Supervisory Control of Robot Manipulator for Gross Motions

by

JONG HYEON PARK

B.S. in Mechanical Engineering
Seoul National University
(1981)

M.S. in Mechanical Engineering
Massachusetts Institute of Technology
(1983)

Submitted to the Department of Mechanical Engineering

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 1991

© Massachusetts Institute of Technology 1991. All rights reserved.

Signature Redacted

Author .....
Department of Mechanical Engineering
June, 1991

Signature Redacted

Certified by .....
Thomas B. Sheridan
Signature Redacted s Supervisor

Accepted by .....
Ain A. Sonin
Chairman, Departmental Committee

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
FEB 20 1992
LIBRARIES
ARCHIVES

Supervisory Control of Robot Manipulator

for Gross Motions

by

JONG HYEON PARK

Submitted to the Department of Mechanical Engineering

on June, 1991, in partial fulfillment of the
requirements for the degree of
DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING

Abstract

A framework for a supervisory approach to robot manipulator collision avoidance in an environment cluttered with three-dimensional obstacles is developed. The algorithm for collision avoidance is triggered by the detection of an impending collision during a motion that is suggested by the operator. Collision detection is done with the clipping hardware of a graphic workstation. Collision avoidance uses heuristics and a world model to generate motions to avoid the impending collision.

Collision detections and avoidance is based upon a world model. A new method, called the fly-and-match method, is proposed to introduce objects into the world model in the context of remote operations where the operator observes objects in the environment through video cameras. In this method, the operator controls a six-d.o.f. joystick and tries to match the camera view of an object with the display of the geometric model of the object as closely as he/she can. An experiment with human subjects introducing objects by matching in a simulated environment was conducted. The concept of virtual obstacle is also introduced in order to model unobserved regions as obstacles in remote operations of the manipulator. Such virtual obstacles are updated by a geometric AND-operation when the camera position changes.

The effect of combining the two processes, i.e. building a world model and executing a task, is evaluated through an experiment with human subjects. Two strategies to introduce objects into the world model using the fly-and-match method are considered in the context of gross motion of the robot manipulator: a crude model and a precise model. A crude world model has a smaller free space for the manipulator to maneuver than a precise one. The results show that the proposed collision avoidance method combined with a crude world model reduces task completion times.

Thesis Committee: Professor Thomas B. Sheridan (Chairman)
Professor David C. Gossard
Professor Tomás Lozano-Pérez

Acknowledgments

I would like to thank all my committee for giving me helps and good suggestions. I thank Professor Thomas B. Sheridan, my thesis adviser, for supporting and overseeing this project, Professor Tomás Lozano-Pérezfor his invaluable help, encouragement and interest in my work, and Professor David C. Gossard for sharing his expertise in Computer Graphics and for giving me many different perspectives. I would also like to thank Dr. Bejczy at JPL for supporting my work for years.

Many heartfelt thanks to my wife, Kyoung, for giving me encouragement when I was down, and for enduring very difficult times with me. I appreciate her participation in experiments. I also thank my son, Richard, who was born in the middle of my study at MIT, for giving me a hope , and for growing up so much. I also thank my mother and father who has been waiting anxiously for the finish of my work at MIT. Yong M. Kim, my brother-in-law, participated experiments for this work.

I would like further acknowledge many people who have given me a special experience at MIT: Chi Cheng Cheng, who participated all the experiments without any complaint; Mike Massimino and Dave Schloerb, my office mates and the keepers of MMSL, who spent their precious time for the experiments, All the previous and other current members of MMSL, in particular: Max B. Mendel, Leonid Charney, James B. Roseborough (a hacker), Hari Das, G. Jagannath Raju, Sam Landsberger, Kan-Ping Chin, Wael Yard, and Nicholas Patrick. With deep affection I thank Professor Nevil Hogan for giving me an opportunity to work with him for 2.151.

This research was supported in part by Jet Propulsion Laboratory, California.

Contents

1 Introduction 1

1.1 Definitions 1

1.2 Supervisory Control 3

1.2.1 Introduction: Man vs. Machine 4

1.2.2 Examples of Supervisory Control 6

1.2.3 Role of Supervisor 6

1.2.4 Why Supervisory Control of Teleoperators? 8

1.3 Collision Avoidance for Robot Manipulators 9

1.3.1 Complexity 10

1.3.2 General Approaches 11

1.4 Uncertainty of Environment 18

1.4.1 What is uncertainty of the environment? 18

1.4.2 Uncertainty of Modeling 18

1.4.3 Sources of Uncertainty of Modeling 19

1.4.4 Uncertainty and Manipulator Motions 19

1.5 Problem Statement and Scope of Work 20

1.6 Outline 21

2 Supervisory Control for Collision Avoidance 23

2.1 Background: Supervisory vs Autonomous Planning 23

2.2 Supervisory Control System 25

2.3 Supervisory Specification of a Robot Path with World Model 26

2.4 Collision Detection 30

2.4.1 Introduction 30
2.4.2 Collision Detection Using Clipping Algorithm 31
2.4.3 Collision Detection in Two-Dimensional Space 31
2.4.4 Collision Detection in Three-Dimensional Space 34
2.4.5 Viewing Transformations 37
2.4.6 Collision Detection for Manipulators 38
2.5 Supervisory Collision Avoidance 40
2.5.1 Introduction 40
2.5.2 Heuristics for Wrist 41
2.5.3 Heuristics for Elbow 49
2.5.4 Other Heuristics 49
2.5.5 Similarity of Motions 50
2.6 Supervisory Control System with Computer Graphics 53
2.6.1 Advantages of Computer Graphics 53
2.6.2 Extra information display to help planning 54

3 Uncertainty **58**
3.1 Virtual Obstacles 58
3.1.1 Uncertainty of Existence 59
3.1.2 Definition of Virtual Obstacles 59
3.1.3 Generation of Virtual Obstacles 60
3.1.4 Camera Motion and Virtual Obstacles 65
3.2 Introduction of Objects by "Flying-and-Matching" 65
3.2.1 Precise Introduction vs. Crude Introduction 67
3.2.2 Precise Introduction 67
3.2.3 Crude Introduction 67
3.2.4 Importance of Accurate Orientational Matching 70
3.2.5 Trade-Off 70

4 Experiments **73**
4.1 Experiment 1: Stacking Operation 73

4.1.1 Objective . . . 73
4.1.2 Setup and Procedure . . . 74
4.1.3 Results . . . 78
4.1.4 Observations on Experiment . . . 81
4.2 Experiment 2: Introduction of Objects to the World Model by Flying-and-Matching . . . 83
4.2.1 Objective . . . 83
4.2.2 Setup and Procedure . . . 83
4.2.3 Results . . . 86
4.3 Experiment 3: Introduction of Objects into World Model and Stacking Operation . . . 94
4.3.1 Objective . . . 94
4.3.2 Setup and Procedure . . . 95
4.3.3 Results . . . 97

5 Conclusions and Recommendations for Future Work 106
5.1 Conclusions . . . 106
5.1.1 Supervisory System . . . 106
5.1.2 Uncertainty . . . 108
5.2 Recommendations . . . 110

A Projection Transformations 112
A.1 Perspective Transformation . . . 113
A.2 Orthogonal Transformation . . . 114

B Three-dimensional solid geometry 115
B.1 Boundary Representations . . . 115
B.2 Edge-based Boundary Models . . . 116
B.3 Spatial Set Operators . . . 119

C Miscellaneous 122
C.1 Coordinate Transformation . . . 122

C.1.1 Position and Orientation 122
C.1.2 Coordinate Transformations 123
C.1.3 Rotation 123
C.2 Joystick Control 125
C.2.1 Translation 125
C.2.2 Rotation 127

D Experiment Analysis and Data **128**
D.1 *T*-test 128
D.2 Data from Experiment 1 130
D.3 Data from Experiment 2 131
D.4 Data from Experiment 3 133

List of Figures

1-1 Potential Approach. 13

1-2 C–space Approach. 16

2-1 Planning. 24

2-2 Block Diagram of the Supervisory Control System. 26

2-3 Block Diagram of the Supervisory Collision Avoidance System. 27

2-4 Interactive Ppath Specification 29

2-5 Two Objects in Collision. 32

2-6 Two Objects in Collision and a Clipping Area. 32

2-7 Collision Detection through a Clipping Area. 33

2-8 Object A and Object B. 35

2-9 Object B and Clipping Boundaries. 35

2-10 Examples of Solids Made with Viewing Transformations 38

2-11 Manipulator Parts for Collision Avoidance. 41

2-12 A Detour Motion. 42

2-13 Motion with Heuristics of Reaching-Out-and-Touching. 43

2-14 The Potential Approach. 45

2-15 A Maze-shaped Obstacle. 46

2-16 Avoidance of an Impending Collision at the Elbow. 48

2-17 A Collision Avoidance Motion When the Elbow Is Fully Stretched. . . . 51

2-18 Collision Avooidance Motions (Graphic Simulation on SGI's Iris). . . . 52

2-19 The Workspace Sphere 56

3-1 Virtual Objects and the Camera Position. 61

3-2 A Virtual Obstacle Is Updated As the Camera Moves Up. 64
3-3 Two Different Strategies To Introduce Objects Into the World Model. . . 68
3-4 Important Correct Angular Alignment. 69

4-1 Learning Effect 77
4-2 Removal of Learning Effect 77
4-3 Time Vs. Subjects 79
4-4 Travel Distance Vs. Subjects 80
4-5 Collisions Vs. Subjects 80
4-6 Collisions Vs. Sequence of the Test 81
4-7 Collisions Vs. Number of Unmodeled Objects 82
4-8 Time Vs. Sequence 82
4-9 Two Objects Used in the Experiment. 84
4-10 Positional RMS Errors in the X- and Y-directions Vs. Field-of-View . . . 88
4-11 Positional RMS Error in the Z-direction Vs. Field-of-View 88
4-12 Effect of Occlusion on the Positional RMS Errors in the X- and Y-directions for the Rectangular solid. 89
4-13 Effect of Occlusion on the Positional RMS Error in the Z-direction. . . . 90
4-14 A Representation of an Angular Error. 90
4-15 A Unit Sphere Is Spanned by Angular Errors. 91
4-16 Directions of Angular Errors. 92
4-17 Angular RMS Error Vs. Field-of-View. 93
4-18 Angular RMS Error Vs. Occlusion. 94
4-19 Time To Introduce 6 Objects. 98
4-20 Collisions Vs. Subjects. 98
4-21 Number of Intermediate Points Vs. Subjects. 99
4-22 Time for PD and PA. 100
4-23 Number of Intermediate Points for PD and PA. 101
4-24 Travel Distance for PD and PA. 101
4-25 Time for PA and CA. 102

4-26 Travel Distance for PA and CA. 103
4-27 Collisions for PA and CA. 103
4-28 Number of Intermediate Points for PA and CA. 104
4-29 Total Time. 105

A-1 Primitives Created by Basic Transformations. 113

B-1 A Curved Object and Its Patched Model. 116
B-2 Two Solids in Collision. 120

C-1 A Joystick Coordinate and the Reference Coordinate. 124

List of Tables

4.1 Six different environments. 76

D.1 Data from Experiment 1. 130

D.2 Effect of Field-of-View. 131

D.3 *T*-Test for the Effect of Field-of-View. 131

D.4 Effects of Occlusion for the Rectangular Solid. 132

D.5 Effects of Occlusion for the Cylindrical Solid. 132

D.6 Rectangular solid vs. Cylindrical solid. 132

D.7 Effects of Occlusion for Rectangular and Cylindrical Solid. 133

D.8 Data for Introducing 6 Objects into a World Model. 133

D.9 Task Performance of Stacking Objects in Environment Model I. 134

D.10 Task Performance of Stacking Objects in Environment Model II. 134

D.11 *T*-Test for Different Environments. 135

Chapter 1

Introduction

This thesis is about new ways of controlling robots with the aid of computers. It addresses how to introduce unmodeled objects into a world model and how to use the world model in order to generate collision-free motions. This chapter starts by defining and explaining the terms that are frequently used in the thesis. Also, some background on the relevant areas of this work is provided.

1.1 Definitions

A *teleoperator* is a general purpose work vehicle controlled remotely by a human operator using video and/or other sensors, power and propulsive actuators for mobility, mechanical hands and arms for manipulation, and possibly a computer for a limited degree of control autonomy. Sometimes the term "teleoperator" is restricted to the manipulator, excluding the system for remotely positioning and orienting a sensor such as a video

camera. In this case, the term *telemanipulator* is preferred.

A *telerobot* is a rapidly emerging class of mechanical devices which are partially automated, i.e., have their own sensors, actuators and closed-loop computer-control, and are partially under remote control by a human operator. By this definition, not only space, undersea and other hazardous environment telemanipulators are becoming telerobots (anthropomorphic ones), but also are aircraft, automobiles, power and chemical plants, home appliances, medical devices, and a variety of other new physical systems (non-anthropomorphic telerobots). The term "telerobot" will be limited in this work only to the anthropomorphic variety.

Telepresence means visual, kinesthetic, tactile or other sensory feedback from the teleoperator to the human operator that is sufficient and properly displayed such that the human feels that he is present at the remote site, and that the teleoperator is an extension of his own body.

Collision detection (more precisely, detection of impending collisions) is a capability or a process for the system to check if the next move of the manipulator may result in collisions with obstacles. If an impending collision is detected, the system can simply stop further manipulator motions, or can plan collision-free motions to complete a given task.

Collision avoidance is a capability or a process for the manipulator to go to a target configuration from an initial configuration without colliding with obstacles in the environment.

Free space is the complement of the obstacles. All parts of the manipulator have to be in free space in order to avoid collisions with obstacles.

When the operator is removed from doing the work with his own hands, operating instead through a computer intermediary, he becomes *remote* in ways which may be more important than simply *spatially remote*. The human supervisor becomes temporally remote due either to:

- the transmission-time delay of a communication channel (the time between sending a command and receiving a sensory message that the command was implemented in a certain way),
- the time gap between sensory samples and/or motor control actions which, as it grows longer, allows memory to fade and control to deteriorate.

1.2 Supervisory Control

First proposed by Ferrell and Sheridan (1967), supervisory control is a mixture of manual control and automatic control. In the strict sense, it indicates that one or more human operators are setting initial conditions for, intermittently adjusting, and receiving information from a computer that itself closes a control loop through external sensors, end effectors, and the task environment. In a broader sense, supervisory control refers to the use of a computer which makes complex transformations of data to produce integrated displays, or retransforms operator commands to generate detailed control actions (Sheridan, 1987).

1.2.1 Introduction: Man vs. Machine

For the reasons mentioned in the previous Section, totally autonomous systems are not suitable in many practical operations, and human involvement is desired to improve the efficiency of operations. By taking advantage of the merits of both humans and machines, the system can be more reliable and efficient. Completely manual control, for example, by joysticks, or a master manipulator as an extreme case, is not desirable. A simple supervisory control scheme using a computer could improve the efficiency of the system dramatically (Brooks, 1979).

Knowledge on what human and machines are each good at is essential to design such a system. There are important differences between human and machine attributes. Attributes for human and machine can be summarized as follows (Bejczy, 1975):

Human. Signal detection is limited to narrow ranges, and input channel capacities can easily be saturated. Information processing is with low speed and narrow bandpass. Poor at monitoring continuous signals. Man is good and flexible (adaptive) signal interpreter, easily recognizes patterns, filters signals from noise. Man is at home in formulating concepts, assumptions, and alternatives, can act on symptoms, makes comprehensive judgements, and good at troubleshooting. Man can easily generalize and make decisions even if faced with incomplete information. Man easily sets goals and priorities, determines risks, recognizes targets and opportunities, and improvises under unforeseen circumstances.

Machine. Multi-channel signal detection and processing can easily be handled simul-

taneously. Signal detection ranges and channel capacities can be wide, and information processing can be at high speed and wide bandpass. Good at monitoring continuous signals or processes. Signal interpretation is very explicit but rigid, and information processing is limited to very specific formats. Pattern recognition and filtering require a great deal of effort and are of limited content. Machines are inflexible versus the unexpected, can hardly deal with symptoms or come up with comprehensive judgements, concepts, assumptions, or alternatives, and cannot meaningfully improvise under unforeseen circumstances. All details(goals, priorities, etc.) must be programmed, and machines usually cannot make decisions if the information is incomplete.

It is difficult for the computer to quickly perceive and analyze the environment. Also, the computer cannot handle unexpected situations well. However, the human is very good at doing such tasks. He is also good at making compromises and determining what are important and what are not. Due to these facts, the human can make a global decisions very efficiently and quickly. As soon as the operator looks at the worksite, he can suggest how the manipulator should move in order to get to a target without collisions. On the other hand, the human is not good at handling details, and sometimes get careless and bored, especially after some period of involvement in a task. This shows that the human is not a good decision maker at a lower level. The computer, however, does not become bored or careless, and is very good at handling detailed information.

1.2.2 Examples of Supervisory Control

A modern automobile may be considered a supervisory controlled system. Multiple microprocessors control fuel injection and the driver controls the vehicle speed by pressing an accelerator pedal. Cruise control can drive the automobile at constant speed but the driver as a supervisor sets and resets the speed. Another example is the laser-guided bomb that homes in on a target at which the operator aims a laser beam. Robotic applications of supervisory control have been mostly in space robots and underwater robots.

In earth orbit and in interplanetary space flight, robot vehicles have been controlled by earth-bound human supervisors in a "move-and-wait" fashion, as dictated by the unavoidable long communications delay. Probably the most sophisticated actual use of robots has been for space exploration, though even here the more recent advances in artificial intelligence and computer control have not yet found application.

1.2.3 Role of Supervisor

A supervisor of human subordinates is normally thought of as an individual who directs the actions of subordinates. The supervisor's function is to *plan* future courses of action for the subordinates, *teach* them the proper method of carrying out these actions, *monitor* their performance, *correct* their actions whenever they do not meet expectations, and *trust* the subordinates to perform the actions as directed.

Similarly supervisory control of a robot system may be said to operate in five roles, normally occurring in the following order and necessitating human interaction with

different parts of the robot system.

Sheridan describes five classes of human function in robot supervisory control scheme (1986). These functions alternate between afferent (the human getting information from the environment) and efferent(the human taking an action to change the environment): planning, monitoring and learning are for the purposes of getting information but do not themselves change the environment. Teaching and intervening are actions to effect changes in the environment.

1. *Planning* is done in anticipation of response to future events (rather than in response to immediate control requirements) and thus is self-paced. It involves consideration of what sensing strategies to employ, then what responses to implement in the case of different sensed inputs, and finally what result will occur in terms of a given objective function or performance criterion. Computer aids may or may not be employed.

2. *Teaching* means conveying necessary and sufficient instructions to the computer to make it run automatically for an interval of time, or make it stop or change to a different program under specified conditions of its sensors. It is also self-paced.

3. *Monitoring* is the allocation of the supervisor's limited attention capability among the displays to ensure that everything is working properly, or to detect and diagnose symptoms of difficulty. The supervisor's pace, forced by the system complexity and the work rate of the robot, at times may be slow and boring, at times very rapid and stressful, or may change rapidly from one pace to another.

4. *Intervention* is when the human operator reverts to a more direct or continuous

mode of control, bypassing higher level executive control. This may be done in emergency, or to make routine tests, or to perform maintenance or repair.

5. *Trusting* is a term which can be used to characterize the overall meta-skill which the operator acquires to learn when and how far he can trust the robot, when to move from one of the above four roles to another, when to move his attention from one robot to another, and when to rest or do other things.

1.2.4 Why Supervisory Control of Teleoperators?

Sheridan (1987) describes necessity of supervisory control in teleoperators as follows.

> For a man to be present and do useful work in space, at ocean depths, in the presence of nuclear radiation, or in chemically toxic environments requires expensive life-support facilities; even so, work in such environments is risky. Yet development of autonomous robots is nowhere near replacing man in unpredictable nonrepetitive situations. Master-slave manipulators, extending man's arm-hand motions one-to-one through electromechanical servomechanisms to a pair of mechanical hands, are common in nuclear laboratories and deep-ocean vessels. In space, when distances are such that transmission-time delays (speed of light) are significant, master-slave manipulation is not practical. Computer-controlled but man-supervised telemanipulators offer a solution for guiding TV and tactile sensors in space, for assembly and maintenance, and for general exploration. Under the sea, and in nuclear and chemical environments, though the time delay is not a problem, com-

puter control still permits higher speeds in semiroutine tasks. It provides protective reflexes such as ones which automatically adjust gripping forces to avoid slip yet prevent crushing objects, and ones which avoid collisions with obstacles, prevent large pushing forces, etc.

The reasons for a supervisory controlled telerobot in preference to an autonomous robot is that autonomous robots at the present time are neither intelligent nor reliable enough to perform any but the simplest routine tasks.

The reasons for supervisory control of a telerobot in preference to direct manual control of teleoperator (e.g. by master-slave or joystick control) are: (1) quicker task completion in spite of telecommunication time delay; (2) quicker automatic reflex responses to unanticipated events; (3) greater accuracy in continuous control of position and force; (4) ability to perform long-duration tasks without continuous attention by the human operator; and (5) increased reliability of task performance.

1.3 Collision Avoidance for Robot Manipulators

Robot manipulator motions can be split into two categories: gross motion and fine motion. The objective of the gross motions of the manipulator is to move the manipulator from one place to another while avoiding collisions with objects (obstacles) in the environment. Gross motions are the ones in which we are interested in this thesis. Fine motions are manipulator motions just before or after some kinds of contacts with objects. Typically, controlling the impedence of the manipulator appropriately is an important

issue during a fine motion. Collision avoidance is the most important constraint that the robot manipulator has for gross motions. Thus collision avoidance is essential for any task planning for manipulator motions.

Avoiding collisions with objects in the environment during gross motions of the manipulator is a research area that many people have been working on. In this section, we are going to review some of the important works regarding collision avoidance for the manipulator. In Section 2.5, the supervisory approach to collision avoidance is explained. For the more complete survey of different approaches to collision avoidance for the manipulator, refer to Whitesides (1985), and Latombe (1991).

1.3.1 Complexity

Schwartz and Sharir (1982) have demonstrated the existence of a polynomial-time algorithm for the Movers' problem with fixed degree of freedom. The classical mover's problem in d-dimensional Euclidean space is to find a sequence of translations and rotations from the initial configuration to the final configuration of an object without contacting any obstacle in the environment which has a set of polyhedral obstacles fixed in Euclidean space. Time to solve the classical Mover's problem using their algorithm is unfortunately order of $O(n^{2d+6})$, where n is polynomially dependent upon the number of faces in the environment, and d is the number of degrees of freedom. The most effecient algorithm to date is one developed by (Canny, 1987), which is singly-exponential in d. This is still too complicated to be implemented in pratical applications. Hence it serves mainly as a proof of existence for a algorithm.

1.3.2 General Approaches

Collision avoidance algorithms for a three-dimensional space, in general, can be grouped into the following classes (Lozano-Pérez and Brooks, 1985):

1. Hypothesize and test.
2. Penalty function.
3. Explicit free space

There are other collision avoidance schemes that combine any of the above methods, and they are called the *hybrid approaches*.

Hypothesize and Test

The basic operation of the hypothesize-and-test method consists of three steps: first, a candidate path between the initial and target configuration of the manipulator is hypothesized; second, a selected set of configurations along the path is tested for possible collisions; third, if a collision is found, an avoidance motion is proposed by examining the obstacle causing the collision. Modifying a proposed path when a possible collision is found is very difficult; typical proposals for path modification rely on drastic approximations of the obstacles, such as enclosing spheres.

Myers and Agin (1982) modeled an object as a primitive object such as a sphere, box, or an infinite plane. Collision detection between two objects is done initially with enclosing objects, i.e., spheres. When an impending collision is detected, a few points

are selected in the vicinity of the colliding object based upon heuristics as candidates for an immediate target position to which the manipulator should move in order to avoid the collision. These points serve as nodes of a search tree when a collision-free path is searched. One of the drawbacks of the scheme is that the selection of intermediate points is based upon a sphere that encloses the obstacle. This, therefore, selects intermediate target points very conservatively. Another drawback is that it doesn't have any special heuristics for collision occurred at the elbow of the manipulator.

These methods work fairly well when the obstacles are sparsely distributed so that they can be dealt with one at a time. When the space is cluttered, however, attempts to avoid a collision with one obstacle may lead to another collision with a different obstacle. The main advantage of a hypothesize-and-test method is its simplicity.

Penalty Function Approach

The penalty function approach defines and uses a penalty for the manipulator for being close to obstacles. Such a penalty encodes the geometric information on an obstacle: it typically becomes infinite in the vicinity of the obstacle surfaces and drops sharply with distance from them. The goal is represented by a valley (or a sink) of the potential field. The total penalty of the robot manipulator at a particular configuration is the sum of penalties for all the obstacles in the environment. The penalty function is a function that relates the total penalty and the manipulator configuration. Figure 1-1 shows a two-dimensional version of potential function used as a penalty function.

Robot motion at any configuration is decided by the *gradient* of the penalty function

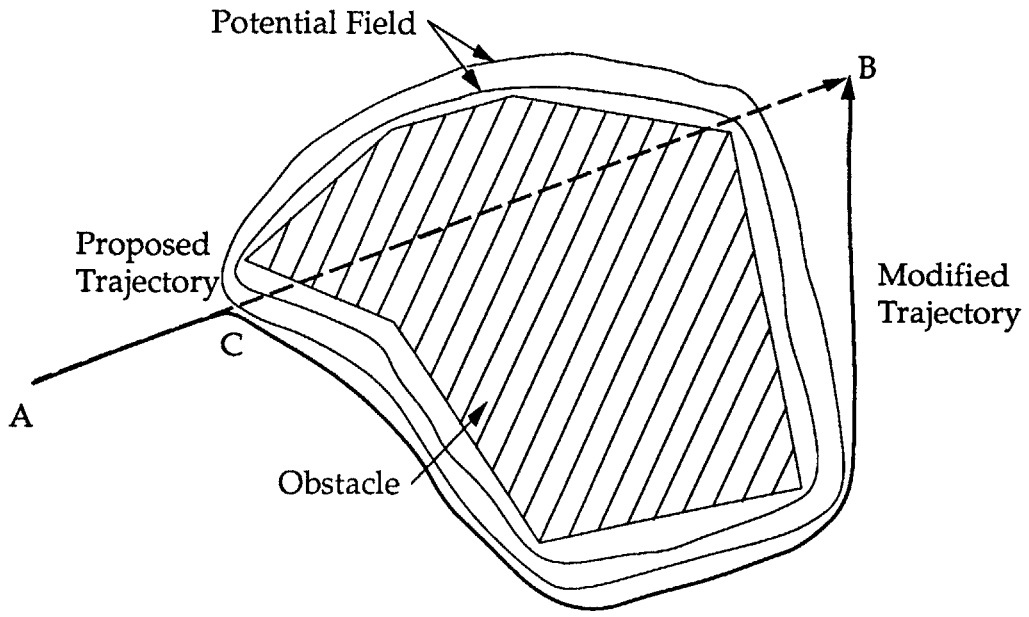


Figure 1-1: Potential Approach.

at the configuration: the manipulator moves in the direction that has the maximum negative gradient of the penalty function. The concept of the penalty function approach is simple.

Hogan (1985) used potential field as part of his impedance control. Khatib (1986) implemented a collision avoidance scheme based on artificial potential field which was based upon the *distance* of the manipulator from the object instead of the position of the manipulator and analytical penalty functions. Potential with an analytical penalty function does not require costly computation of the distance, but is difficult to use for asymmetric obstacles where the separation between an obstacle's surface and equipotential surfaces can vary widely. Khatib's method creates local minima points of the potential field, which could trap the robot (Khosla and Volpe, 1988).

The most significant drawback of such a penalty function approach is that it uses only local information during path searching. Following local minima does not guar-

antee that it will reach the global minimum, the manipulator target location. Another drawback is that the manipulator cannot approach an obstacle without collisions because the potential field does not allow it. Depending upon the actual implementation, it may show bouncing around obstacles.

Explicit Free Space Approach

The explicit free space approaches builds explicit representations of subsets of robot manipulator configurations that are free of collisions. These are called *free space.* Obstacle avoidance is to find a continuous path connecting the initial configuration and the target configuration within the free space.

