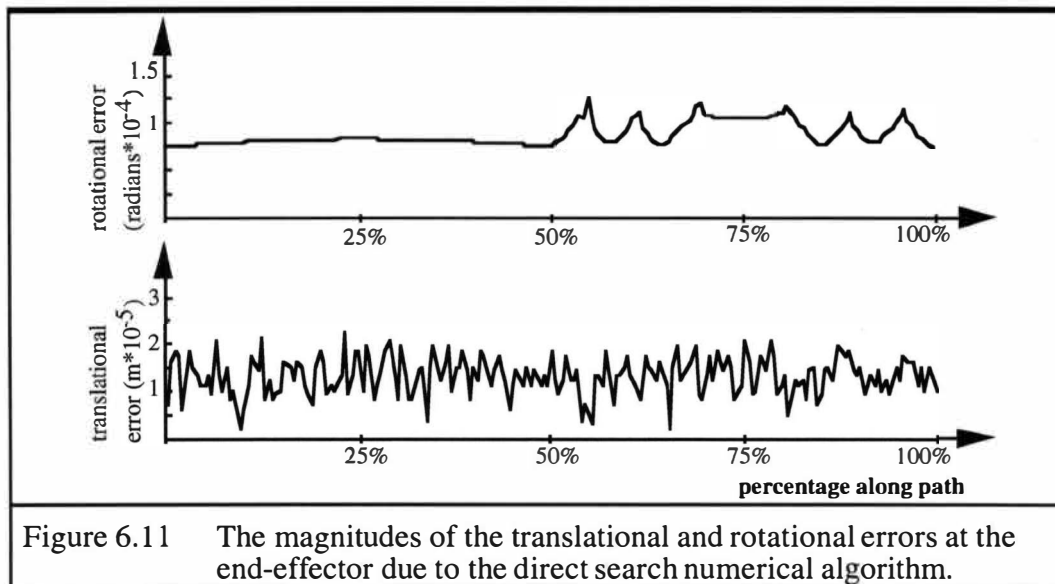


Figure 6.11 The magnitudes of the translational and rotational errors at the end-effector due to the direct search numerical algorithm.

The maximum translational error is .000022 meters (.00087 inches) and the maximum rotational error is .000122 radians. If these errors are unacceptable, the step-size can be reduced and a more accurate solution found, but at the cost of a slower solution speed.

6.3 PUMA

This example demonstrates a direct search inverse kinematics algorithm converging at a singularity. An end-effector path forces the familiar Puma geometry to pass through an exact singularity. Tracking the Cartesian error establishes the direct search converged to the singular solution.

An exhaustive exploration pattern combined with opportunistic decision making formed the direct search strategy for this example. The Cartesian error formed the objective function and the direct search proceeded at a joint-level resolution of .001 degrees. A scaling factor for the rotational elements in the

objective function was derived using the global scaling procedure from Chapter 4. As it has just been described, the direct search algorithm executed on a Silicon Graphics Indigo R4000 personal workstation at an approximate rate of 2 Hertz.

Table 6.3 Search parameters and results for the Puma simulation	
exploration pattern -	exhaustive
decision making strategy -	opportunistic
objective function -	Cartesian error
search resolution -	.001 degrees
scaling method -	global
solution speed -	2 Hertz
maximum translational error -	.000058 meters (.0023 inches)
maximum rotational error -	.000011 radians
degrees of freedom -	6

The Puma geometry used in this example has six degrees of freedom and a spherical wrist. This is a very simple geometry and closed form solutions are easily derived. It is chosen mainly because many researchers are familiar with this geometry and its associated singularities. The singularity of interest in this example occurs when the fifth joint passes through the center of its travel. At this point, joint axes four and six are collinear and the robot is in a singular position.

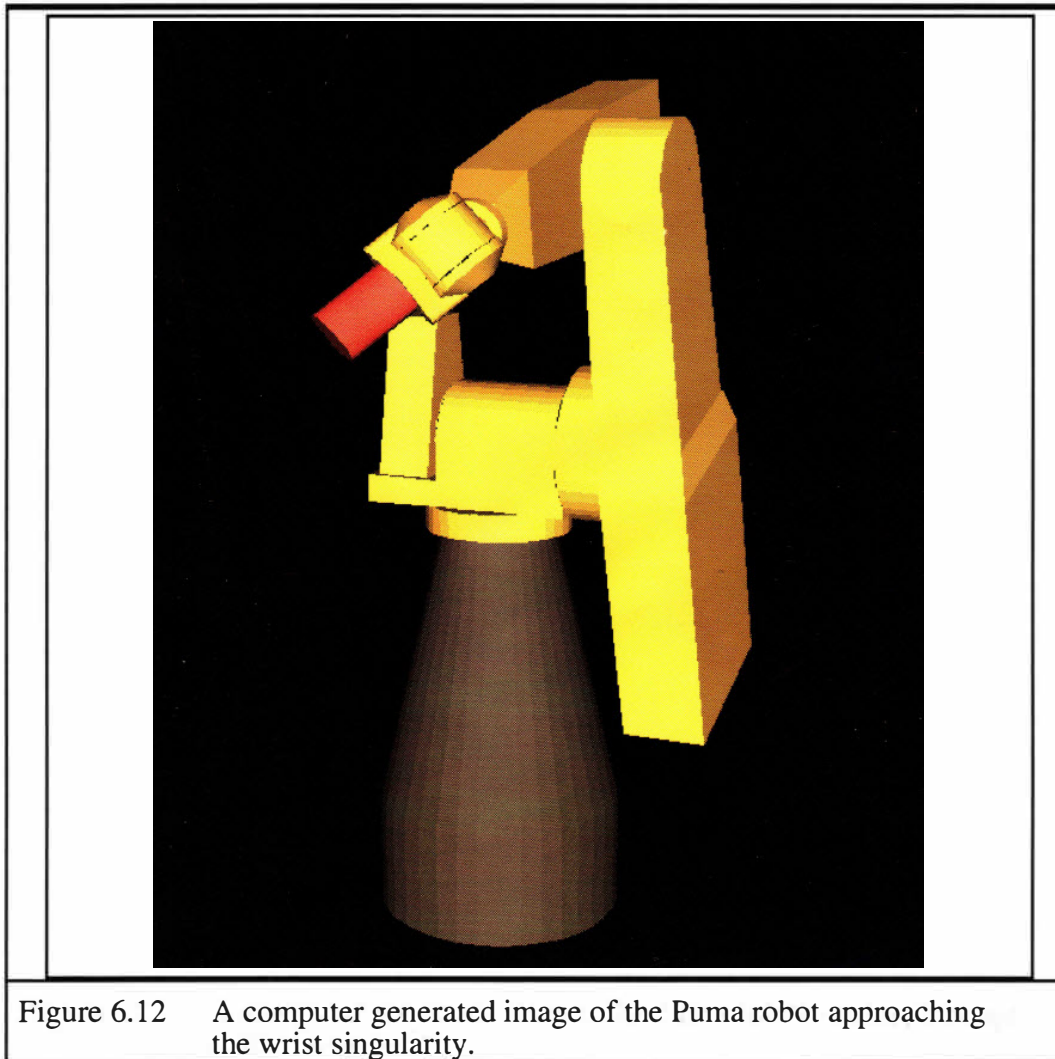
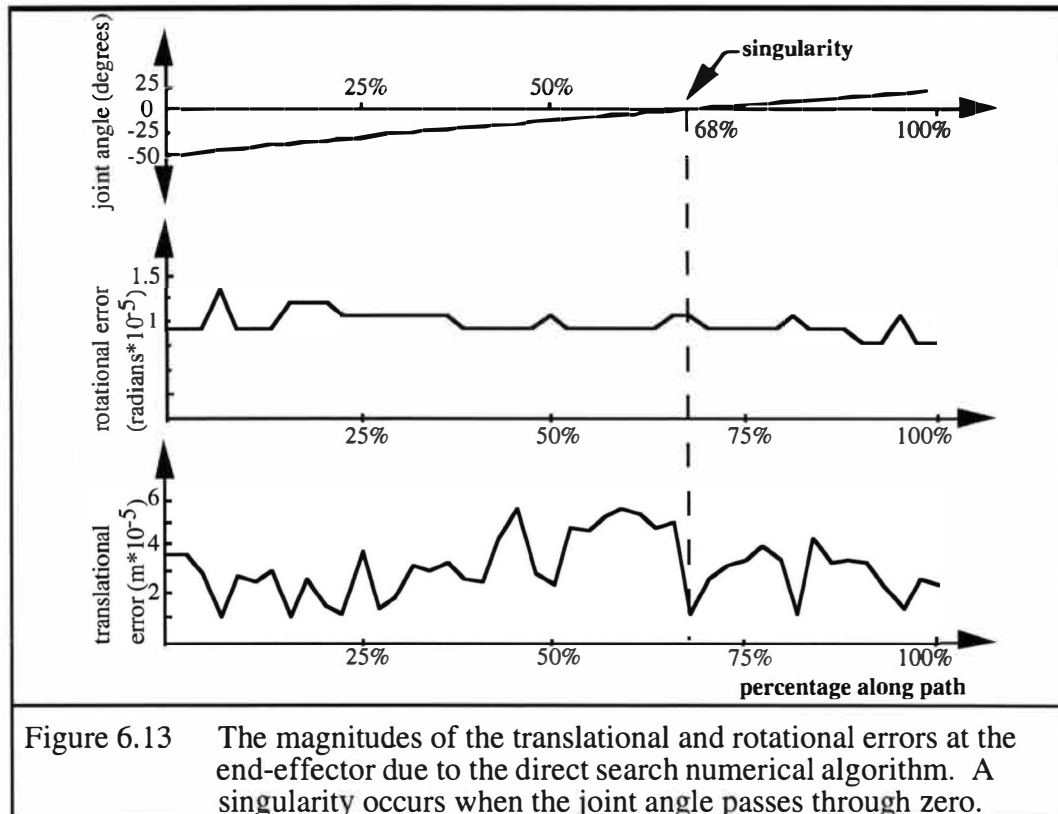


Figure 6.12 A computer generated image of the Puma robot approaching the wrist singularity.

Figure 6.12, shows the simulated Puma robot in the straight line path that passes directly through the wrist singularity described above. Figure 6.13 shows the fifth joint angle, the rotational error, and the translational error from top to bottom respectively. The singularity occurs when the fifth joint angle passes through zero. Examination of the translational and rotational errors shows

that the direct search algorithm did not exhibit any unusual errors as it successfully converged to, and passed through, the singularity.



6.4 ALPHA

The University of Texas Robotics Research group is currently designing and building an advanced seven degree of freedom modular robot. The robot is called ALPHA, which stands for Advanced Lightweight Prototype High-power Arm. The generality of the direct search inverse kinematics method has allowed it to play an important part in the design phase of this robot. The ALPHA robot,

once completed, will also be an important test platform for future multicriteria inverse kinematics efforts.

The University of Texas Robotics Research Group believes that by bringing together all of their research history and expertise; they can build a truly advanced and revolutionary robot. A large part of the available expertise is in the area of dynamic modeling. Dynamic modeling determines the forces and torques throughout the robot. The forces and torques, when coupled with the robot's design, determine the deflections, stresses and actuator requirements. Before the dynamic modeling can take place, however, the joint's displacements, speeds, and accelerations must be determined for a given test path. This is essentially the inverse kinematics problem. The generality of the direct search method allows its application to many different designs as the design team evaluates them.

The ALPHA robot will also be an important test platform for many different leading-edge technologies being developed by the University of Texas Robotics Research Group. Multicriteria inverse kinematics is one of these technologies. Figure 6.14 shows one of the designs for the ALPHA robot currently being considered. The main kinematic feature of interest in this design is the spherical wrist. A robot has a spherical wrist if its last three joint axis intersect at a point. A spherical wrist allows an important geometric simplification in the inverse kinematics solution (discussed later in this section). This simplification significantly increases the solution speed of a direct search inverse kinematics solution. The only issue is solution speed – not solution tractability. The increase in speed should allow direct search to optimize multiple performance criteria in real-time with common computing hardware. These

geometric simplifications detract from the generality of the direct search, but they are currently required if optimizing anything more than very simple performance criteria in a real-time application.

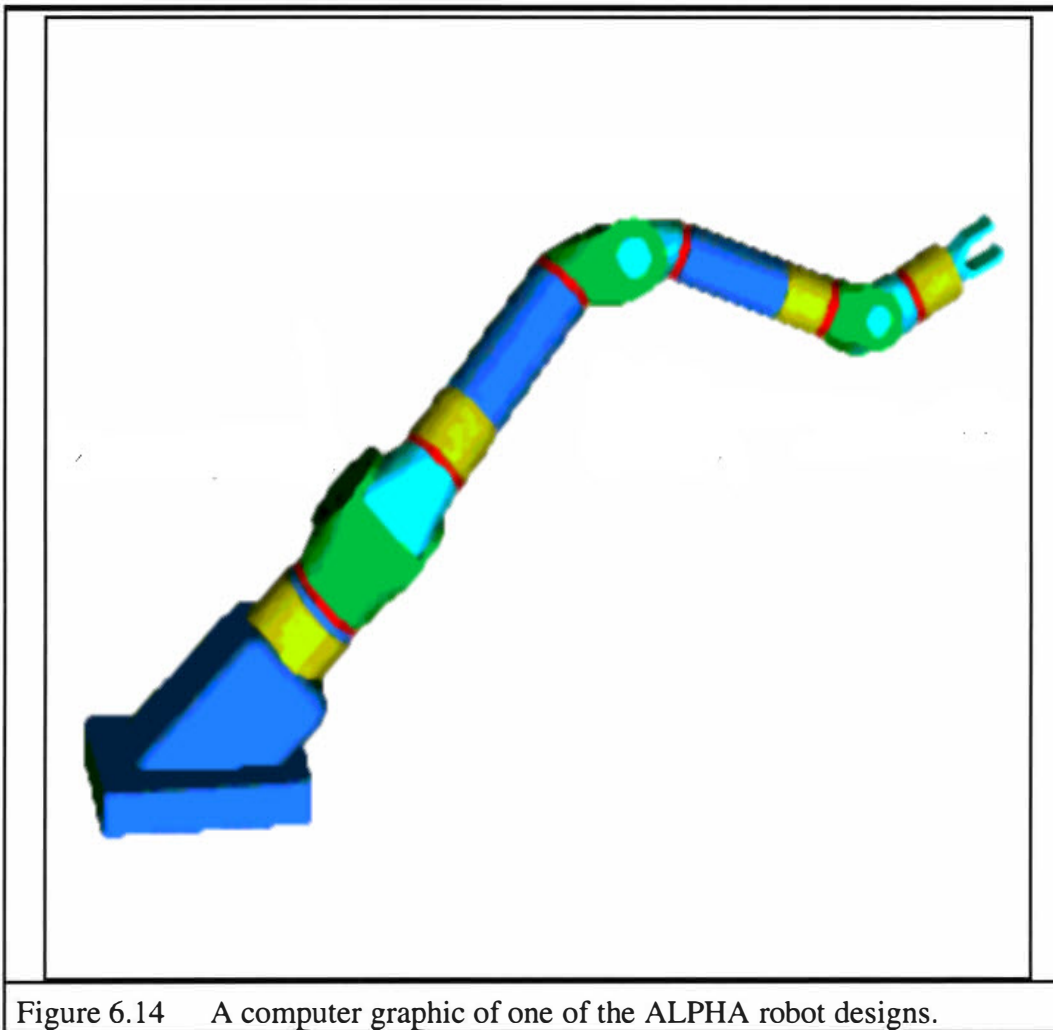


Figure 6.14 A computer graphic of one of the ALPHA robot designs.

This example discusses two different test cases for the ALPHA robot. For both cases, an exhaustive exploration pattern combined with opportunistic decision making to form the direct search strategy. This combination guarantees

local optimality. In the first case, the objective function contained the translational error, the rotational error, and the two-norm of the joint speeds. A direct search performed through all seven joints found the solution. In this case the solution speed was about .6 hertz on a Silicon Graphics Indigo R4000 personal workstation. In the second case, simple geometry calculated the three wrist angles in closed-form. The three wrist angles then satisfied the three rotational end-effector constraints. Since the rotational end-effector constraints are satisfied in closed form, they are removed from the objective function, which now contains just the translation error and the two-norm of joint speeds. Note that all of the joint speeds, including those at the wrist, are included in the objective function and hence the optimization. The wrist angles are simply found in closed form. Thus, the direct search must only search through the first four joints. This simplification increases the speed of execution to about thirty Hertz on the same Indigo computer. This is approximately the speed required for real-time performance, though only one very simple performance criteria is being considered.

For a six degree of freedom robot, both spherical shoulder and spherical wrist geometries have known closed-form solutions. This means that six out of the seven joint angles can be found in closed-form and the search need only be performed over one joint variable. This should allow the real-time application of a direct search inverse kinematics algorithm incorporating multiple complex performance criteria and a sophisticated decision-making strategy. Most of the geometric generality of the direct search inverse kinematics method will be lost, but the closed-form solution can be formulated so as to allow minor changes to

the robot's geometry (such as changing link lengths). Fault tolerance capabilities are also sacrificed because a fault in one of the joints will likely change the closed-form solution. This strategy maintains generality with respect to the performance criteria.

6.5 TITAN 7F

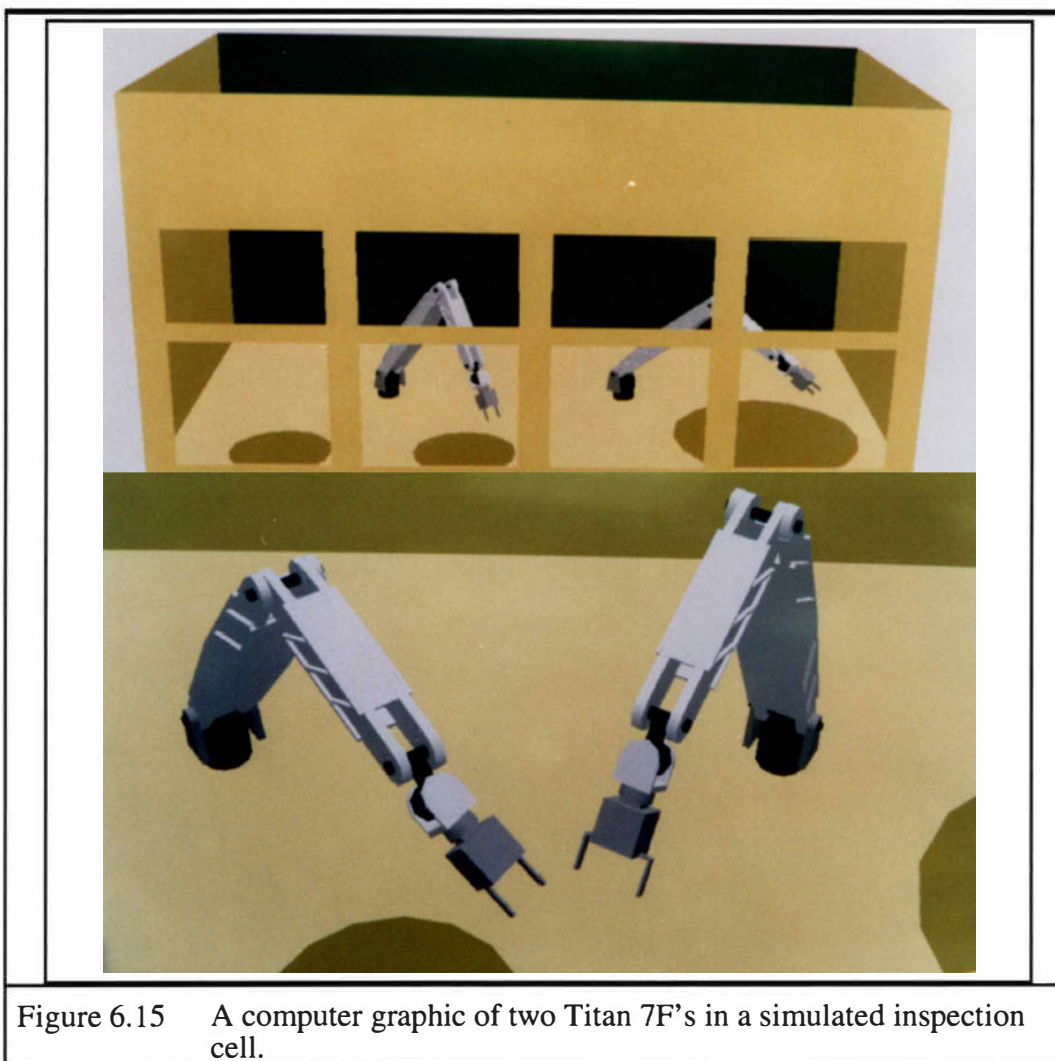


Figure 6.15 A computer graphic of two Titan 7F's in a simulated inspection cell.

A simulation of a robot inspecting nuclear waste was developed in conjunction with Argonne National Laboratory. The robot is a Schilling Titan 7F, which has six degrees of freedom. The task is to push a thin gas-sampling probe through a vent in the top of a fifty-five gallon drum and then withdraw the probe. The probe is $1/16$ th of an inch and the vent $1/8$ of an inch in diameter. Figure 6.15 is one frame of an animation showing the robot performing this task. The gas-sampling task is described in end-effector space. A direct search algorithm that combined exhaustive exploration and opportunistic decision making translated the task into joint space. The interesting aspect of this example is the geometry of the robot. It is actually quite complex. Typically, industrial robots have at least two, and usually three, joint axes intersecting. This is so the robots have known closed-form inverse kinematic solutions.

The Titan 7F has no joint axes intersecting. These robots are supplied with master-slave manual controllers. This type of manual controller is a small kinematic replica of the actual robot. The robot exactly duplicates, in joint space, the motions of the manual controller. The joint angles are simply measured at the manual controller and relayed to the robot. This simulation, however, was created as part of a cell-layout effort. Cell-layout requires task simulation before the robot is put into service.

6.6 WAFER HANDLING

A direct search inverse kinematics algorithm aided in the development of a robotic semiconductor wafer handling simulation. The simulation was developed in conjunction with Sematech as part of an advanced semiconductor

manufacturing effort. The simulation shows three robots operating in one workcell. The robots move a number of wafers between storage and process equipment. Figure 6.16 shows one frame of an animation of this task. The robots are used instead of humans because of the desire for an extremely clean environment. The wafer handling task was described in end-effector coordinates, so this simulation required inverse kinematics to translate the task into joint space.

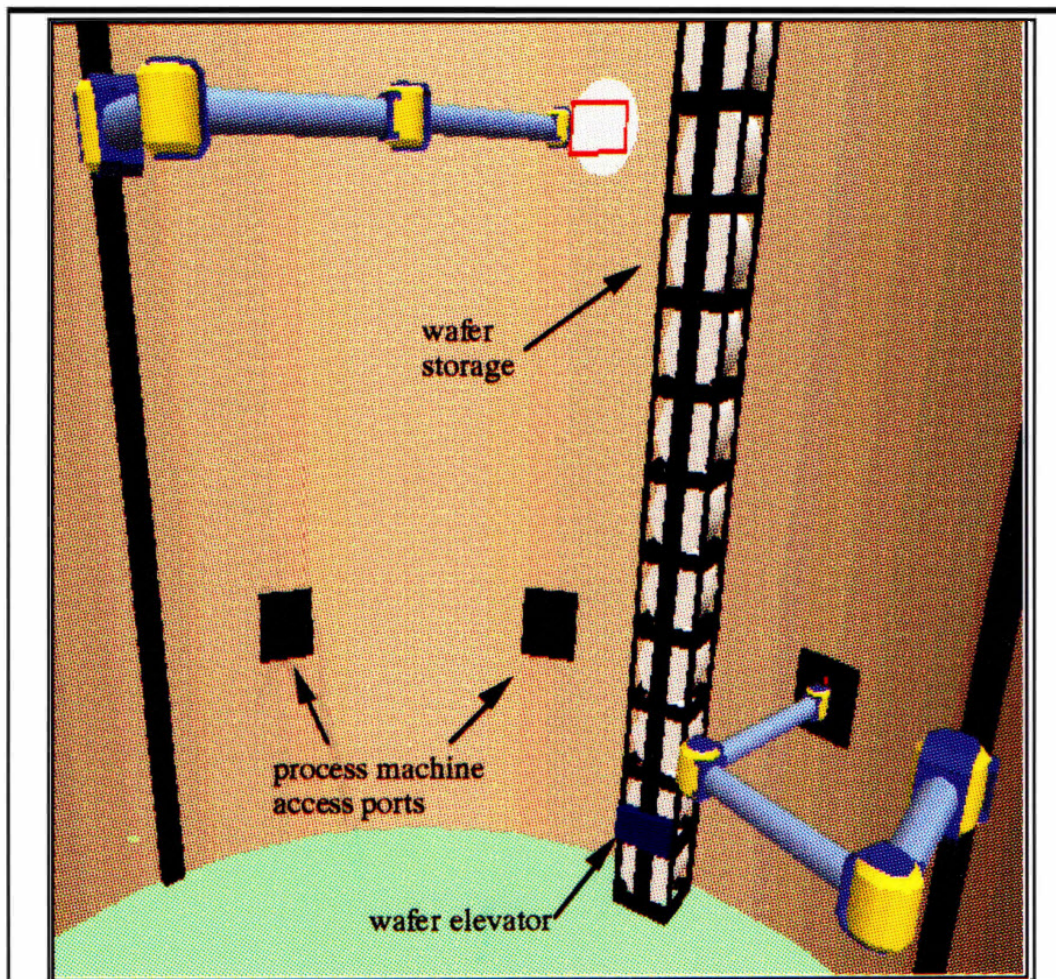


Figure 6.16 A computer graphic of two wafer-inspection robots in a simulated advanced semiconductor manufacturing facility.

6.7 DEXTEROUS ROBOT WITH NINE DEGREES OF FREEDOM

There are two main features of interest in this example. The first is the robot, which has nine degrees of freedom and therefore three degrees of redundancy. The second feature is the direct search algorithm, which uses a sub-optimal exploration strategy.

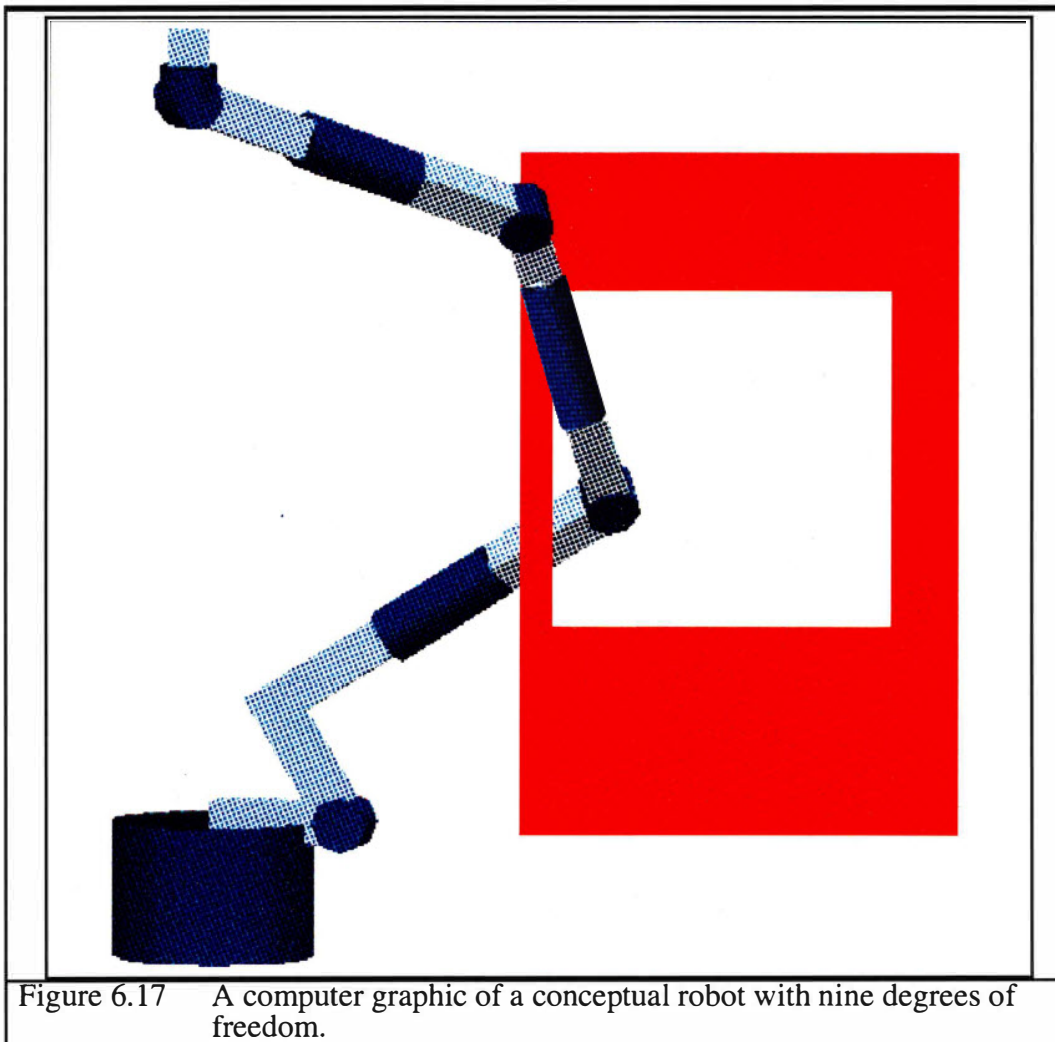


Figure 6.17 A computer graphic of a conceptual robot with nine degrees of freedom.

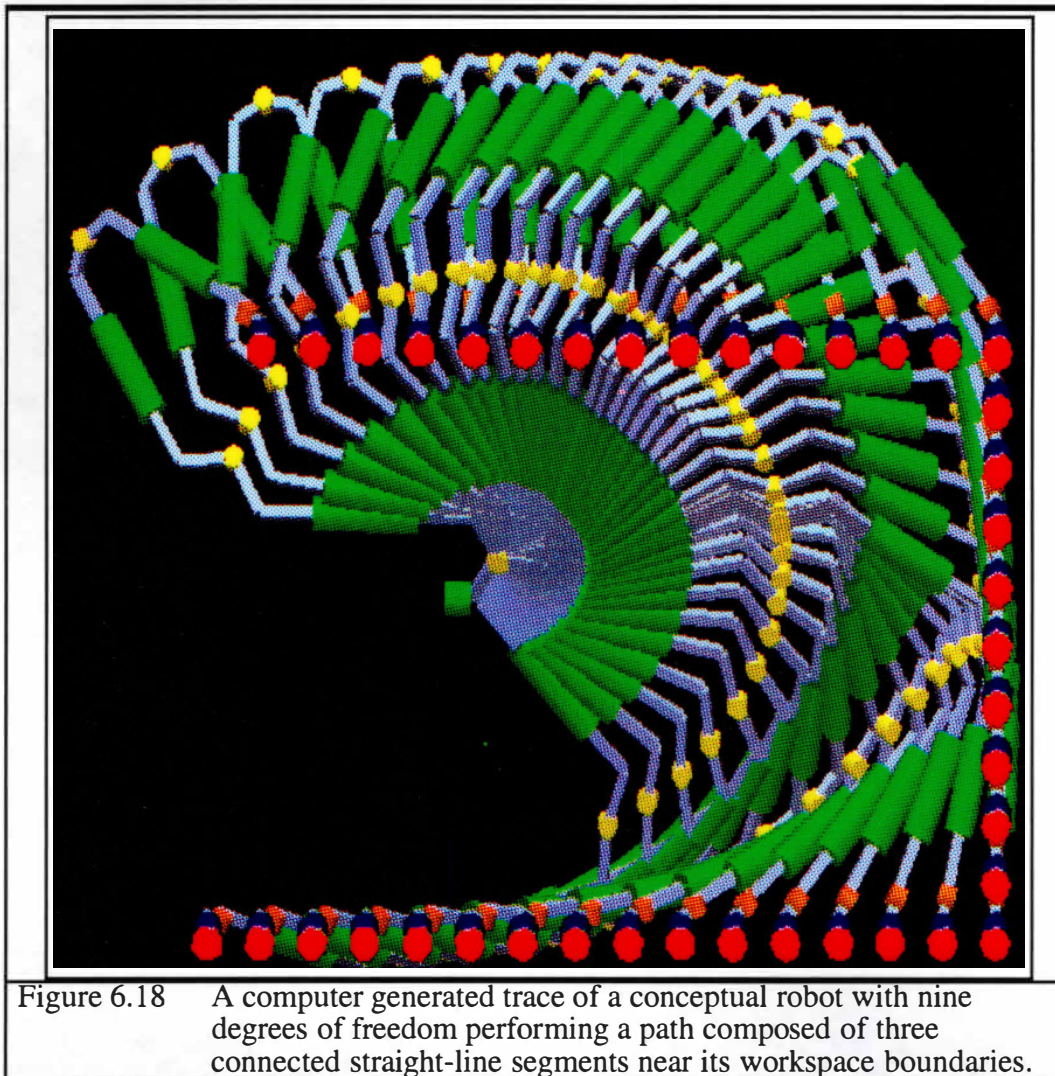
Figure 6.17 shows the conceptual robot with nine degrees of freedom. It is a sequence of alternating roll and pivot joints. The geometry is very much like many research robots with seven degrees of freedom, but with one extra pivot and one extra roll. Besides being highly redundant, this geometry is also complex because offsets at the pivot joints prevent any of the joint axes from intersecting.

Table 6.4 Search parameters and results for the simulation of the dexterous robot with nine degrees of freedom.	
exploration pattern -	simple
decision making strategy -	opportunistic
objective function -	Cartesian error and the two-norm of joint angles
search resolution -	.001 degrees
scaling method -	search-based with forced convergence
solution speed -	10 Hertz
maximum translational error -	.000046 meters (.0018 inches)
maximum rotational error -	.00004 radians
degrees of freedom -	9

A simple exploration pattern combined with opportunistic decision making to form the direct search strategy. The simple exploration pattern does not guarantee local optimization. The objective function for the search contained the rotational and translational error as well as the two-norm of the joint displacements. The joint displacement criterion discourages the robot from approaching its joint limits while following the path. An infinite barrier constraint enforced the joint limits. The resolution for the search was .001 degrees. The

search-based scaling procedure generated the initial scaling factors. Because of the suboptimal search strategy, the scaling factors were modified during the search to force convergence. Chapter 4 discusses this procedure. Essentially, after each time the search terminates without finding a solution, the scaling factors of any unsatisfied constraints are increased by a power of ten. Only two constraints are included in the objective function. They are the magnitudes of the translational and the rotational errors. The translational constraint was satisfied if the error was less than .0000254 meters (.001 inches). This is ten times less than the resolution of a typical industrial robot. The global scaling method determined the corresponding factor for the rotational constraints as .0000318 radians.

A path described in end-effector space tested the direct search algorithm by recording the rotational and translational error at each point along the path. The end-effector path was three connected straight-line segments near the boundaries of the robot's workspace. Figure 6.18 shows a computer generated trace of the robot following the path. Figure 6.19 shows the error due to the direct search inverse kinematics algorithm. The maximum translational error is .000046 meters (.0018 inches) and the average is .000022 meters (.00087 inches). The maximum rotational error is .000040 radians and the average is .000016 radians. At this resolution, the direct search algorithm's solution speed was about ten Hertz on a Silicon Graphics Indigo R4000.



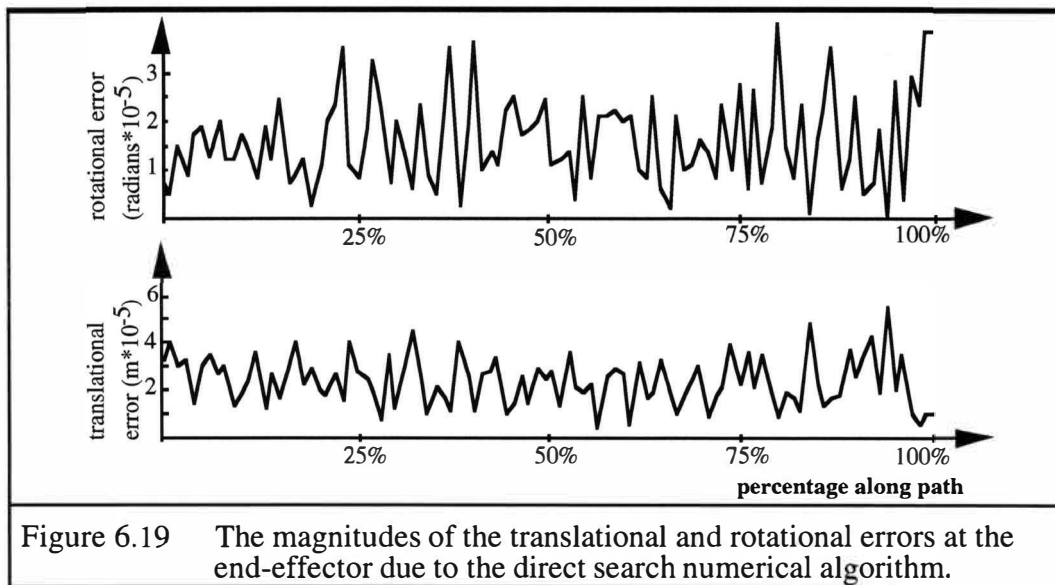


Figure 6.19 The magnitudes of the translational and rotational errors at the end-effector due to the direct search numerical algorithm.

6.8 FAULT-TOLERANT ROBOT WITH TWENTY-ONE DEGREES OF FREEDOM

This final example simulates a hyper-redundant and fault-tolerant robot. The robot has twenty-one degrees of freedom (Figure 6.20). A robot with very many extra degrees of freedom is often called hyper-redundant. Hyper-redundancy enables the robot to continue operating effectively even if several joints fail. The application of direct search inverse kinematics as part of a fault-tolerant robotic system is shown to be particularly straightforward. Suddenly locking three of the joints while the robot is following a straight-line path simulates a fault. Though it is realized that suddenly locking any joints will have serious dynamic repercussions, this example only considers kinematics. The performance of the direct search algorithm during a brief interval immediately after the fault is particularly interesting.