A method by Brooks (1983) creates free-space from high-level description of the empty corridors between prismatic obstacles, which is based upon the use of generalized cones). He used the generalized-cone concept for a manipulator with 5 or 6 revolute joints by describing free space in two ways: as freeways for the hand and payload ensemble and as freeways for the upper arm. However, the best-known methods are ones based upon *configuration space.* Configuration space (*C-space*) of a rigid solid is defined as a space formed by six-dimensional vectors that represent the positions and orientations of the solid. Configuration space of a 6 d.o.f. revolute manipulator is the joint angle space, Θ, which *uniquely* determines the configuration of the manipulator. Udupa (1977) introduced a new way of computing the free space for the Stanford Arm using two cylinder models, one for the boom and the other for the forearm. Lozano-Pérez developed the idea of Udupa further and developed an algorithm for computing the free configu-

ration space for a general polyhedral object (Lozano-Pérez and Wesley, 1979). He also presented approximate solutions for a Cartesian manipulator with 6 degrees of freedom which consider three-dimensional slice-projections of C-space (Lozano-Pérez, 1981). It is a very clever algorithm in the sense that it solves two-dimensional collision avoidance very easily and efficiently. Figure 1-2 illustrates the method in a two-dimensional space for a finding collision-free path for a polygon (a triangle). It is assumed that the polygon can only move in the space without rotation. A point, A, on the moving object is arbitrarily chosen as a reference point. Then each obstacle is grown to compensate precisely for shrinking the moving polygons to point A, and a new configuration space is formed for point A. Collision-free path for the polygon in the two-dimensional space corresponds to a collision-free path for point A in the new configuration space. The shortest collision-free path for point A is a piecewise linear path connecting the start and the goal configuration via the vertices of the grown obstacles. A graph, called a visibility graph, is formed by connecting all pairs of vertices of configuration space obstacles (and the start and goal) that can be connected by a straight line that does not intersect any of the obstacles. Then the shortest path is selected by searching through the viewgraph.

A three-dimensional problem with rotation results in a six-dimensional configuration space. As the overall dimensions increases, it is difficult to handle a complicated configuration space. A rotational motion of the manipulator with revolute joints could be handled by defining several free-space representations, each using a manipulator model that represents the volume swept out by the rotational links over some range of joint angles.

Path searching in such a high dimensional space is not so easy as it is in a two-

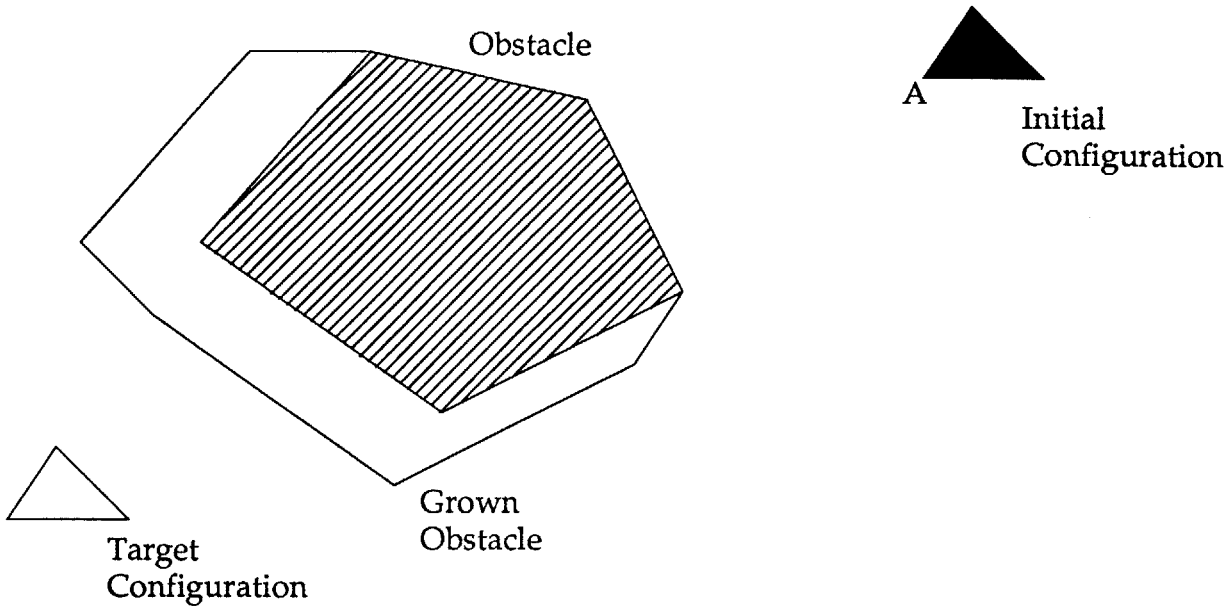


Figure 1-2: C–space Approach.

dimensional case. Moreover, searching in a viewgraph, for example, has several drawbacks when obstacles are three-dimensional:

1. Shortest paths do not typically traverse the vertices of the grown obstacles.

2. There may be no paths via vertices, within the enclosing polyhedral region.

Another way to build a search tree is to divide the space into a set of cells rather than generating a viewgraph. Brooks and Lozano-Pérez (1982) divided C-space into rectangloid. Then each cell is determined to be *full*, i.e., completely occupied by an obstacle, *empty*, i.e., completely occupied by no obstacle, and *mixed*, i.e., part *full* and part *empty*, which can be further expanded into other *empty*, *mixed*, and *full* cells. Searching is executed in order to find a set of *empty* cells from an initial configuration to a target configuration which are adjacent one another. If such an empty cell path cannot be found in the initial division, mixed cells on the path are subdivided and each resulting

cell is appropriately labeled as *empty, full,* or *mixed.* This process is repeated until a collision-free path connecting an initial configuration and a target configuration through a series of *empty* cells, is found.

Hybrid Approach

There are some works that combines more than one approaches. Faverjon (1989) proposed a hybrid approach by combining the global and local approaches. At the global level, the configuration space is discretized in quite big cells. For the global-level planning, probability for the local planning to succeed in moving the robot from one cell to another adjacent cell that are learned during execution of planned trajectories are used. At the local-level planning, local free configuration space is generated based upon the the velocity of the manipulator and the distance of the manipulator from obstacles, and minimization of some measure, as in the potential field approach, is performed with the constraints in the configuration space.

Cheung and Lumelsky (1988, 1990) used a sensitive "skin" (proximity sensors) and heuristic algorithm in configuration space in order to avoid manipulator collisions with obstacles.

1.4 Uncertainty of Environment

1.4.1 What is uncertainty of the environment?

The real world is filled with uncertainty. There are many areas in the context of robot applications that uncertainty is important, such as object and environment modeling, task planning, sensor integration, and kinematics of the manipulator. However, uncertainty of the environment to be dealt in this thesis is restricted to one in object and environment modeling.

1.4.2 Uncertainty of Modeling

The most important uncertainty in this category is inaccuracy of information on spatial relationships among objects. Another type of uncertainty of modeling is so-called "uncertainty of existence," which means that it is not known if a certain object exists.

There are two different approaches to handle the uncertainty due to inaccuracy of positions and orientations of objects. One is the *probabilistic* approach that explicitly represents and manipulates the uncertainty associated with transformations of objects. The most important advantage of this approach is that it can represent uncertain geometric features in a consistent and homogeneous manner, which allows these uncertainties to be manipulated easily. Smith (1987) and Smith et al. (1987) modeled uncertainty of the transformation of an object by estimating the first two moments of its probability distribution, and showed how to compose and merge, using a Kalman filter, such

transformations. Durrant-Whyte (1987) developed a description of uncertain geometric features as families of parameterized functions together with a distribution function defined on the associated parameter vector, and analyzed effects of changing locations or feature descriptions.

The other approach assumes error bounds on geometrical parameters. This results in "error manifolds" and "uncertainty cones", over which the error has a uniform distribution. A drawback of this scheme is that results can become very conservative. Another drawback is that they are not so easy to manipulate mathematically as the probabilistic approach. Some works on error propagation were based on this method (Brooks, 1982a).

1.4.3 Sources of Uncertainty of Modeling

Uncertainty of the position and orientation of an object is inserted into the model of the object as errors of position and orientation when it is introduced into the world model. One of the reasons why errors, in many cases, cannot be avoided is that accurate measurements on position and orientation of an object are not possible, feasible, or economical. Uncertainty is an intrinsic part of modeling, no matter how small it is.

1.4.4 Uncertainty and Manipulator Motions

As uncertainty of the environment increases, the guaranteed free space of the manipulator shrinks. This, in general, would make the manipulator move over a longer distance, and make planning for collision-free motions more difficult, resulting in slow comple-

tion of a task. On the other hand, reducing uncertainty of the environment costs time and resources. More sophisticated equipment and more elaborated model building are required. So there exist trade-offs between uncertainty of the environment and the speed of completion of a task.

1.5 Problem Statement and Scope of Work

The main goal of this work is to develop a human *supervisory approach* to the collision avoidance problem in a three-dimensional world with some uncertainty in order to overcome the problems that other approaches have, and to evaluate the usefulness of the system through realistic experiments of robot operations in different environmental scenarios. Another goal of this work is to develop a method to build a world model with available geometric data on the environment, and to evaluate the trade-offs between having a crude world model and having a precise world model in the context of manipulator collision avoidance.

The scope of the work, including assumptions, is as follows.

1. The goal is to enable the human operator to solve the collision avoidance problem quickly by cooperation with the computer.

2. The above includes improving the operator's perception of both the environment and the manipulator kinematics with a display of visual information.

3. Only the kinematics of the manipulator is considered; the dynamics of the manipulator is neglected.

4. Only gross motions of the manipulator are considered; Fine motions of the manipulator are not considered.

5. All the objects considered here are assumed to be convex polyhedra. (Any concave polyhedra can be represented by a multiple convex polyhedra.)

6. Experiments are done in a graphically simulated environment in order to evaluate system performance. The capability of avoiding collisions automatically is compared with the capability of detecting collisions and allowing the human operator to command the avoidance movements.

7. A supervisory method of registering objects to the world model is suggested and an experiment is performed to evaluate the method.

1.6 Outline

Chapter 1 describes supervisory control of robots from the perspective of the man-machine interface and previous works in the collision avoidance, and sets the frame work for a supervisory approach to collision-avoidance scheme that follows. There are 4 chapters following this chapter. Chapter 2 describes a method for detecting collisions using clipping hardware, a collision avoidance scheme based upon collision detection and heuristics, and graphic aids for the human operator. Chapter 3 describes uncertainty in the environment. It shows what virtual obstacles are, and how they are used in the context of collision avoidance. A supervisory method to introduce an object into the world model by flying-and-matching is also suggested in this chapter. Chapter 4 reports on an

evaluation of the supervisory approach to collision avoidance through experiments and introducing objects into the world model by the "flying-and-matching" method. This chapter shows the results from the experiments and describes how the system could be improved. Finally, in Chapter 5, the work is summarized and areas of further research are recommended.

Chapter 2

Supervisory Control for Collision Avoidance

2.1 Background: Supervisory vs Autonomous Planning

In totally autonomous systems, manipulator paths are generated by a hierarchical planner. Typically a planner consists of a high-level planner[1] and a low-level planner. Consider an example of mounting a motor on a frame as shown in Fig. 2-1. In the initial state, the frame (A), the motor (B) and two bolts (C and D) are separated. The goal is to mount the motor as shown in Fig. 2-1. High-level planners are supposed to come up with a plan of the form: move the motor underneath the frame and align the hole with the shaft of the motor. While holding the motor, insert bolts and tighten them one by one. Based upon the high-level plan, lower-level planners, using world models, generate

[1]Examples of the high-level planner are STRIPS (Fikes and Nilsson, 1971) and NOAH (Sacerdoti, 1975), a nonlinear planner, generate high-level plans.

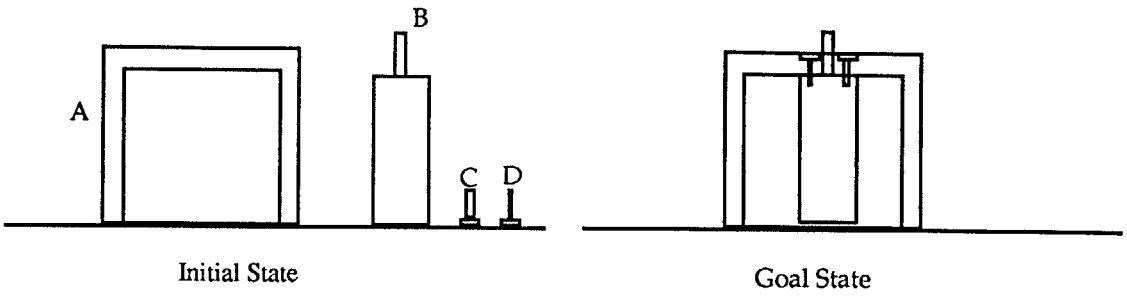


Figure 2-1: Planning.

detailed plans for the manipulators to achieve given goals. A detailed plan may be, in the context of above example, a path of the manipulator to move the motor underneath the frame without a collision between the motor and the frame. Whereas high-level plans are very conceptual, low-level plans, for example how to maneuver the motor so that it does not collide with the frame, have details and are executable; thus, low-level planners have to deal with world models more extensively than high-level ones.

There are a number of problems with this kind of approach. First, world models do not always match the actual environments which they purport to model. They may be inaccurate and incomplete. Models are incomplete because they are defined to be incomplete and to represent corresponding actual objects in a certain limited way. However, the meaning of incompleteness of the world model used here is that some objects that exist in the environment are not included in the world model. More likely, models get incomplete when unexpected situations occur. In many cases of telerobotics, the environment has both modeled objects and unmodeled objects. In such an environment, the autonomously generated manipulator paths are not guaranteed to be free of collisions.

Second, planning collision-free motion is computationally expensive. For example,

collision avoidance algorithms using configuration space (C-space)[2] require mapping of all the geometric data on obstacles from the world coordinate to configuration space before performing a search operation, which is, in most cases, a joint vector space. Mapping geometric data takes long time and is required frequently if the manipulator has a mobile base. The C-space for the manipulator with 6 degrees of freedom revolute joints is too complicated to handle.

2.2 Supervisory Control System

For the reasons explained in Section 1.2.1, a supervisory approach where the computer and human work together can improve the efficiency of a task. Figure 2-2 shows the involvement of the human operator in action. In the context of collision avoidance, the human operator takes charge of making a global decisions as to how the manipulator should move in order to avoid collisions with obstacles in the environment, and suggests a desired path to the computer. The computer, then, simply executes the path suggested by the human operator, and make corrections if it finds the path is not collision-free. The operator does not have to be involved in making such detailed corrections. Instead, he can spend time in planning subsequent paths.

Figure 2-3 shows the structure of the overall system. The operator is offered visual information regarding the manipulator and the environment. He can get visual information from (simulated) actual cameras and the world model in the computer. A command issued by the operator is processed or *filtered* in the graphic simulation block

[2]Collision avoidance using C-space was explained in more detail in Section 1.3.

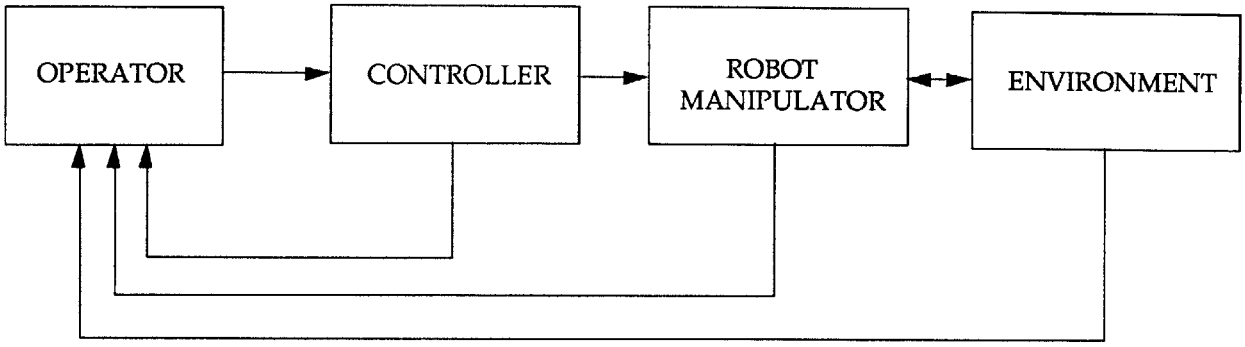


Figure 2-2: Block Diagram of the Supervisory Control System.

with aids of collision detection, avoidance and world model sub-blocks. Only motion commands that would not result in collision of the manipulator are passed to the actual manipulator.[3]

2.3 Supervisory Specification of a Robot Path with World Model

In the supervisory mode, the operator specifies intermediate points of a manipulator path, and then the manipulator moves in such a way that its tip moves straight from one intermediate point to another until it reaches the target position. If the operator predicts that a collision may occur during any straight motions of a given path, he/she could select and input other intermediate points that the hand tip should go through in order to avoid the collision.

This supervisory approach to path specification has advantages over full autonomy. If there are unmodeled obstacles in the way, the operator may want to propose a path

[3]The actual manipulator is also graphically simulated.

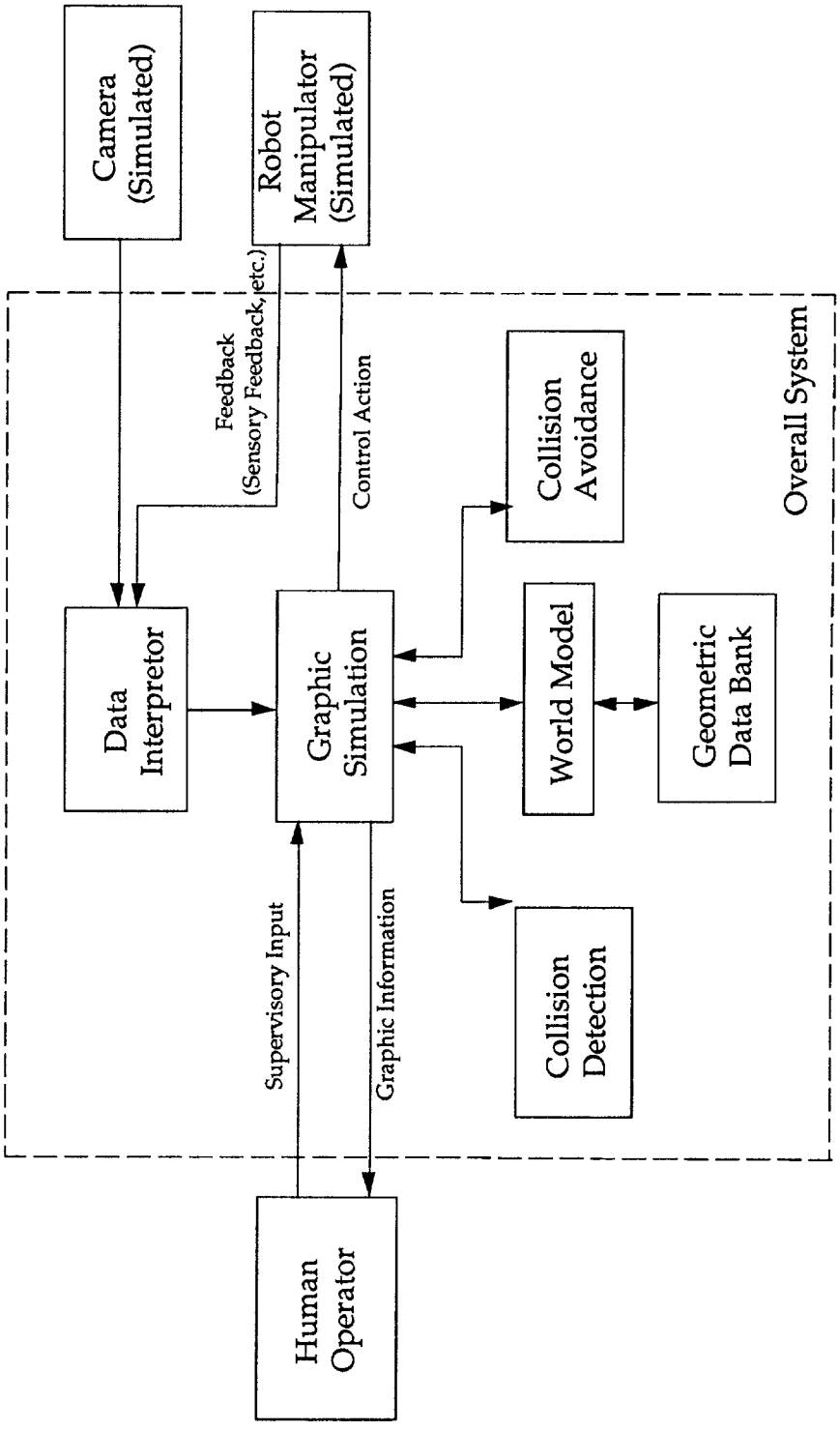


Figure 2-3: Block Diagram of the Supervisory Collision Avoidance System.

rather than letting a computer algorithm, which is ignorant of those obstacles, generate a path. Another advantage is that planning a path takes little time simply because human beings are "better" planners, at least at a global level.

Intermediate locations that the operator specifies and that the manipulator tip passes through, can be represented by three-dimensional positional vectors. A method to specify such a three-dimensional vector using a mouse (a two-dimensional input device) was developed after observing a simple fact: the vertical orientation of the operator is fixed and aligned with the earth gravitational field, but the horizontal orientation is not fixed. This is because human beings have been accustomed to the gravitational field. For example, "above A" means a spatial location that is farther away from the earth than A but the meaning of "in front of A" depends upon the position of the speaker or the position of A. In general, a datum of the vertical orientation is a horizontal surface that people and/or objects are standing upon. In horizontal orientations, it is difficult to find a reasonable datum of the orientation. It is possible to define mathematically an orientation (x and y axis) and a datum (origin) on a horizontal surface. Then any location in the surface can be represented by a set of two numbers, x and y. However, using this kind of mathematical scheme is not feasible because the origin and the orientation of the horizontal surface cannot always be accurately located by the operator. The operator may see a scene where the orientation is not shown. Even if it can be located, the relative distance of a location from the origin is not precisely measurable because the operator does not have precise depth perception. Therefore, it is not practical in a teleoperation to issue an instruction such as "move (5, 0, 10) relative" or "move to (10, 20, 21).

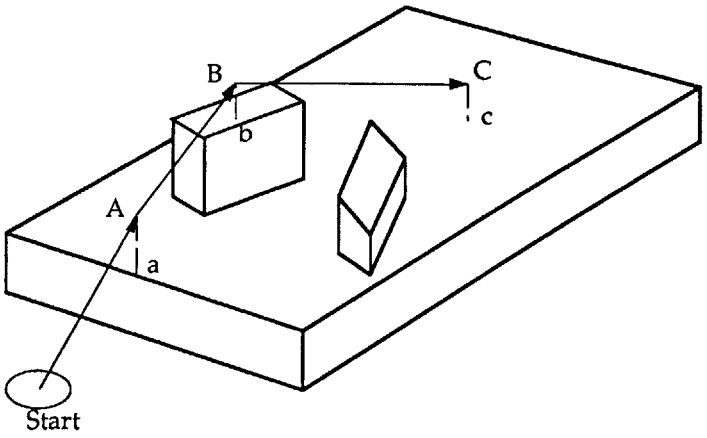


Figure 2-4: An Interactive Path Specification Using the Geometric Model of the Environment.

In the proposed method of describing a three-dimensional location, the operator uses the mouse to point to a location on a horizontal surface such as a floor, and then specifies a vertical distance from the point by dragging the mouse. Figure 2-4 shows a typical example of specifying three-dimensional locations. Suppose the operator wants to instruct the manipulator to move its hand tip from a point Start to a point C, via points A and B. The operator uses the mouse to indicate point em a on a horizontal surface, which in this case is one of the surfaces of another object. The operator then clicks a mouse button, and drags the mouse until the mouse points to A. While dragging the mouse, a vertical line from a point *a* to some point that is the closest to the mouse cursor is interactively displayed so that the operator knows the height of the point that he/she is specifying. Similarly, the operator specifies point B and point C to complete the specification of the path.

2.4 Collision Detection

2.4.1 Introduction

In many industrial environments and computer simulations, it is often necessary to ascertain whether there is a collision between objects. Algorithms to detect collisions between objects vary depending upon the type of objects and the constraints on them. For example, consider two perfectly aligned cubic objects of the same size that are subject to translation without any rotation. In this case, the simplest method to check collisions between the two objects is to calculate the vector which points to the center of one object from the center of the other. If all components of the vector in the direction of surface normal vectors are larger than the size of an edge of the cubes, they do not collide with one another. If any of the components is less than the size of an edge, the two cubes collide with one another.

For more complex objects than simple cubes, more systemic methods should be used for collision checking. Collision checking between objects that are modeled as polyhedra can be done by analyzing the topological relationship of the two objects (Boyce, 1979). For curved surfaces, piecewise planar approximations to them are adequate in many collision checking applications. Any surface can be approximated as closely as desired by polygon patches. To get a very close approximation to a curved surface, however, requires a large number of facets, which in turn implies a large amount of storage and long processing times for collision checking.

Another method to detect collisions between two objects uses the clipping algorithm

of computer graphics (Smith, 1985; Hoffman and Hebert, 1986). This is a very convenient way to solve the collision checking problem, especially when the geometric models of objects are already available. Clipping is a process which cuts off the lines and polygons that are outside a window so that only the lines and polygons within the window are displayed. Many different algorithms are available for clipping. For example, an algorithm developed by Sutherland and Hodgman can clip polygons, simple lines, and characters (Harrington, 1983).

2.4.2 Collision Detection Using Clipping Algorithm

Collision detection is performed using the capability of an Iris workstation to detect drawing an object within the clipping boundaries. Clipping boundaries are defined by a projection transformation. The relationship between clipping boundaries and projection transformations is explained in Appendix A.

2.4.3 Collision Detection in Two-Dimensional Space

A collision detection scheme for two-dimensional objects, which is simpler, is presented first, and then collision detection scheme for three-dimensional objects with simple shapes is presented, followed by a generalization of the scheme to objects with more complicated shape.

Collision checking for two-dimensional objects using clipping hardware can be described more easily with an example, as shown in Fig. 2-5. It is assumed that there

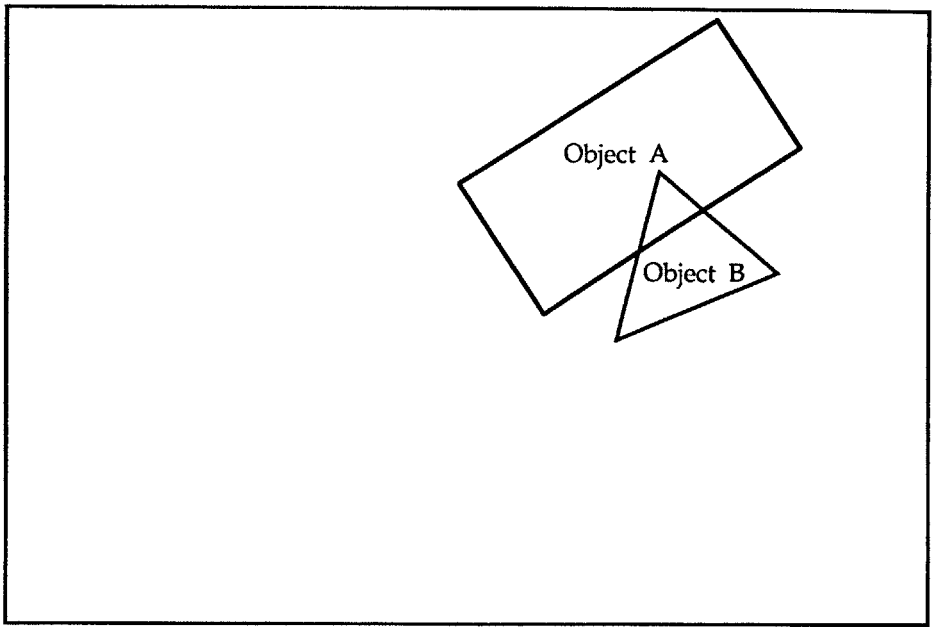


Figure 2-5: Two Objects in Collision.

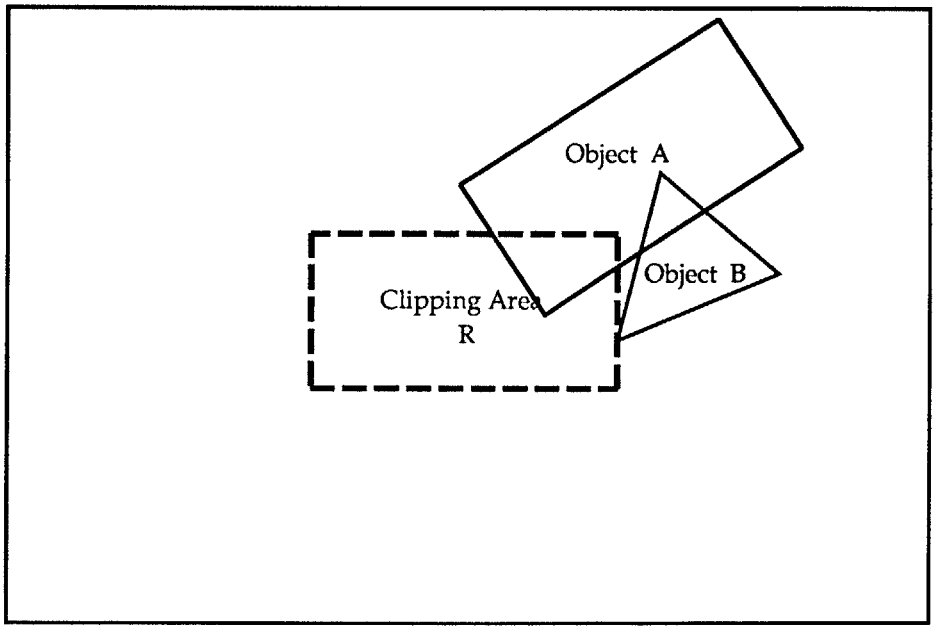


Figure 2-6: Two Objects in Collision and a Clipping Area.