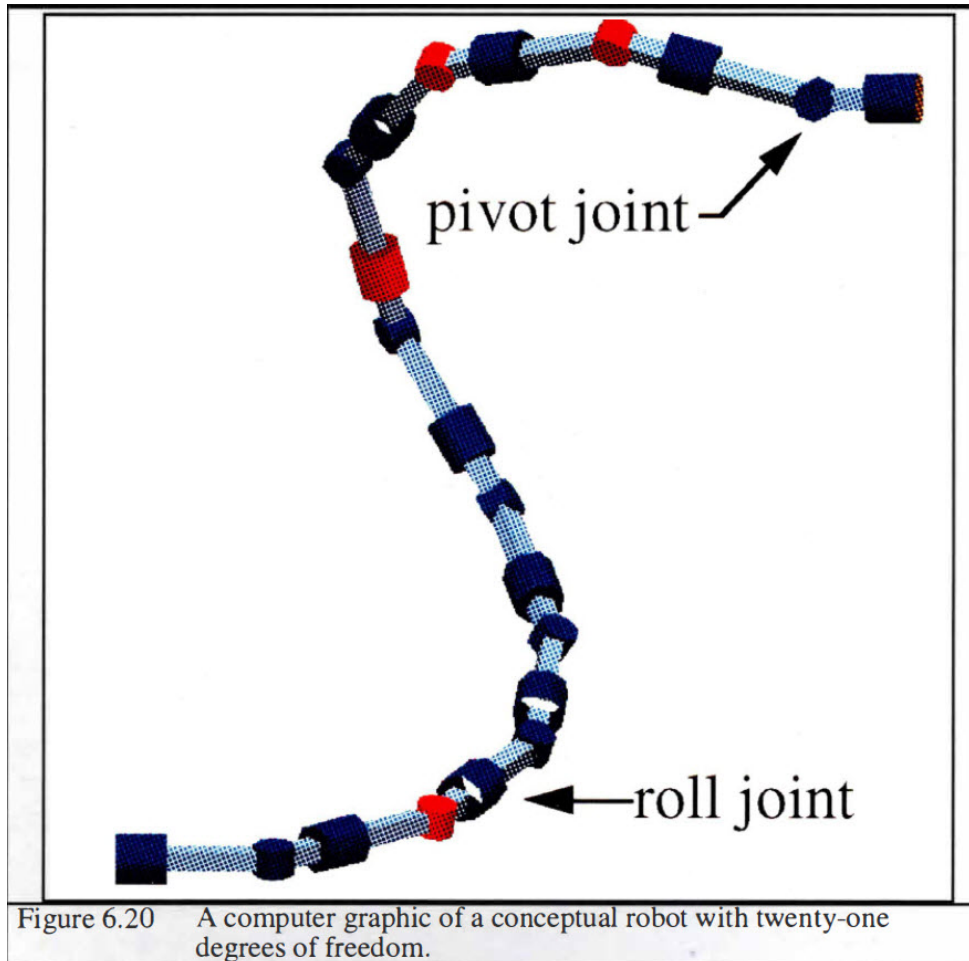


Figure 6.20 A computer graphic of a conceptual robot with twenty-one degrees of freedom.

A simple exploration pattern combined with opportunistic decision making to form the direct search inverse kinematics strategy. This combination does not guarantee a local optimum. This example used four different objective functions. They all contained terms for the Cartesian error. One was only the Cartesian error while the other three also included either the two-norm of the joint speeds, the two-norm of the joint accelerations, or both. These norms represent

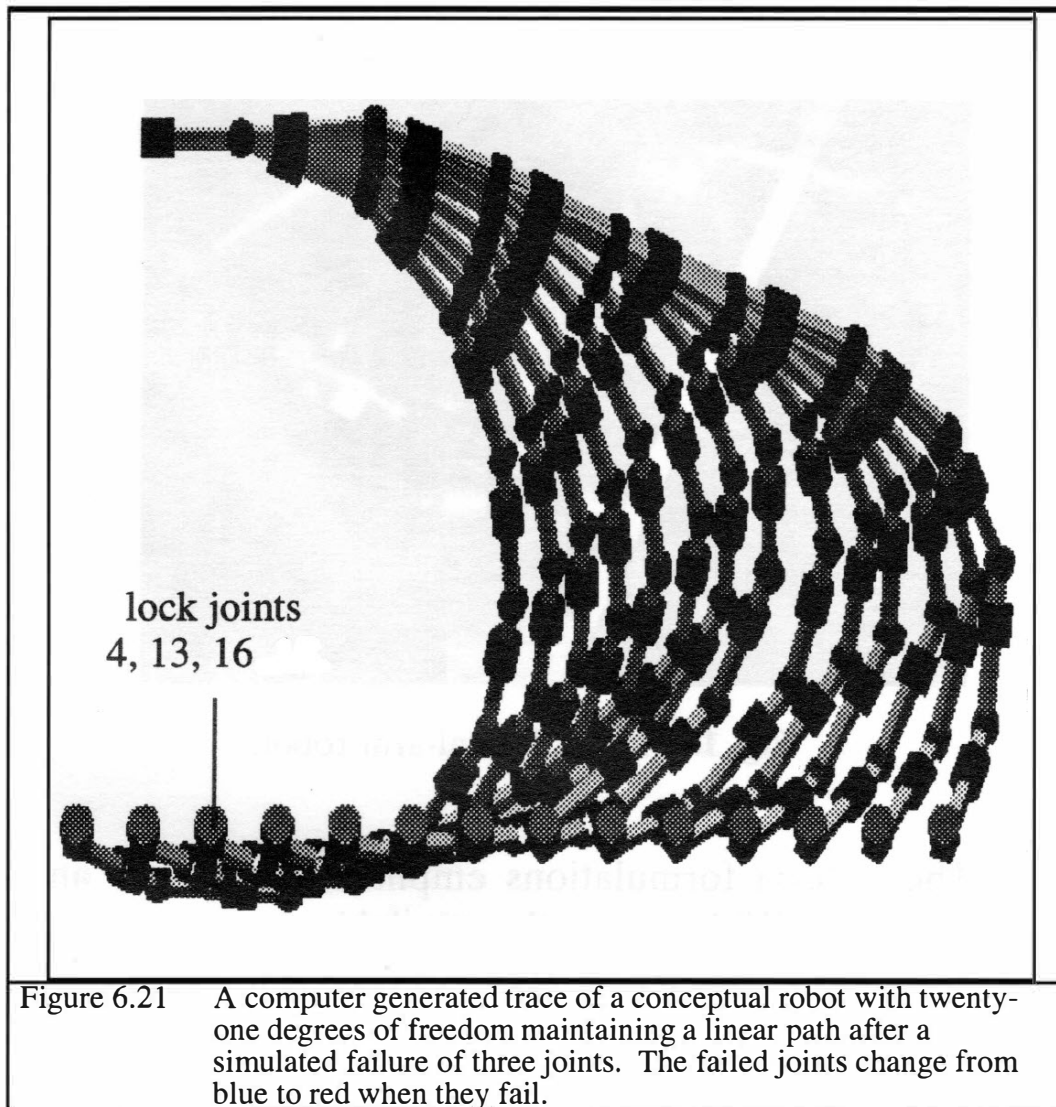
very simple performance criteria; one at the velocity level and one at the acceleration level. The search-based scaling procedure generated the scaling parameters. Regardless of the objective function, the direct search algorithm's solution speed was about five Hertz.

Table 6.5 Search parameters and results from the simulation of the fault-tolerant robot with twenty-one degrees of freedom	
exploration pattern -	simple
decision making strategy -	opportunistic
objective function (four cases) -	Cartesian error
	Cartesian error and two-norm of joint speeds
	Cartesian error and two-norm joint accelerations
	Cartesian error, two-norm of joint speeds, and two-norm of joint accelerations
search resolution -	.001
scaling method -	search based with forced convergence
solution speed -	5 Hertz
maximum translational error -	.000024 meters (.00095 inches)
maximum rotational error -	.00044 radians
degrees of freedom -	21 (before fault, 18 after)

Serial kinematic redundancy is one way of achieving fault tolerance. The robot in this example has enough extra joints to allow continued operation even if several joints fail. A properly chosen geometry will also allow a robot with less than this degree of redundancy, perhaps a robot with ten joints, to achieve fault-tolerance through serial kinematic redundancy. In this fault-tolerance scenario, a faulty joint is locked and prevented from turning. The faulty joint essentially

becomes part of the robot's structure. The incorporation of this fault-tolerance strategy as part of the direct search is straightforward. The failed joint is simply removed from the search space and the exploration process generates no perturbations of this joint. Thus, the decision making process is not presented with choices associated with moving the failed joint.

The robot in this example is interesting because of its very large number of joints. Application of a pseudoinverse would require the inversion of a twenty-one by twenty-one matrix; a process not likely to be performed in real-time (at least in the very near future). The robot's joints are arranged as the familiar sequence of roll and pivot, but there are no offsets at the pivot joints. This is very important because, without the offsets, every time one of the pivot joints passes through the center of its travel, the axes of the roll joints on either side are collinear. When these roll axes are collinear; the robot is in a singularity. For this robot, however, the singularities do not represent any loss of dexterity. The robot would be expected to pass through several of these singularities while following even a modestly demanding path. Avoiding these singularities would severely limit the dexterity of the robot. These singularities are not a problem for the direct search algorithm this example demonstrates. The algorithm does not explicitly address singularities at all. The decision making process simply evaluates the singular solutions in the same manner it evaluates any choices the exploration pattern generates.



This experiment simulated a fault by suddenly locking three separate joints. The simulated fault occurred while the robot was following a straight-line path. The Cartesian error, the two-norm of the joint speeds, and the two-norm of the change in joint speeds with respect to time, were recorded as the robot continued to follow the path after the fault. This experiment was performed four

different times, with the objective function for the direct search being changed each time as described above. In each case, the robot was in the same state before the fault. After the fault, the different objective functions cause the simulated robot to follow different joint-level paths, even though the end-effector path did not change.

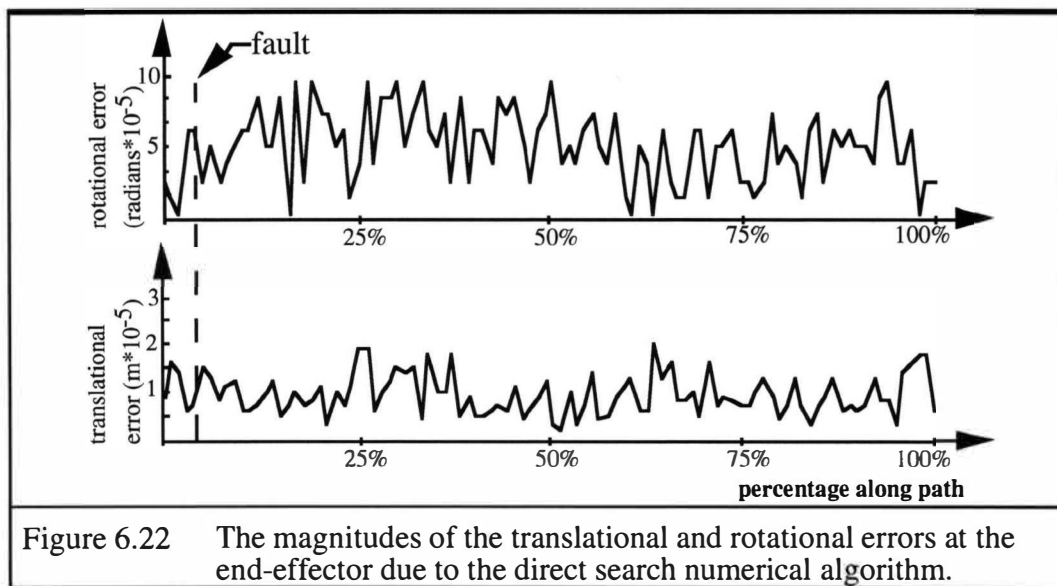


Figure 6.22 The magnitudes of the translational and rotational errors at the end-effector due to the direct search numerical algorithm.

Figure 6.22 graphs the rotational and translational errors for the path considering both speed and acceleration criteria. The fault had no effect on the ability of the direct search algorithm to properly place the robot's end-effector. For each of the four paths discussed in this example, the solution was always within the desired error bounds both before and after the fault. In some sense, solving the inverse kinematics problem for a robot with this many extra joints is less difficult than solving the problem for a robot with fewer degrees of freedom. The proliferation of solutions actually makes the solution more tractable.

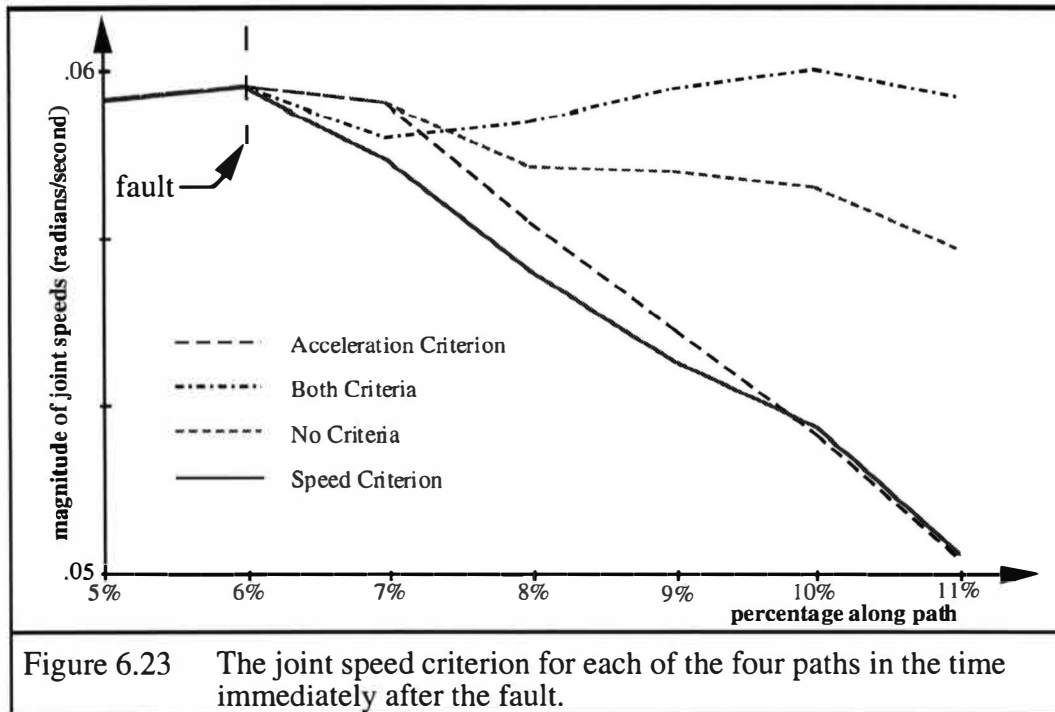


Figure 6.23 shows the two-norms of the joint speeds in the time immediately after the fault for each of the four objective functions. The step size seems to be coarse because of the brief time interval being examined. During the first few steps after the fault, the objective function which only considers the two-norm of joint speeds is the best at minimizing the joint speed criterion. This is to be expected, and helps to validate that the direct search optimization scheme is working.

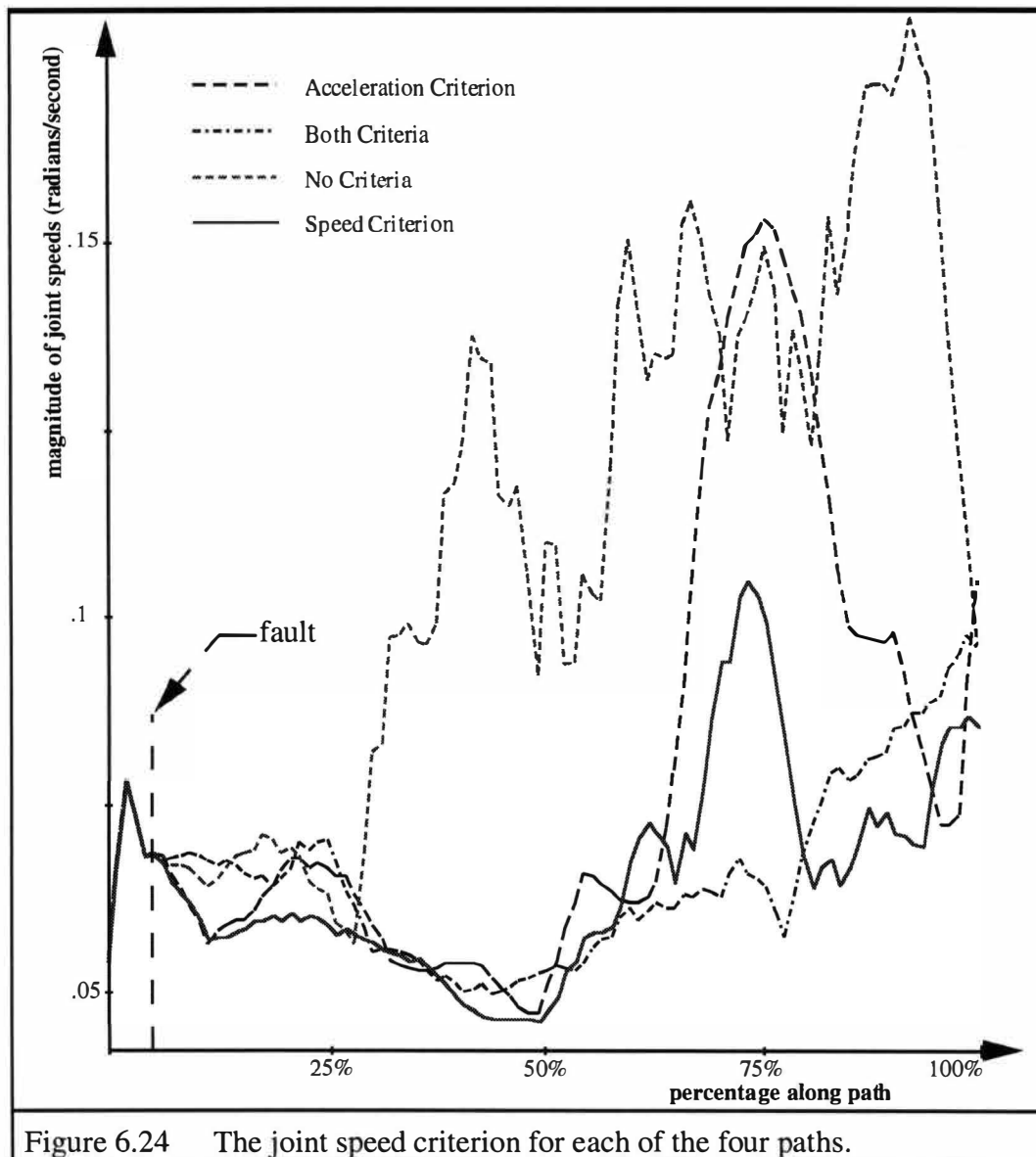


Figure 6.24 The joint speed criterion for each of the four paths.

Once the paths diverge there is no guarantee that they also won't cross. In other words, the best first step does not guarantee the best whole path. This is because of the local nature of the direct search inverse kinematics method. Figure 6.24 shows the path corresponding to the velocity criterion actually crosses the path corresponding to the mixed criterion several times.

Chapter 7: Road Map

This chapter outlines a vision for the future of generalized inverse kinematics and its application to advanced robotic systems. Envisioning the future is by nature a risky activity. Technological changes happen virtually overnight and can render any vision obsolete. This chapter discusses the vision in terms of ten advanced robot applications with the hope that applications will ensure relevance, at least into the near future. Each application possesses a degree of difficulty that will likely require generalized inverse kinematics. Generalized inverse kinematics cannot, however, be applied without the benefit of supporting technology. This chapter therefore discusses a broad spectrum of supporting technologies in terms of their relevance to the application of advanced robotic systems.

7.1 ADVANCED APPLICATIONS

There are several features distinguishing an advanced application. Some of these applications are very demanding kinematically. The robot must operate in very cluttered or constrained environments. These applications call for a redundant robot with at least several extra degrees of freedom. Highly redundant robots require an inverse kinematics method insensitive to singularities. Other advanced applications demand a modular robot. The modular robot is reconfigurable to perform a variety of tasks. Reconfigurability is important in manufacturing operations and in space or nuclear environments. Reconfigurable robots require an inverse kinematics method general with respect to geometry. Applications requiring very precise calibrated robots also require an inverse kinematics method general

with respect to geometry. Fault tolerance is very important in space and nuclear applications. Serial kinematic redundancy is one way of achieving fault tolerance. Achieving fault tolerance through serial kinematic redundancy requires an inverse kinematics method insensitive to singularities and general with respect to geometry.

Table 7.1 A brief description of ten advanced robotic applications which would benefit from generalized inverse kinematics.

Application	Description
complex assembly	characterized by the need to operate in cluttered environments
food packaging	relatively simple tasks which must be performed at very high speed
micromanipulation	characterized by very precise operation in a small workspace
in-space assembly and construction	representative tasks include assembling and mating truss sections and solar arrays
satellite retrieval and repair	docking with the faulty satellite, repairing it, and then placing it back into orbit
E.V.A. site preparation	this decreases the amount of time the astronauts must spend outside of the spacecraft
semiconductor wafer handling	moving semiconductor wafers amongst process machines and storage in a clean-room environment
long reach arm	heavy-duty tasks in very large, yet complex work environments for waste-site cleanup and construction
inspection at nuclear facilities	inspection of drums, plumbing, tanks, and pumps for radiation, leaking, temperature increases, and wear in a radioactive environment
decontamination and dismantlement	stabilizing, disassembling, and storing decommissioned nuclear facilities

7.1.1 Complex Assembly

The need to operate in cluttered environments characterizes complex assembly tasks. Perhaps the robot must reach into a “tight” space or reach around an intermediate object. Installing wiring harnesses in an automobile is one example of a complex assembly task. This is not to say that other assembly tasks do not require precision. Complex assembly tasks are simply characterized by the need to operate in very constrained or obstacle-strewn environments.

Assembly tasks of any sort are by nature repetitious. This is what makes them good candidates for robot application. These repetitious tasks are boring for humans and may lead to repetitive motion injuries. Because of increased worker's compensation costs and increased health insurance costs, it is becoming more and more expensive to employ humans to perform these tasks. Also, a robot can typically replace three shifts of workers.

The obstacle avoidance and constrained environments which characterize complex assembly tasks essentially necessitate a redundant robot. The robot must avoid obstacles while still following the specified end-effector path. Also, the end-effector path will likely be quite constrained, at least at the position level. This will make it much more difficult to plan end-effector paths that avoid joint limits and singularities. A complex assembly task will thus require inverse kinematics based on multiple performance criteria as well as a robot with several redundant degrees of freedom. The extra degrees of freedom allow the robot to avoid obstacles and joint limits while still following the specified end-effector path. An inverse

kinematics algorithm insensitive to singularities is desirable when working with a redundant robot.

Representative assembly tasks include: force-fit part mating, controlled stressing of materials, joining parts by impact, and fastening. Performing these tasks in constrained environments adds complexity. These assembly tasks are in the very wide medium-duty range and involve forces of from 1 to 25 pounds. The global accuracy is in the range of from 1 millimeter to 100 micrometers and the required resolution is from .1 to 10 micrometers. Work volumes for these tasks are on the order of from 10 cubic centimeters to 1 cubic meter. This range of tasks recommends the use of a few modules to make a large population of assembly robots for this class of assembly tasks.

7.1.2 Food Packaging

The food industry represents a tremendous opportunity for applying robotic systems. In particular, packaging in the food industry has been identified as a promising application for robots. It is not difficult to imagine the typical packaging tasks in the food industry. One representative task involves putting stacks of wrapped candy bars into multiple bar packs. Another task is to put packages of cookies into their shipping cartons. The prominent feature of these tasks is that they are not very challenging kinematically, yet they must be performed at a relatively high rate of speed.

In the food industry, robots must compete with both hard-automation and humans for packaging applications. Versatility is the main benefit to using robots over hard-automation in these applications. For instance, if the size and shape of

the product changes, the robot can be reprogrammed while the hard-automation may need to be replaced or refitted. This versatility is becoming more and more important as marketing demands for packaging updates and regional variations in packaging increase.

The main benefit to using robots over humans in the food packaging application is associated with the repetitiveness of the tasks. Wrist and back problems are common. Also, these tasks are incredibly boring, involving thousands of packages a day. Finally, these tasks are relatively easy for robots to do.

Since packaging tasks in the food industry are not that challenging, it is reasonable to question the application of advanced robotic systems employing generalized inverse kinematics. The answer is modularity. These are ideal tasks for a modular robot. For instance, many of the tasks are so simple kinematically that a robot with two degrees of freedom will suffice. This robot will almost certainly be lighter and faster than a competing robot with more degrees of freedom performing the same task. A modular robot assembled from standard joint and link modules is cheaper as well. If the task changes significantly, the modules can be rearranged, or more modules can be added to the system. Fulfilling this vision of a truly modular and reconfigurable robot will require an inverse kinematics approach entirely general with respect to the robot's geometry. It will also require virtually all of the supporting technology described later in this chapter.

The food packaging tasks involve light loads, such as a 1 pound box of cookies. Typical forces on the end-effector are from 1 to 10 pounds. These tasks do not require extremely accurate operations. For instance, loading sacks of potato chips into packing boxes requires an accuracy of .5 centimeter. The range of global

accuracy requirements is from 1 to 10 millimeters and the range of required resolutions is from .1 to 1 millimeter. Handling individual pieces of small food products, for instance putting individual pieces of fine chocolates into one or two pound boxes, requires the most accuracy. Food packaging tasks are performed in work volumes of from .1 to 1 cubic meter. The main feature of interest is the very high speed of operation required by these food packaging tasks. Speeds of 1 to 10 cycles per second are required. These high speed requirements suggest that mechanical design will be of the utmost importance for robots involved in food packaging.

7.1.3 Micromanipulation

Micromanipulation tasks are characterized by the need for very precise operation within a very small workspace. Soldering wires from a semiconductor's silicon substrate to the pin-outs is one example of a micromanipulation task. Manipulating individual cells in the biological or animal sciences is another example. Microsurgery is also be a micromanipulation task. In these very precise applications, tolerances during the manufacture of the robot may be important. Manufacturing tolerances cause the actual geometry of the robot to deviate from the specified geometry. Robot metrology measures the actual geometry of the robot after it has been manufactured. The kinematic description of the robot is then modified to include the manufacturing errors. In essence, the robot has been calibrated.

The need for kinematic calibration of a manipulator in turn necessitates an inverse kinematics method general with respect to the robot's geometry. Current

manipulators are designed to have geometries with closed-form inverse kinematic solutions. These closed-form solution methods do not allow for deviations from the ideal geometry. Hence, a solution general with respect to geometry is required for calibrated robots.

Microsurgery is becoming increasingly common. Microsurgery is involved in ophthalmology, neurosurgery, otolaryngology, and cardiovascular. The main benefits associated with using robotics in microsurgery involve increased precision during the operation. The change in motion scales is on the order of 100 to 1 and the change in force scales is on the order of 20 to 1. The forces will be very small on the order of from .01 to 1 ounce. This work must be very precise, with a resolution on the order of .1 to 1 micrometers. For instance, suture material for fine operations is about 5 micrometers. The accuracy required for radial keratotomy is 2.5 micrometers. Repairing a blood vessel of 100 micrometers in diameter requires 6 to 10 sutures of 10 micrometers in diameter. The work volume will be approximately 100 cubic centimeters. Clearly, manipulation on this small scale will require an extremely precise robot.

7.1.4 In-Space Assembly and Construction

Assembling the space station is one example of constructing a large space system. Typical tasks could include putting together truss sections and erecting solar arrays. Attaching habitation modules is another example of a typical task during the construction of large space systems. Mating payloads to booster rockets is also an example of in-space assembly. Smaller-scale tasks include screwing, bolting, clamping, winding, coiling, and locking. Safety considerations lead to

preferring robots over humans in space applications. It is extremely dangerous for humans to be in outer space, thus the more tasks performed by robots or other automation the better. Tasks involving large systems would be very difficult for one person to perform. Several astronauts would likely be required, thus magnifying the danger.

This range of tasks suggest the robotic systems will include small, medium, and large scale robots with capabilities ranging from low to very high. The global accuracy may range from 10 cm to 10 μ m and the resolution may range from .1 cm to 1 μ m. The work volume may range from about 1 cubic meter to 1000 cubic meters. The loads on the robots' end-effectors will vary from about 1 pound to 250 pounds.

The wide range of tasks expected for in-space assembly and construction requires the use of variety of robots each differing in scale. This recommends the use of modular robots that can be assembled from a finite number of joint and link modules. The use of modular robots will reduce costs, increase performance capabilities, and increase availability of the robot systems. The use of modular robots also requires an inverse kinematics algorithm general with respect to the robot's geometry.

Several other aspects of the construction of large space systems call for a very general inverse kinematics solution. Fault tolerance is very important in space applications and general inverse kinematics is essential for third-level fault tolerance. The complexity of the work environment for this application will likely require obstacle avoidance as well. The varied nature of the tasks may well recommend a robot that can be reconfigured to better perform the required tasks.

An inverse kinematics approach general with respect to geometry is required for reconfigurable robots.

7.1.5 Satellite Retrieval and Repair

Satellite retrieval and repair may be divided into three sub-tasks. The first is docking with the faulty satellite. Then the broken satellite needs to be repaired. This might be performed in orbit by astronauts or robots. The satellite could also be stowed for repair on the ground. The final sub task is placing the satellite back into orbit. This subtask will be much easier if the repair is performed in space.

The reasons for wanting to use robots instead of astronauts in the retrieval phase of the satellite repair operation are clear. This phase will require intensive extra-vehicular activity if humans must do all of it. Retrieving satellites using extra-vehicular activity is a very dangerous thing to do with astronauts.

Exactly when a satellite failure will occur, or what the satellite will be doing at the time, is unpredictable. This unpredictability may warrant a robot with several extra degrees of freedom to cope with obstacle avoidance or unforeseen requirements. Fault tolerance is clearly very important in the satellite retrieval application. A modular and reconfigurable robot would also have the advantage that it could be reconfigured to handle a wider range of retrieval operations. There is also another scenario in which a modular and reconfigurable robot could be of use in a satellite retrieval and repair mission. For the retrieval phase, a large number of modules, perhaps including ten or more joints, would be arranged in a serial chain to form a very long arm with a large workspace. This configuration is for the retrieval phase of the operation. For the repair phase. The ten modules could be

divided into two separate arms, perhaps of six and four degrees of freedom. Each of the arms would be mounted to the repair station. The four degree of freedom arm could be used to present the satellite to the six degree of freedom arm. This arrangement would maintain a good amount of dexterity while allowing more precise operation than would a strictly serial robot. Clearly, this type of retrieval and repair scenario is only possible if modularity and reconfigurability are approached from the highest level.

The satellite retrieval and repair operational requirements are similar to the in-space assembly and construction tasks except that after the retrieval phase, the work volume will be smaller. The global accuracy may range from 10 cm to 10 μm and the resolution may range from .1 cm to 1 μm . The work volume may range from about 1 cubic meter to 10 cubic meters. The loads on the robots' end-effectors will vary from about 1 pound to 250 pounds.

7.1.6 EVA Site Preparation

EVA is an acronym for extra vehicular activity; essentially, an astronaut leaves the space craft. Preparing the site of the task is desirable to minimize the amount of time that the astronaut will spend outside of the space craft. Common EVA site preparation tasks involve unstowing tools and bringing them to the work site. A robot could unload the toolbox, bring it to the work site, and then open it. Installing foot restraints and power fixtures are other EVA site preparation tasks. These implements must also be replaced in storage after the EVA task is complete.

The EVA site preparation task is a natural function for robots to perform. The large workspace and variations in the tasks recommend a serial robot. Also,

performing the tasks will likely only place modest demands on the manipulator. This should allow the robots to operate autonomously, or under ground control, to save the astronauts intra-vehicular time as well.

As with any application in space, fault tolerance is an important issue in EVA site preparation. Serial kinematic redundancy is one way of achieving fault tolerance. An inverse kinematics approach general with respect to the robot's geometry is required to achieve this level of fault tolerance. Also, some obstacle avoidance capabilities would likely be desirable for EVA site preparation. Finally, a robot capable of very dexterous motion may relax some design constraints on systems the robot must interact with. Meeting these dexterity and obstacle avoidance goals may require a robot with several redundant degrees of freedom.

The operational requirements on the EVA site preparation task are very modest. The precision will be on the order of 1 centimeter and the resolution does not have to be significantly better than that. The loads will be very low, on the order of from 1 to 5 pounds. The workspace will be large, about 100 cubic meters.

7.1.7 Semiconductor Wafer Handling

Numerous separate operations are required during the semiconductor manufacturing process. Between almost every operation the wafers must be moved among process machines. Complex integrated circuits, such as microprocessors, currently require about one hundred separate operations as the circuit is formed on the silicon substrate. This represents a significant amount of handling operations as the wafers are being processed. Logistics typically dictate that the wafers must also spend some time in storage as they are being processed. The movement into and

out of storage increases the handling load. The bulk of the wafer processing takes place in a clean room environment. The purity of this environment is very important. To reduce contamination of the clean room, the humans who are currently performing these handling operations must always wear special clothing.

The clean room environment is the key factor recommending robots rather than humans for these handling operations. Because the robots do not have to leave the clean room, they do not introduce particulate matter carried in from outside. Also, the robots can operate in a near-vacuum environment, which has been suggested as a method of reducing particulate contamination. Another factor to consider is the effect working in a clean room environment has on a human. The special clothing that must be worn is restricting and uncomfortable. Masks must be worn to cover the face. The handling task itself is very repetitive. All these factors recommend using robots for semiconductor handling tasks.

The semiconductor wafer handling application imposes fault tolerance requirements on a robotic system. This is because each robot must fulfill the handling function for several process machines. While the robot is inoperable, the capabilities of several process machines will be lost. This could interrupt the production of several hundred wafers. Fault tolerant robots, which can continue to perform their tasks even under fault conditions, are required so that the manufacturing process will not be interrupted until servicing is arranged.

The actual semiconductor manufacturing processes (masking, lithography, CVD, PVD, etc.) require extremely high tolerances. These submicron tolerances are far beyond the capabilities of any serial robot arm. Transferring wafers between specialized process machines is the main application for robots in the semiconductor

manufacturing process. This transfer process requires a precision of .1 to .01 centimeters for loading the wafers into susceptor ports in the machines. There is essentially no difference in the requirements for precision and resolution. Each wafer may require 100 or more separate processes during its manufacture. To accommodate this many process machines, the total workspace is large, on the order of 1000 cubic meters. Though each wafer is very light, the robot may be asked to carry a cassette of multiple wafers which weighs on the order of 10 pounds.

7.1.8 Long Reach Arm

This long reach arm concept is in response to the DOE plan to develop key technologies for efficient, timely, and cost-effective clean-up of existing nuclear waste sites and the decommissioning of various nuclear fuel facilities. This long reach arm concept is part of a larger national effort structured at DOE headquarters through ORNL and Sandia National Lab to create a special class of robot systems to work in the hazardous environment of nuclear waste sites with minimum human intervention to bring down the level of ORE that would be experienced by direct operator involvement. This concept effort expands on technology base activities at The University of Texas (approaching \$1,800,000/year of funding by DOE and other federal and state agencies) in order to produce prototype technology which would otherwise not be achievable because of limits in resources. Specifically, strong partnerships with industrial development groups (Lockheed Austin Division), government labs (ORNL, Sandia) and other universities (Michigan, Tennessee, Florida) are being integrated in this team to respond to tasks associated with

inspection, sampling, transport, etc. in clean-up tasks associated with restricted spaces in nuclear waste tanks, low level waste barrel storage, etc.

The prototype long reach arm will be based on a modular structure of very lightweight (but stiff) tetrahedra some of which will be actuated by 7 specially designed motor spindle drives which are unusually lightweight, stiff, compact, and nearly frictionless. There will be 3 rotary actuators used (based on an existing fault tolerant prototype) at the base, at the wrist, and at the end-effector. The design will be highly modular such that the failure of any one actuator will not prevent continued operation (until the system can be taken off-line for maintenance) and it can be reconfigured (by interchanging links, actuators, tools, etc.) and up-dated without major changes to the operating systems (SFW and HDW). This level of technology will dramatically accelerate the implementation of long reach arm technology ultimately reducing its costs, making modular repairs feasible, and reducing the threat of obsolescence by allowing rapid "tech mods" to the system.

The actuator that drives the joints is the fundamental component of a robot and the long reach arm will put special demands on the actuator. The actuators in this robot will be loaded with extremely large forces primarily in tension. Because of these very large forces, the actuators for this robot must be almost frictionless. Also, the length of this robot will produce tremendous amplification of small errors at the actuators to very large errors at the end-effector. This amplification demands the use of a very precise actuator. To meet these demands, the University of Texas will build on its experience with advanced robot actuators to specialize a revolutionary design initially developed at the German Aerospace Establishment (DLR), with which the University of Texas Robotics Research group has a long

standing and very good relationship. This design uses screw planets and a spindle to provide an extremely compact, lightweight, and powerful linear actuator. The resulting actuator will be precise, fault-tolerant, and capable of developing the extremely large forces required by the long reach arm. Some specific characteristics include: high reduction (600:1), very fine thread pitch (0.2 mm/revolution), and 96% improvement in efficiency over standard spindle-nut. The long reach arm will work in a volume of about 10000 cubic meters. The arm will have capabilities on the order of a course industrial robot. The loads will be from 1 to 250 pounds. The required global accuracy is about 10 cm and the resolution 1 cm.

7.1.9 Inspection at Nuclear Facilities

Inspection operations are very common at modern nuclear facilities. Drums, plumbing, tanks, valves, and pumps all must be routinely inspected. The sensing part of the inspection process may be fairly complex and involve checking for many different signs of problems. Radiation tests will clearly be common. Certain equipment will be checked for cracks or other signs of excess stresses. Stored waste may need to be checked for excess heat generation or gas leakage. The actual path that the sensors must follow during the inspection process is quite complex. Many of the sensors need to be moved directly over the entire exterior surface of the equipment being tested.

The dangers associated with human exposure to radiation is the main reason robots are desirable in nuclear environments. There are federally mandated limits on the amount of radiation to which a worker may be exposed. As the inspection tasks are routine and must be performed regularly, the total exposure of workers to

radiation continues to accumulate over time. Some of the inspections, especially those of the waste, will have to be repeated for a long time into the future. The cumulative exposure factor also recommends robots for inspection tasks at nuclear facilities.

There are several reasons only a very advanced robot will be capable of performing many of the inspection tasks at nuclear facilities. Just following the required sensor path requires a very dexterous arm. The complex and cluttered work environment necessitates advanced obstacle avoidance capabilities. Reaching through port holes or passageways may also be required as part of some inspection tasks. Fault tolerance issues will be important in almost all applications of robots in nuclear facilities. Finally, the inspection tasks will likely be so varied that no one robot geometry will suffice. A spectrum of robots with varied scale, yet based on a standard set of joint and link modules, could be assembled to perform this wide range of inspection tasks. This wide range of demanding requirements will only be satisfied by a very advanced system of robots, of which a general inverse kinematics approach will certainly be a part. The inspection process requires the robot to carry a suite of sensors. The combined weight of these sensors will represent a load of from 10 to 25 pounds. The inspection process is low precision, requiring a global accuracy of 1 cm and a resolution of .1 cm. The work volume may be as large as the volume of the plant, which is approximately 100,000 cubic meters.

7.1.10 Decontamination and Dismantlement

Decontamination and Dismantlement at nuclear facilities is part of the environmental restoration and waste minimization (ER&WM) process. The

physical tasks associated with ER&WM will range from simple inspection and transport to complex disassembly, cutting, welding, rigging, packaging, repair of failed equipment, etc. Some of the tasks will be underground (piping), in large volumes (tanks), in cluttered environments (reactors), etc., and will require a diverse class of manipulator systems. Some of these systems can be quite simple (3-DOF planar devices) to multiple arms of extra DOF (8 to 10 DOF) capable of precision tasks under varying forces in a cluttered environment, all operating remotely sometimes with human intervention through a manual controller or through supervisory software taught to perform repetitive tasks by human input. The work environment for the slave may contain high levels of moisture, corrosive chemicals, radiation, dust, etc. Hence, decontamination, maintenance, replacement of failed parts, etc., must be planned as part of the manipulator technology in its foundation architecture.