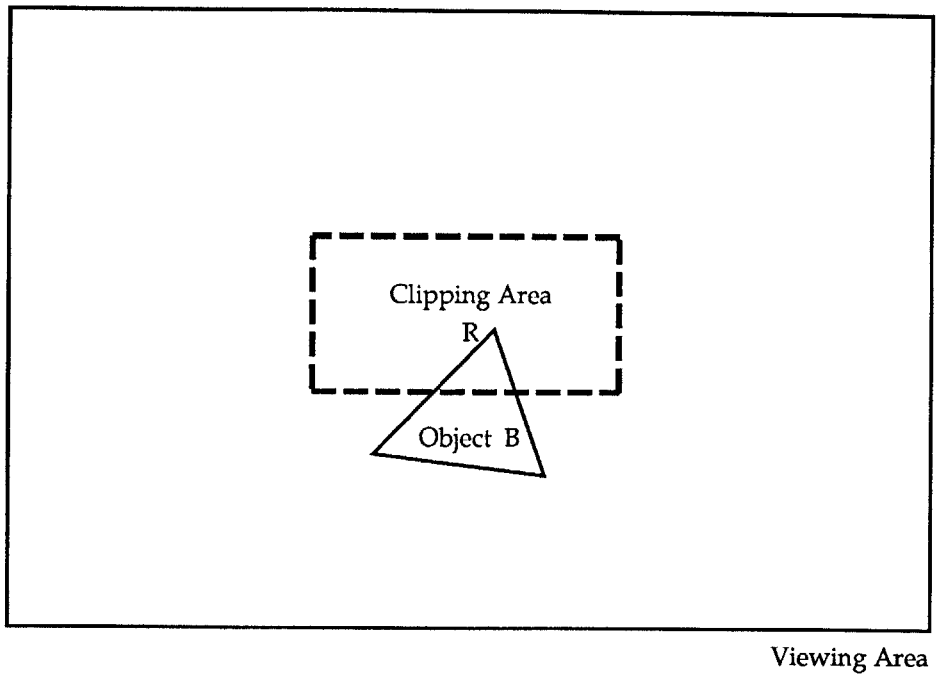


Figure 2-7: Collision Detection through a Clipping Area.

are two objects, object A and object B, and that they are in collision with each other, meaning there is an overlap between the area that object A occupies and the area that object B occupies. The task is to detect the collision between object A and object B. More precisely, it is checked if object B intrudes into the area that object A occupies. The first step for collision detection is to create clipping boundaries that enclose a region that has the same shape and size as object A. In general, clipping boundaries are created in the middle of the screen window. In Fig. 2-6, a clipping area, R, is enclosed by clipping boundaries, which form a rectangle. Since object A has the same shape and size as the clipping area R, object A could be rotated and translated such that it geometrically exactly matches the clipping area. Let θ and x denote the rotation angle and translation vector of object A, respectively, that would result in the matching between object A and the clipping area. The next step is to rotate and translate the world excluding object A (therefore, object B) by θ and x, respectively, and then draw it. Figure 2-7 shows the

position of object B after rotation by θ and translation by x. Note that the position of object B relative to the clipping area is the same as the position of object B relative to object A shown in Fig. 2-5. Now, object B is in collision with the clipping area in the same way as object B is in collision with object A. Since the graphic workstation can detect drawing of object B within the clipping area, it can detect the collision between object A and object B.

2.4.4 Collision Detection in Three-Dimensional Space

The basic idea of collision detection in a two-dimensional space, described in the previous section, can be extended to collision detection in a three-dimensional space. Let's use transformation matrices in order to make the description simpler, rather than dealing with translational vectors and rotation angles as in a two-dimensional space. Figure 2-8 shows two objects whose coordinate frames are A and B, respectively. Coordinate E is the base coordinate of graphic rendering and the viewer is considered to be at the origin of the coordinate system looking at the world. Conventionally, the positive z-axis of coordinate system E points toward the viewer. The position and orientation of object A and object B relative to the coordinate system E can be described by a transformation matrix, ${}^E_A T$ and ${}^E_B T$, respectively. In order to draw object A and object B in graphic simulation, the coordinates of the objects should be transformed by ${}^E_A T$ and ${}^E_B T$ from coordinate E, respectively, before actual drawing of the objects take place.

The first step for detecting (possible) collision between object A and object B is to create three-dimensional clipping boundaries at the origin of coordinate system E. In

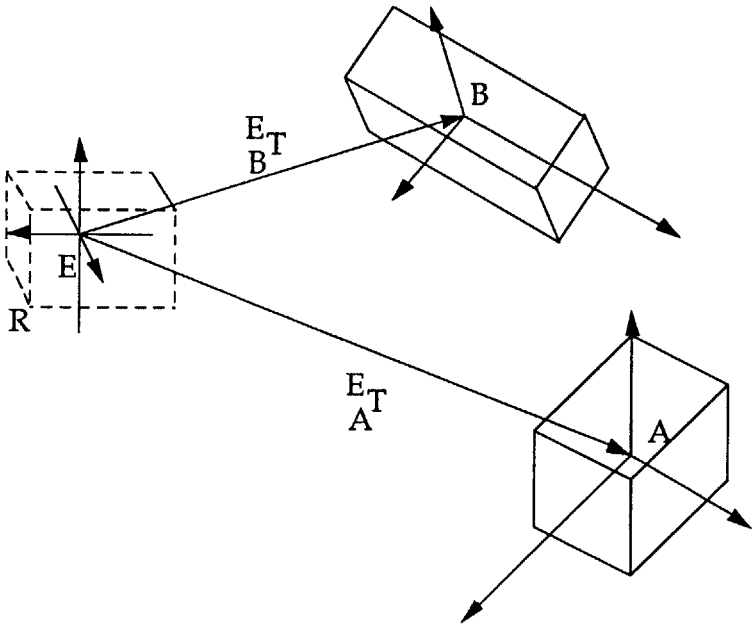


Figure 2-8: Object A and Object B.

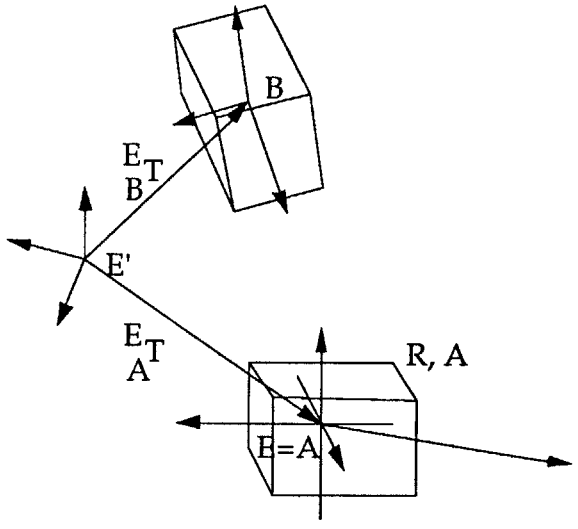


Figure 2-9: Object B and Clipping Boundaries.

general, clipping boundaries, and thus clipping volumes, are created by viewing transformations. The size and the shape of the clipping volume should be exactly the same as that of object A. Denote the object defined by the boundaries as object R. Then the next step is to find a transformation that would bring object A back to the origin of E such that the volumes of object A and object R are matched to one another, resulting in

$$\mathcal{V}(\mathcal{A}) = \mathcal{V}(\mathcal{R}) \,, \tag{2.1}$$

where the volumes occupied by object A and object R are denoted as $\mathcal{V}(\mathcal{A})$ and $\mathcal{V}(\mathcal{R})$, respectively. That transformation is simply an inverse of ${}^{E}_{A}T$, or ${}^{A}_{E}T$. This means that if the coordinate E is transformed by ${}^{E}_{A}T^{-1}$ and then object A is drawn normally, which means that object A is drawn after being transformed by ${}^{E}_{A}T$, object A would fit to object R. Therefore, if the whole world, which consists of object A and object B in this example, is drawn after its original coordinate frame is transformed by ${}^{E}_{A}T^{-1}$, object A would be located such that $\mathcal{V}(\mathcal{A}) = \mathcal{V}(\mathcal{R})$ as in Fig. 2-9. When the world is drawn after such transformation of its coordinate frame, the relative position of object A to object B remains the same. This means that the relative position of object A and object B in the world is the same as the relative position of object R and object B. Thus, a collision between object A and object B in the world would result in intrusion of object B into the clipping boundaries. Likewise, if there is no collision between object A and object B, there would be no intrusion of object B into the clipping boundaries. Such an intrusion into the clipping boundaries is detected by the hardware of the workstation.

2.4.5 Viewing Transformations

Orthogonal transformations and perspective transformations are the best–known forms of transformations. For more details, refer to Appendix A.

General Objects

Clipping boundaries are defined by a viewing transformation. If the volume of an object cannot be represented by a single viewing transformation, the object can be represented by a set of different viewing transformations each of which represents a sub-volume of the object. As long as the union of sub-volumes of the object is equal to the volume of the object, the object can be divided into the sub-volumes. In order to check if the object collides with an obstacle, each sub-volume has to be checked for collisions with the obstacle. A collision of one of the sub-volumes with an obstacle guarantees a collision of the original object with the obstacle.

Some examples of objects that can be represented by a set of viewing transformations are explored below. An object shown in Fig. 2-10 is a hexagonal solid. It can be represented by three box-shaped volumes, each with 60 degrees of angle apart, which can be represented by three identical orthogonal transformations. This idea can be extended to such solids with more complicated surfaces. A solid with an octagonal cross section, for example, can be represented by a set of four identical orthogonal transformations. And a cylindrical object can be represented by a set of many (theoretically, an infinite number of) identical orthogonal transformations.

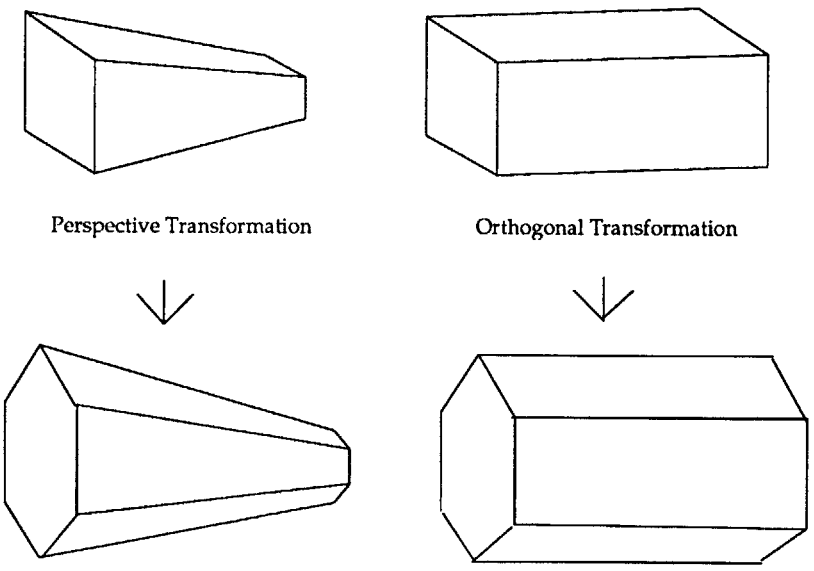


Figure 2-10: Solids Can Be Constructed with a Perspective or Orthogonal Transformations.

Perspective transformations are suitable for representing tapered objects. A tapered object shown in Fig. 2-10 is represented by a set of four identical perspective transformations. Again, a tapered cylinder can be theoretically represented by an infinite number of perspective transformations.

2.4.6 Collision Detection for Manipulators

For the purpose of detecting collisions, a manipulator motion can be considered as a sequential series of different geometric configurations of the manipulator, each with sequential incremental changes of configuration variables such as joint angles. For example, a motion resulting in a straight path at the manipulator tip consists of a series of geometric configurations with incremental changes, in each of which the manipulator tip is on the straight line and a certain distance from the previous one. In order to prevent manipulator collisions, collision checking is performed for each and every geometric

configuration of the manipulator. Moreover, the dimension of incremental configuration changes should be kept small to reduce the possibility that collision occurs in the middle of an incremental change of the configuration.

In general, there are two kinds of manipulator collisions, depending upon the subjects involved in the collision: collision against itself and collision against obstacles in the environment. An example of the first is a collision between the manipulator tip and its shoulder. As far as collision is concerned, other parts of the manipulator body are considered as obstacles. In order to assure that the manipulator in a particular geometric configuration has no collision, each part of the manipulator has to be checked as to whether it collides against other parts of the manipulator *and* if it collides against any obstacles in the environment.

It is possible to determine whether a certain configuration of the manipulator would result in collisions with objects(or obstacles) in the environment. There are two kinds of situations with respect to the availability of manipulator configurations for collision checking. One is off-line programming, where all the manipulator configurations are available before actual motion. In this case, collision checking for all the configurations can be performed once before the actual motion. The other is in manual control, where the operator uses input devices such as a joystick or on-line programming. Here collision checking should be done as the manipulator moves, to specify an incremental position vector or a velocity vector of the manipulator tip. In this case, the only configuration available for collision checking at a certain moment is the next set of incremental changes from the current configuration. Unless collisions are found for the impending manipulator configuration, the manipulator moves to that configuration.

2.5 Supervisory Collision Avoidance

In this section, a supervisory approach to collision avoidance is presented. This approach is based upon human-machine cooperation. The characteristics of humans and machines are different as explained in Chapter 2, and the supervisory approach exploits the difference to solve the collision avoidance problem more efficiently.

The operator decides on robot manipulator paths that he thinks do not cause any collisions. Then the operator instructs the manipulator to follow the paths. The collision avoidance algorithm is triggered by detection of impending collisions during the motions.

2.5.1 Introduction

The collision avoidance algorithm uses heuristics to give an alternative path to avoid impending collision. Different heuristics are used depending upon where the collision occurred in the manipulator body and the configuration of the obstacle that caused the collision. The manipulator is assumed to have two main parts for the purpose of collision avoidance: the wrist and the elbow. The wrist part includes the gripper, the wrist, and the half of the forearm that is attached to the wrist. The elbow part includes the other half of the forearm and the upper arm. They are illustrated in Fig. 2-11.

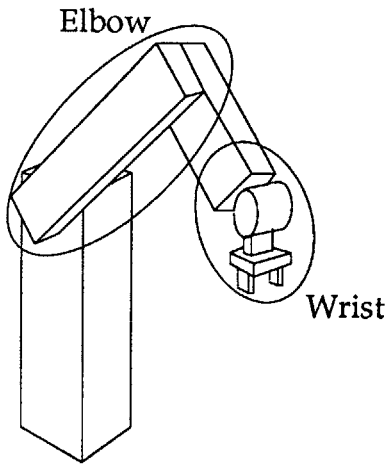


Figure 2-11: Manipulator Parts for Collision Avoidance.

2.5.2 Heuristics for Wrist

If the impending collision occurs in the manipulator wrist, it has four options for the direction of the next motion. The manipulator may simply go over or underneath the obstacle. The other two options are going to the right of the obstacle, and going to the left of the obstacle. In most cases, objects are located upon some other objects and the option to go underneath the obstacle is rarely available.

These four directions form a branch for a search tree if searching is employed in the algorithm. However, the searching process in this domain is not time-efficient and additional time spent in searching would not bring significant improvement of the system capability. It turns out that selecting only one moving direction based upon heuristics gives satisfactory results without expensive time wasted for searching.

The heuristics used for selecting the moving direction are based upon an estimate of the distance of a detour motion in each direction. Estimating the distance of a detour

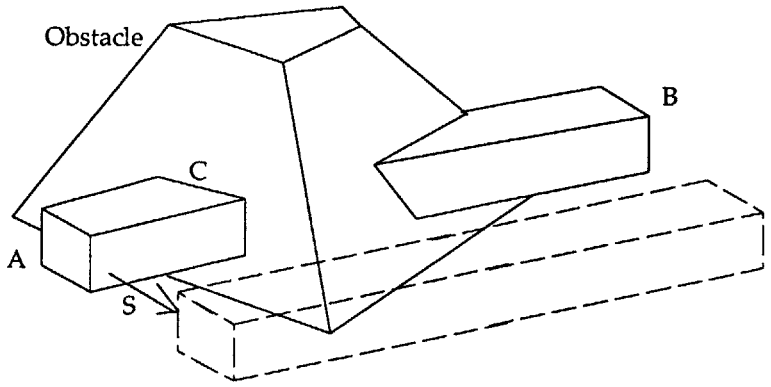


Figure 2-12: A Detour Motion.

motion can be easily described using an example shown in Fig. 2-12. Points A and B may be intermediate locations that the operator specifies, the initial location, or the goal location. An impending collision is detected at point C during the straight motion from point A to point B. As explained above, there are three options considered for the moving direction of the manipulator. Upon collision detection, an imaginary column with square cross-section, whose size is compatible to the size of the wrist and the gripper combined and with length of $\overline{AB}$, is created and located along line $\overline{AB}$. Note that this imaginary column is in collision with the obstacle. The column is moved little by little perpendicular to line $\overline{AB}$ to the right of the obstacle until it does not collide with the obstacle. The distance of the translation movement perpendicular to line $\overline{AB}$, denoted as S in Fig. 2-12, is used as an *estimate* of a detour distance *to the right* of the obstacle. Similarly, estimates of detour distances in the other directions can be determined. Then, the manipulator moves in the direction that gives the shortest estimate of a detour motion. During the motion, the manipulator hand continually reaches out and touches the obstacle to follow the contour of the obstacle. This is similar to the way in which a blind person tries to avoid an obstacle located in front of him/her by using a stick or a hand. This would

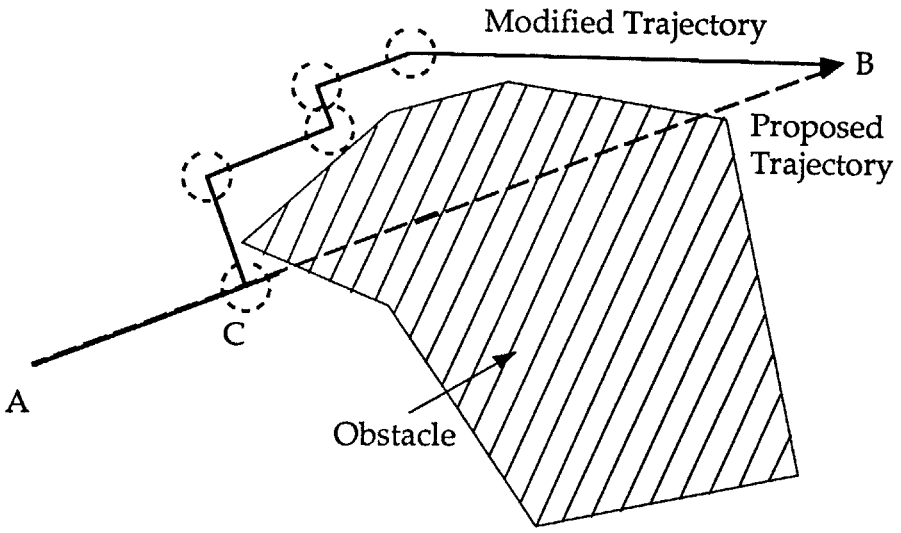


Figure 2-13: Motion with Heuristics of Reaching-Out-and-Touching.

make it possible to maneuver in the region with tight free space.

A two-dimensional example of this kind of motion is illustrated in Fig. 2-13. Again impending collision is detected at point C during a straight motion from point A to point B. Moving to the left of the obstacle gives the shortest estimate of a detour and the manipulator tries to move in that direction. First the manipulator moves a small distance to the left of the obstacle, and then it tries to move straight toward the intermediate goal location, point B. If another impending collision is detected for the motion to point B, it moves a small distance to the left of the obstacle again. This procedure is repeated until straight motion toward to point B is possible, which is point D. Then it moves straight toward point B until another impending collision is detected at point E. Using the procedure, the manipulator reaches point F, from which straight motion to point B can be completed without further impending collision.

The motion strategy can be illustrated in a flow-chart. Figure 2-14 shows a flow-chart that describes a motion strategy for the manipulator motion to the left of the obstacle

when an impending collision is detected. The complexity of the detour motion, i.e. the complexity of the geometry of the obstacle, is expressed by an index, ζ. For convex obstacles, ζ remains zero during any detour motion.

Figure 2-15 shows a two-dimensional obstacle part of which is a maze. Assume that the intended path is from point A to point B. An impending collision is detected at point C and the collision avoidance algorithm is triggered. Assume that it selected moving to the left of the obstacle as the direction of detour. Thus the manipulator initially goes to the left until it reaches point D. Then the manipulator cannot move forward or move to the left; therefore, it moves backward until it reaches point E. Applying the rule shown in Fig. 2-14 in a similar fashion, the manipulator reaches point F, where it starts moving forward again. From point F and point G, the value of ζ keeps changing from zero to one. From point G, where the manipulator starts moving to the left, ζ is one. Eventually, the manipulator moves through points H, I, J, K, L, M, N, O, and P. Note that ζ becomes zero again when the manipulator reaches point M, and remains zero afterwards. The application of the rule illustrated in the flow chart ends at point P, where it tries to move straight to point B, an intermediate point, or a target. Thus another impending collision is detected at point P and the same rule is applied for avoiding the collision. However, the resulting detour this time is simply a straight motion to point Q, from which the manipulator moves straight to point B without further collision, thus completing detour motions from point C to point B. The example shown in Fig. 2-15 is an extreme case where geometric complexity is extraordinarily high. The geometrically complex object is selected in this example to demonstrate simply how the rules are triggered and operate upon an impending collision. Of course, in most cases, obstacles would have much

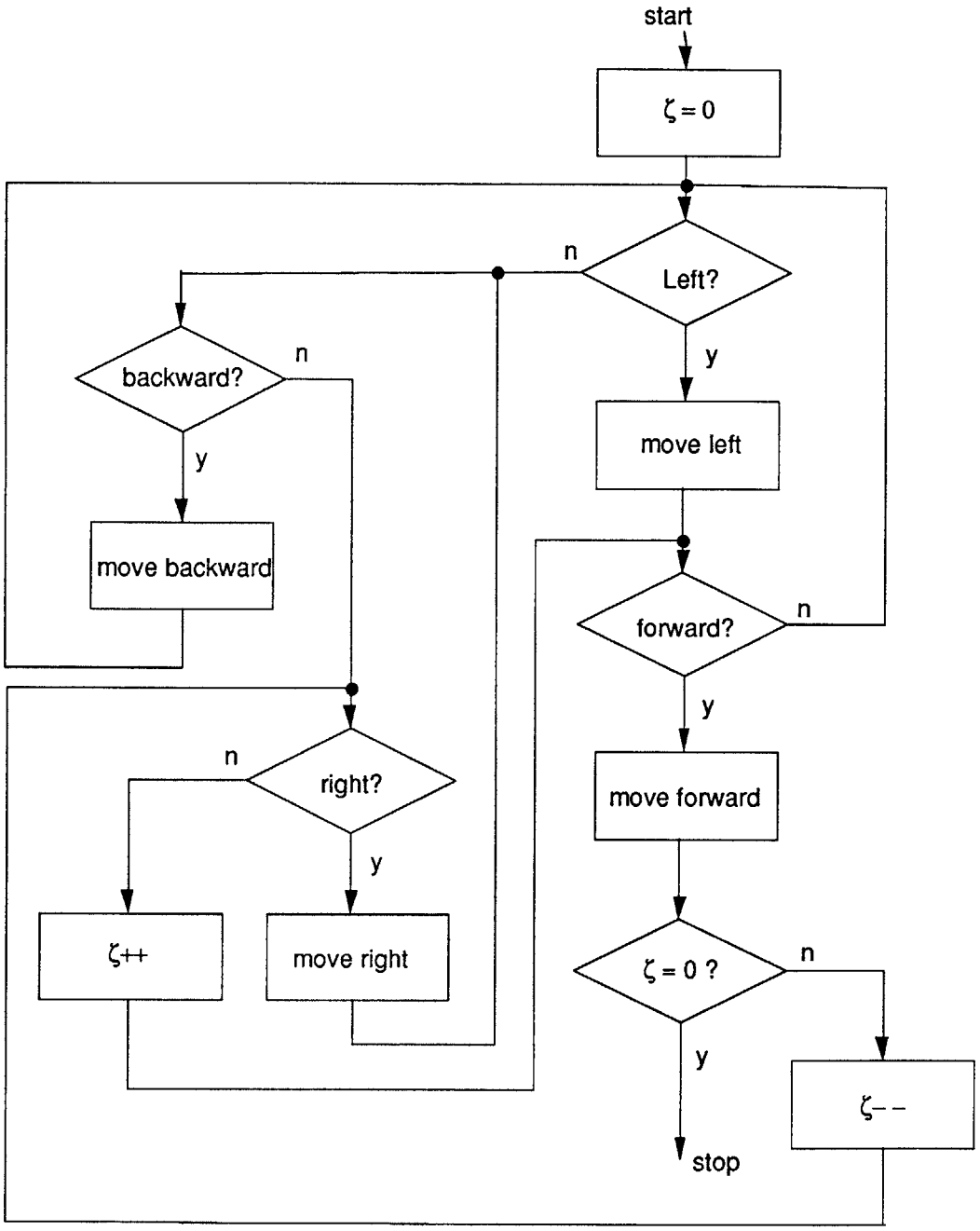


Figure 2-14: The Potential Approach.

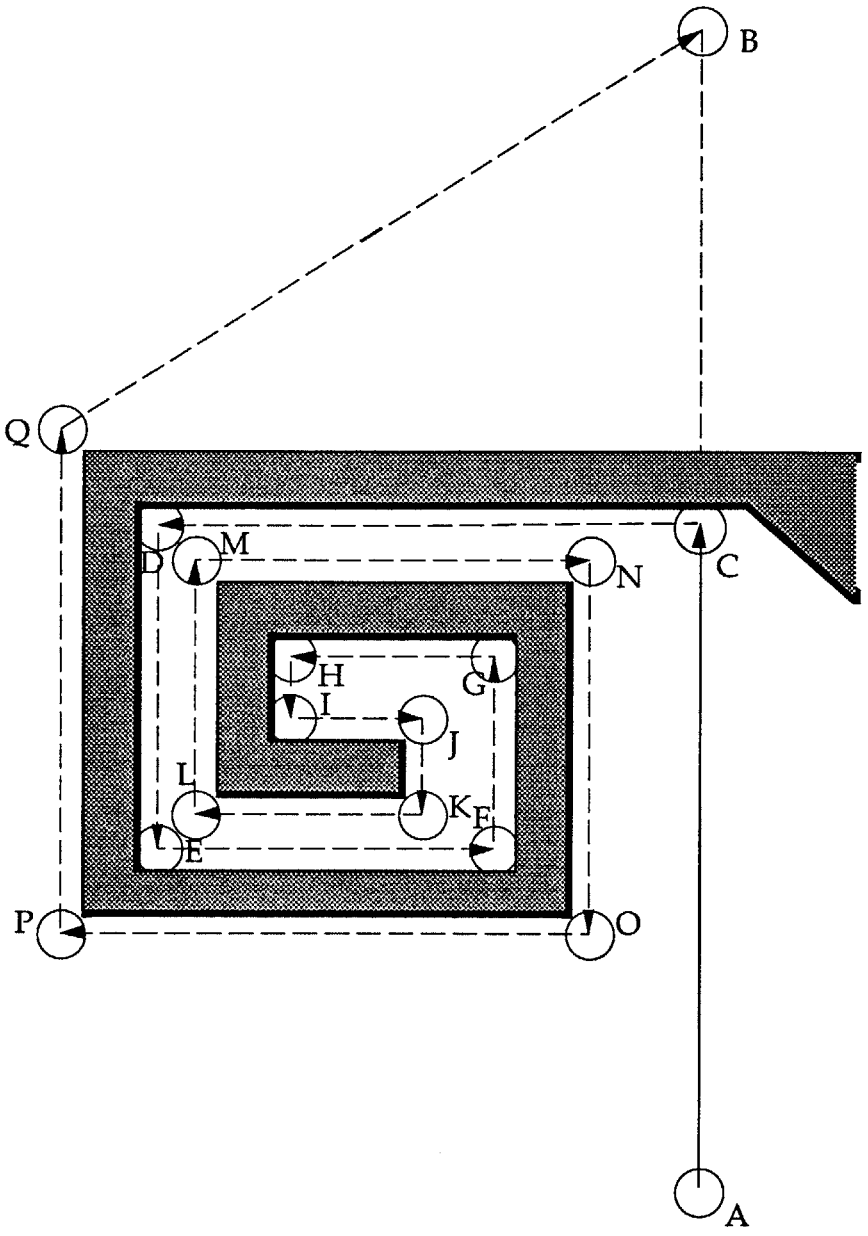


Figure 2-15: A Maze-shaped Obstacle.

simpler geometric configurations than the one shown in Fig. 2-15.