The ER&WM tasks addressed here involve the deployment of robots for the selective removal of equipment. This part of ER&WM presents many challenges due to the complex environment and demanding tasks. ORNL developed the robotic Dual Arm Work Module as a vehicle for developing the technology necessary for robots to perform this task. The DAWM has two arms, each with multiple redundant degrees of freedom. This robot has the ability to perform a task in a variety of ways while simultaneously obeying hardware and environmental constraints. Due to the complexity of the application there are a variety of important criteria that impact the robot's ability to perform its task. This list of potentially important criteria includes: joint travel and torque limits, load-carrying capabilities, precision operation, obstacle avoidance, and dexterity issues. These

multiple criteria provide a more balanced basis for decision making that ultimately improves computational reliability and task performance capabilities.

The most challenging task for the robot in the selective removal of equipment is dismantling the equipment. This dismantling may involve forces as high as 250 pounds. The task is moderately precise, requiring a global accuracy of 1 mm and a resolution of .1 mm. Though the work may be performed throughout the nuclear facility, the work volume for each dismantling task will be approximately 10 cubic meters.

Table 7.2 This table presents a summary of the requirements for the ten advanced applications.				
Application	Load (lbf)	Precision (m)	Resolution (m)	Work Space (m³)
complex assembly	1 - 25	1 - .1 *10 ⁻³	.1 - .01 *10 ⁻³	.1 - 1
food packaging	1 - 10	.01 - .001	1 - .1 *10 ⁻³	.1 - 1
micromanipulation	.001 - .1	1 - .1 *10 ⁻⁶	.1 - .01 *10 ⁻⁶	1 *10 ⁻⁴
in-space assembly and construction	1 - 250	.1 - .0001	1 - .00001	1 - 1000
satellite retrieval and repair	1 - 250	.1 - .0001	1 - .00001	1 - 100
E.V.A. site preparation	1 - 5	.01	.01	100
semiconductor wafer handling	1 - 10	1 - .1 *10 ⁻³	1 - .1 *10 ⁻³	1000
long reach arm	1 - 250	.1	.01	10000
inspection at nuclear facilities	10 - 25	.01	.001	100000
decontamination and dismantlement	1 - 250	1 *10 ⁻³	.1 *10 ⁻³	10

7.2 SUPPORTING TECHNOLOGIES

Table 7.3 A brief description of ten supporting technologies required for the application of generalized inverse kinematics.

Technology	Description
computer hardware	the computing machinery on which the inverse kinematics algorithms will be developed and executed
computer architecture	the structure of the computer, both hardware and software
computer software	the language used to communicate the inverse kinematics algorithms to the computer
performance criteria development	continued development of performance criteria primarily in the area of physical interpretation
sensors	sensors extract information about the robot and its environment
data communication	information exchange amongst the sensors, the controller, and the actuators
control	the software and hardware which must operate and interface with the actual robot
mechanical architecture	the structure and granularity of the robot, primarily issues of modularity
mechanical design	the conception and detail of the robot's physical structure
robot actuators	structural units supplying the mechanical power at the robot's joints

The application of generalized inverse kinematics depends on the support of these ten technologies, which fall into four basic groups. The computer technologies include: the hardware, software, and architecture required for the actual execution of the inverse kinematics algorithms. The electronic technologies include the robot's sensors, controllers, and communications systems. The mechanical technologies include the robot's mechanical design, actuators, and

construction. Intellectual development is the final technology group. This group includes computer algorithms and performance criteria. This discussion of the technologies includes: the performance required for the application of generalized inverse kinematics, the performance available, and the changes in performance levels over the past ten years. The discussion also includes features of the technologies particularly important to their support of generalized inverse kinematics.

7.2.1 Computer Hardware

Any algorithm for solving the generalized inverse kinematics problem will ultimately execute on some sort of computer hardware. Today's most common architecture for computer hardware consists of three major subsystems. These three major subsystems are the central processing unit, the long-term storage device, and the input/output pathways. The "brains" of the computer is the central processing unit, which is commonly called the CPU. This discussion considers RAM (short-term memory used by the CPU) part of the CPU. A long-term storage device, such as a hard disk, is required to save programs and data for future use. The final subsystem performs what is called input and output. Input and output, or I/O, is where information and instructions are exchanged with the computer. A user will typically instruct the computer through the keyboard, a pointing device, or voice input. The computer responds through a video display or audio feedback. The visual display of information through computer graphics is the major I/O function this section describes. Each of these three subsystems will be discussed in terms of the performance required by a generalized inverse kinematics implementation, the

performance which is currently available, and the changes in performance levels which have occurred over the last several years.

Choosing some reasonable numbers for the system's parameters allows the calculation of the CPU performance required for the real-time application of generalized inverse kinematics. Thirty Hertz is commonly cited as the required frequency for solving the inverse kinematics problem in real-time (Hernandez and Tesar, 1989), and thus will be used as the definition of "real-time" for this discussion. A geometry with seven degrees of freedom will be used in these calculations for two reasons. The first is that a geometry with seven degrees of freedom is a common example found in the literature. The second is that criteria calculation performance benchmarks are available for the Robotics Research Corporation's kb1207 manipulator, which has seven degrees of freedom. The following calculations are made for a direct search inverse kinematics algorithm general with respect to the robot's geometry. Geometric simplification, such as that provided by a spherical wrist or shoulder, would significantly reduce the demands on the CPU. The inverse kinematics algorithm used in this example incorporates a factorial perturbation pattern and steepest descent decision making.

The direct search method of solving the generalized inverse kinematics problem may be broken into two major parts, the local exploration and the decision making. Generating the joint perturbations during the exploration is simply a short series of addition and subtraction. The decision making, in contrast, involves calculating the performance criteria values for each of the trial solutions and represents far more computational overhead. The computational demand depends upon which performance criteria the solution incorporates. Van Doren and Tesar

(1992) present a set of benchmarks that separate the criteria into several categories. The benchmarks are for the Robotics Research kb1207 robot and the Silicon Graphics 25TG personal workstation. The amount of time it took for one calculation of an entire criteria category is presented. For instance, the set of Jacobian-based criteria took .00822 seconds to calculate. The set of Hessian-based criteria took .01614 seconds to calculate and the entire suite of twenty-nine criteria took .1594 seconds to calculate (Van Doren and Tesar, 1992). The 25TG personal workstation is rated at 1.6 MFLOPS, so these timing figures may be written as seconds per MFLOP as shown in Table 7.3.

Table 7.4 The amount of time per MFLOP for each calculation of the values associated with three criteria sets	
Criteria Set	Benchmarks
Jacobian-based	.00514 seconds/MFLOP
Hessian-based	.01009 seconds/MFLOP
29 criteria	.09963 seconds/MFLOP

For a robot with seven degrees of freedom, the factorial perturbation pattern generates 128 trial solutions. Each of these trial solutions must be addressed at least thirty times a second for real-time performance. The demand, in MFLOPS, on the CPU may now be calculated. For Jacobian-based criteria only, the required CPU performance is

$$\text{MFLOPS} = (128 * 30) \frac{\text{calculations}}{\text{second}} * .00514 \frac{\text{seconds}}{\text{calculation}} = 19.7$$

For Hessian-based criteria only, the required CPU performance is

$$\text{MFLOPS} = (128*30) \frac{\text{calculations}}{\text{second}} * .01009 \frac{\text{seconds}}{\text{calculation}} = 38.7$$

For all twenty-nine criteria, the required CPU performance is

$$\text{MFLOPS} = (128*30) \frac{\text{calculations}}{\text{second}} * .09963 \frac{\text{seconds}}{\text{calculation}} = 382.6$$

These calculations assumed that all of the criteria must be calculated at each point along the path. Thus, in a sense these numbers represent a path planning argument. Nonetheless, these calculations do give an impression of the magnitude of CPU performance required to address the generalized inverse kinematics problem in real-time. In a larger sense, this type of calculation is useful for determining the performance requirements for addressing any multiple criteria decision making problem. For instance, if one-hundred choices must be faced at a rate of thirty Hertz and each choice takes .1 seconds per MFLOP to evaluate, then the required CPU performance is

$$\text{MFLOPS} = (100*30) \frac{\text{evaluations}}{\text{second}} * .1 \frac{\text{seconds}}{\text{evaluation}} = 300$$

The CPU performance available on the market today may be gleaned from available manufacturer's data. The MIPS corporation's 100 MHz R4400 CPU is rated at 22 MFLOPS (Gorey, 1992). The DEC corporation's 200 MHz Alpha CPU is rated at 200 MFLOPS and the Intel corporation's 50MHz i860 CPU is rated at 100 MFLOPS (Comerford, 1992). There are many other CPUs available from a variety of manufacturers, but these three give an impression of the range of CPU performance typically available to today's engineer. Each of these CPUs is found in computers costing from \$5,000.00 to \$100,000.00. The main difference in the cost

is not due to the CPU, but rather due to other computer hardware, primarily for graphics display.

Comparing the available CPU performance with the performance required for solving the generalized inverse kinematics problem shows that today's CPU's are capable of incorporating some criteria into real-time solutions.

Though incorporating a full-complement of performance criteria in a general solution will take more CPU performance than is currently available, the recent pace of CPU advancements is truly amazing. For example, in May of 1990, the top-of-the-line MIPS CPU was the 25MHz R2000 which was rated at 3.3 MFLOPS (Gorey, 1990). This CPU was only available in computers which cost at least \$50,000.00. Today, the 100MHz R4400 is rated at 22 MFLOPS and is available in computers costing as little as \$5000.00. The price-to-performance improvement in only three years is almost one hundred times! Even if this rate of improvement decreases dramatically, and there are no indications that it will, incorporating a full-complement of performance criteria into the real-time solution of a multicriteria decision making problem should be possible within the next few years.

An alternative to repeatedly calculating performance criteria values for each trial solution is to calculate the values once, over the entire workspace, and then save the values in a permanent storage device. The main concern here is the amount of data that will need to be stored. The amount of data is dependent upon the number of degrees of freedom, the range of motion at each joint, the resolution over the range, and the precision with which each of the data points is represented. The total amount of storage space required is then

$$\text{storage space} = \left(\text{range} * \frac{\text{precision}}{\text{resolution}} \right)^{\text{DOF}} \quad (7.1)$$

Substituting reasonable numbers gives an impression of the storage space requirements. For instance, a robot with seven rotational joints each having a range of 180 degrees, is representative of current research robots. A precision of four bytes allows each data point to be represented to seven decimal digits (assuming the common digital representation of one sign bit, seven exponent bits, and a twenty-four bit mantissa). With these choices (DOF = 7, range = 180 degrees, and precision = 4 bytes) the storage space requirements for three different resolutions will be calculated. For a resolution of five degrees, the storage space required is

$$\text{storage space} = \left(180 * \frac{4}{5} \right)^7 = 1.28 * 10^{15} \text{ bytes}$$

For a resolution of one degrees the storage space required is

$$\text{storage space} = \left(180 * \frac{4}{1} \right)^7 = 1.00 * 10^{20} \text{ bytes}$$

For a resolution of .001 degrees the storage space required is

$$\text{storage space} = \left(180 * \frac{4}{.001} \right)^7 = 1.00 * 10^{41} \text{ bytes}$$

Of these examples, only the resolution of .001 degrees is sufficient for directly controlling a robot. The other two resolutions, five and one degrees, would require interpolation.

Unfortunately, these storage space requirements are far beyond what is available with the current state-of-the-art. Recently, The IBM corporation announced that they had successfully demonstrated optical storage densities of 387

million bits per square centimeter (Zorpette, 1993). This density will allow approximately 6.5 gigabytes of information to be stored on one 5.25 inch disc. This is still many orders of magnitude less than what is required to store criteria values generated over a complete workspace, even at very coarse resolutions.

It is, however, interesting to note the recent advancements in data storage techniques. During the mid-1980's, the standard storage medium was a 5.25 inch magnetic disc that could store about 500 kilobytes of data. Assuming that the 6.5 gigabytes of storage on a 5.25 inch optical disc becomes available in the mid-1990's, the storage capacity will have increased by four orders of magnitude in just ten years. The ability to store the amount of data represented by Equation (7.1), even at coarse resolutions, in the near future is unlikely.

Computer graphics is an essential part of a complete generalized inverse kinematics system. As an interface, computer graphics queries for information and provides visual feedback. The most demanding form of computer graphics is computer animation. For a smooth animated display, specialized graphics hardware is currently required. The capabilities of the graphics hardware dictate the amount of information that can be displayed.

A graphical user-interface is an efficient and intuitive means of querying for information. Even the most basic personal computer is capable of supporting a graphical user interface. One method of providing visual feedback shows an animation of the robot while simultaneously displaying criteria values in the form of a bar graph, line graph, or numerical display. This type of display requires graphics performance of approximately 30×10^3 polygons/second (Hooper and Tesar, 1990). Another type of display shows parameter values by varying the shading along the

surface of the animated robot. For instance, stress levels along the robot can be shown with artificial coloring of the animated robot's surface. This type of display might also show how kinetic or compliance energy is circulating throughout the system. For an animated robot, parameter display by surface shading is estimated to require graphics performance of 30×10^6 polygons/second (Hooper and Tesar, 1990).

Computers built by the Silicon Graphics Corporation provide a benchmark for available graphics hardware. The Silicon Graphics company is the currently acknowledged leader in computer graphics hardware. The highest performance graphics hardware available from Silicon Graphics is the Reality Engine, which is capable of 1.1×10^6 polygons/second (Gorey, 1992). This graphics performance is easily capable of displaying an animated robot and parameter values via a graphical user interface. More advanced display of criteria values through surface shading requires further hardware development.

Unfortunately, the state of the art in graphics performance, at least as measured by polygons per second, has recently been stagnant. For instance, in May of 1990, 1.0×10^6 polygons per second was available from Silicon Graphics (Gorey, 1990). Today, the performance has only increased to 1.1×10^6 polygons/second (Gorey, 1992). Even the cost of these machines has not changed much in the last several years. In 1990, the minimum cost for 1.0×10^6 polygons/second was \$94,900.00 (Gorey, 1990). Today, the minimum cost for 1.1×10^6 polygons per second is \$42,900.00 (Gorey, 1992). It may be several years before the real-time display of criteria values by surface shading, estimated to require 30×10^6 polygons/second, is possible.

7.2.2 Computer Architecture

Any algorithm for solving the generalized inverse kinematics problem will ultimately execute on a computer. This section addresses the question of whether there is a specific computer architecture especially suited for solving the generalized inverse kinematics problem in the context of multicriteria decision making.

Computer architecture includes such things as: the layout of the bus, digital logic, multiple processing units, and system input/output. Computer architecture includes both hardware and software issues. The computer scientist is interested in all of these subjects. The subject of multiple processing units has the greatest impact on the speed of solving the multicriteria decision making problem. Typically, multiple processing units are arranged in either serial or parallel architectures. The serial architecture is often called a pipeline. A computer pipeline is roughly analogous to an automobile assembly line. Many automobiles are assembled at the same time, but different tasks are performed at each workstation. The automobile is finished when it comes off the end of the of the assembly line. A computer pipeline is suited for problems where many different operations must be performed on each data point; with each operation depending upon the result from the previous operation. Computer pipelines are often used as graphics subsystems because displaying each pixel requires several sequential operations, such as: scaling, clipping, and shading. On the other hand, a parallel architecture is suited for problems that may be decomposed into a set of smaller subproblems that do not depend on results from each other. The parallel architecture is best for multicriteria decision making problems.

The parallelizable percentage of the problem is one indicator of the improvement expected from a parallel architecture. The percentage is measured relative to the amount of computation time spent on each part of the problem. For instance, if parallelization makes half the problem twice as fast, the overall improvement in speed is twenty-five percent. The performance increase as a function of the number of parallel processors is the other major indicator of improvement possibility. In the ideal case, there is a one-to-one correspondence. Two processors solve the problem twice as fast and three processors solve the problem three times as fast, etc. Judging by both of these indicators, the multiple performance criteria decision making problem is ideally suited to a parallel computer architecture. This is because the decision making strategies spend the vast majority of computation time calculating criteria values for each of the choices. The calculations in Section 7.2.1 assume that the algorithms spend all of their time calculating criteria values.

Both of the above observations bode very well for using parallel computers to solve multicriteria decision making problems. The first result is important because parallel computers with several processors are beginning to become available to the average engineer. For example, the Silicon Graphics company recently introduced the Onyx line of parallel computers that have from two to twenty-four parallel processors. The Onyx computers are engineering workstations with state-of-the-art microprocessors. These computers cost from \$100,000 to \$750,000. The second result is important because massively parallel computer architectures are currently dominating the supercomputing industry. Both the Cray corporation and Thinking Machines Inc. are currently developing parallel computers

with hundreds of microprocessors. A multicriteria decision making problem which must face a very large number of choices seems ideally suited to this type of computer architecture. Though these massively parallel computers are currently only found in the supercomputer domain, the recent trend has been that supercomputer performance comes to an engineer's desktop in only a few years. For instance, the Cray X/MP with one vector processor has a peak performance of 240 MFLOPS. This was state-of-the-art in the 1980's. Today, a single ALPHA CPU has virtually this same peak performance, 200 MFLOPS. As has been discussed, the alpha CPU is found in engineering workstations costing less than \$10,000.00.

7.2.3 Computer Software

The inverse kinematics algorithms and multicriteria decision making strategies are coded into computer software for simulation and application. Developing software is by nature time-consuming. Thus it is important to get the most benefit from any software development effort. Certain measures correspond to the relative benefit expected from a given software development effort. These measures include the software's generality, portability, and efficiency.

The ability to expand and upgrade the software to include new performance criteria and new decision making strategies measures the generality of the software. In the sense that developing software is both time-consuming and expensive, software development may be compared to robot development. A modular robot architecture has been proposed as a method of decreasing the time and expense associated with developing a robot. A modular approach to software development

has similar benefits. Essentially, the modular approach allows the incorporation of new software modules without redesigning the entire software package. Keeping the software modules separate instills a degree of reconfigurability. For instance, one module might calculate the Jacobian based criteria values while another module could calculate the factorial perturbation pattern. Different criteria modules could be arranged with different decision making modules to best configure the software given a specific task requirement. The software can incorporate new performance criteria and decision making strategies as they are developed. Clearly, the interfaces to the software modules are of the utmost importance. The inputs and outputs of each module must be carefully designed and documented. Reduced obsolescence of the software package is the reward for this effort. Maintaining a separation of various software packages also increases generality. The various software packages can communicate with one another through data files, shared memory, or by some other means. Several examples of the benefits associated with maintaining a separation of software packages exist in the University of Texas Robotics Research group. The robo_cad software, for instance, was originally designed as an interactive configuration environment for modular robots. Because the software incorporates the ability to communicate through datafiles and shared memory, it has since been used for real-time animation, design optimization, obstacle avoidance research, path-planning, and inverse kinematics development. Another example is the NDOF dynamic modeling software. Since this software is general, with the specific robot and path characteristics communicated through data files, it remains useful after almost ten years.

Software is portable if it runs on different computing platforms without modification. Due to the rapid pace of computer hardware improvements, software portability is currently of the utmost importance. Portable software takes advantage of improvements in hardware automatically. Portable software may sacrifice some execution speed at first, but before long, portable software will likely run much faster than any software bound to an obsolete computing platform. Of the current computer languages, “C” is probably the most portable. There are “C” compilers for personal computers, specialized controllers, engineering workstations and supercomputers. It is also possible to take advantage of specialized computer hardware while still maintaining portable software. The Code 2.0 graphical programming language allows software to take advantage of parallel architectures while still maintaining portability.

The efficiency of the software relates to execution speed without sacrificing generality or portability. Efficiency may be loosely divided into algorithmic efficiency and execution efficiency. Algorithmic efficiency is associated with the formulation and coding of the algorithm itself. For instance direct search can achieve some performance increase by not calculating criteria values for trial solutions that have already been explored. Also, many different performance criteria depend on the singular value decomposition of a matrix. Significant performance improvement is achieved if the singular value decomposition is performed once and the results distributed to the various criteria calculation modules. It is probably unwise to seek these performance improvements if they come at the expense of generality or portability. Careful formulation of the computer code and the underlying numerical representation often improves

execution efficiency. Integer performance, for instance, is often faster than floating point performance. Formulating an algorithm using integers, when possible, is a way to increase performance without sacrificing generality or portability.

As another example of improving computational speed without sacrificing portability, consider Table 7.5. This table shows that the rapidly changing criteria need to be sampled the most frequently, but slower changing criteria need to be sampled less frequently. The order of the criteria directly influences the speed at which the criteria are changing. The higher order criteria change more rapidly and must be sampled more frequently. This must be balanced against the relative importance of the criteria to the application as determined by the robot's operator. Taking into account the order of the criteria to determine the sampling rate will improve computational speed without sacrificing portability.

Table 7.5 The relative sampling frequency for the performance criteria as determined by their order.		
Criteria Basis	Sampling Rate	Order of Criteria
position	low	zero order
$\underline{g}$	modest	first order
$\underline{h}$	medium	second order
kinetic energy	medium	second order
$\underline{g}\underline{h}$ product	high	third order

7.2.4 Performance Criteria Development

Clearly, performance criteria are a fundamental part of generalized inverse kinematics. The University of Texas Robotics Research Group has made a

considerable investment in criteria development. The results of this effort are over thirty geometric, inertial, kinetic, and compliance criteria. There are two avenues for continued performance criteria development. The first is to generate more criteria and the second is to improve the criteria that have already been identified. Both of these avenues are worthy of exploration. Of these, the most important is developing multiple physical, yet independent, meanings. As for generating more criteria, the largest void is the lack of criteria associated with obstacle avoidance. This is not for lack of trying. Two Masters theses within the University of Texas Robotics Research Group have been dedicated wholly to obstacle avoidance and two more are in progress. Currently, the major results from this work is that the obstacle avoidance issue is very difficult to formulate mathematically and very difficult to solve numerically. This has led to the realization that progress on the obstacle avoidance issue can only be made by building on a solid foundation of fundamental principals. As far as improving the existing criteria, a physical interpretation of the criteria's meaning is still missing.

The importance of obstacle avoidance criteria to the application of advanced robotics cannot be overemphasized. Most of the applications discussed in this chapter require reaching around objects or through small passageways. These applications require obstacle avoidance by nature. The use of hyper-redundant robots to achieve fault-tolerance also requires obstacle avoidance. This is because the portion of the robot between the base and the end-effector is relatively long and could easily experience collisions even though the end-effector follows a collision-free path. Controlling both the end-effector of the robot as well as the intermediate joints and links represents a tremendous demand during teleoperation. The human

should control the end-effector while collisions involving the intermediate portion of the robot are avoided automatically. If the costs of the robot's actuators and their controllers decrease, then the benefit/cost factor will increase and redundant robots will become much more realistic for applications. Further advances should decrease the weight and volume of the actuators as well. If the robot is slimmer and lighter, then obstacle avoidance becomes much more feasible.

Assessing the need for obstacle avoidance criteria is not nearly as difficult as formulating a course of action for developing useful criteria. Developing useful obstacle avoidance criteria will involve both analytic and abstract reasoning. Analysis is required to express the obstacle avoidance criteria in terms of fundamental properties. First, second, and even higher order geometric properties will likely be required. Incremental improvements towards the realization of real-time obstacle avoidance can only be achieved by building on a sound foundation of fundamental principles. Developing obstacle avoidance criteria also requires a degree of abstract reasoning. For instance, when should an obstacle in the workspace even influence the decisions? If there is no collision, the obstacle does not create a problem. How close should the manipulator be allowed to come to the obstacle? Current efforts address these questions by simply operating the robot at very low speeds and decreased work volumes (NASA, 1992). This approach limits the robot's performance to extremely modest tasks. In some cases, it might be desirable that the manipulator even push against an obstacle. The issue of reaching through an access port is also important. For instance, if the manipulator is not allowed close to an obstacle, a wall with an access port in it may look the same as a

wall with no port. These are just a few of the issues that must be addressed on an abstract as well as an analytic level.

Incorporating performance criteria will only result in enhanced performance if one or more physical interpretations are attached to each of the criteria. These interpretations are the main area yet to be addressed in performance criteria development. A physical interpretation allows the operator to intelligently choose and rank the performance criteria for a given application. For instance, kinetic energy partition values presented in purely geometric terms have several important physical interpretations that can guide the operator. Rapidly changing kinetic energy partition values indicate rapid energy movement from one part of the system to another. This destructive movement implies: large internal forces, high torque demands, high force-induced deflections, and poor system responsiveness. Table 7.6 lists some more important physical interpretations that may be associated with the performance criteria.

Table 7.6 Physical interpretations that may be associated with the criteria.
Deflection Energy Partition Values
Backlash, Hysteresis, and Frictional Distribution
Sensitivity to Temperature Variations
Internal Force Magnification
Criteria Affecting Minimum Step
Criteria Affecting Resolution
Criteria Affecting Accuracy
Criteria Affecting Control (Saturation, Overshoot, Oscillations, etc.)

Achieving a valid physical interpretation of the performance criteria requires a focused effort. One result of this effort will be the use of lessons-learned from experts as part of a supporting expert system. Ultimately, the physical interpretation requires experimental validation. This experimental validation involves an extensive instrumentation and metrology effort.

7.2.5 Sensors

Accurate sensor feedback is crucial for the application of advanced redundant robots. Sensors extract information about physical realities and typically use a transduction process. For instance, a sensor may generate a voltage level corresponding to a displacement. The voltage is interpreted by a computer. Because sensors provide information about the physical environment, sensor inaccuracies may result in an incorrect response. This section further discusses the fundamental importance of sensors to the application of generalized inverse kinematics. Some basics concerning sensor design, sensor placement, and the interpretation of sensor data are also discussed.

Integrating advanced sensor technology is very much intertwined with applying generalized inverse kinematics. Thus far, the inverse kinematics problem has been discussed in an open loop context. That is, a given set of joint displacements results in a known displacement of the end-effector. Unfortunately, this is not the case in an actual application. Errors due to manufacturing tolerances, temperature variations, and unmodeled phenomena (such as backlash) cause a deviation between the predicted and the actual response of the robot. Since errors accumulate in a serial structure, a redundant robot tends to accumulate more errors.

A hyper-redundant robot accumulates even more errors. Sensors will identify these deviations from the predicted response. A course of action may then nullify the errors. The generality of the inverse kinematics algorithms developed in this dissertation allows the incorporation of a wide variety of sensor data. For instance, temperature sensors along the links of the robot can identify changes in the geometry of the robot due to variations in temperature. The sensors may then communicate these changes to the inverse kinematics algorithm that will incorporate this information while finding the required set of joint displacements.

There are many factors to consider when choosing a sensor. Resolution, range, precision, stability, and noise immunity are common measures of a sensor's performance. The resolution is the smallest reliably detected change in input. The range is the difference between the maximum and minimum input states. The precision is given by the equation

$$\text{resolution} * \text{range} = \text{precision} \quad (7.2)$$

Equation 7.2 can be used to choose a sensor when the resolution and range are determined by the application. For instance the range for one full revolution is 360°. Assuming the application requires positioning the joint to within .001 degrees, the required precision is

$$1000 \frac{\text{counts}}{\text{degree}} * 360 \text{ degrees} = 360,000 \text{ counts} \quad (7.3)$$

Thus, the required precision is 360,000 decimal digits or 19 binary bits. Though in actual operation the precision at the robot's end-effector is the important concern, calculating the precision at the end-effector relating to the precision at a joint is specific to both the robot and the robot's position within its workspace. The sensor

must be stable and not drift during normal operation. Finally, the sensor should be immune to noise interference. This is because a system is just as sensitive to noise coupled through the sensor as it is to the controlling input variable. Figure 7.1 shows the block diagram of a simple control loop with noise in the feedback.

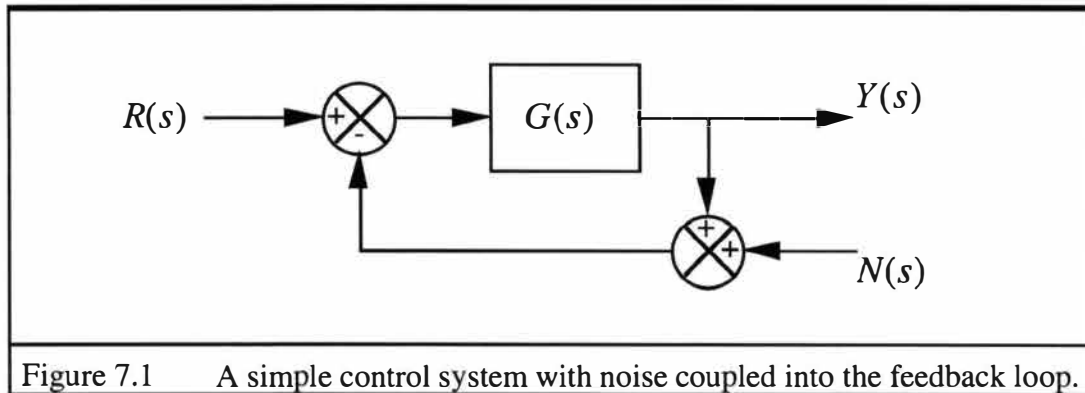


Figure 7.1 A simple control system with noise coupled into the feedback loop.

The transfer function between the controlling input, $R(s)$, and the output, $Y(s)$, is

$$\frac{Y(s)}{R(s)} = \frac{G(s)}{1 + G(s)} \quad (7.4)$$

The transfer function between the noise, $N(s)$, and the output, $Y(s)$, is

$$\frac{Y(s)}{N(s)} = \frac{G(s)}{1 + G(s)} \quad (7.5)$$

The transfer function is the same for the noise as it is for the controlling input variable. Thus the noise has effectively the same control over the system as the input. The noise can be removed with filters if it is in a frequency band different than the frequencies of interest. Preventing the noise from entering the system is a better approach. This may be accomplished by choosing sensors, such as optical sensors, which are by nature immune to noise interference during the transduction

process. Careful shielding and grounding then minimizes the noise corrupting the signal after transduction.

The layout of the sensors within the system is also important. Essentially, the sensors are placed so as to measure the parameters of interest. Typical sensors at the joints measure positions, velocities, forces, and torques. Sensors at the end-effector are commonly of the tactile, proximity, and force/torque variety. External sensors, such as ultrasound or laser, measure the position and orientation of the end-effector in workspace coordinates. Sometimes there may be more than one sensor of the same variety in almost the same location. For instance, angular sensors at the joints might be placed on both sides of the gear train. The sensor on the motor-side of the gear train provides stability in the control loop (backlash in the gears adversely affects stability), while the sensor on the output side measures the actual angular output of the joint.

Finally, the issue of sensor fusion must be addressed. The use of multiple sensors often provides redundant information. For example, position sensors at the joints and external end-effector sensors both provide information about the position and orientation of the end-effector. The most accurate sensor data should be chosen. Choices based on intuition may be acceptable when there are only a few redundant sensors. Statistical techniques are common when there are many redundant sensors.

7.2.6 Data Communication

Data communication within a robot having many joints and sensors is clearly an issue. The signal carriers must be routed over a demanding physical

pathway that includes turning joints and many tight spaces. Reconfigurable robots compound the problem by requiring connections at each of the module interfaces. The total number of pathways required to carry the information must be addressed. Also, the noise immunity of the pathways and their ability to withstand the demands of the robotic environment are important considerations. Data is typically carried by radio waves, conducting wires, and fiber optics.

The total number of conductors devoted to carrying information in even a simple industrial robot is amazingly large. The Cincinnati Milacron T3-776, for example, has over seventy-five conductors devoted exclusively to carrying data. This robot has six joints and the only sensors are one tachometer and one course/fine resolver. An advanced redundant robot with a wide variety of sensors could easily require hundreds of data pathways. Routing a single conductor for each of these data pathways would be absurd. The solution is to multiplex the information and transmit it over fewer physical carriers. The bandwidth of the carrier determines the amount of information placed on each line. Redundant lines may be required for fault tolerance.

Radio waves are one method of carrying data from place to place. The elimination of routing and connection problems is the main benefit associated with using radio waves. There is also no requirement for multiple lines for fault tolerance, just multiple transmitters and receivers. Because there is essentially only one transmission medium, only a limited amount of data can be carried over radio waves. There also may be problems associated with interference.

Electrical conductors are the most common method of carrying data through robots. Because a tremendous variety of machines use conductors, there has been

considerable development of conductor technology. This includes very flexible conductors capable of withstanding the twisting and turning of revolving joints. Because conductors usually require connectors, there has also been considerable connector development. Multiplexing techniques allow conductors to carry many signals and extra conductors can be used if there is more information than will fit on a single conductor. Carefully shielded conductors have good noise immunity.

A single fiber optic can carry very many signals. This is because fiber optics operate at frequencies in the light spectrum. The exact number of signals depends on the signals' bandwidth, but for signals with a bandwidth to 10 KHz a single fiber optic can carry about 100 individual signals. Fiber optics also have a very good noise immunity. The difficulty with using fiber optics is associated with the stress on the fiber due to flexing. Movements of the robot's joints cause flexing. The problems with flexing require very careful routing of the fiber through physical pathways designed specifically for fiber optics. Making fiber optic connections is also difficult. Imperfect connections distort the optical signal. These connections typically use a special fiber optic glue. Reconfigurable robots require connectors that can be repeatedly connected and disconnected (no glue).

7.2.7 Control

An advanced robot places tremendous demands on its controller. Almost every characteristic that makes a robot advanced corresponds to increased demands on the controller. These characteristics include: fault tolerance, kinematic redundancy, modularity, and multiple sensing capabilities.

Fault tolerance requires an extremely intelligent controller. The controller must first identify the fault. Once a fault is detected, the controller must respond appropriately. This requires a course of action determined by both the location and the type of fault. For instance, consider a short-circuit in one actuator of a redundantly actuated joint. The drive power to the faulty actuator must be interrupted, since supplying power into a short-circuit is a dangerous activity. The controller must then disengage the failed actuator from the joint with a clutch. Finally, the controller must notify other components in the system of the fault so that they can respond appropriately. Upon being notified of the reduced force capabilities in the faulty joint, the inverse kinematics algorithm can redistribute the torque demands among the joints. Thus, a controller for a fault-tolerant robot must identify a fault, initiate a course of action, and communicate news of the fault to other components in the system.

The addition of kinematic redundancy increases the demands on the controller. Clearly, the controller must be more powerful to drive the extra joints. Since errors accumulate in a serial robot, a robot with more joints accumulates more errors. A very accurate controller is necessary to minimize the errors. There also may be increased demands on the stability of the controller, especially in the case of a hyper-redundant manipulator, due to dynamic coupling among the joints.

Modular and reconfigurable robots place a unique set of demands on the controller. The controller must automatically adjust to a reconfiguration of the system. For this, the controller will act in concert with the automatic plant description system described in Chapter 5. The controller also must control a wide

variety of modules. Finally, the controller should be easily expandable if more modules are added to the robot.

A robot with a wide variety of sensors also increases the demand on the controller. The controller must interface and communicate with all of the sensors. Once the sensor data has been accumulated, it must be interpreted and the appropriate response undertaken.

7.2.8 Mechanical Architecture

A modular mechanical architecture will almost certainly form the foundation of any truly advanced robotic system. Some advanced applications for robotic systems explicitly call for modular robots. Other advanced applications implicitly require modularity. Numerous journal articles, master's theses, and doctoral dissertations detail the benefits of modular robots. This discussion focuses on modularity from an applications perspective.

A truly modular and reconfigurable robot requires a tremendous amount of supporting technology. Virtually every component of the system must be designed with modularity in mind. The interface requirements alone are staggering. Both mechanical and electrical connections must be established quickly and easily as the modules are assembled to form the robot. Fiber optic, pneumatic, and hydraulic carriers may be connected as well. The operational software and system controller must also be easily reconfigurable. Without these, and other, supporting technologies, the modular robot will not be truly reconfigurable.

Other advanced applications require a very precise and dependable robot with multiple sensing capabilities and kinematic redundancy. The question is how

to develop these robots, because they aren't available now. One avenue follows the example set by past robot development efforts. Unfortunately, this path would be expensive, slow, and likely unsuccessful. For example, the Cincinnati Milacron T3 series of manipulators required seven million dollars of development expense and took seven years to bring to market (Tesar and Butler, 1989a). The NASA space shuttle manipulator was delivered at a total cost of one hundred million dollars (Tesar and Butler, 1989b). The FTS robot, which was supposed to be the next-generation of advanced space robot, was canceled after almost four hundred million dollars in development costs (NASA, 1992). Clearly these are not the models to emulate.

The personal computer is a much more successful model. The personal computer was introduced about ten years ago. The architecture of this personal computer has become very familiar. It is a microprocessor and RAM connected to other system components, such as the hard-drive and video display, through a bus. Because these fundamental components were kept separate, progress was made on each one of them independently. These separate components were improved incrementally without the tremendous expense associated with redesigning the entire system. This modular approach made possible today's personal computers; which have very much the same architecture as those from ten years ago, yet are hundreds of times better by almost any measure. During this same ten years, the mechanical design of robots remained stagnant. Even though not all of the advanced applications discussed in this chapter explicitly call for reconfigurability, the requirements they place on a robot will only be met by adopting a modular mechanical architecture.

7.2.9 Mechanical Design

The truly advanced robot's mechanical design is another hurdle remaining. This advanced robot will certainly incorporate a wide-variety of sensors. Fault-tolerance capabilities may call for clutches to disengage faulty actuators and, like common industrial robots, the advanced robot requires brakes, motors, and gears. A kinematically redundant robot also requires the set of components associated with each additional joint. These components can all be purchased "off-the-shelf" and incorporated into the mechanical design of the robot, or they can be designed specifically as integral elements. There are advantages and disadvantages associated with either approach.

Using standard off-the-shelf components is certainly easier. The monetary cost of these components is less because their manufacturer spreads the design expense over a wider variety of applications. Finally, using standard components that are based on proven and tested designs involves less risk. The advantages associated with ease of use, lower expense, and lower risk come with some serious disadvantages. The main disadvantage is that standard components include their own enclosures and mounting hardware. This makes them both heavy and bulky. Given that an advanced robot incorporates a large number of sensors and other components, this added bulk and weight is likely unacceptable. In any case, added bulk and weight certainly reduce the performance capabilities of the robot.

The alternative to using standard components is integrating the functions they perform as part of the robot's design. This requires a large concerted effort encompassing several engineering disciplines. A well-planned and systematic

design effort is essential. The initial expense of this effort will likely be greater than designing a robot using standard components. There is also a significant amount of risk involved since the failure of even one of the components will jeopardize the success of the entire effort. If, however, the designs are successful and perform as they are intended, the benefits will be tremendous. The robot will be much lighter and more compact. The savings in weight and bulk come from eliminating the enclosures and mounting hardware associated with off-the-shelf components. The robot should also be stiffer and more precise since intermediate bearings and shafts are eliminated. Finally, the sensors should gather data of a higher fidelity since they can be located to more precisely measure the physical phenomena of interest.