The rule governing the manipulator behavior when the detour direction is to the right of the obstacle, has the same structure as the rule shown in Fig. 2-14. The only differences are that the word "left" is replaced by the word "right" and that the word "right" is replaced by the word "left." The rule governing detour motion to the right of the obstacle is, in a sense, rotated 180 degrees about the original straight path from the rule governing detour motion to the left of the obstacle.

Also, the rule governing the manipulator behavior when the detour direction is over the obstacle, has the same structure as the rule shown in Fig. 2-14. The only differences are that the word "left" is replaced by the word "up" and that the word "right" is replaced by the word "down." Similarly, the rule governing detour motion over the obstacle is rotated 90 degrees about the original straight path from the rule governing detour motion to the left of the obstacle.

The motion resulting from the reaching-out-and-touch [ROT] strategy is different from that resulting from so called potential field approach. The latter blindly follows the path that minimizes the value of a potential function without considering the configuration of the obstacle. Figure 1-1 shows the difference between the potential approach and the ROT strategy. In the particular case shown in Fig. 1-1 the path from the ROT strategy is much more reasonable than that from the potential approach.

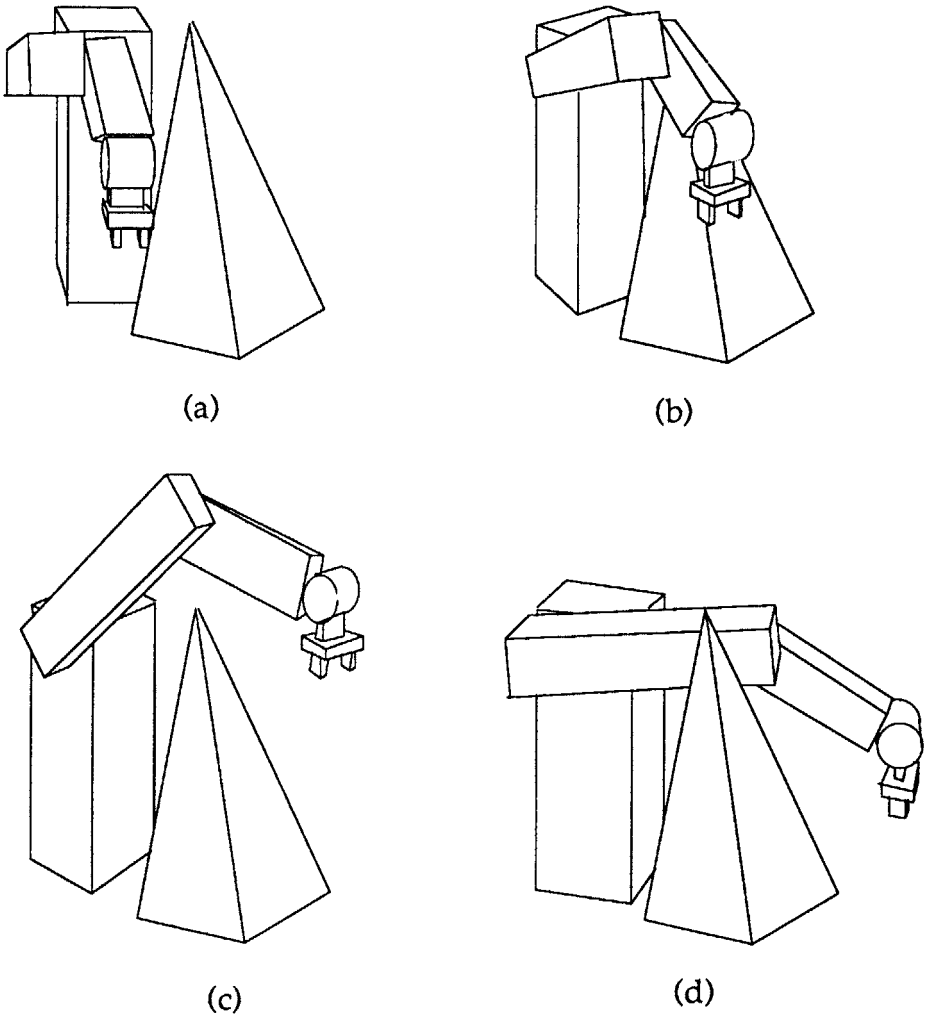


Figure 2-16: Avoidance of an Impending Collision at the Elbow.

2.5.3 Heuristics for Elbow

If the elbow part is involved in an impending collision, the manipulator simply tries to rotate its shoulder around and make a circular motion of the elbow. This is again similar to the way in which a person avoids an elbow collision with an object during a motion of an arm. If the elbow is to collide with an object, a person would simply lift the elbow while moving in the direction of the original motion. If most obstacles in the environment are located upon other object(s), not hanging in the sky, a circular motion which results in lowering the elbow would not be considered. Figrure 2-16 shows how the manipulator moves to avoid collisions between its elbow and the obstacle.

2.5.4 Other Heuristics

The heuristics explained so far can solve most impending collision avoidance problems. However, there are some cases where the manipulator cannot move and gets stuck near an obstacle during a detour motion to avoid impending collision. This phenomenon happens when the location of the obstacle involved in the impending collision is near the boundary of the manipulator work space. During a detour motion, the manipulator pushes itself against the boundary of the workspace[4], resulting in a workspace limit. The manipulator cannot move further from that location, and thus gets stuck. At this workspace limit, the forearm is fully stretched (with no angle limits). In order to solve this problem, different heuristics are used for the configuration where the forearm is fully

[4]A volume of space where solutions of the inverse kinematics exist. In plain English, a volume of space which the end-effector of the manipulator can reach.

stretched. Since motions that require elbow motion are not permissible due to the loss of freedom at the elbow, it is natural to use the shoulder to avoid the impending collision. This suggests that when the elbow is fully stretched heuristics for elbow collisions are triggered to lift the elbow regardless of the location of the impending collision in the manipulator body. Figure 2-17 shows how this heuristic works. When an impending collision occurs, the manipulator tries to move to the right of the obstacle, which is the direction that gives the smallest estimate of detour distance. At one point during the motion to the right of the obstacle(a), the manipulator stretches the elbow fully and cannot continue to move to the right of the obstacle. Therefore, it lifts the elbow and the wrist (b and c) to avoid an impending collision, which occurs at the wrist and reaches to the target location (d).

This scheme is basically a three-dimensional but can be extended to a six dimensional one by using more heuristics about wrist motions. Figure 2-18 shows how the manipulator moves while avoiding collisions with polyhedral objects in the environment.

2.5.5 Similarity of Motions

The collision avoiding motions for the manipulator when the heuristic scheme is used are similar to those generated by the algorithm developed by Cheung and Lumelsky (1990), even though the latter uses a sensitive "skin" (proximity sensors) to detect contacts with obstacles and solves the collision avoidance problem in the configuration space rather than in the world coordinate.

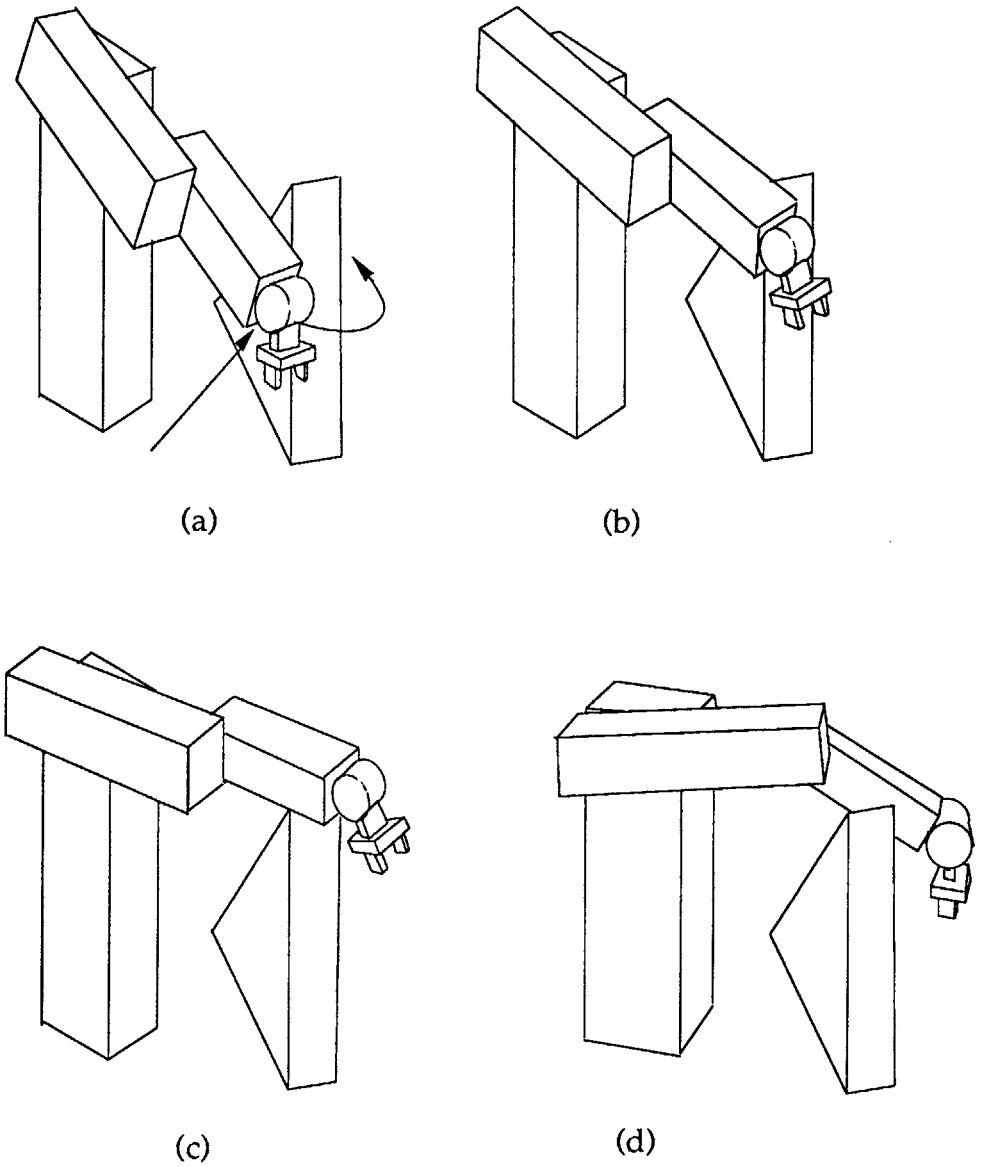


Figure 2-17: A Collision Avoidance Motion When the Elbow Is Fully Stretched.

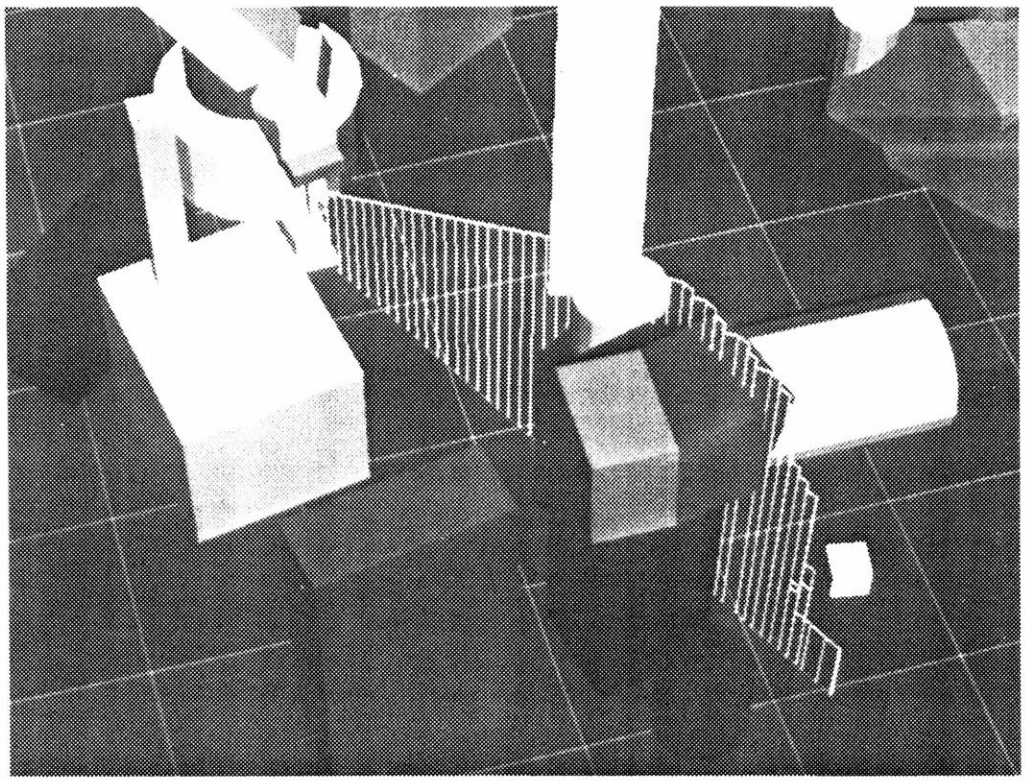


Figure 2-18: Collision Avooidance Motions (Graphic Simulation on SGI's Iris).

2.6 Supervisory Control System with Computer Graphics

2.6.1 Advantages of Computer Graphics

In general, the human operator performs teleoperation based upon visual information on the environment which is obtained by video cameras. If communication of the video signal suffers problems such as limited bandwidth of signal, noisy video signal, a time delay, and malfunction of video equipment, etc., the operation is significantly affected and deteriorated. If a geometric model of the environment is available, the operation can be protected from influence by those factors. For example, when the operator controls a remotely located robot, there may be a time delay due to the remoteness and in this case the operator typically takes a "move-and-wait" approach. If computer graphic environmental model is available, the operator can get visual information on the environment and a prediction of the result of his commands from computer simulation, and does not have to take a "move-and-wait" approach. Availability of visual information about the environment and predicted results of the action taken help the operator make the next plan more quickly.

Information on the environment can be manipulated and displayed in any way the operator wants. The operator can change the viewing point arbitrarily in order to watch regions that he cannot observe through the cameras. This would help the operator operate the robot in such hidden areas if there are no hidden objects.

2.6.2 Extra information display to help planning

Information Display to Enhance Human Perception

Computer rendering is the process of presenting three-dimensional geometric information on a two-dimensional screen. There are two types of visual displays: monoscopic and stereoscopic. The stereoscopic display provides two slightly different perspective views for the human operator's right and left eyes, enabling the human operator to perceive depth by providing a distinct binocular depth cue called stereo disparity. A few studies indicated that stereo performance in general is superior to mono under most conditions tested. Three dimensional tracking tests show that monoscopic displays allow equivalent performance when they were defined with appropriate perspective parameters and provided with adequate visual enhancement depth cues such as reference lines. Similar results have been obtained through experiments for pick-and-place operations (Kim et al., 1987).

Grid and Projection Line

In the graphic simulation, a grid for horizontal floor and a reference line for the manipulator tip were used. The system was also capable to draw vertical reference lines from each vertex of an object. This gave the operator the cue of the object's vertical location from the floor or from another object.

lines connecting intermediate points of a path

During the interactive specification of a manipulator path, the line connecting the previous intermediate point and the current point being specified was displayed. This enabled the operator to visualize a straight path for the manipulator tip and to avoid specifying an intermediate point that would force the manipulator to collide with obstacles at its wrist. This line was shown interactively as the operator changed the position of an intermediate point by dragging the mouse as explained in Section 2.3. If the intermediate point that was specified by the operator was the first one, a line from the current manipulator tip to the point was drawn interactively.

a workspace sphere

During the preliminary test of the system, many people experienced few minor problems which were overcome quickly. One problem that some of them had was that they could not predict whether a given straight path would force the manipulator to go out of the workspace and thus resulting in a workspace limit of the manipulator at the boundary of the workspace Due to the special shape of the outer boundaries of the workspace, the operators did not specify paths that would force the manipulator out of the workspace (The system warned if the operator specified an intermediate point out of the workspace.) Such a workspace limit that annoyed the operators occurred while the manipulator was moving from one intermediate point to another point that was within the workspace when the forearm of the manipulator was folded to align with the upper arm. Figure 2-19 shows a situation that will result in such a workspace limit of the manipulator. Note

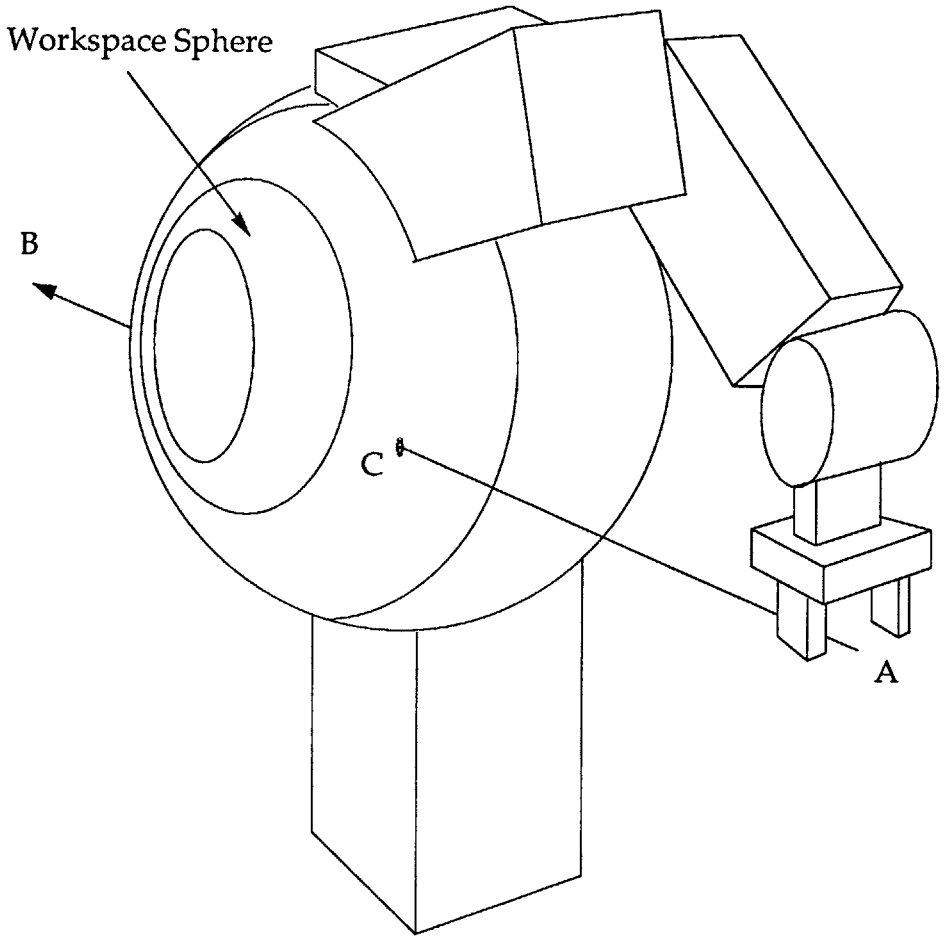


Figure 2-19: Whether a Part of a Path May Be Outside the Workspace Can Be Seen Easily.

that the end-effector of the manipulator, point A, is within the workspace and the next intermediate point, point B, to which the end-effector of the manipulator is about to go, is also within the workspace. However, in order to move straight from point A to point B, the end-effector has to pass through a region that is out of the workspace. The manipulator tip is at point C in the figure just before the end-effector goes out of the workspace.

In order to help the operator avoid specifying an intermediate point that would result in a workspace limit such as point B in Fig. 2-19, graphic information regarding the workspace limits is displayed.

What is displayed to the operator during path planning is the volume that has no intersection with the workspace and that is encircled by the workspace. The shape of the volume is always a sphere, but the location of the volume is dependent upon the orientation of the end-effector of the manipulator. Let's call this sphere a *workspace sphere*. Figure 2-19 shows a workspace sphere for a particular orientation of the end-effector. Note that point C in the figure appears on the sphere. If the end-effector is on the surface of a workspace sphere, a workspace limit occurs.

When the operator specifies a path for the end-effector, the operator sees interactively a straight line connecting the previous intermediate point (or the current position of the end-effector) and the intermediate point that he is currently specifying. This enables him to see if the line is penetrating through the workspace sphere. Thus, he can specify an intermediate point such that a workspace limit does not occur during the straight motion from the previous intermediate point.

The idea of the workspace sphere can be expanded and used as a virtual object. The workspace sphere is regarded as an obstacle in manipulator motions. If an impending collision with the workspace sphere is detected during a manipulator motion, a collision avoidance algorithm can be triggered and engaged to prevent collisions, which amounts going out of the workspace, resulting in a workspace limit. The manipulator collision checking with the workspace sphere should be done with the wrist parts.

Chapter 3

Uncertainty

3.1 Virtual Obstacles

In many robot operations, there exist uncertainties. These uncertainties are there because they cannot be removed or because there are costs associated with removing them. There are many types of uncertainties such as uncertainty on object attitude, uncertainty of object existence, and so on. The type of the uncertainty covered in this chapter is the uncertainty of existence. Section 3.1.1 explains the uncertainty and need for a tool to deal with it. In Section 3.1.2, the definition of virtual obstacles is given. In Section 3.1.3, it is explained how to generate virtual obstacles and how to update them as the viewing point changes.

3.1.1 Uncertainty of Existence

The fact that a region has uncertainty of existence means that the operator has no information on the region. If it is put in the context of collision detection and avoidance, this means that it cannot be assured that there is no obstacle in the region because observations in the region have never been performed. A typical situation with this kind of uncertainty is when objects in the environment block the view of the region behind, whose model does not exist, so that the operator has no idea whether there are some other obstacles in that region. The operator may not want to have the manipulator move into the uncertainty region, since that might result in collisions.

The operator can reduce the uncertain regions by changing the camera locations and observing new scenery. Whenever the operator changes the camera location, he/she can see parts of previously unobserved regions and the uncertain regions shrink.

However, in most cases, the cameras cannot be moved around as freely as the observer wishes. So there are limits on the amount of uncertainty the observer can reduce by using the cameras. Moreover, there are certainly costs associated with moving the cameras. The observer would not try to reduce all the uncertainty. Therefore, there are always uncertainties that have to be dealt with.

3.1.2 Definition of Virtual Obstacles

Virtual obstacles are introduced to represent regions with such uncertainty. They are used as obstacles in the collision detection and avoidance, keeping the manipulator

from moving into a region with uncertainty of existence.

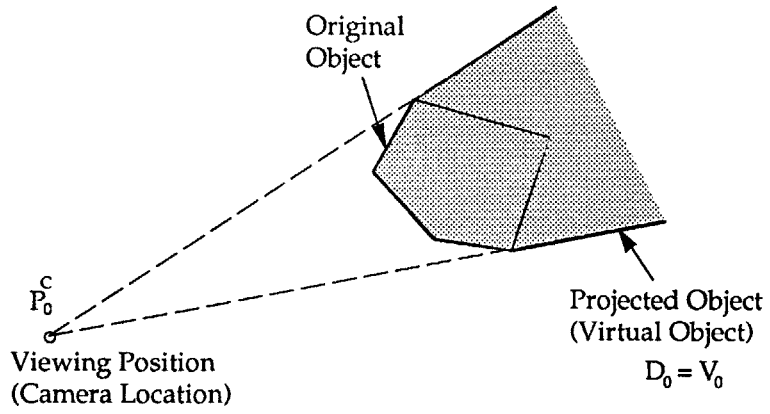
Virtual objects are imaginary objects that occupy spatial volumes occupied by objects, and their neighboring spatial volumes that haven't been observed. The observer in this case can be a vision system or the operator viewing through a camera. For convenience, it is assumed from now on that the observer is the operator.

Objects in the environment are obstacles when the robot manipulator moves around to avoid collisions with them. Similarly, virtual objects are virtual obstacles when they are referred in the context of collision detection and avoidance. Meanings of the virtual object and the virtual obstacle are the same except for context in which they are used. Therefore, they may be used interchangeably in the rest of the thesis.

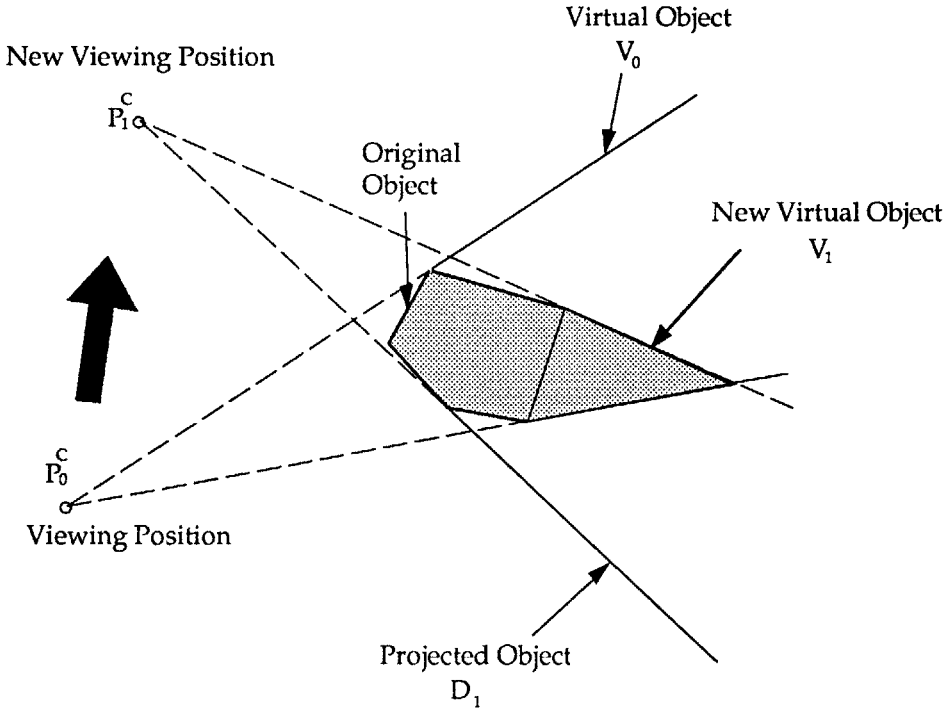
3.1.3 Generation of Virtual Obstacles

In general, every object in the environment has a corresponding virtual object. The virtual object at any moment occupies the spatial volume that has not been observed by the operator because it is occluded by the original object. At the start of a robot operation, the size of the virtual object is large, but gets smaller as the camera moves around, and more previously unobserved regions are observed.

To show the principles behind virtual objects and the relation between objects and the cameras, consider an example of a two-dimensional object . For simplicity, let's assume that there is only one object in the world and that only one camera is used for visual feedback as shown in Fig. 3-1 a).



a)



b)

Figure 3-1: Virtual Objects and the Camera Position.

If a light source is located at the camera location, P_0^C, the object creates its own shadow in the opposite direction to the light source. The shadow would be infinite in that direction. A *projected object* is defined as an imaginary object that occupies the region of the original object and the region of its shadow. Note that the shape of a projected object depends not only upon the original object but also upon the location of the camera. In Fig. 3-1 a), projected object D_0 is bounded by the original object facing the camera location, P_0^C, and two straight infinite lines. Mathematically, projected objects have no finite bounds in size. In the beginning of robot operation, the virtual object, V_0, is the projected object, D_0, itself. It is simply because the region occupied by projected object D_0 cannot be observed and has never been observed by the operator. Thus,

$$V_0 = D_0. \tag{3.1}$$

Now, let's assume that the camera is moved to a new location, P_1^C, as shown in Fig. 3-1 b). Then, a projected object for the new location can be created. Let's call it D_1. Object D_1 represents the region that cannot be observed from the camera located at P_1^C because the original object blocks the view of the camera. Then, the virtual object at that moment is the region that cannot be observed from point P_1^C *and* P_1^C. Note that those two locations are the only observation locations. Thus, the virtual object V_1 represents the common area of D_0 and D_1.

$$V_1 = D_0 \cap D_1. \tag{3.2}$$

If the camera is moved to another location, P_2^C, the virtual obstacle, V_2, would be

the one that occupies the region that cannot be observed from this new location and the two previous locations.

$$V_2 = D_0 \cap D_1 \cap D_2 = V_1 \cap D_2. \tag{3.3}$$

This can be generalized for any new camera location after a number of changes in camera location. The virtual obstacle for the i-th view point is

$$V_i = V_{i-1} \cap D_i. \tag{3.4}$$

V_{i-1} denotes the virtual obstacle in the previous camera location, and D_i denotes a projected object for the current camera location. Thus, a virtual obstacle for a particular camera location can be obtained by performing a logical AND operation on two objects.

We said that a projected object, in theory, expands infinitely in the direction of projection and has an infinite volume. However, if the camera moves through a sufficiently large distance a few times perpendicular to the axis of viewing, the virtual object would have a finite volume. In the actual implementation of the virtual object, it is assumed that the projection distance is finite and each projected object has a finite volume. By doing so, a mathematical representation for a projected object remains consistent with other polyhedra and complies with Euler's law.[1] See Fig. 3-2.