7.2.10 Robot Actuators

Robot actuators are the final supporting technology this chapter discusses. The actuators supply the power that moves the robot's joints. A robot actuator has two main parts. The first is the prime-mover, which actually develops the power. The second part is the drive elements which are between the prime mover and the motion output of the actuator. By nature, kinematically redundant robots have more actuators than their non-redundant counterparts. Unfortunately, actuator technology is probably the largest factor limiting robots. The actuators are typically the heaviest and most expensive part of the robot.

A comparison between robot's and human's prime-movers gives an impression of why actuators are currently limiting the performance of robots. The actuators in the human biomechanical system are the individual muscles. The biceps muscle of the arm is a representative muscle that has been extensively

studied. The biceps muscle weighs, on average, about .27 pounds and can generate 14.75 foot pounds of torque (Winters and Stark, 1988). This gives the biceps muscle a torque-to-weight ratio of 55.66 foot-pounds per pound of actuator. For comparison the N9M4 electric motor from PMI Corporation, which is made with current state-of-the-art rare-earth magnets, weighs 3.1 pounds and generates .36 foot pounds of torque (PMI, 1991). This gives the N9M4 electric motor a torque-to-weight ratio .116 foot-pounds per pound of actuator. Using the torque-to-weight ratio as the sole measure of performance, the human muscle is about 480 times better than the current state-of-the-art in robot prime movers.

Basing the above comparison solely on torque-to-weight ratio is not entirely fair. Gear trains and other drives will improve an actuator's torque-to-weight ratio. Also, humans have a limited ability to produce a controlled force and cannot endure heavy loads for very long. Finally, mechanical and electromechanical actuators can develop much higher total forces than can any human. For instance, a large robot can carry and manipulate an engine block from a car. Most human's cannot carry this heavy of a load with their muscles.

The drive elements in robot actuators, which are very poor, compound the deficiencies of the prime-mover. The drive elements connect the prime mover to the motion output of the actuator, typically with a gear drive or other speed reducer. The prime movers and speed reducers are almost without exception in separate cases with separate shafts mounted on separate bearings. These separate cases, shafts, and bearings all contribute excess weight. There is compliance, and the associated imprecision, contributed by the extra shafts. The speed reducers also contribute compliance and, especially if they are gears, backlash. Backlash is a

major source of imprecision and makes the actuator's controller less stable. In keeping with the biomechanics comparison, the muscles are connected to their output at the joints via tendons with no backlash and very little compliance.

Clearly, the actuators in robots are not very good, especially when compared to the actuators in humans. What can be done to improve robot actuators? Basically, the torque-to-weight ratio and the precision must be improved. Designing the prime-movers to generate more power, perhaps with forced cooling or superconducting technology, will improve the torque capabilities. Eliminating the excess cases, shafts, and bearings will decrease the weight. Ultimately, the actuator should be designed as a single modular unit.

7.3 PRIORITIZATION OF THE COMPONENT TECHNOLOGIES

This prioritization focuses on generalized inverse kinematics and the ten advanced applications discussed in this chapter. The estimation of these numbers took into account the previous discussion of the ten advanced applications and supporting technologies.

The features that distinguish an application as being advanced also influence its most important supporting technologies. Those applications dependent upon generalized inverse kinematics in turn are especially dependent upon computer hardware and computer software for executing the algorithms. Performance criteria are clearly an indispensable element of generalized inverse kinematics. Robots with the extra degrees of freedom necessary to make the most use of generalized inverse kinematics place extremely high demands on their actuators. Obstacle avoidance is dependent upon generalized inverse kinematics, but also includes requirements for

sensors to identify obstacles and data communication to support these sensors. Applications requiring fault tolerance place the most demands on the supporting technologies. Fault tolerance places high demands on computer software, sensors, data communication, control, mechanical architecture, and actuators. Applications for modular robots demand generalized inverse kinematics, mechanical architecture, mechanical design, and actuators. Tables 7.7 prioritizes and presents an estimate of the possible improvements in the ten supporting technologies over the next five years. Table 7.8 presents some numerical support for the above observations relating the supporting technologies to the advanced applications.

Table 7.7 The prioritization and short-term improvement possible for the ten component technologies.

Technology	Relative Importance	Possible Improvement Over 5 Years
computer hardware	8.0	100+
computer architecture	6.5	8
computer software	5.5	3
performance criteria development	9.0	2
sensors	5.0	7
data communication	9.0	10
control	5.0	2
mechanical architecture	7.5	4
mechanical design	7.0	2
robot actuators	10.0	100

Table 7.8 Relative demands placed on each of the supporting technologies by each of the advanced applications.

	computer hardware	computer architecture	computer software	criteria development	sensors	data communication	control	mechanical architecture	mechanical design	robot actuators	
complex assembly	7	4	6	8	6	5	7	5	5	10	6.3
food packaging	5	3	5	2	2	3	9	9	5	7	5.0
micromanipulation	3	3	3	7	2	2	6	2	9	9	4.6
in-space assembly and construction	8	8	9	7	8	8	7	8	8	9	8.0
satellite retrieval and repair	8	8	10	8	9	10	8	9	9	10	8.9
E.V.A. site preparation	2	2	2	2	3	4	5	6	5	5	3.6
semiconductor wafer handling	3	3	6	5	4	4	2	7	6	5	4.5
long reach arm	6	6	7	8	8	5	6	8	10	10	7.4
inspection at nuclear facilities	7	7	8	6	9	5	5	8	6	6	6.7
decontamination and dismantlement	8	7	8	8	9	8	8	9	9	9	8.3

Chapter 8: Conclusion

This dissertation introduced generalized inverse kinematics for serial robots. The development of the direct search inverse kinematics method is the most important contribution of this work. The direct search inverse kinematics method is valid for all serial robots and allows for the incorporation of an unlimited number of performance criteria. The direct search method also has the important quality that it converges at singularities. Direct search addresses the generalized inverse kinematics problem as a decision making problem with an unlimited number of performance criteria, equality constraints and inequality constraints. The direct search seeks to optimize the performance criteria while simultaneously satisfying the constraints. Another important contribution of this work is the design and implementation of an automatic plant description system. The direct search inverse kinematics algorithm uses the automatic plant description system to generate performance criteria values during the course of the solution. The automatic plant description system is also useful for robot design, robot selection, and the configuration of modular robots in response to task requirements. Direct search was tested in simulation for a wide variety of robots and was tested experimentally on a modern research robot with seven degrees of freedom. Finally, this dissertation included a road map that discussed the future of generalized inverse kinematics in the context of ten advanced applications for robots and ten crucial supporting technologies.

Direct search is a method of solving problems numerically using sets of trial solutions to guide a search. The direct search inverse kinematics method begins at an estimate of the solution. This estimate serves as an initial base point for the search. A pattern of explorations about the base point generates a set of trial solutions. A decision making process evaluates the trial solutions and chooses one as the next base point for the search. The search generates another pattern of perturbations about this new base point and again chooses one as the next base point. This process of local exploration and decision making continues until the search converges to a solution.

The direct search inverse kinematics method is general with respect to the robot's geometry. The exploration and decision making processes do not require geometric or algebraic simplifications particular to any given robot. The direct search method may be applied to any robot with any number of degrees of freedom.

The direct search method allows for the incorporation of an unlimited number of performance criteria. Each trial solution completely determines the state of the robot. This allows all performance criteria, equality constraints, and inequality constraints to be explicitly calculated and used during the decision making process.

Direct search's ability to converge at singularities is unique among inverse kinematics methods. Because all robots have singularities, this is an extremely important quality. Direct search converges at singularities because generating trial solutions and making decisions does not require any scalar or matrix inversion.

Direct search addresses the generalized inverse kinematics problem as a decision making problem with an unlimited number of performance criteria, equality constraints, and inequality constraints. The criteria and constraints must be functions of the joint displacements and/or their time derivatives only. The direct search method does not constrain the definition of optimality. The multicriteria inverse kinematics chapter discusses two definitions of optimality. The first is a non-dominated solution and the second is a minimum of a composite performance index.

The performance criteria are employed when choosing an optimum form among the set of trial solutions. These criteria may be dependent or independent of one another. Performance criteria independent of one another may give conflicting information as to the optimum solution. Dependent performance criteria lead to “double-counting of their shared effects. These complications are simply some of the realities of the generalized inverse kinematics problem.

The equality constraints represent the heart of the inverse kinematics problem. The fundamental equality constraints are the position and orientation constraints on the placement of the end-effector. Solving for the joint displacements given these equality constraints is the inverse kinematics problem. There may also be equality constraints on the higher order properties of the end-effector state.

Inequality constraints typically become important during the operation of an actual robot. The physical travel limits on the joint displacements (joint limits) are examples of equality constraints. All robot actuators have a finite maximum speed which represents another inequality constraint. The actual joint speeds

must be less than or equal to the maximum speed of the actuators. Acceleration, torque, and force limits are all examples of inequality constraints.

A non-dominated solution is one method of defining optimality. A non-dominated solution is superior to all other solutions with respect to at least one criteria and at least as good with respect to all the remaining criteria. The main benefit of this definition of optimality is that scaling difficulties are essentially eliminated. There is no summing of effects and no mixing of units, though the criteria may be ranked relative to one another. This allows the operator some control over the ultimate solution. Once the criteria are ranked, however, an extremely simple rule determines the choice of an optimum solution.

A minimum of a composite performance index is the other definition of optimality this dissertation considered. A composite performance index is essentially a weighted sum of all the criteria values. In summing the effects of different criteria, scaling issues are paramount. This dissertation develops scaling procedures in the context of direct search that produce criteria values of comparable value. This scaling procedure draws on previous work in the area of criteria normalization (Bevill and Tesar, 1990). A procedure for transforming both equality and inequality constraints and expressing them as performance criteria was also developed.

Sequential filters, originally developed for mechanism synthesis, were combined with direct search to form a powerful generalized inverse kinematics method. Sequential filters sorts and ranks the trial solutions to reduce the number of choices presented to the decision making process. The addition of sequential filters both increases the solution speed and eliminates problems associated with

extraneous minimum. Most importantly, sequential filters allow the robot's operator to stay "in the loop" and influence the course of the solution.

Another important contribution of this doctoral work is the design and implementation of an automatic plant description system. The plant description is essentially a dynamic model of the robot. In its current form, this plant description system calculates performance criteria values, the joint forces or torques, and the deflection of the robot's end-effector. The system is automatic because it produces its numerical outputs given its parametric inputs without further user intervention. The direct search inverse kinematics algorithm uses the automatic plant description system to generate performance criteria values during the course of the solution. The automatic plant description system is also useful for robot design, robot selection, and the configuration of modular robots in response to task requirements.

The design began by identifying the system's inputs and outputs. The inputs are: the externally applied loads, a joint-level description of the robot's path, and a description of the robot. The outputs are: the robot's geometry and other pertinent parameters, the joint torques and forces, the end-effector deflection, and the criteria values. The inputs and outputs determined a set of specifications for the system. To meet these specifications, the design process divided the plant description system into three subsystems: formulation, calculation, and database.

The calculation subsystem is where several of the outputs of the automatic plant description system are actually generated. The input to this system is a parametric description of the robot and the task that the robot is to perform. The

outputs are the joint torques or forces, the end-effector deflections, and performance criteria values. This subsystem could reside on a general-purpose computer for simulation applications, or it could reside on a specialized computer for real-time control applications. The important issue is the identification and description of the subsystem's inputs. The proper inputs can then be communicated to the calculation subsystem wherever it resides. There are two general types of inputs. The robot-specific inputs describe the robot's physical parameters and the task-specific inputs describe the task.

The automatic plant description system's database keeps a record of the available modules and associates with each module a variety of information. The modeling subsystem accesses the database and gathers the information required by the calculation subsystem. Other processes may access the database as well. For instance, the automatic configuration of a modular robot in response to task requirements requires information about the available modules. The design process identified four different types of databases: list, hierarchical, relational, and object-oriented. The list database was chosen for the implementation because it matched the requirements of the automatic plant description system and could be written entirely "in-house".

The model formulation subsystem is the heart of the automatic plant description system. This subsystem's input is a description of the robot. This description may be of either a modular or monolithic robot. Using the robot's description and information in the database, this subsystem assembles the parameters and formulates a model of the robot. The model of the robot is generated and transferred to the calculation subsystem for analysis. The model

must be expressed in either a modular or Denavit and Hartenberg format depending upon the analysis software. There are four combinations of input and output formats: a modular robot with module-based analysis, a modular robot with Denavit and Hartenberg-based analysis, a monolithic robot with module-based analysis, and a monolithic robot with Denavit and Hartenberg-based analysis. A procedure for extracting Denavit and Hartenberg parameters from a modular robot description was the major development during the design of this subsystem.

An example of the automatic plant description system was presented for a modular robot with seven degrees of freedom. The example shows the steps presented to the user and the necessary input parameters. The example shows the system's outputs using representative plots and gives run times on an inexpensive personal workstation and on a specialized array processor.

Experimentation and simulation verified the effectiveness of the direct search inverse kinematics method. Several different robots having especially difficult or interesting geometries were tested. The robots cover a spectrum, from the industrial variety with six degrees of freedom to a conceptual hyper-redundant robot with twenty-one degrees of freedom. The examples include a graph showing the solution's accuracy and timing results for the solution speed. In each case direct search was successful in finding solutions to the required level of accuracy. One of the examples showed that the search was successful even though the robot was forced to pass through a singularity.

For the optimal direct search strategies with completely general formulations, the direct search method was about ten times too slow for real-time

application. The sub-optimal search strategy with a general formulation was, however, fast enough for real-time application. Geometric simplification in the form of a spherical wrist also increased the solution speed enough for real-time application. These simplifications are solely for the purpose of increasing the solution speed and will not be necessary as computers become more powerful in the next few years.

For every researcher who has seriously worked on the inverse kinematics problem, there is probably a different solution method. Now there is one more, direct search. The goal for this doctoral work – inverse kinematics general with respect to both the robot's geometry and performance criteria – very much determined the choice of direct search. The direct search method achieves this goal of generality by formulating the exploration procedure for a general serial robot and considering any number or variety of performance criteria in the decision making strategy.

This dissertation has discussed the benefits associated with generality, but what are the costs? With computer software, the cost of generality is almost always decreased solution speed. Just as a race car is faster than a family sedan, specialized computer software is faster than its generalized counterpart. This is not to say that direct search is a family sedan. It's more like an armored tank, impervious to changing geometries and varied criteria.

The important issue is the effect this sacrifice in solution speed has on the intended applications of the direct search inverse kinematics method. For the off-line applications; robot design, monolithic robot selection, modular robot configuration, and general simulation; direct search is already fast enough. The

problem is with real-time application. For the suboptimal search strategies, the solution speed is probably fast enough. Section 6.4 presented simulation results showing that geometric simplification will also increase the solution speed enough for real-time application. Geometric simplification sacrifices the generality of direct search with respect to the robot's geometry. For most applications this sacrifice is not too great. The simplification is particular to the robot and the direct search remains general with respect to the performance criteria. Thus the direct search will still optimize different criteria to enhance the robot's performance in a variety of tasks. Reconfigurable robots and fault-tolerant robots are the applications absolutely requiring inverse kinematics general with respect to the robot's geometry. For these two applications, an optimal direct search algorithm in real-time is still several years into the future.

This doctoral work began with generality. The general solution methods it developed are easily specialized to give increased solution speed for a particular robot. If the work had begun in search of a specialized solution, one likely would have been found.

References

Ambrose, R. O., and Tesar, D., 1992, "Design, Construction and Demonstration of Modular, Reconfigurable Robots," University of Texas at Austin, Report to the U.S. Department of Energy under Grant No. DE-FG-02-86NE37966.

Ambrose, G., Hansen, D. R., and Duckstein, L., 1982, Multiobjective Decision Analysis with Engineering and Business Applications, John Wiley and Sons, New York.

Angeles, J., 1985, "On the Numerical Solution of the Inverse Kinematic Problem," *The International Journal of Robotics Research*, Vol. 4, No. 2, pp. 21-37.

Angeles, J., 1982, Spatial Kinematic Chains, Springer-Verlag, Berlin.

Baillicul, J., 1986, "Avoiding Obstacles and Resolving Kinematic Redundancy," *Proceedings, 1986 IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 1698-1704.

Barton, L., 1984, Mechanism Analysis: Simplified Analytical and Graphical Techniques, John Wiley and Sons, New York.

Bedrossian, N., and Flueckiger, K., 1991, "Characterizing Spatial Redundant Manipulator Singularities," *Proceedings, 1991 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 714-719.

Benedict, C. E., and Tesar, D., 1978, "Model Formulation of Complex Mechanisms With Multiple Inputs," *Journal of Mechanical Design*, Vol. 100, pp. 747-761.

Benhabib, B., Zak, G., and Lipton, M. G., 1989, "A Generalized Kinematic Modeling Method for Modular Robots," *Journal of Robotic Systems*, Vol. 6, No. 5, pp. 545-571.

Bevill, P. J., and Tesar, D., 1990, "Criteria Normalization to Support Decision Making in Intelligent Machines," University of Texas at Austin, Report to the U.S. Department of Energy under Grant No. DE-FG-02-86NE37966.

Burdick, J., 1991, "A Classification of 3R Regional Manipulator Singularities and Geometries," *Proceedings, 1991 IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 2670-2675.

Burdick, J., 1989, "On the Inverse Kinematics of Redundant Manipulators: Characterization of the Self-Motion Manifolds," *Proceedings, 1989 IEEE Conference on Robotics and Automation*, Vol. 8, pp. 264-270.

Carnignan, C., 1991, "Trajectory Optimization for Kinematically Redundant Arms," *Journal of Robotic Systems*, Vol. 6, No. 5, pp. 221-248.

Cenedese, A., Gallerano, F., and Misiti, A., 1987, "Multiobjective Analysis in Optimal Solution of Hydraulic Networks," *Journal of Hydraulic Engineering*, Vol. 113, No. 9, pp. 1133-1143.

Chang, P., 1986, "A Closed-Form Solution for the Control of Manipulators with Kinematic Redundancy," *Proceedings, 1986 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 9-14.

Cheng, H., and Gupta, K. C., 1991, "A Study of Robot Inverse Kinematics Based Upon the Solution of Differential Equations," *Journal of Robotic Systems*, Vol. 8, No. 2, pp. 159-175.

Chevallereau, C., and Khalil, W., 1987, "Efficient Method for the Calculation of the Pseudo Inverse Kinematic Problem," *Proceedings, 1987 IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 1842-1848.

Cleary, K., and Tesar, D., 1990, "Incorporating Multiple Criteria in the Operation of Redundant Manipulators," *Proceedings, 1990 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 618-623.

Comerford, R., 1992, "How DEC Developed the Alpha," *IEEE Spectrum*, Vol. 29, No. 7, pp. 26-31.

Craig, J. J., 1989, Introduction to Robotics, Addison Wesley, Massachusetts.

Crane, C., Duffy, J., and Carnahan, T., 1991, "A Kinematic Analysis of the Space Station Remote Manipulator System (SSRMS)," *Journal of Robotic Systems*, Vol. 8, No. 5, pp. 637-658.

Das, H., Slotine, J-J. E., and Sheridan, T. B., 1988, "Inverse Kinematic Algorithms for Redundant Systems," *Proceedings, 1988 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 43-48.

Dimentberg, F. M., 1948, "A General Method for the Investigation of Finite Displacements of Spatial Mechanisms and Certain Cases of Passive Constraints," *Trudi Seminara po Teorii Mashin i Mekhanizmov*, Akademiia Nauk USSR, Vol. 5, pp. 5-39.

Dubey, R., and Luh, J., 1988, "Redundant Robot Control Using Task Based Performance Measures," *Journal of Robotic Systems*, Vol. 5, No. 5, pp. 409-432.

Duffy, J., 1990, "The Fallacy of Modern Hybrid Control Theory that is Based on "Orthogonal Complements" of Twist and Wrench Spaces," *Journal of Robotic Systems*, Vol. 7, No. 2, pp. 139-144.

Duffy, J., 1980, Analysis of Mechanisms and Robot Manipulators, Wiley, New York.

Duffy, J., and Derby, S., 1979, "Displacement Analysis of a Spatial 7R Mechanism – A Generalized Lobster's Arm," *Journal of Mechanical Design*, Vol. 101, pp. 224-231.

Duffy, J., and Rooney, J., 1975, "A Foundation for a Unified Theory of Analysis for Spatial Mechanisms," *ASME Journal of Engineering for Industry*, Vol. 97, No. 4, pp. 1159-1164.

Eckmiller, R., Beckman, J., Werntges, H., and Lades, M., 1989, "Neural Kinematics Net for a Redundant Robot Arm," *International Joint Conference on Neural Networks*, Vol. 2, pp. 333-339.

Egeland, O., Sagli, J. R., and Spangelo, I., 1991, "A Damped Least-Squares Solution to Redundancy Resolution," *Proceedings, 1991 IEEE International Conference of Robotics and Automation*, Vol. 1, pp. 945-950.

Eschenbach, P. W., and Tesar, D., 1969, "Optimization of Four-Bar Linkages Satisfying Four Generalized Coplanar Positions," *Journal of Engineering for Industry*, February, 1969, pp. 75-82.

Euler, J. A., Dubey, R. V., and Babcock, S. M., 1989, "Self Motion Determination Based on Actuator Velocity Bounds for Redundant Manipulators," *Journal of Robotic Systems*, Vol. 6, No. 4, pp. 417-425.

Euler, J. A., and Dubey, R. V., 1989, "A Comparison of Two Real-Time Control Schemes for Redundant Manipulators with Bounded Joint Velocities," *1989 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 106-112.

Everett, L., and McCarroll, D., 1986, "Using Finite Element Methods to Approximate Kinematic Solutions of Robot Manipulators when Closed Form Solutions are Unobtainable," *Proceedings, 1986 IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1164-1167.

Fresonke, D. A., Hernandez, E., and Tesar, D., 1988, "Deflection Prediction for Serial Manipulators," *Proceedings, 1988 IEEE International Conference on Robotics and Automation*, Vol. 1., pp. 482-487.

Freudenstein, F., 1972, "Kinematics: Past, Present and Future," *Mechanism and Machine Theory*, Vol. 8, No. 2, pp. 151-160.

Freudenstein, F., and Sandor, G. N., 1959, "Synthesis of Path-Generating Mechanisms by Means of a Programmed Digital Computer," *Journal of Engineering for Industry*, May, 1959, pp. 159-168.

Galicki, M., 1991, "A Closed Solution to the Inverse Kinematics of Redundant Manipulators," *Mechanism and Machine Theory*, Vol. 26, No. 2, pp. 221-226.

Gluss, B., 1972, An Elementary Introduction to Dynamic Programming, A State Equation Approach, Allyn and Bacon, Boston.

Goldenberg, A. A., and Lawrence, D. L., 1985, "A Generalized Solution to the Inverse Kinematics of Robotic Manipulators," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 107, pp. 103-106.

Golinski, J., 1973, "An Adaptive Optimization System Applied to Machine Synthesis," *Mechanism and Machine Theory*, Vol. 8, pp. 419-436.

Gorey, K., August 6, 1992, "Periodic Table of the IRISes," Silicon Graphics Co., Mountain View, California.

Gorey, K., May 7, 1990, "Periodic Table of the IRISes," Silicon Graphics Co., Mountain View, California.

Gu, Y-L., and Ho, J-S., 1990, "A Unified Kinematic Representation and its Applications to Robotic Control," *Proceedings, 1990 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 98-103.

Gupta, K. C., and Kazerounian, K., 1985, "Improved Numerical Solutions of Inverse Kinematics of Robots," *1985 IEEE International Conference on Robotics and Automation*, March, 1985, pp. 743-748.

Gupta, K. C., 1984, "A Note on Position Analysis of Manipulators," *Mechanism and Machine Theory*, Vol. 19, No. 1, pp. 5-8.

Hakala, J., Steiner, R., and Eckmiller, R., 1990, "Quasi-Local Solution for Inverse Kinematics of a Redundant Robot Arm," *International Joint Conference on Neural Networks*, Vol. 3, pp. 321-326.

Hemami, A., 1989, "Joint Velocity Uniformity in Redundant Robot Manipulators," *Robotica*, Vol. 8, pp. 69-72.

Hemmerle, J., and Prinz, F., 1991, "Optimal Path Placement for Kinematically Redundant Manipulators," *Proceedings, 1991 IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1234-1238.

Hernandez, E., and Tesar, D., 1989, "Efficient Algorithms and Real-Time Software for Quasi-Static Manipulator Deflection Modeling," The University of Texas at Austin, Report to NASA under Grant No. NAG 9-188, and DOE under Grant No. DE-FG02-86NE37966.

Hicks, C., 1967, Fundamental Concepts in the Design of Experiments, Springer-Verlag, Berlin.

Hirose, S., and Ma, S., 1989, "Redundancy Decomposition Control for Multi-Joint Manipulator," *1989 Conference on Robotics and Automation*, Vol. 1, pp. 119-124.

Hooke, R., and Jeeves, T. A., 1961, "'Direct-Search' Solution of Numerical and Statistical Problems," *Journal of the Association of Computing Machinery*, Vol. 8, pp. 212-229.

Hooper, R., and Tesar, D., 1994, "Computer-Aided Configuration of Modular Robots", *Journal of Computing and Control*, Accepted for Publication.

Hooper, R., and Tesar, D., 1994, "Generalized Inverse Kinematics Algorithm," Library of Congress, Copyright Office, The University of Texas at Austin - claimant.

Hooper, R., and Tesar, D., 1993, "robo_cad," Library of Congress, Copyright Office, The University of Texas at Austin - claimant.

Hooper, R., and Tesar, D., 1990, "The Interactive Assembly and Computer Animation of Modular Robotic Systems," University of Texas at Austin, Report to the U.S. Department of Energy under Grant No. DE-FG-02-86NE37966.

Hsia, T. C., and Guo, Z. Y., 1991, "New Inverse Kinematic Algorithms for Redundant Robots," *Journal of Robotic Systems*, Vol. 8, No. 1, pp. 117-132.

Ji, Z., and Leu, M. C., 1990, "Inverse Kinematics of Calibrated Robots," *Journal of Robotic Systems*, Vol. 7, No. 5, pp. 675-687.

Kang, H. J., Tesar, D., and Freedman, R. A., 1992, "Dynamic Modeling and Optimal Joint Torque Coordination of Advanced Robotic Systems," The University of Texas at Austin, Report to the U.S. Department of Energy under Grant No. DE-FG-02-86NE37966.

Karpak, B., Zionts, S., 1987, Multiple Criteria Decision Making and Risk Analysis Using Microcomputers, Springer-Verlag, Berlin.

Kazerounian, K., and Wang, Z., 1988, "On the Singularity Avoidance and Uniqueness of Solutions in Redundant Industrial Robots," *Computers in Engineering 1988*, Vol. 2, pp. 293-296.

Kazerounian, K., 1987, "On the Numerical Inverse Kinematics of Robotic Manipulators," *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 109, pp. 8-13.

Kelmar, L., and Khosla, P. K., 1990, "Automatic Generation of Forward and Inverse Kinematics for a Reconfigurable Modular Manipulator System," *Journal of Robotic Systems*, Vol. 7, No. 4, pp. 599-619.

Kelmar, L., and Khosla, P. K., 1988 "Automatic Generation of Kinematics for a Reconfigurable Modular Manipulator System," *Proceedings, 1988 IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 663-668.

Kieffer, J., and Litvin, F. L., 1990, "Swept Volume Determination and Interference Detection for Moving 3-D Solids," *Flexible Mechanism, Dynamics, and Robot Trajectories*, Vol. 24, pp. 309-318.

Kim, J.-O., Khosla, P. K., and Chung, W.-K., 1990, "Modeling Static Behavior of Redundant Manipulators," *Robotics and Manufacturing*, Vol. 3, pp. 465-471.

Kohli, D., and Osvatic, M., 1992, The Inverse Kinematics of General Serial Manipulators, The University of Wisconsin Press, Wisconsin.

Lee, S., and Kil, R. M., 1990, "Robot Kinematic Control Based on Bidirectional Mapping Neural Network," *International Joint Conference on Neural Networks*, Vol. 3, pp. 327-335.

Liegeois, A., 1977, "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-7, No. 12, pp. 868-871.

Leitmann, G., and Marzollo, A., 1975, Multicriteria Decision Making, Springer-Verlag, Wien.

Lin, W., and Duffy, J., 1989, "Displacement Analysis of Two Special Cases of the TTTR Mechanisms," *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 3, pp. 309-314.

Lipkin, H., 1990, "Invariant Properties of the Pseudoinverse in Robotics," *Proceedings of NSP Design and Manufacturing Systems Conference*, Arizona State University, Tempe, Arizona, January 8-12.

Lipkin, H., and Park, Y. S., 1988, "Root-Locus Analysis of Robot Kinematics," *Trends and Developments in Mechanisms, Machines, and Robotics – 1988*, Vol. 15-3, pp. 291-298.

Lipkin, H., and Duffy, J., 1985, "A Vector Analysis of Robot Manipulators," Recent Advances in Robotics, John Wiley and Sons, New York, pp. 175-241.

Litvin, F. L., Fanghella, P., Tan, J., and Zhang, Y., 1986, "Singularities in Motion and Displacement Functions of Spatial Linkages," *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 108, pp. 516-523.

Litvin, F. L., and Castelli, V. P., 1985, "Configurations of Robot's Manipulators and Their Identification, and the Execution of Prescribed Trajectories," *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 107, pp. 170-178.

Llinares, J., and Page, A., 1984, "Position Analysis of Spatial Mechanisms," *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 106, pp. 252-294.

Luo, R. C., 1990, "Special Issue on Multisensor Integration and Fusion for Intelligent Robots," *Journal of Robotic Systems*, Vol. 7, No. 3, pp. 291-294.

Maciejewski, A., 1989, "Kinetic Limitations on the Use of Redundancy in Robotic Manipulators," *Proceedings, 1989 IEEE Conference on Robotics and Automation*, Vol. 1, pp. 113-118.

Maciejewski, A., and Klein, C. A., 1988, "Numerical Filtering for the Operation of Robotic Manipulators through Kinematically Singular Configurations," *Journal of Robotic Systems*, Vol. 5, No. 6, pp. 527-552.

Martin, J. J., 1967, Bayesian Decision Problems And Markov Chains, Wiley, New York.

Martin, H. L., and Satterlee, P. E., 1987, "Kinematic Redundancy – Key to Dexterous Manipulation in Confined Environments," *Robots 11, 17th International Symposium on Industrial Robots*, Vol. 1, pp. 37-45.

Mayorga, R. V., and Wong, A. K. C., 1990, "A Singularities Prevention Approach for Redundant Robot Manipulators," *Proceedings, 1990 IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 812-817.

Meldman, M. J., McLeod, D. J., Pellicore, R. J., and Squire, M., 1978, RISS: A Relational Data Base Management System For Minicomputers, Van Nostrand Reinhold Company, New York.

Mussa-Ivaldi, F. A., and Hogan, N., 1989, "Solving Kinematic Redundancy with Impedance Control: A Class of Integrable Pseudoinverses," *Proceedings, 1989 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 283-288.

NASA, 1992, "Flight Telerobotic Servicer Lessons Learned and Technology Evolution," App. B, ATAC Report No. 14.

Nelson, B., and Donath, M., 1990, "Optimizing the Location of Assembly Tasks in a Manipulator's Workspace," *Journal of Robotic Systems*, Vol. 7, No. 6, pp. 791-811.

Nenchev, D. N., 1989, "Redundancy Resolution through Local Optimization: A Review," *Journal of Robotic Systems*, Vol. 6, No. 6, pp. 769-798.

Novakovic, Z. R., 1990, "Solving the Inverse Kinematics Problem via the Principle of Self-Support," *Journal of Robotic Systems*, Vol. 7, No. 4, pp. 621-647.

Pahl, G., and Beitz, W., 1988, Engineering Design A Systematic Approach, Springer-Verlag, The Design Council, Berlin.

Pamanes, J. A., and Zeghloul, S., 1991, "Optimal Placement of Robotic Manipulators Using Multiple Kinematic Criteria," *Proceedings, 1991 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 933-938.

Parker, J. K., and Goldberg, D. E., 1989, "Inverse Kinematics of Redundant Robots Using Genetic Algorithms," *Proceedings, 1989 IEEE Conference on Robotics and Automation*, Vol. 1, pp. 271-276.

Pieper, D., and Roth, B., 1969, "The Kinematics of Manipulators Under Computer Control," *Proceedings, II International Congress on the Theory of Machines and Mechanisms*, Vol. 2, pp. 159-168.

PMI Motion Technologies, March, 1991, "DC Servo Motors," *Catalog Number 165*.

Potkonjak, V., 1990, "Distributed Positioning for Redundant Robotic Systems," *Robotica*, Vol. 8, pp. 61-67.

Pourboghraat, F., 1989, "Neural Networks for Learning Inverse-Kinematics of Redundant Manipulators," *Proceedings, 32nd Midwest Symposium on Circuits and Systems*, Vol. 2, pp. 760-762.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., 1991, Numerical Recipes in C The Art of Scientific Computing, Cambridge University Press, Cambridge, England.

Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M., 1983, Engineering Optimization Methods and Applications, John Wiley and Sons, New York.

Ropponen, T., and Nakamura, Y., 1990, "Singularity-Free Parameterization and Performance Analysis of Actuation Redundancy," *Proceedings, 1990 IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 806-811.

Roth, B., 1979, Theoretical Kinematics, Wiley and Sons, New York.

Roth, B., Rastegar, J., and Scheinman, V., 1973, "On the Design of Computer Controlled Manipulators." *On The Theory and Practice of Robots and Manipulators*, First CISM-IFTOMM Symposium, Vol. 1, pp. 93-113.

Roth, B., 1967 "The Kinematics of Motion Through Finitely Separated Positions," *Journal of Applied Mechanics*, Vol. 34, pp. 591-598.

Roth, B., and Freudenstein, F., 1963, "Synthesis of Path-Generating Mechanisms by Numerical Methods," *Journal of Engineering for Industry*, Vol. 85, pp. 298-306.

Smith, D. R., and Lipkin, H., 1990, "Analysis of Fourth Order Manipulator Kinematics using Conic Sections," *Proceedings, 1990 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 274-278.

Spang, H. A., 1962, "A Review of Minimization Techniques for Nonlinear Functions," *SIAM Review*, Vol. 4., pp. 343-365.

Sreevijayan, D., and Tesar, D., 1993, "On the Design of Fault-Tolerant Robotic Manipulator Systems," *Proceedings of the Third International Symposium on Measurement and Control in Robotics*, Session Cs.I-27.

Tesar, D., and Sreevijayan, D., 1990, "Four-Level Fault Tolerance in Manipulator Design for Space Operations," Presented at First International Symposium on Measurement and Control in Robotics (ISMCR '90).

Tesar, D., and Butler, M. S., 1989a, "A Generalized Modular Architecture for Robot Structures," *Manufacturing Review*, Vol. 2, No. 2, pp. 91-118.

Tesar, D., and Butler, M. S., 1989b, "20-Year Forecast of NASA Robotics Requirements for Space Exploration," *Proceedings NASA Satellite Servicing Workshop*.

Tesar, D., 1989, "Thirty-Year Forecast: The Concept of a Fifth Generation of Robotics - The Super Robot," *Manufacturing Review*, Vol. 2, No. 1, pp. 16-25.

Thomas, M., and Tesar, D., 1982, "Dynamic Modeling of Serial Manipulator Arms," *Journal of Dynamic Systems, Measurement, and Control*, Vol. 104, pp. 218-228.

Thomopoulos, S., and Tam, R., 1991, "An Iterative Solution to the Inverse Kinematics of Robotic Manipulators," *Mechanism and Machine Theory*, Vol. 26, No. 4, pp. 359-368.

Tourassis, V. D., 1989, "A Modular Architecture for Inverse Robot Kinematics," *Proceedings, IEEE Transactions on Robotics and Automation*, Vol. 5, No. 5, pp. 555-568.

Tsai, L.-W., and Morgan, A. P., 1985, "Solving the Kinematics of the Most General Six- and Five-Degree-of-Freedom Manipulators by Continuation Methods," *Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 107, pp. 189-199.

Uicker, J. J., Denavit, J., and Hartenberg, R. S., 1964, "An Iterative Method for the Displacement Analysis of Spatial Mechanisms," *Journal of Applied Mechanics*, Vol. 31, pp. 309-314.

Van Doren, M. J., and Tesar, D., 1992, "Criteria Development to Support Decision Making Software for Modular, Reconfigurable Robotic Manipulators," Report to the U.S. Department of Energy under Grant No. DE-FG-02-86NE37966, and Texas Higher Education Coordinating Board for Advanced Technology Program under Project No. 003658-156.

Walker, I. D., and Marcus, S. I., 1986, "An Approach to the Control of Kinematically Redundant Robot Manipulators," *1986 American Control Conference*, Vol. 5, pp. 1907-1912.

Wallace, D. M., and Freudenstein, F., 1970, "The Displacement Analysis of the Generalized Tracta Coupling," *Journal of Applied Mechanics*, Vol. 37, pp. 713-719.

Wampler, C. W., 1989, "Inverse Kinematic Functions for Redundant Spherical Wrists," *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 1, pp. 106-111.

Wander, J. P., and Tesar, D., 1987, "Pipelined Computation of Manipulator Modeling Matrices," *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 6, pp. 556-566.

Wellman, J., and Tesar, D., 1991, "Kinesthetic Feedback for Manual Control Using Intersecting Volumes," Report to the U.S. Department of Energy under Grant No. DE-FG-02-86NE37966, and Texas Higher Education Coordinating Board for Advanced Technology Program under Project No. 003658-156.

Whitney, D. E., 1969, "Resolved Motion Rate Control of Manipulators and Human Prostheses," *IEEE Transactions on Man-Machine Systems*, Vol. MMS-10, No. 2, pp. 47-53.

Wiederhold, G., 1983, Database Design, McGraw-Hill Company, New York.

Winters, M., and Stark, L., 1988, "Estimated Mechanical Properties of Synergistic Muscles Involved in Movements of a Variety of Human Joints," *Journal of Biomechanics*, Vol. 21, No. 12, pp. 1027-1041.

Yang, A. T., and Freudenstein, F., 1964, "Application of Dual-Number Quaternion Algebra to the Analysis of Spatial Mechanisms," *Journal of Applied Mechanics*, Vol. 31, pp. 300-308.

Yates, F., 1937, Design and Analysis of Factorial Experiments, London Imperial Bureau of Soil Sciences, London, England.

Yuan, Freudenstein, Woo, 1971, "Kinematic Analysis of Spatial Mechanisms by Means of Screw Coordinates. Part 2 - Analysis of Spatial Mechanisms," *Journal of Engineering for Industry*, Vol. 93, pp. 67-73.

Yuan, M. S. C., and Freudenstein, F., 1971, "Kinematic Analysis of Spatial Mechanisms by Means of Screw Coordinates. Part 1 - Screw Coordinates," *Journal of Engineering for Industry*, Vol. 93, pp. 61-66.

Zdonik, S. B., Maier, D., 1990, Readings in Object-Oriented Database Systems, Morgan Kaufmann Publishers, San Mateo.

Zorpette, G., 1993, "Big Blue Pushes Blue-light Storage," *The Institute*, Vol. 17, No. 5, pp. 1&4.