[1]Euler's law describes the relationship between the number of faces (F), edges (E), and vertices (V) of a polyhedron: $F - E + V = 2$.

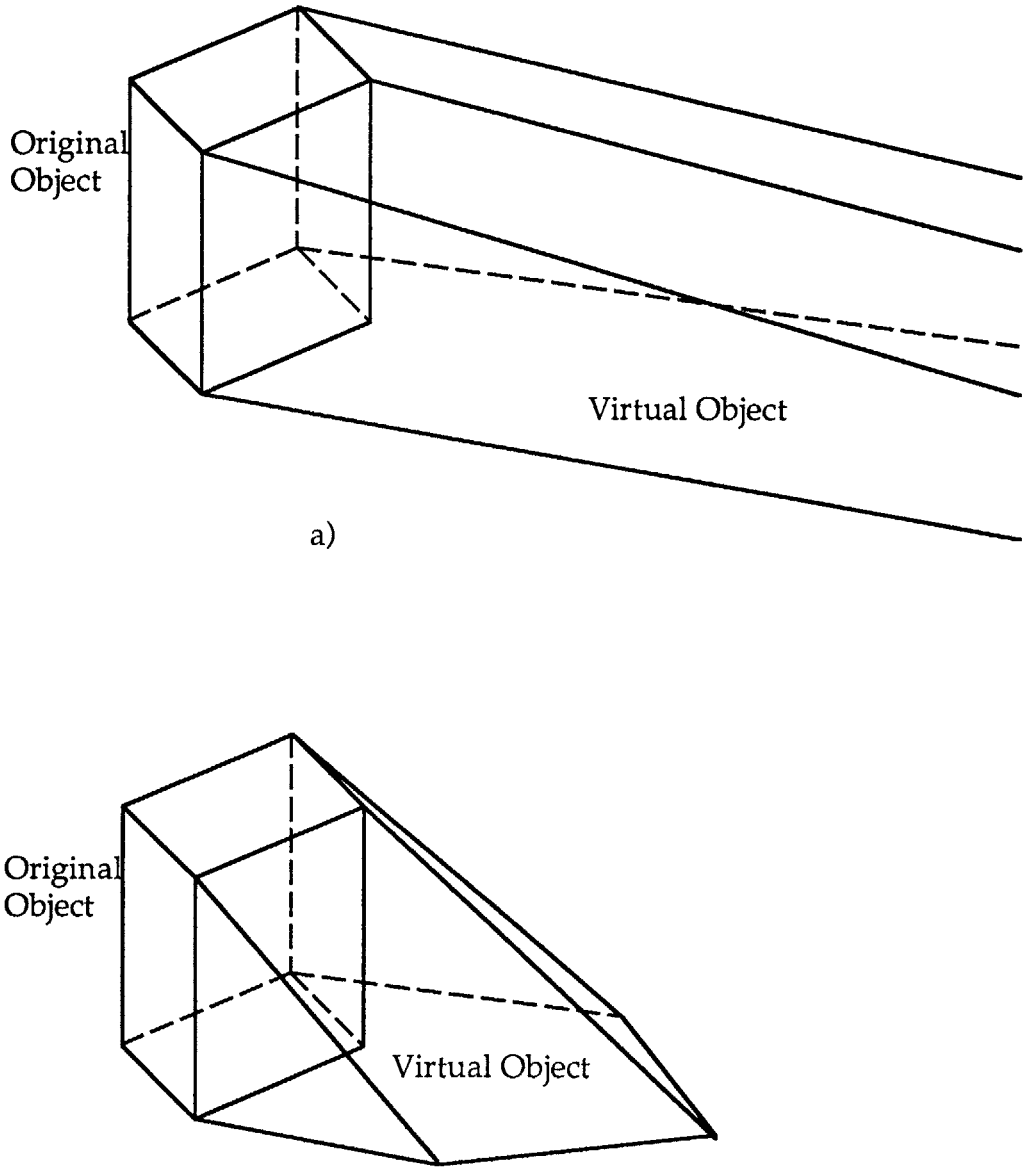


Figure 3-2: A Virtual Obstacle Is Updated As the Camera Moves Up.

3.1.4 Camera Motion and Virtual Obstacles

As shown in Section 3.1.3, new virtual obstacles have to be updated based upon the previous virtual obstacles and the camera location when the camera location changes. However, not all camera motions require updating virtual obstacles. Panning, tilting, and rolling the camera wouldn't require the updates. In general, moving the camera closer to the object does not change the virtual obstacle because the projected object can contain the previous virtual obstacle inside its volume. However, moving the camera away from the object requires the update, because such a motion reduces the viewing angle for the object, shrinking the projected object.

3.2 Introduction of Objects by "Flying-and-Matching"

There have been many works in locating automatically objects with their geometric models (Roberts, 1965; Faugeras and Hebert, 1983; Grimson and Lozano-Pérez, 1984; Haralick et al., 1989). They have to deal with signal noise, which is difficult. Moreover, locating an obscured object turned out to be more difficult (Fisher, 1983).

This section proposes and explains a new way of introducing objects into the world model, called "flying-and-matching," and introduces two different strategies to use the method.

There are a few assumptions to make before going into details. First, it is assumed that the geometric model of the object that the operator is to introduce into the world model is available. Because this method is based upon matching between the view of

an actual object and the display of its model, this assumption is required. Second, it is assumed that the superposition of the actual camera view and graphic rendering are available.

The procedure to use this method is simple. When the operator wants to introduce an object into the world model, he makes the system display the geometric model of the object on the screen, which is superposed with the actual camera view. Then he uses a joystick to "fly" the display of the object model until it matches the actual camera view of the object as closely as he can. While controlling the position and orientation of the model object with a six d.o.f. joystick such as the one used in this thesis, the operator can control the joystick as if he grasps and moves the actual object.

This is a supervisory method to introduce objects into the world model. The operator is in charge of the whole process. One advantage is that it does not need any sophisticated device such as a laser gun. Only needed are the capability of superposition of a camera view and a graphic rendering, and a joystick (preferably a six-dimensional one). Another advantage is that this method can be used for objects that do not have any distinct vertices or edges. Some others require that objects have distinct vertices.

The operator does not have any cue in the viewing direction except the size of the object. The size of the object changes depending upon its location: an object would appear larger in a closer distance. It is expected that the error in the z-direction of the viewing coordinate system is larger than those in other directions. However, such errors would be small if more than one camera at different locations are used for introduction of objects.

3.2.1 Precise Introduction vs. Crude Introduction

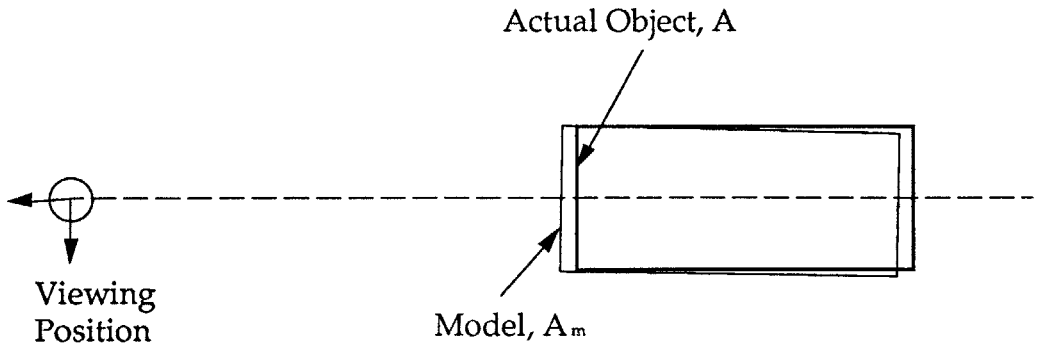
There are two different strategies as to how to use this method when introducing objects into the world model: a precise introduction and a crude introduction.

3.2.2 Precise Introduction

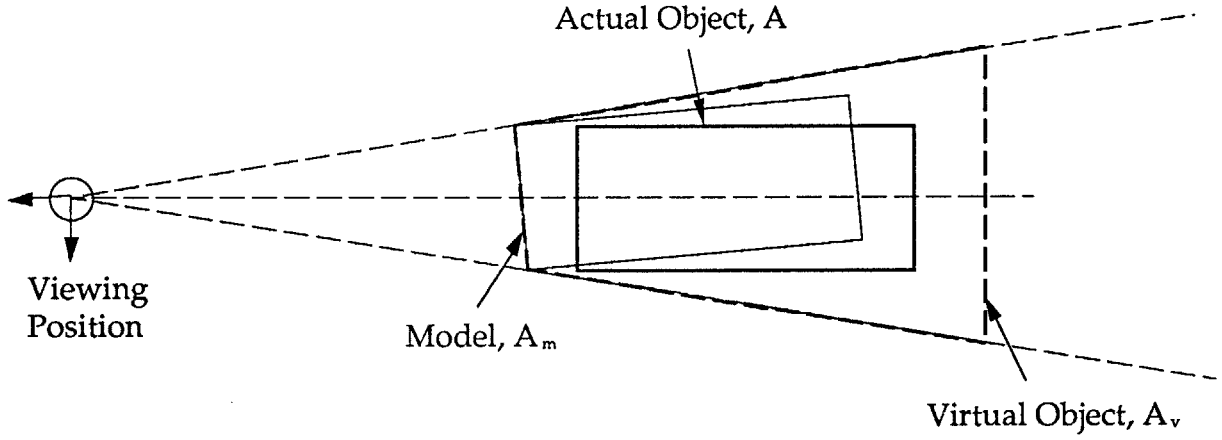
In a precise introduction, the operator tries to have the best match he can between the view of the actual object and the display of its model. When the operator has more than one camera, he can introduce objects very precisely. However, it is difficult to introduce objects precisely with a single camera because the operator has limited depth cues.

3.2.3 Crude Introduction

In a crude introduction, the operator does not have to have such a precise match as in precise introduction but uses the concept of virtual objects to model the errors of the position and orientation of the object model. In order to use this strategy, the operator should make sure that the model is a bit closer to the viewing point than the actual object is so that the virtual object encompasses the actual object. Figure 3-3 shows two different strategies of introducing an object, A, with a single camera view in a two-dimensional world.



a)



b)

Figure 3-3: Two Different Strategies To Introduce Objects Into the World Model.

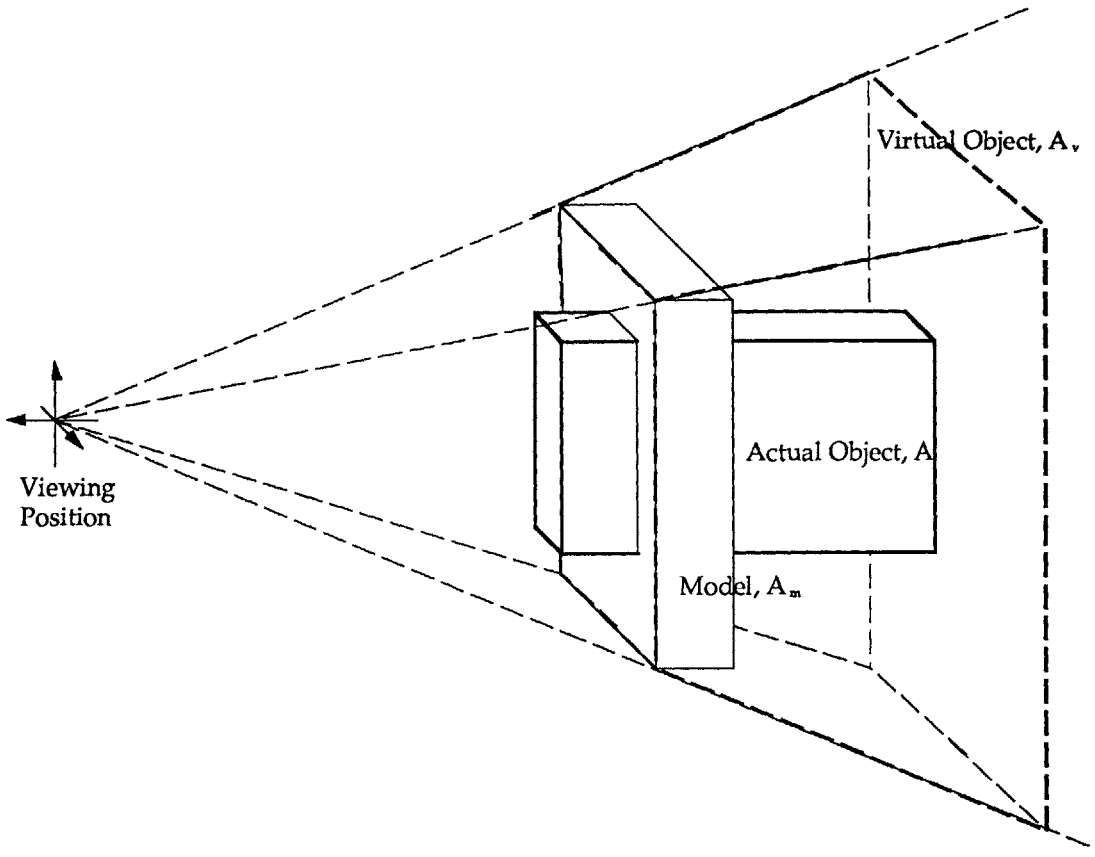


Figure 3-4: Important Correct Angular Alignment.

3.2.4 Importance of Accurate Orientational Matching

In order to have a proper introduction, it is crucial to have a close match in orientation of the actual object and the model, especially when only a single camera is available for the task. When the error in the orientation of the object and the model is large, the virtual object may not encompass the corresponding object even though it looks as if it does. Figure 3-4 shows an extreme example where there is a significant error in orientation of the actual object, A, and its model object, A_m. For this configuration, object A is not encompassed by the virtual object A_v that is based upon A_m. If the model object (a solid model) is rendered on the screen after the camera view is displayed on the screen, object A would appear encompassed by virtual object A_v to the operator. In the two-dimensional screen, object A would be totally occluded by object A_m, or object A_v. If more than one camera are available, a large orientation error, however, could be tolerated because a wrong configuration as shown in Fig. 3-4 could be detected by other cameras.

3.2.5 Trade-Off

The advantage of a crude introduction is in speed. It takes a shorter time to introduce objects crudely into the world model than it does precisely because the operator does not have to spend time in getting a precise match. The disadvantage of crude introductions is that it shrinks the free space for the manipulator to move since virtual objects are larger than the corresponding actual objects. Thus in order to choose the strategy of introduction a trade-off is made between the amount of free space and the time spent

for introducing objects.

When the task consists of two subtasks, i.e. introducing unmodeled objects into the world model and operating the manipulator, the total time spent for the task is the combination of the introduction time and the manipulator operation time:

$$T_{\text{task}} = T_{\text{introduction}} + T_{\text{operation}} \ . \tag{3.5}$$

And,

$$T_{\text{introduction}} = N \cdot \bar{T}_{\text{int}} \ , \tag{3.6}$$

where N is the number of unmodeled objects to be introduced into the world model and $\bar{T}_{\text{int}}$ is the average time to introduce one unmodeled object into the world model.

The time *earned* by using a crude introduction, $\Delta T_{\text{introduction}}$, is

$$\Delta T_{\text{introduction}} = N \cdot \left(\bar{T}_{\text{int, precise}} - \bar{T}_{\text{int, crude}} \right) \ . \tag{3.7}$$

Operation time *lost* for having a crude world model rather than a precise world model is a complicated function of the environment, the degrees of crudeness of the model and etc.

$$\Delta T_{\text{operation}} = f(\text{environment, crudeness, } \ldots) \ , \tag{3.8}$$

where f is a unknown complicated function. Thus the total time *saved* by using the crude modeling is

$$\Delta T_{\text{operation}} = N \cdot \left(\bar{T}_{\text{int, precise}} - \bar{T}_{\text{int, crude}} \right) - f(\text{environment, crudeness, } \ldots) \ . \tag{3.9}$$

The task time saved by using a crude world model depends upon N, the number of objects that need to be introduced, $\bar{T}_{\text{int, precise}} - \bar{T}_{\text{int, crude}}$, the difference of average introduction between two methods, and f. If there is no difference between $\bar{T}_{\text{int, precise}}$ and $\bar{T}_{\text{int, crude}}$, which is an extreme case, it is better to use a precise world model. If there is a large difference between $\bar{T}_{\text{int, precise}}$ and $\bar{T}_{\text{int, crude}}$, the crude model would have an edge over the precise modeling.

Equation 3.9 suggests there is a threshold $N_{\text{threshold}}$ which makes it indifferent to choosing one method over the other.

$$N_{\text{threshold}} = \frac{f(\text{environment, crudeness, } \ldots)}{\bar{T}_{\text{int, precise}} - \bar{T}_{\text{int, crude}}} \tag{3.10}$$

Two extreme cases can be considered here. When there is no difference in introduction time between the crude and precise modeling, $N_{\text{threshold}}$ becomes ∞; when there is an infinite difference in introduction time, $N_{\text{threshold}}$ becomes 0. It is expected that $N_{\text{threshold}}$, in most cases, is expected to be between the two extremes.

Chapter 4

Experiments

4.1 Experiment 1: Stacking Operation

4.1.1 Objective

The objective of this experiment is to evaluate the feasibility of operating the supervisory system where some of objects are not modeled in the world model, and to determine operating patterns by the subjects and important design factors in order to improve the system design.

4.1.2 Setup and Procedure

Task

Subjects were seated in front of the computer display. On the display, a Puma-style manipulator and the environment filled with objects were shown with a command menu. The subjects were asked to use the command menu and the graphic display to control the manipulator to pick and stack five small cubic blocks that were scattered in the environment, while avoiding collisions with other objects. These five blocks had distinctive colors for easy recognition by the subjects. The sequence of stacking the blocks was not fixed and was up to each subject.

There were three viewing options that the subject could select by clicking a mouse button. The first option is a camera view[1] which was assumed to show all the objects as a real camera does. The second option was a graphic rendering of the world model in the computer. In the rendering, the subject could change his viewing conditions anyway he wanted—he could zoom in and out, turn the environment, and pan-and-tilt. The third option was the *simulation* of the actual camera view, which was the graphic rendering of the world model viewed from the camera position. This view showed only objects that were *registered* in the world model. If there was an object(obstacle) unregistered in the world model, the subject(operator) could detect the discrepancy between the world model and the actual world by comparing the camera view with the camera simulation. In the experiment, it was assumed that only one camera was available and

[1]The camera view was actually a graphic simulation; however, the subjects were told that he/she should think that it was a view from the actual camera.

it was immobile. Also assumed was that the display of the workspace sphere was not available to the subjects.

Environment

Each subject was asked to do the same task of stacking five blocks for six different environment sets. Each environment set had a number of objects from three different categories. The objects in the first category(Category I) were the small five cubic objects that the manipulator would have to move to make a stack of blocks at the target location. Their positions and orientations as well as their geometries were known and stored in the world model. The objects in the second category(Category II) were the objects that were simply obstacles in the tasks. Their positions and orientations as well as their geometries were also known and stored in the world model. The collision detection algorithm could prevent any manipulator collisions with the objects(or obstacles) in the two categories since they were registered in the world model. The objects in the third category(Category III) were the objects whose models were not registered in the world model. This meant that the position and the orientations as well as the existence of the objects in this category were not known to the computer. The subjects were warned that there may be, in the environments, some objects in this category and that it was their responsibility to prevent manipulator collisions with these objects(obstacles).

All the six different environments had five objects from Category I. The only differences between them were the number of objects(obstacles) that belonged to Category II and Category III. Table 4.1 shows the six environments with different numbers of objects

		Number of Objects(Obstacles) in Category III		
		0	3	6
Number of Objects (Obstacles) in Category II	16	I	II	III
	23	IV	V	VI

Table 4.1: Six different environments.

from each of these two categories.

Learning

In order to make the environment with more objects more difficult than than one with fewer objects, more obstacles were simply added to the less crowded environment. The similarity of the environments had an adverse effect on the experiment. The subjects who did complete the task in one environment *learned* quickly about the other environments because of their similarity. When the sequence of experiments was fixed and the subjects[2] had to face the simpler environments before the more complicated environments, a consistent learning effect was shown in their performance. Figure 4-1 shows a pattern of learning by the subjects when each subject encountered the environments in order I, IV, V, II, III, and VI. In order to get rid of the effect of learning, a Latin-square method was used in counterbalancing subjects and experimental order of environments. The result shown in Figure 4-2 indicated that learning effect was largely removed.

[2]Three subjects participated in this part of the experiment.

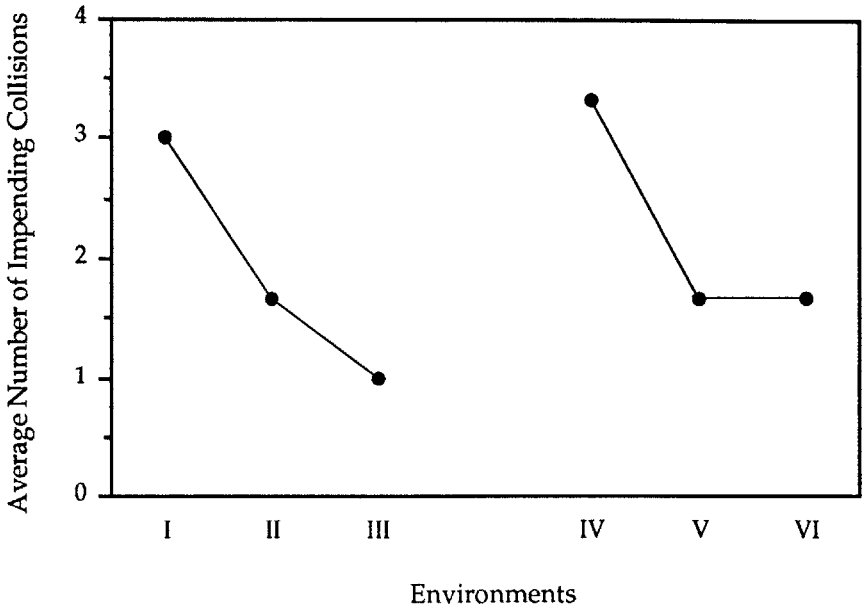


Figure 4-1: As the Subjects Went on the Experiment, the Number of Impending Collisions Got Smaller Due to the Learning Effect, Even Though the Actual Environment Got More Cluttered.

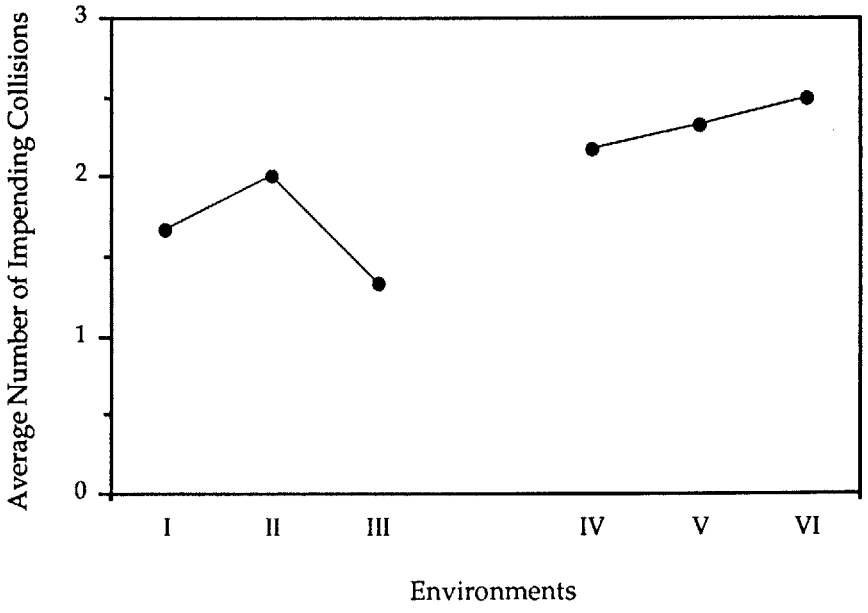


Figure 4-2: The number of Impending Collisions Got Larger(Except for Environment III) As the Actual Environment Got More Cluttered.

Subjects

A total of 6 subjects participated in this experiment. They were trained until they were quite familiar with the system. Each subject had a different background in the context of using computers and mice. One subject had no experience in using a computer at all. Another subject had much experience with computers but no experience with a mouse. The other three subjects had extensive experience of using a computer with a mouse, and one of them had some experience in using a Puma manipulator.

4.1.3 Results

The subjects used different strategies. Figure 4-3 shows the time spent by the subject to analyze the environment while the manipulator was stationary, and the time spent only for manipulator motions. For example, Subject 3 took a very long time to analyze the environment in order to avoid collisions, and took a moderate time to execute the task, meaning that total travel distance by the manipulator was moderate. Subject 2 took a relative short time to analyze the environment but took a long time to execute the task, meaning that the manipulator traveled longer than average as can be seen from Fig. 4-4. Subject 4 spent very little time to analyze the environment but managed to spend almost the same time as Subject 3 for the manipulator motions. Figure 4-5 shows that there are differences in the numbers of different types of collision for each subject. Subject 6 has no experience on computers and robot manipulators and managed to have less than 1 collision on average. But in order to do that, the subject had to spend a lot of time to analyze the environment. The important fact to notice is that every subject



Figure 4-3: This Figure Shows that Each Subject Took a Different approach from the Others.

experienced some kind of impending collision. Without collision detection or avoidance, all subjects would have had manipulator collisions with the obstacles. A critical collision is a collision with obstacle unregistered in the world model. The computer cannot detect such collisions, and thus cannot avoid them.

Learning by the subjects is well shown in Fig. 4-6. The most collisions occurred in the environment which each subject encountered first. Again, subjects learned quickly about the other environments which were similar to the first environment and had fewer number of collisions in those environments. Each subject experienced, on average, 2 impending collisions.

Each subject experienced a singular configuration of the manipulator at the workspace boundary more than once. Even though the numbers of singularities are small, the subjects had to be constantly concerned with the possibility of commanding a wrong

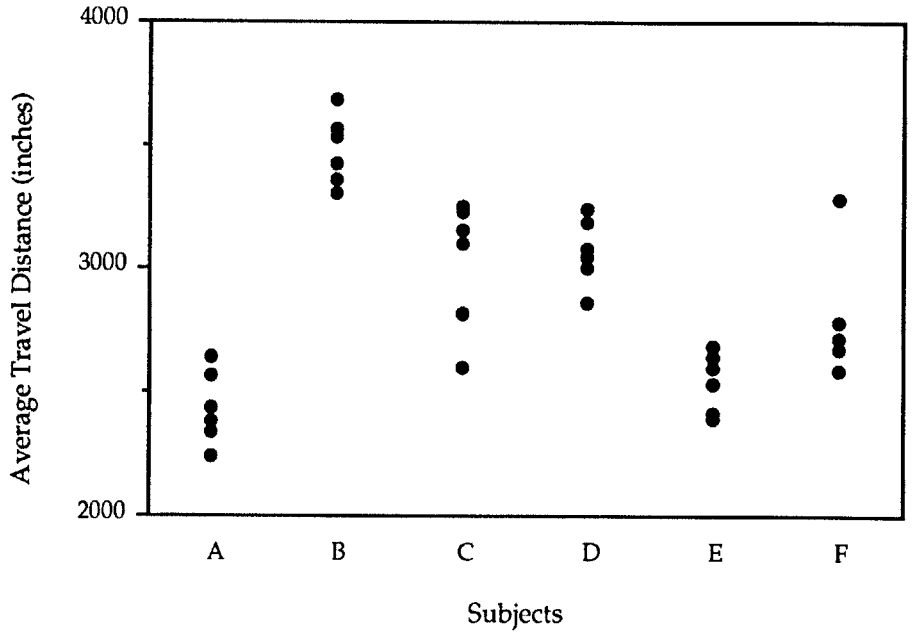


Figure 4-4: The Travel Distance of the Manipulator Was Consistently Different from Subject to Subject.

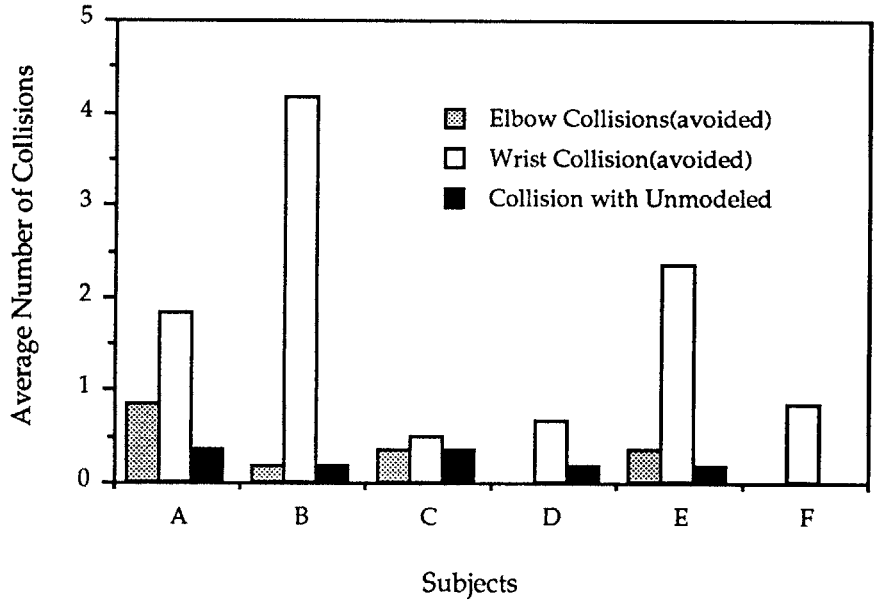


Figure 4-5: Different Types of Collisions for Each Subject.

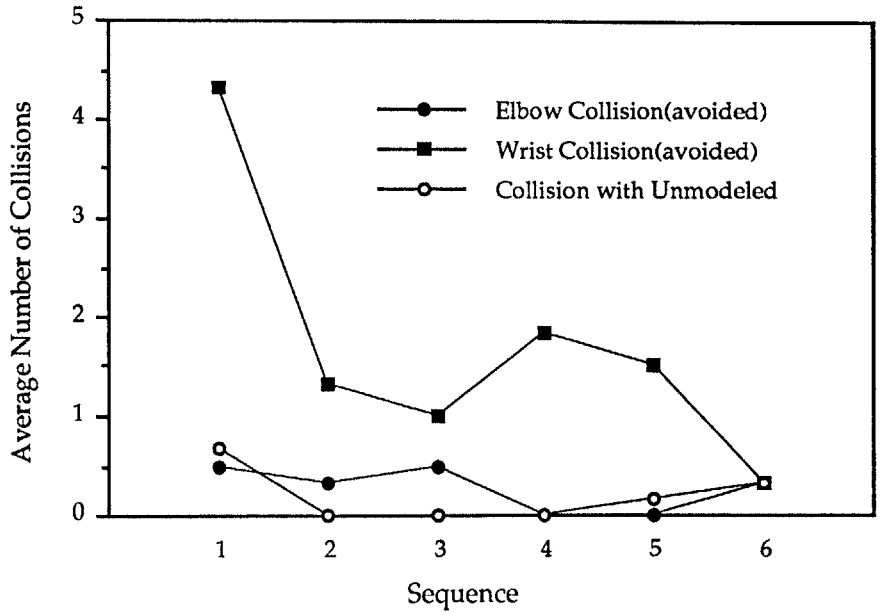


Figure 4-6: As the Subjects Learned More About the Environments, the Total Number of Collisions Shrank. There Was a Drastic Reduction on the Numbers Between the First and the Second Environments.

trajectory that would result in a singularity.

Figure 4-7 shows how the number of different types of collisions depended upon the number of objects that were not registered in the world model.

4.1.4 Observations on Experiment

It was difficult for the subjects to predict if and when an elbow collision of the manipulator would occur. Wrist collision was easier. However, the majority of the collisions occurred at the wrist of the manipulator. Most of them were glancing collisions with obstacles.

The subjects got tired at the end of the experiment. Figure 4-8 shows an increase of observing time even though they had environments very similar to earlier ones.

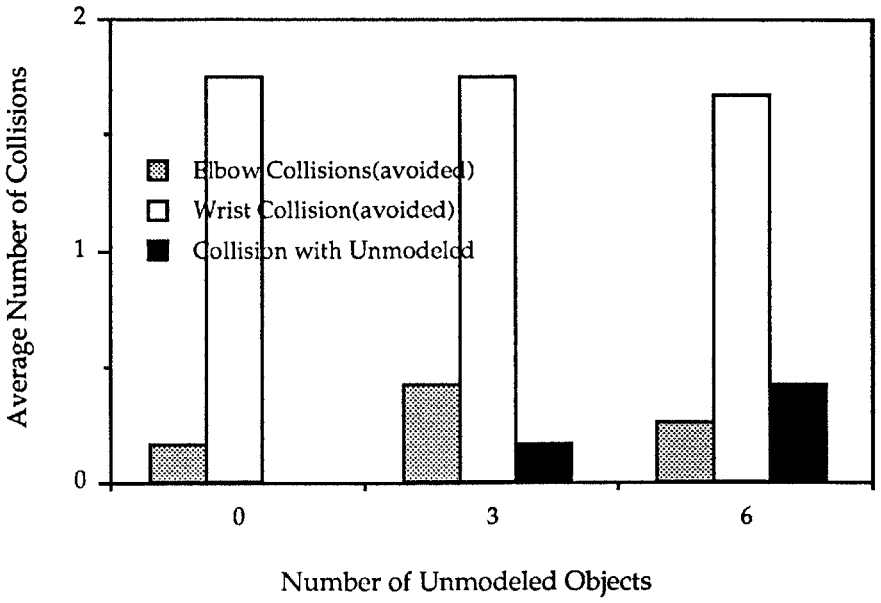


Figure 4-7: Frequency of Collisions with Unmodeled Objects Increased As the Number of Unmodeled Objects Increased, As Expected.

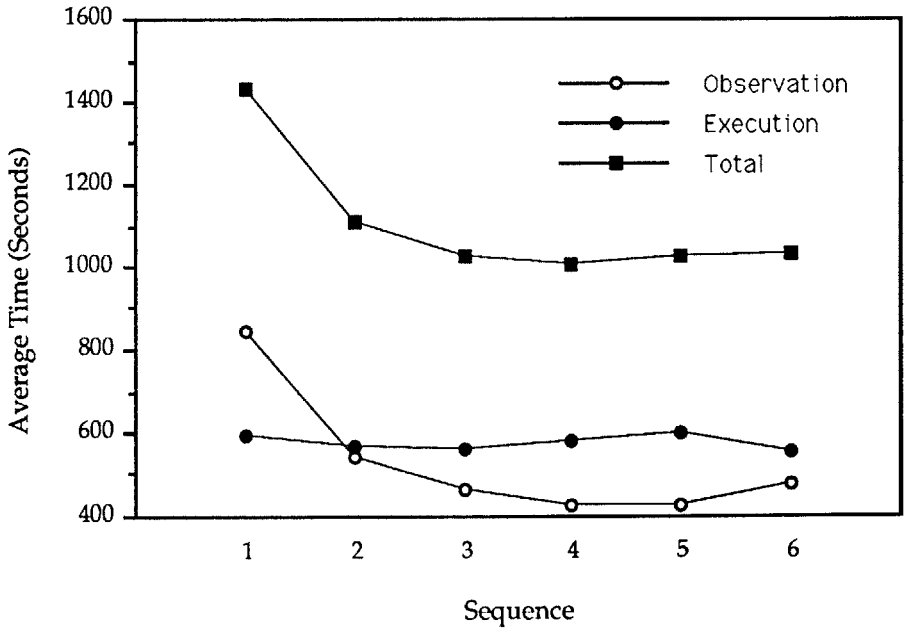


Figure 4-8: As the Subjects Learned More About the Environments, the Time Spent for Observation Was Reduced Until the Last Two Environments.

Another difficulty in planning was to predict when and where, during manipulator motions, the manipulator would go out of the workspace and result in a singular configuration of the manipulator . All such singular configurations in the experiment occurred when the upper arm got stuck with the forearm. Even if the starting and end points were within the workspace, a straight path between them could result in a singular configuration. More discussion on this matter was in Chapter 3.

4.2 Experiment 2: Introduction of Objects to the World Model by Flying-and-Matching

4.2.1 Objective

The objective of this experiment was to evaluate the proposed method of introducing objects into the world model by flying as described below.

4.2.2 Setup and Procedure

Setup

Subjects were seated in front of the display. On the display, the solid model for the unmodeled object was superimposed on the camera view which shows the unmodeled objects and others. The subjects were asked to control the 6-degree-of-freedom joystick to "fly" the solid model so as to match with the camera display of the "actual" object in all six degrees of freedom. Size was the cue for depth. The joystick coordinates were

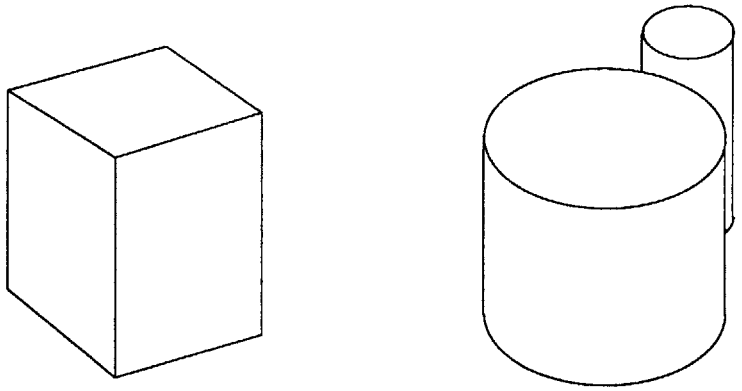


Figure 4-9: Two Objects Used in the Experiment.

transformed to coincide with the screen coordinate. Pulling the joystick out of the screen, for example, would move the solid model out of the screen.

Objects

The unmodeled objects used in this experiment were a rectangular solid whose size was 30 x 40 x 70(inches), and a cylindrical solid which had two parts, one 60 (inches) diameter and 70 (inches) long, and the other 30 (inches) diameter and 70 (inches) long, as shown in Fig. 4-9. The centerlines of the two parts were offset by 10 (inches) in longitudal direction in order to give a unique configuration for a particular display of the object. Each face of the actual unmodeled object had a different color. Each surface of the solid model of the object had a different color which corresponded to the "actual" object. This prevented confusion due to geometric similarities in different positions and orientations. For example, an object rotated 180 degrees about any axis would appear to be the same as the original object, if all the surfaces had the same color.

Viewing Conditions

The camera was located 707 inches from the objects. The orientation of the object was determined by a random number generator. The display of the "actual" object was displayed at the center of the screen. The size of the screen was 14 inches (900 by 1024 pixels). The position of the display of the solid model of the object was also determined by a random number generator. The deviation of the initial position of the solid model from the position of the "actual" object was limited within 100 inches. The objects were displayed in a *perspective* view.(Note that the size of the display of an object in an *orthogonal* view would be constant regardless of the object distance from the camera.) The experiment was done with different fields-of-view(10, 20, 30 degrees) [3] and visibility(100% and 50%).[4] The different fields-of-view simulated zoom lenses with different focal lengths. The relationship between the fields-of-view and the computer display is explained further in Appendix A.

Variations

In actual applications of this method of introducing objects, some objects that the operator wanted to introduce to the world model may be partially hidden by other objects. In order to evaluate any change of the performance when the operator cannot see all portions of the objects, the visibility of the object was varied. The 50% reduction of the visibility was achieved by displaying an appropriately-sized opaque object in front of the viewing coordinate frame.

[3]For the rectangular solid only.
[4]For both the rectangular solid and the cylindrical solid.

Each subject was asked to do five runs of the task with the same object, each run with a different location and orientation of the object. The subjects were given at most 3 minutes for one run for a rectangular solid, and 4 minutes for one run for a cylindrical solid. During preliminary experiments, the subjects indicated that matching the cylindrical solid is more difficult than matching the rectangular solid. If the subjects decided within the given time for a run that the displays of the actual unmodeled object and its solid model matched well, he hit a key and went on with the next run.

Subjects

Two young adult male subjects participated in the experiment. They already had a good deal of experience with the 6-degree-of-freedom joystick. Each of them was trained for at least 30 minutes before the experiments to saturate the learning effect. During the training period, completion times were regularly checked to see whether the subject reached an asymptotic, steady-state performance. However, during the actual experiment, mean completion times were not checked until all the experimental runs were completed.

4.2.3 Results

Even though there was a time limit of 3 minutes (or 4 minutes) for each run of the task, the subjects finished most runs before the time limit. The accuracy of the position and orientation of the object were used as the performance measures. The positional and orientational errors were transformed relative to the eye coordinate(or the viewing coordinate).

Positional Errors

Positional errors are expressed in the viewing coordinate system. Positional errors got larger as the field of view got larger as shown in Fig. 4-10 for the rectangular solid. The scale of the errors is normalized by Z_o, the distance between the camera and the object. Since the x-components and y-components of positional errors are small, the errors are equivalent to angles that are formed by the actual position of the object, the camera position, and the estimated position of the object. Namely,

$$\alpha = \frac{\Delta x}{Z_o} \tag{4.1}$$

The z-components of positional errors also got larger as the field of view increased as shown in Fig. 4-11. It was clear, as observed from Fig. 4-10 and Fig. 4-11, that the z-components of positional errors were significantly larger than those in the x- and y-components. This was expected because most depth cues of the object were lost when the object was displayed on the two-dimensional screen. The only depth cue the subjects could use for locating the object was the size of the display of the object on the screen. As the object got near the viewing coordinate frame, it looked larger. The subjects had difficulty using this depth cue effectively, which resulted in poor estimates in the z-direction.

When 50% of the object was occluded, the x- and y-components of position errors got large as shown in Fig. 4-12 for a rectangular solid and a cylindrical solid. The z-components of positional errors also got large when the object was occluded. Figure 4-13 shows how error in the z-components of position errors changed as the object got oc-

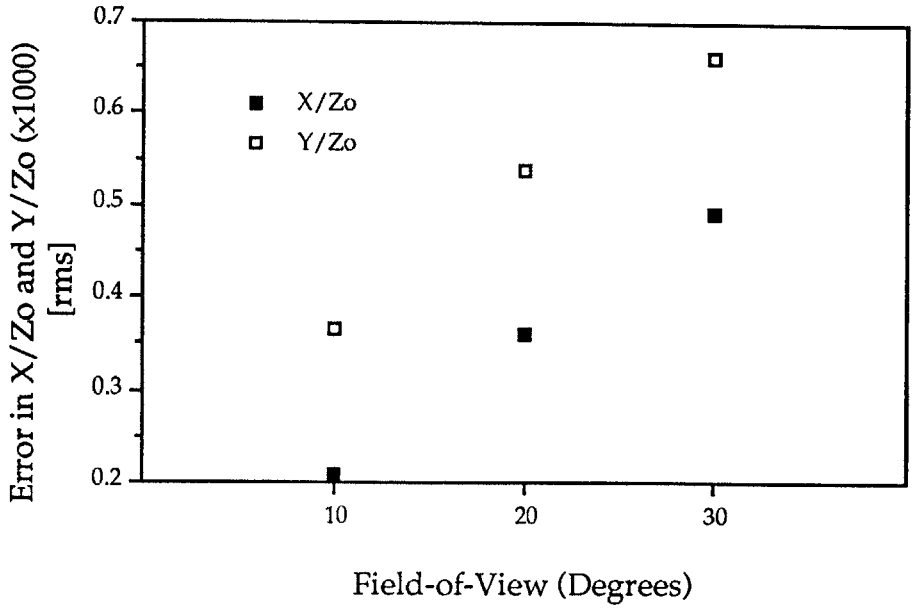


Figure 4-10: As the Field-of-View Increased, the Positional RMS Errors in X- and Y-directions Increased for a Rectangular Solid.

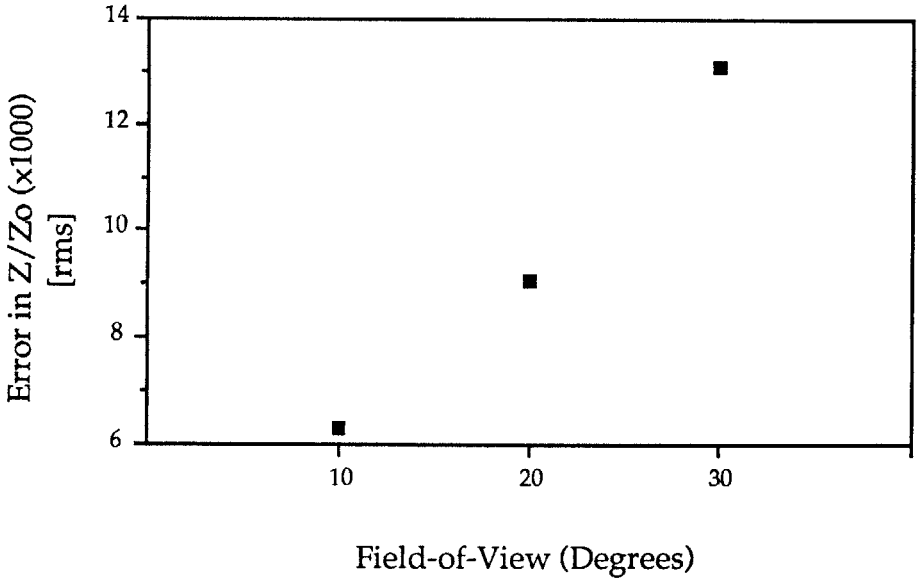


Figure 4-11: As the Field-of-View Increased, the Positional RMS Errors in the Z-direction Increased for a Rectangular Solid.

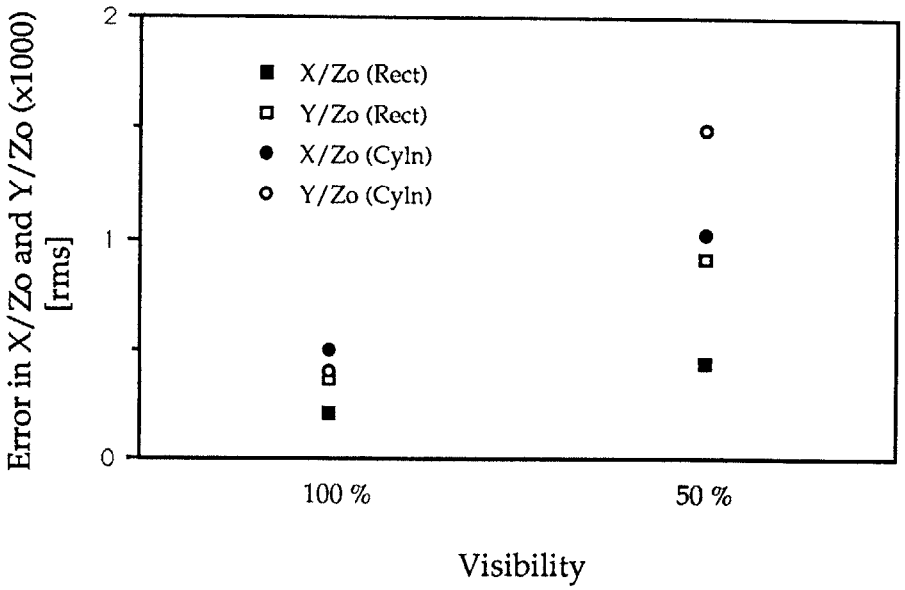


Figure 4-12: The Positional RMS Errors in the X- and Y-directions Was Approximately Doubled As 50% of the Target Rectangular Solid Was Occluded.

cluded, and for different objects. When the objects were not occluded, the z-components of positional errors for the cylindrical solid were larger than those for a rectangular solid. However, when the objects were occluded, the difference got insignificant.

Representation of Orientational Errors

In general, angular error can be represented in many different forms such as Euler angles and quaternions. However, a different angular error representation is used. Figure 4-14 shows two objects that are misaligned to each other. The angular error can be represented by a unit vector, $\hat{k}$ and an an angle, Θ. Unit vector $\hat{k}$ indicates only the direction of rotation to be made in order to align the two objects (axis of rotation), and angle Θ denotes the amount of the rotation.

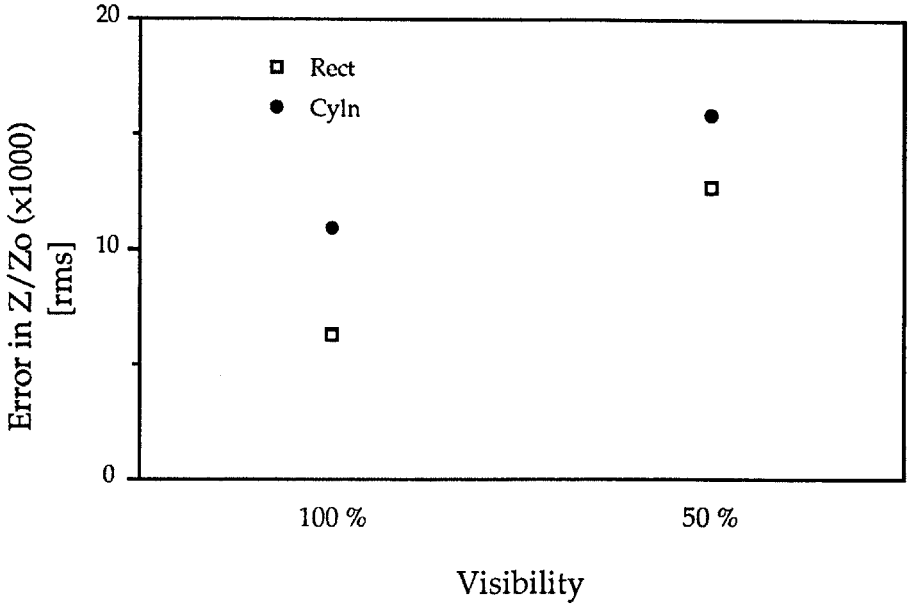


Figure 4-13: The Positional RMS Error in the Z-direction Increased As 50% of the Target Was Occluded.

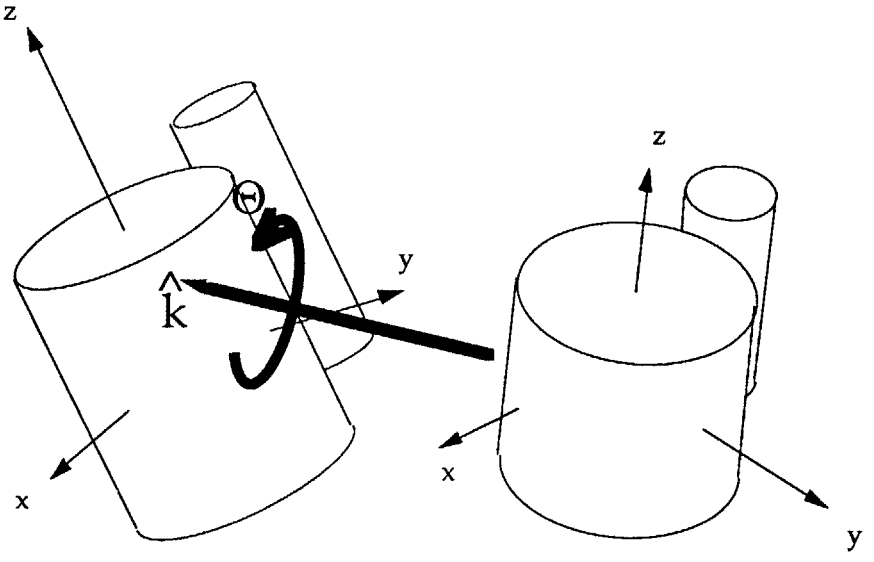


Figure 4-14: An Angular Error Is Represented by a Unit Vector and an Angle.

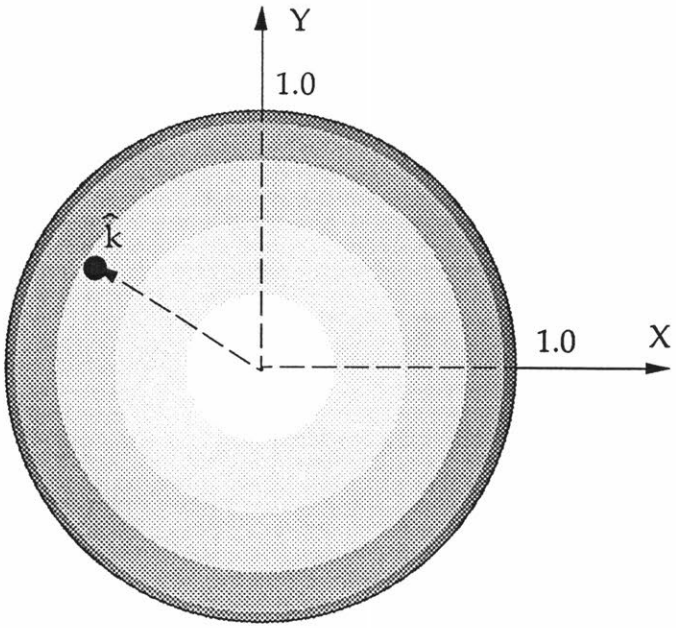
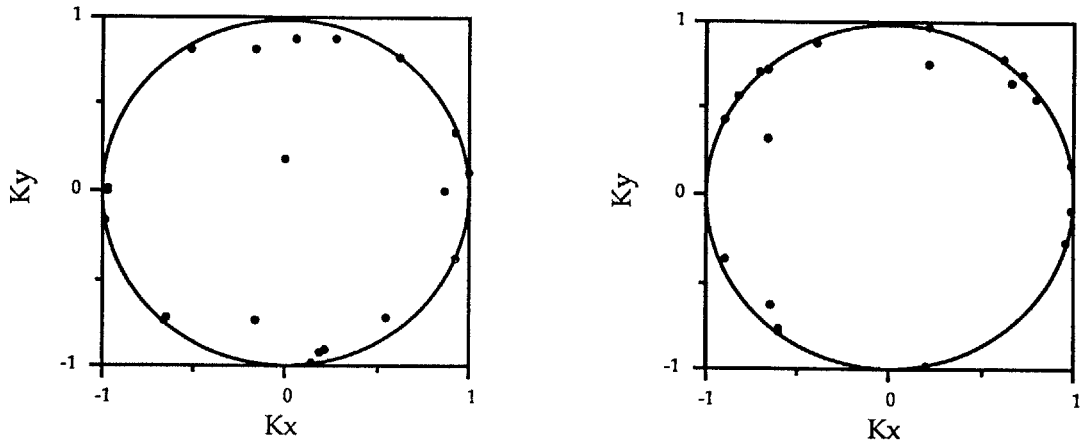


Figure 4-15: A Unit Sphere Is Spanned by Angular Errors.

If directions of angular errors are random, $\hat{k}$ would be uniformly distributed on a unit sphere. Figure 4-15 shows a unit sphere viewed in the z-axis of the viewing coordinate system. An angular error in the z-axis only would have $\hat{k}$ pointing either in the direction of the z-axis or in the direction of the negative z-axis. This could be represented by a point in the middle of a unit (transparent) sphere when it is viewed in the z-axis direction.

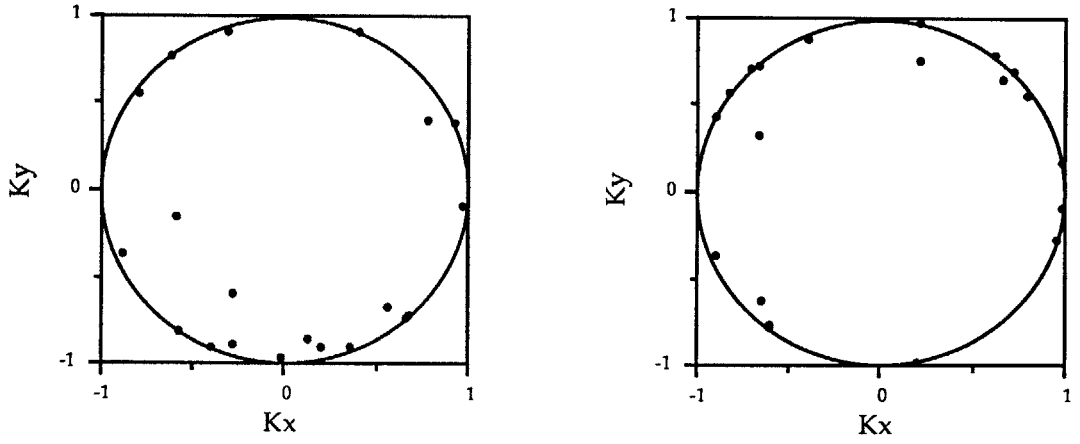
Orientational Errors

Angular error directions under different conditions are shown in Fig. 4-16. They show that there was no significant difference in the pattern of data whether objects had distinct vertices or not, and whether objects were occluded or not. They also show that angular errors in the z-direction (the viewing direction) were very small compared to those in the x-direction and the y-direction.



(a) Unoccluded (b) 50% occluded

Rectangular Solid



(a) Unoccluded (b) 50% occluded

Cylindrical Solid

Figure 4-16: Most Directions of Orientational Errors Were Close to the Normal to the Viewing Direction.

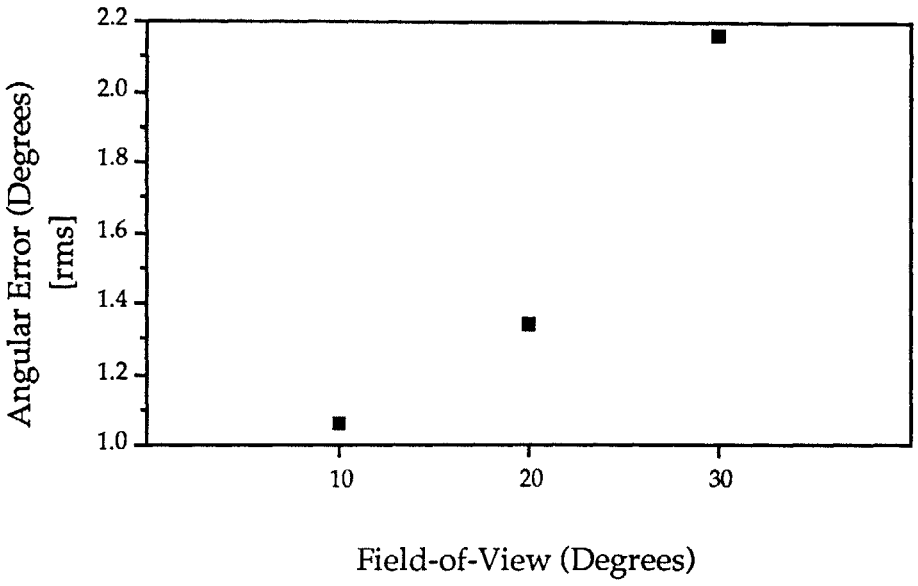


Figure 4-17: As the Field-of-View Increased, the Angular RMS Error Increased for a Rectangular Solid.

Different experimental conditions resulted in different magnitudes of angular errors. The angular error got larger as the field of view got larger for a rectangular solid as shown in Fig. 4-17. However, the mean error was less than 1 degrees when the field of view was 10 degrees. Vertical bars in the figure indicate standard deviations[5]. Figure 4-18 shows how the angular errors changed when the objects were occluded. When 50% of the objects were occluded, the angular errors got large for both objects. It is also clear from the figure that angular errors for the cylindrical solid were larger than those for the rectangular solid when they were not occluded regardless of whether they were occluded or not.

---

[5]standard deviation(s) = $\sqrt{\left(\sum_{i=1}^{N}(\epsilon_i - \bar{\epsilon})^2\right)/(N-1)}$ whereas rms error(r) = $\sqrt{\left(\sum_{i=1}^{N}\epsilon_i^2\right)/N}$. Therefore, $s = \sqrt{\frac{N}{N-1}(r^2 - \bar{\epsilon}^2)}$

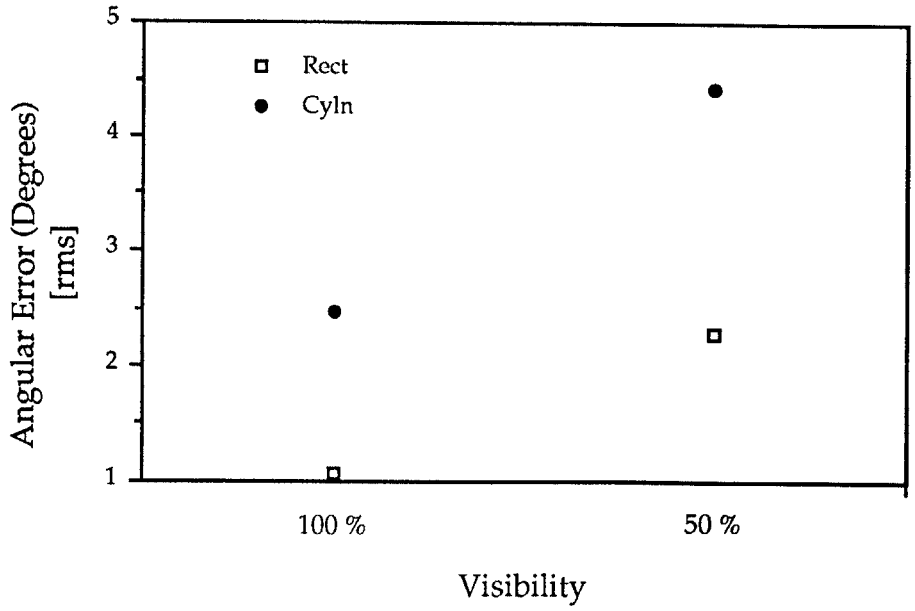


Figure 4-18: Angular RMS Error Got Larger When 50% of the Target Object Was Occluded.

4.3 Experiment 3: Introduction of Objects into World Model and Stacking Operation

4.3.1 Objective

There were two objectives of this experiment. One was to compare the system with and without the capability of collision avoidance through a realistic task of pick-and-place. The other objective was to compare the performances of building a world model and completing the pick-and-place task when different world models were used: a crude world model and a precise world model.

4.3.2 Setup and Procedure

Task

The subjects were asked to do two tasks in sequence: first, introduction of six objects into the world model, and then a manipulator operation of stacking five cubic blocks at a given target location. For the introduction task, the subjects were asked to introduce 6 objects into a world model using the flying-and-matching method which was introduced in Section 3.2. The manipulator operation task was similar to the one used in Section 4.1.

Experimental Conditions

Each subject performed the tasks in three different conditions. One was with a precise world model and without collision avoidance (PD). Another was with a precise world model and with collision avoidance (PA). The other was with a crude world model and with collision avoidance (CA). When the capability of avoiding impending collisions was not available, the capability of detecting collisions was assumed.

For PD and PA, the subjects were asked to introduce objects as closely as possible using two camera views.[6] It was possible to introduce objects precisely into the world model because there were two cameras available. The subjects could pan and tilt them, and change their positions. Two camera views could be simultaneously displayed on the screen, which made it easy for the subjects to introduce objects precisely into the world model. It was assumed that the introduction phase was the same for both PD and

[6]The author observed how accurately an introduction was done and judged when it was good enough to stop.

PA, and precise introductions of 6 objects were done only once for PD and PA. For CA, the subjects were asked to introduce objects in a crude fashion ("Actual" object should be enclosed by their virtual model). The concept of this kind of crude modeling was introduced in Section 3.2.

The manipulator operation was almost the same as the one in Section 4.1. One difference was that the subjects in this experiment did not have to be concerned with unmodeled objects because any initially unmodeled object was registered into the world model. This meant that the subjects did not have to swap the camera view and the graphic simulation, and that they could solely depend on the graphic simulation of the world model. Another difference from Experiment 1 was that the display of the workspace sphere was available in this experiment.

Environment

Each subject was asked to do the same task of stacking five blocks in two different environments: Model I and Model II. In Model I, many objects (or obstacles) were in the way of the manipulator, and long maneuvering of the manipulator was needed (even without the obstacles) to complete the task. Model II was simpler in this sense.

Subjects

A total of 4 subjects participated in this experiment. They were well trained particularly for using the six-d.o.f. joystick before participating in the experiment. Two of them had substantially long experience of using the joystick. The others had at least 5 hours of training for the joystick.

4.3.3 Results

Introduction of Objects

It took less time to introduce objects crudely than precisely as shown in Fig. 4-19. "P" and "C" stand for a precise model and a crude model, respectively. There was more consistency in the crude introduction than the precise introduction. The most difficult part in introducing objects by "flying-and-matching" was fine manipulations of objects to get precise matches, which required skills to control the 6-degree-of-freedom joystick well. Even after the long training, there were differences in the level of subject skills to control the joystick. Since a crude introduction does not require intensive fine manipulations that requires such skills, times taken for crude introductions of objects were more consistent. In the figure, Subject D was the author, who had the most experience with the joystick. Without the data for Subject D, the differences between the precise modeling and the crude model would have been more significant.

Task

Due to the graphic aid which assisted the subject to avoid specifying a trajectory part of which was outside the workspace, no singularity configuration at the workspace boundary occurred. This certainly relieved the subject of pressure to come up with a trajectory that did not result in a singular configuration at the boundary.

The subjects could not totally prevent impending collisions even though they knew that the process of avoiding impending collision would take time, slowing down the

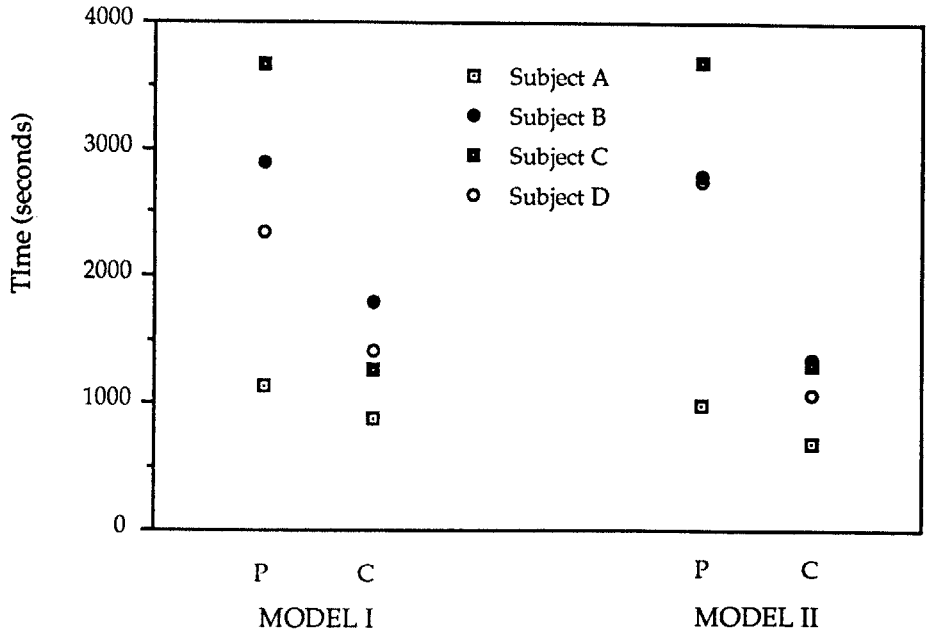


Figure 4-19: It Took More Time To Introduce 6 Objects Precisely into the World Model Than Introducing Them Crudely.

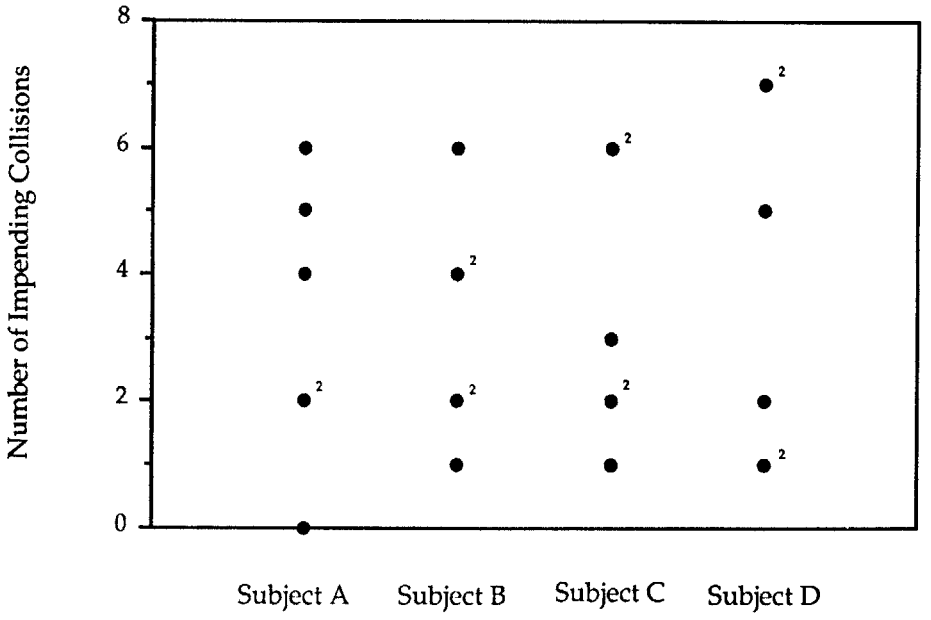


Figure 4-20: Each Subject Had About 3.6 Collisions Per Task on Average.

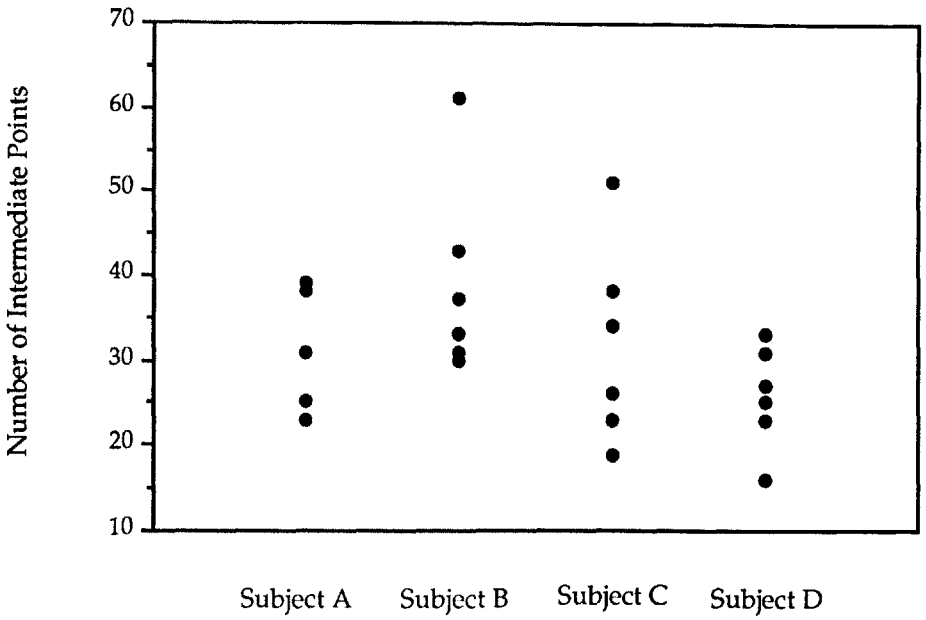


Figure 4-21: The Subjects Took Different Strategies in Terms of Specifying Intermediate Points.

operation speed. Figure 4-20 shows how many impending collisions each subject experienced for each task. There were 3.375 impending collisions per task per person on average. Figure 4-21 shows how many intermediate points each subject pointed to in order to specify desired trajectories for the manipulator. This shows that each subject took a different strategy. Subject B, for example, was very cautious and pointed to more intermediate points than any other subject.

Collision Detection Only vs. Collision Avoidance

It took more time to complete the task when only collision detection was available than when collision avoidance was available, as shown in Fig. 4-22. This was mainly due to the fact that with the system that had no capability of avoiding collisions, the subject had to spend time in specifying a new trajectory when the manipulator simply stopped upon detecting an impending collision. Figure 4-23 shows that the subjects specified

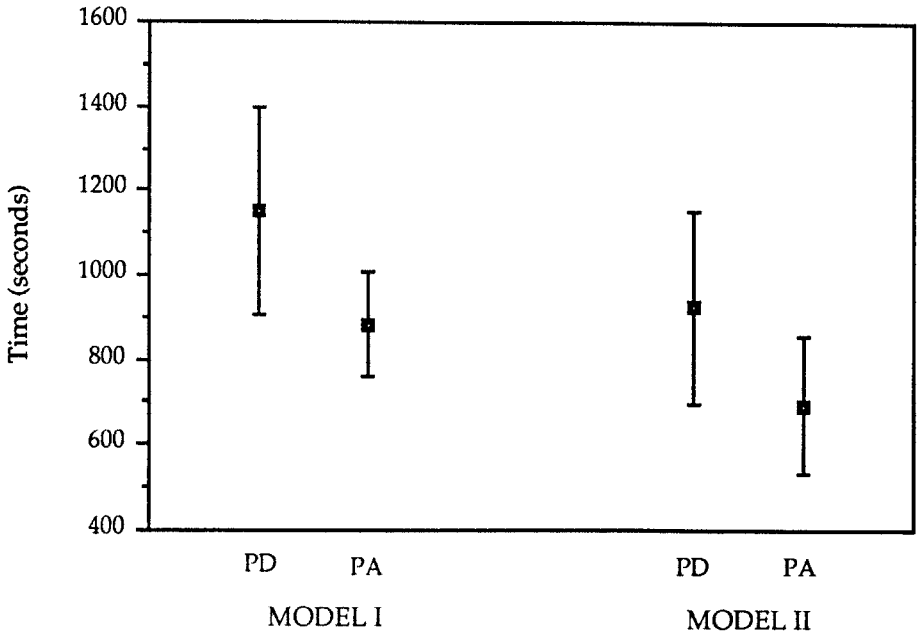


Figure 4-22: It Took a Less Time To Complete a Task of Stacking 5 Blocks When the System Was Capable of Avoiding Impending Collisions.

more intermediate points when the system had no capability of avoiding collisions. This showed that it was better to have a collision avoiding capability in the system.

Figure 4-24 shows that the average travel distance by the manipulator got shorter for the system with a capability of collision avoidance than for the system without it. This is because the subjects got more conservative in specifying trajectories of the manipulator and instructed trajectories to remain at some distance from obstacles when the system didn't have the capability of collision avoidance.

Precise Model vs. Crude Model

The results showed that it took more time to complete the task with a crude world model than with a precise model as shown in Fig. 4-25. There are a few reasons that could explain the result. First, a crude world model had less free space than a precise

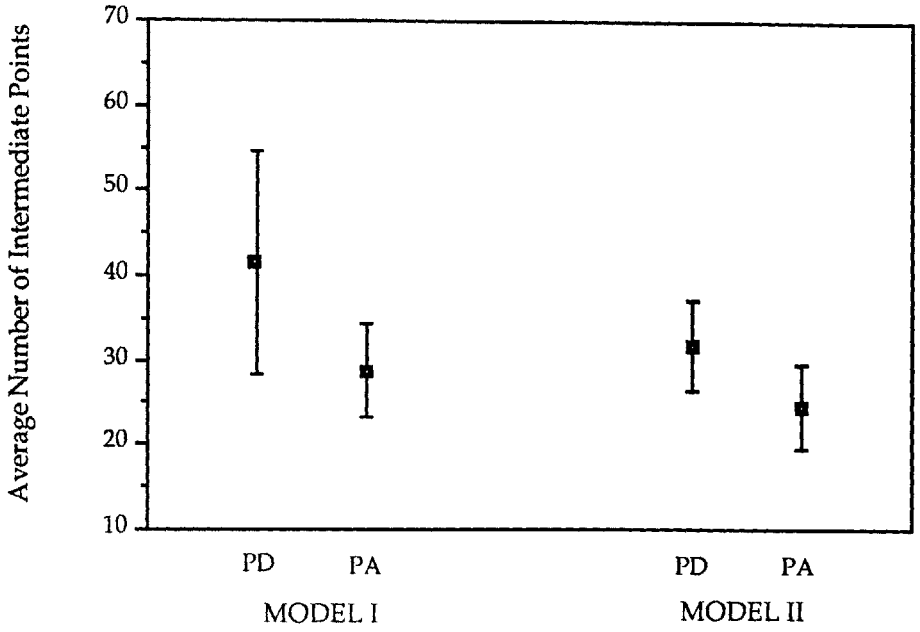


Figure 4-23: The Subjects Specified a Fewer Number of Intermediate Points When the System Was Capable of Avoiding Impending Collisions.

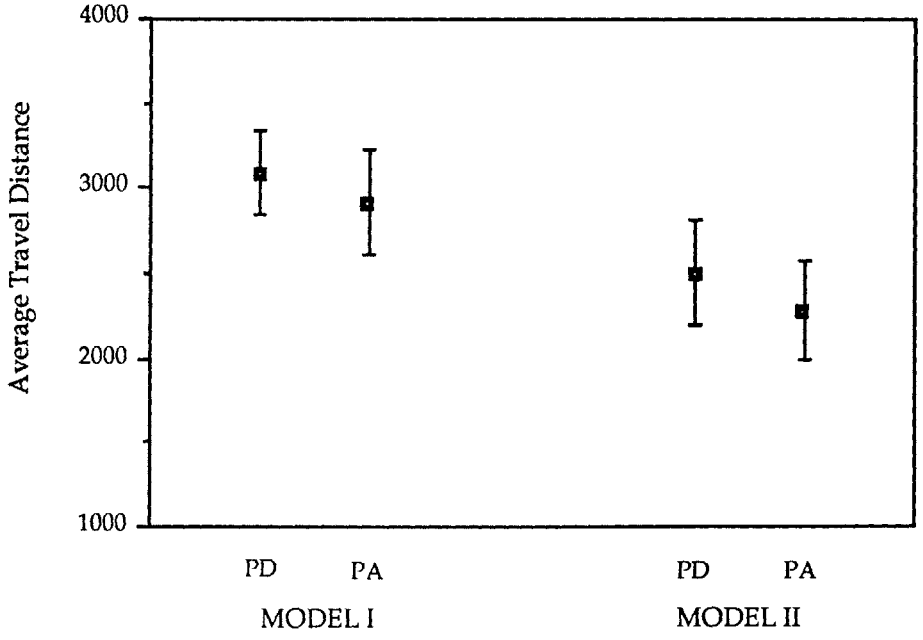


Figure 4-24: The Manipulator Traveled Less Distance When It Was Capable of Avoiding Impending Collisions.

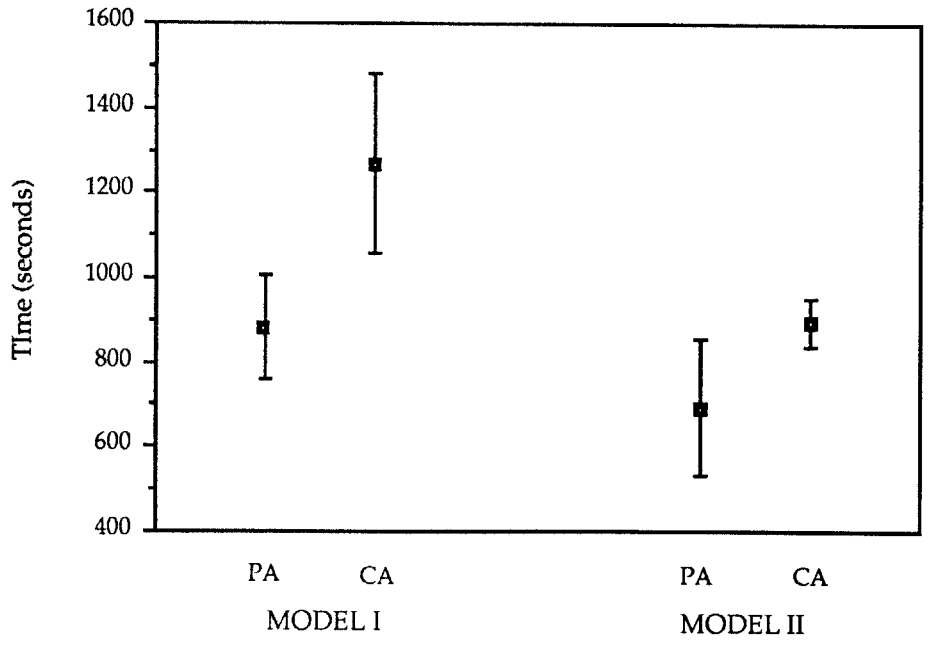


Figure 4-25: It ook a More Time To Complete a Task with a Crude World Model.

model; therefore, the manipulator had to travel longer distances. Figure 4-26 shows that in Model I it took more travel distance by the manipulator for a crude world model than for a precise model. In Model II, the differences between the crude model and the precise model were negligible due to the simple structure of the environment. Second, avoiding collision took time; therefore, the more collisions occurred, the more time it would take. Figure 4-27 shows that more collisions occurred with a crude world model. The objects in the world model looked significantly different from the nominal objects because some of them were virtual objects. Therefore, it was difficult for the subjects to perceive the shapes the objects. This resulted in more collisions. Third, the subject in a crude model had a greater number of faces, and drawing more polygons in the graphics workstation took more time.

Figure 4-28 shows how many intermediate points were specified depending upon the world model. For Model I, the subjects specified more intermediate points with a

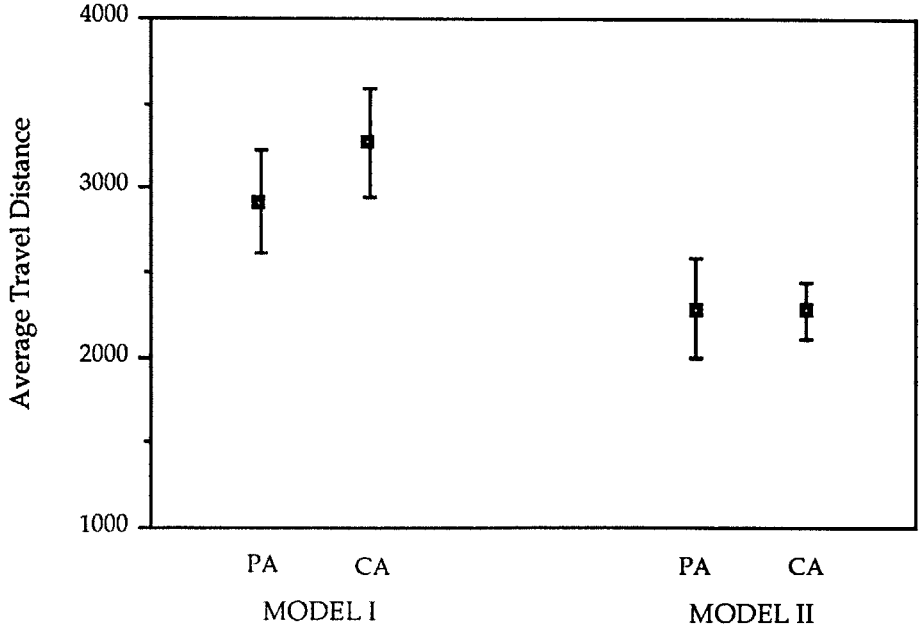


Figure 4-26: When a Crude World Model Was Used, the Manipulator Moved Longer Distance in Model I Due to a Small Free Space.

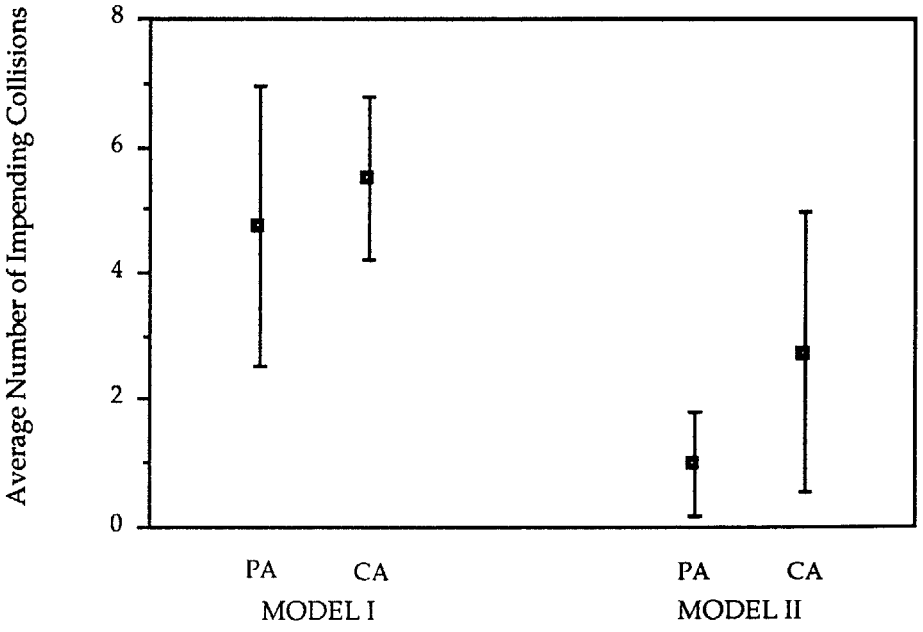


Figure 4-27: The Average Number of Collisions Increased for both Model I and Model II When a Crude World Model Was Used.

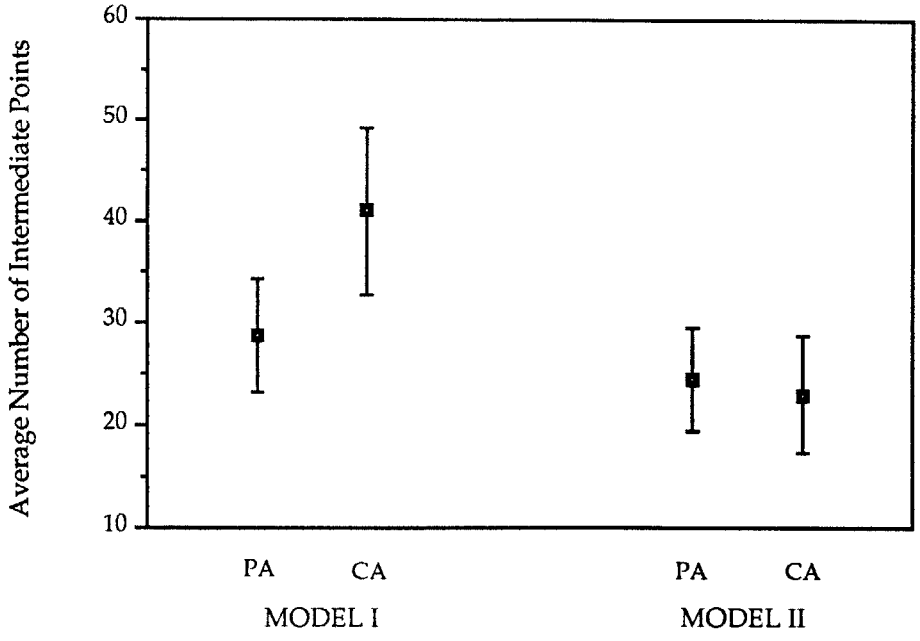


Figure 4-28: The Subjects Specified More Intermediate Points in Model I When a Crude World Model Was Used.

crude model than with a precise model. For Model II, there was no significant difference in the number of intermediate points due to the simple structure of the environment. The crude model environment was filled with strange-looking obstacles, and subjects tried to instruct collision-free trajectories by specifying more intermediate points.

Effect of Modeling Method on the Pick-and-Place Task

The total time of completing the task, combination of introducing 6 unmodeled objects and operating the manipulator, was dictated by the introduction time because there were a large difference in introduction times between a crude model and a precise model. The result was that crude modeling was the better choise.

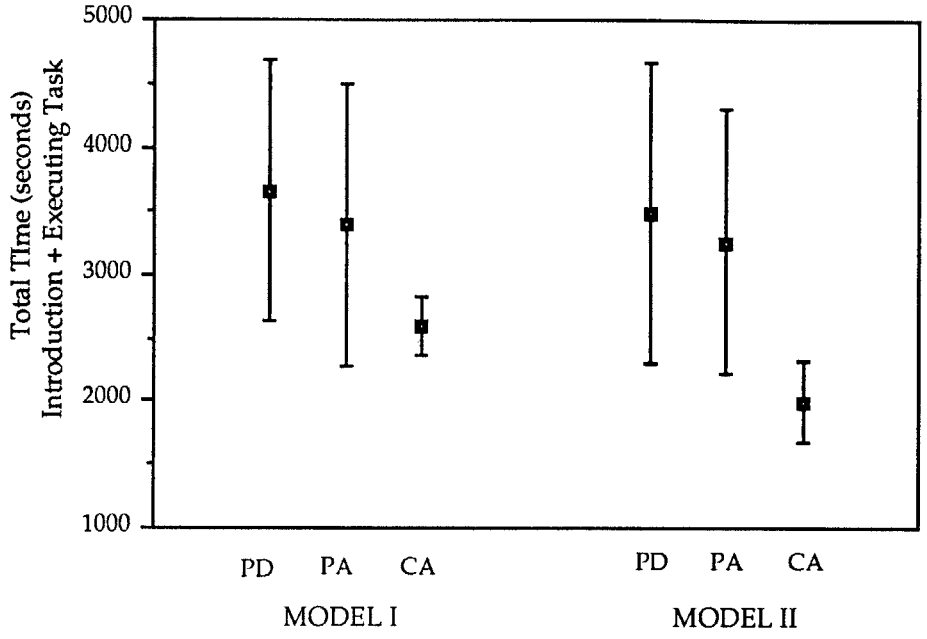


Figure 4-29: Total Time Spent for Introducing 6 Objects into the World Model and Completing the Task of Stacking 5 Blocks Was Significantly Reduced When the System Was Capable of Avoiding Collisions and When a Crude World Model Was Constructed.

Chapter 5

Conclusions and Recommendations for Future Work

5.1 Conclusions

5.1.1 Supervisory System

Supervisory Collision Avoidance

A supervisory approach to collision avoidance for the manipulator in a three-dimensional space was developed and implemented in a SGI's Iris graphics workstation. The approach is based upon cooperation between the operator and the computer. This allows, in principle, the manipulator to avoid unmodeled obstacles because the operator makes

global decisions regarding paths of the manipulator. However, it was observed from experiment that the subjects sometimes failed to consider unmodeled obstacles when making a decision for a path, resulting in manipulator collisions with unmodeled obstacles. The conclusion that can be drawn from the experiment is that introducing unmodeled objects (or obstacles) into the world model is necessary in order to surely prevent manipulator collisions with them.

Graphic Aids

It is experimentally proven that graphical aids to the operator help him make global decisions. Presenting a workspace sphere to the operator was able to prevent him from making a plan that would force the manipulator out of its workspace boundaries and, therefore, result in a singular configuration of the manipulator at the boundaries.

Advantage of Collision Avoidance

It is suggested through out the experiments that the system with the capability of avoiding collisions automatically can complete a task more quickly than one with the capability of detecting collisions only. The system with capability of avoiding collisions automatically also relieves the operator of the pressure to generate a collision-free path. Without automatical collision avoidance, the subjects tried hard to generate a collision-free path because they did not want to spend time to correct a path if it had an impending collision.

5.1.2 Uncertainty

Virtual Obstacles

The concept of virtual obstacles was introduced to model uncertain regions that have not been observed by the operator because they are occluded by objects in the environment. Using virtual obstacles as part of the world model for collision avoidance, prevents the manipulator from intruding into the uncertain regions and thus removes the possibility of manipulator collisions there.

Introduction of Objects

A new way of introducing unmodeled objects into the world model based on a geometry data base is developed and implemented on the workstation. It is called the "flying-and-matching" method. The operator "flies" a model object with a joystick, and matches it on the graphic screen with the corresponding "actual" object. This method does not require any special equipment other than the capability of superposition of a camera signal and the graphic renderings. This method can be used for objects that have no distinct vertices.

The position accuracy of the method is lower in the z-direction of the viewing coordinate system than in the other directions. The position accuracy depends on the field-of-view of the display, the shape of the object, and occlusion of the object. Significantly smaller accuracy is obtained for objects without distinctive vertices such as cylindrical solids. With a smaller field-of-view, the display of the object gets larger,

and more accurate positioning is possible. When the objects get occluded by 50%, the accuracy deteriorates a little bit.

There is no particular pattern of angular error direction except that angular error in the z-direction of the viewing coordinate system is less than that of the other directions: x and y. Like position error in the z-direction, the size of an angular error depends on the field-of-view of the display, the shape of the object, and occlusion of the object. With a smaller field-of-view, the display of the object gets larger, and more accurate positioning is possible. When the objects get occluded by 50%, the angular accuracy deteriorates *significantly*. When the objects are not occluded, the accuracy for objects without a distinct vertex is far less than that for ones with. However, there is not so much difference in accuracy for objects with and without a distinct vertex when they are occluded by 50%.

Two Different Strategies of Introductions

Two different strategies of introducing unmodeled objects into the world model are proposed: precise introduction and crude introduction. They were used and compared to each other in a realistic experiment of a robot operation. The crude introduction is quicker than the precise introduction. Despite less free space for the manipulator, the operator can complete a task in a less time when the crude introductions are used.

5.2 Recommendations

Following are the areas recommended for further research.

1. I have implemented the system in graphic simulation. The ultimate test of the system is to implement it on an actual robot manipulator. The simulation done here could be repeated on an actual environment. However, there may be problems, such as a system calibration, to solve before implementing the system on an actual environment.

2. It has been assumed that dynamics of the manipulator are negligible. However, dynamics become important when the speed of the manipulator increases. Therefore, it is desirable to consider dynamics of the manipulator, which may require modification of the world model and the manipulator model.

3. I have suggested and implemented a scheme of introducing objects with corresponding models into the world model. But it cannot be used when the objects to be introduced do not have corresponding models. Some systems have schemes of introduction based on primitive models; however, they take up an excessive amount of space, leaving less free space. Thus, a quick and interactive introduction scheme should be developed to introduce objects without corresponding models. A possible scenario is that the operator points, with a mouse cursor, to all the vertices of a solid in order to provide the computer with information needed to build a polyhedral model.

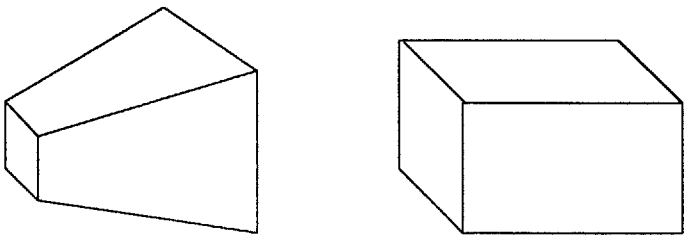
4. I have conducted experiments as a preliminary investigation of how accurately subjects could introduce objects into the world model with the "flying-and-matching" method. More strategies should be developed to use this method in different conditions of operation: shapes of objects, lighting conditions, colors of display, occlusion and fields-of-view. The result could be used to improve the implementation of the scheme.

5. In the system designed for supervisory collision avoidance, the human operator assists the computer in solving a collision-free path for the manipulator by suggesting a desired path. If many similar movements of the manipulator in one environment are needed in order to complete a task, it could be a boring task for the operator to instruct all the paths. If the system can *learn* about the motion patterns, it could generate a reasonably short path for the manipulator.

Appendix A

Projection Transformations

Viewing items in perspective on the computer screen is like looking through a rectangular piece of perfectly transparent glass at the items. Imagine drawing a line from your eye through a glass until it hits the item, then imagine coloring a dot on the glass where the line passes through the same color as the item that the line intersects. If this were done for all possible lines through the glass, the coloring were perfect, and the eye not allowed to move, then the picture painted on the glass would be indistinguishable from the true scene. The collection of all the lines leaving your eye and passing through the glass would form an infinite four-sided pyramid with its apex at your eye. Anything outside the pyramid would not appear on the glass, so the four planes passing through your eye and the edges of the glass would *clip* out invisible items. These are called *clipping planes*. The geometry hardware also provides two other clipping planes that eliminate anything too far from the eye or too near the eye. They are also *clipping planes*. Thus for a perspective view, the region of the world that is visible looks like an Egyptian pyramid



a) Perspective b) Orthogonal

Figure A-1: Primitives Created by Basic Transformations.

with the top sliced off. The technical name for this is a *frustum,* or *rectangular viewing frustum.* Figure A-1 a) shows a frustum specified by a perspective transformation.

A.1 Perspective Transformation

Perspective transformation defines the viewing pyramid by indicating the field-of-view angle (θ) in the y direction of the eye coordinate system, the aspect ratio (a) which determines the field of view in the x direction, and the location of the near and far clipping planes (n and f) in the z direction. The clipping planes are the boundaries of the viewing pyramid. Perspective transformation is defined as

$$Perspective(\theta, a, n, f) = \begin{bmatrix} (1/a)cot(\theta/2) & 0 & 0 & 0 \\ 0 & cot(\theta/2) & 0 & 0 \\ 0 & 0 & -(f+n)/(f-n) & -1 \\ 0 & 0 & -2fn/(f-n) & 0 \end{bmatrix}. \quad \text{(A.1)}$$

A.2 Orthogonal Transformation

Orthogonal transformation defines a box-shaped enclosure in the eye coordinate system. Its viewing volumes are rectangular parallelepipeds. They correspond to the limiting case of a perspective frustum as the eye is moved infinitely far away and the field of view decreases appropriately. Another way to think of the orthogonal is that the geometry outside the box is clipped out, and then the geometry inside is projected parallel to the z axis onto a face parallel to the $x-y$ plane. Figure A-1 b) shows a box-shaped enclosure specified by an orthogonal transformation. Orthogonal transformation is defined by left plane(l), right plane(r), bottom plane(b), top plane(t), near plane(n), and far plane(f) as

$$Orthogonal(l,r,b,t,n,f) = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & -\frac{2}{f-n} & 0 \\ -\frac{r+l}{r-l} & -\frac{t+b}{t-b} & -\frac{f+n}{f-n} & 1 \end{bmatrix}. \qquad \text{(A.2)}$$

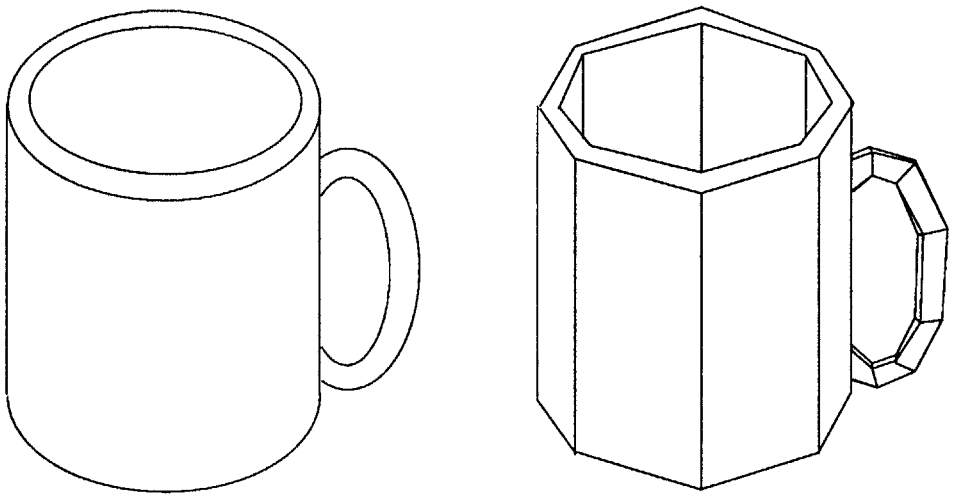
Appendix B

Three-dimensional solid geometry

B.1 Boundary Representations

Boundary representations of solids are used here because they are used in computer graphics. The main virtue of boundary representations lies in the ready availability of representation for faces, edges, and the relations between them; these are important for generating line drawings and graphic displays, for supporting graphic interaction, and for other purposes as well. This is why boundary representations have been used extensively in computer graphics.

A solid is represented by segmenting its boundary into a finite number of bounded subsets usually called "faces" or "patches," and representing each face by (for example) its bounding edges and vertices. One drawback of the representation is that it is not



a) b)

Figure B-1: A Curved Object and Its Patched Model.

intuitively clear what the faces of curved objects, such as a cup shown in Fig. B-1 a) are. However, any curved object can be approximated by an object with a finite set of small planar faces. Figure B-1 b) shows an object with many planar faces that corresponds to a sphere. Of course, this approximation does not preserve the topological properties of the original polyhedra. But, the topological properties do not play an important role in the context of collision detection and avoidance.

B.2 Edge-based Boundary Models

There are many varieties of boundary representations. They differ in the redundant data they provide to speed up specific algorithms. In most cases, redunt information takes the form of adjacency relations between faces, edges, and vertices. Such information sometimes is stored to avoid expensive searches that certain algorithms require.

In the paper, the hybrid edge model, an edge-based model is used to represent polyhedral objects. Drawing an object when it is represented by an edge-based model requires the same number of polygon drawings as the number of its faces. Some other methods such as one based upon faces with three edges (triangles) require different number of polygon drawings depending upon the shape of the faces.

Winged Edge Model

Baumgart developed this model in order to take advantage of edge-topological information. The structure of the model contains all the necessary topological information for any polyhedra.

```
struct edge {
     struct solid *s;
     struct edge *next[2];
     struct face *f[2];
     vertex *v[4];
};
```

The most significant problem with this model is that one data structure includes two different roles of edges. This problem is apparent when all the edges of one face have to be visited throught a face-traversal procedure. Face traversal is, by definition, a unidirectional operation, due to the orientability of the bounding surface. However, each edge is used for bounding two faces; therefore, it participates in two face traversals.

These traversal are opposite to each other, according to Mobius' law. Thus, each edge can be traversed in two opposite directions, depending upon the particular face being operated on. It is necessary, therefore, to check and determine the direction in which individual edges must be traversed at each traversal step, a process that considerably increases traversal cost.

Split Edge Model

In order to get rid of this problem, Eastman has developed the split edge model, where the two roles of edges have been separated.

```
struct edge {
     struct solid *s;
     struct edge *next;
     struct face *f;
     vertex *v;
     struct edge *opposite;
};
```

Winged edge model and split edge models are designed for representing edges that link adjacent faces. They are too specialized for representing 3-d polyhedra to be adequate when considered candidates for implementing a more general model, capable of representing solids, individual polygons, and even line segments. The reason for this deficiency, in part, lies in the uniformity of the requirements that are met by edges of

solids; they always bound pairs of adjacent faces and are always traversed bidirectionally. However, the directionality and adjacency requirements of polygons and lines are different.

Hybrid Edge Model

A solution to the problem of representing edges with dual purposes can be achived by separating the connectivity role of edges from their role as carriers of directionality information.

```
struct edge {
     struct solid *s;
     struct edge *next;
     struct segment *seg;
};
```

B.3 Spatial Set Operators

Spatial set operators, which are also known as shape operators, facilitate interactive generation of complex forms simply by direct additions and subtractions of volumes. From a mathematical point of view, the spatial set operators are the set operators of union, intersection, and difference with spatial sets of points as their operands.

Given that polyhedra are represented by a set of polygons, their set operations can be implemented by polygon set operators with certain extentions. For example, let's

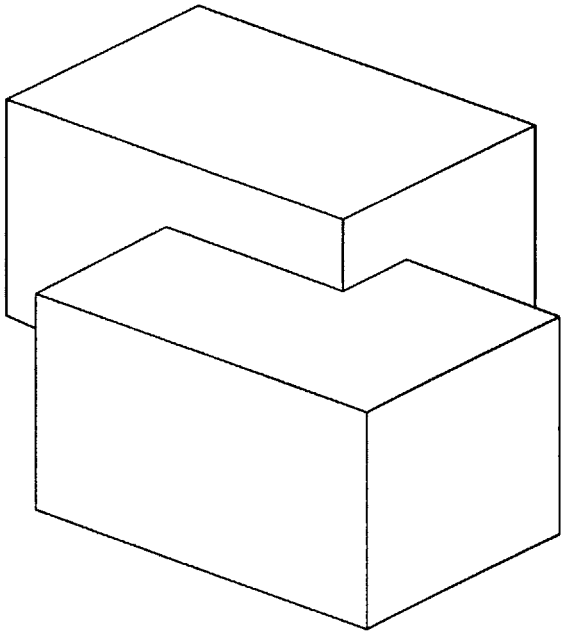


Figure B-2: Two Solids in Collision.

consider a situation where two polyhedra intersect each other, as shown in Fig. B-2. Each of the polyhedra faces (polygons) are partitioned by faces of the other polyheda. The next opereration is to decide whether each partitioned face is included in the resulting polyhedron. This process depends upon the operation involved, as follows:

- If the operation involved is, for example, a **UNION(A,B), A OR B,** the faces of polyhedron A that are *inside* polyhedron B, and the faces of polyhedron B that are *inside* polyheron A should participate in the resulting polyhedron.

- If the operation involved is, for example, an **INTERSECTION(A,B), A AND B,** the faces of polyhedron A that are *outside* polyhedron B, and the faces of polyhedron B that are *outside* polyheron A, should participate in the resulting polyhedron.

- If the operation involved is, for example, a **DIFFERENCE(A,B),** or **A - B,** the faces of polyhedron A that are *outside* polyhedron B, and the faces of polyhedron

B that are *inside* polyheron A should participate in the resulting polyhedron.

After the faces that should participate in the resulting polyhedron are identified, they should be glued together appropriately to form a valid polyhedron. Note that the orientations of the faces joined from polyhedron A and polyhedron B have to be inverted so that all the face normal vectors of the resulting polyheron point outward.

Appendix C

Miscellaneous

C.1 Coordinate Transformation

C.1.1 Position and Orientation

Position of point P relative to frame {A} can be represented by ${}^{A}P$, a vector relative to frame {A}.

$$ {}^{A}P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} . \tag{C.1} $$

In order to describe the orientation of a body, a coordinate system is attached to the body and this coordinate system is described relative to the reference system. One way to describe the body-attached coordinate system, {B}, is to write the unit vectors of its

three principal axes in terms of the coordinate system {A}. This can be represented by a single 3x3 matrix ${}^A_B R$, a *rotation matrix.*

$$ {}^A_B R = \left[\begin{array}{c|c|c} {}^A\hat{x}_B & {}^A\hat{y}_B & {}^A\hat{z}_B \end{array} \right] = \begin{bmatrix} \hat{x}_B \cdot \hat{x}_A & \hat{y}_B \cdot \hat{x}_A & \hat{z}_B \cdot \hat{x}_A \\ \hat{x}_B \cdot \hat{y}_A & \hat{y}_B \cdot \hat{y}_A & \hat{z}_B \cdot \hat{y}_A \\ \hat{x}_B \cdot \hat{z}_A & \hat{y}_B \cdot \hat{z}_A & \hat{z}_B \cdot \hat{z}_A \end{bmatrix} . \tag{C.2} $$

C.1.2 Coordinate Transformations

Point P relative to frame {A} can be expressed in terms of its position relative to frame {B} and the relative position and orientation of frame {A} and frame {B}:

$$ {}^A P = {}^A_B R \, {}^B P + {}^A_B P_{B,orig} . \tag{C.3} $$

Equation C.3 can be written in the matrix form using a 4x4 matrix, and 4x1 vectors:

$$ \left[\begin{array}{c} {}^A P \\ \hline 1 \end{array} \right] = \left[\begin{array}{c|c} {}^A_B R & {}^A P_{B,orig} \\ \hline 0 & 1 \end{array} \right] \left[\begin{array}{c} {}^B P \\ \hline 1 \end{array} \right] . \tag{C.4} $$

The 4x4 matrix in Eq. C.4 is called a *homogeneous transformation.*

C.1.3 Rotation

Rotation of frame {B} about vector ${}^A\hat{v}$ by an angle, θ, according to the right-hand rule can be represented by a rotation matrix $R_{\hat{v}}(\theta)$.

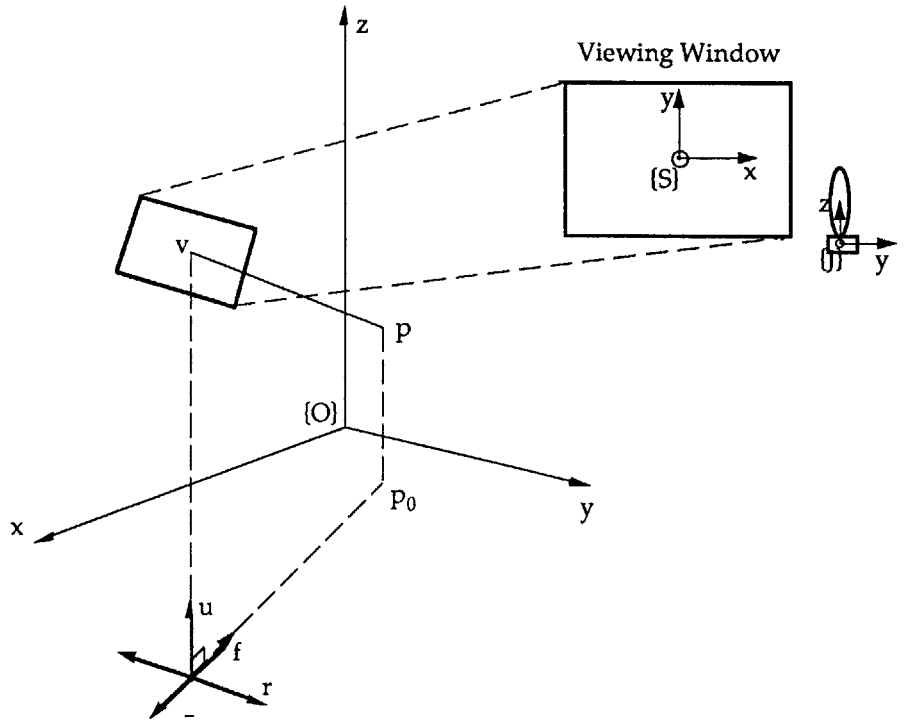


Figure C-1: A Joystick Coordinate and the Reference Coordinate.

$$R_{\hat{v}}(\theta) = \begin{bmatrix} v_x v_x a_\theta + c_\theta & v_x v_y a_\theta - v_z s_\theta & v_x v_z a_\theta + v_y s_\theta \\ v_x v_y a_\theta + v_z s_\theta & v_y v_y a_\theta + c_\theta & v_y v_z a_\theta - v_x s_\theta \\ v_x v_z a_\theta - v_y s_\theta & v_y v_z a_\theta + v_x s_\theta & v_z v_z a_\theta + c_\theta \end{bmatrix}, \tag{C.5}$$

where $c_\theta = \cos\theta$, $s_\theta = \sin\theta$, $a_\theta = 1 - \cos\theta$, and ${}^A\hat{v} = [v_x\ v_y\ v_z]^T$.

There are many other representations for rotation such as Euler angles and fixed X-Y-Z angles (Craig, 1986). The quaternion representations of rotation (Markley and Lerner, 1985; Pletincks, 1988) are efficient and well suited to solve rotation and orientation problems in optimization, computer graphics, and animation.

C.2 Joystick Control

It is desirable that input from a joystick during the manual mode be associated with the viewing position and orientation. For example, pulling a joystick out of the graphic screen would force the manipulator to move toward the viewer, resulting in a motion out of the screen. Figure C-1 shows the relationships among the viewing location (position and orientation), a reference frame, and a frame attached to the joystick. Point v is the viewing position and point p is the viewing target, and thus vector $p - v$ is the viewing direction. Then, a coordinate frame whose three unit vectors are in the direction of right, upward, and forward relative to the viewer, can be defined at location v_0 as shown in Fig. C-1. Vector u_f, u_r, and u_u represent *forward* vector, right vector, and *upward* vector of the viewer, respectively. Note that the *upward* vector is defined to be always pointing in the $+z$ direction. This is due to the fact that human operators always associate the upward direction with the vertical direction in the earth gravitational field.

C.2.1 Translation

Let's consider translation first and assume the translational vector input from the joystick to be ${}^{J}t = (t_x \ t_y \ t_z)^T$ in the joystick coordinate. Let's also assume that the joystick coordinate frame {J} can be represented relatively to the screen coordinate frame {S} by rotation matrix ${}^{S}_{J}R$. Then the joystick input vector can be expressed relative to the screen coordinate frame {S}:

$$^{S}t = {}^{S}_{J}R\,{}^{J}t\,. \tag{C.6}$$

As mentioned above, pulling the joystick out of the screen should result in motion in the direction of vector $-u_f$. Therefore, the z-component of the ^{S}t, i.e. $^{S}t_z$ should be associated with vector $-u_f$. Similarly, $^{S}t_x$ and the $^{S}t_y$ should be associated with vector u_r and u_u, respectively. Note again that pulling the joystick upward results in motion in z direction of the reference frame {O}. Thus,

$$^{O}t = u_r\,{}^{S}t_x + u_u\,{}^{S}t_y - u_f\,{}^{S}t_z = {}^{O}_{S}R\,{}^{S}t\,, \tag{C.7}$$

where

$$^{O}_{S}R = \left[\begin{array}{c|c|c} & & \\ u_r & u_u & -u_f \\ & & \end{array}\right]\,. \tag{C.8}$$

With Eq. C.6,

$$^{O}t = {}^{O}_{S}R\,{}^{S}_{J}R\,{}^{J}t\,. \tag{C.9}$$

If the operator moves point P located at $(p_x\ p_y\ p_z)$ in frame {O}, using the joystick, the new P should be located at

$$(p_x\ p_y\ p_z)^{T}_{new} = (p_x\ p_y\ p_z)^{T} + {}^{O}t\,. \tag{C.10}$$

C.2.2 Rotation

For rotation vector of a joystick input ${}^{J}r = ({}^{J}r_x \; {}^{J}r_y \; {}^{J}r_z)^T$ in the joystick coordinate, as in the translation case,

$$ {}^{O}r = {}^{O}_{S}R \, {}^{S}_{J}R \, {}^{J}r \quad . \tag{C.11} $$

${}^{O}r$ can be separated into its magnititude, θ, and its direction, r, an unit vector:

$$ \theta = \|{}^{O}r\| = \|{}^{S}r\| = \|{}^{J}r\| \, , \tag{C.12} $$

$$ r = \frac{{}^{O}r}{\|{}^{O}r\|} \quad . \tag{C.13} $$

A vector, P has to be rotated about r by θ. The rotation matrix that represents rotation about r by θ is given in Eq. C.5. Thus vector new P is

$$ (p_x \; p_y \; p_z)^T_{new} = R_{\hat{r}}(v) \; (p_x \; p_y \; p_z)^T \, . \tag{C.14} $$

Appendix D

Experiment Analysis and Data

D.1 *T*-test

Hypothesis testing using the t-test is presented with test results from the experiment, which are shown in Tables D.3, D.4, D.5, D.6, and D.7. The t-test is a standard way to test a null hypothesis regarding the mean of one sample set or the means of two *independent*[1] sample sets when the variances are *unknown*.

It will be assumed from now on that there are two sets of sample data. The typical null hypothesis for two sets of sample data is that the means of the two sample sets are equal. Among different alternative hypotheses, the alternative hypothesis that the means of the two sample sets are different, will be used here:

$$H_0 : \mu_1 = \mu_x, \tag{D.1}$$

[1]The paired t-test is used for two *dependent* sample sets.

versus (D.2)

$$H_1 : \mu_1 \neq \mu_y. \tag{D.3}$$

Under the assumption that the two variances are not equal[2], the t-value is defined as

$$t = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{s_x^2}{n} + \frac{s_y^2}{m}}}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of x and y, s_x and s_y are the sample standard deviations and n and m are the numbers of the sample data for the respective sets. Then, the t-value shows the Student t-distribution with ν degrees of freedom, where, if $a_x = s_x^2/n$ and $a_y = s_y^2/m$,

$$\nu = (a_x + a_y)^2 / \left(\frac{a_x^2}{n-1} + \frac{a_y^2}{m-1} \right).$$

The decision as to whether to accept or reject H_0 depends upon the confidence interval of the t-test. If the t-value exceeds the confidence interval of the Student t-distribution, H_0 should be rejected and H_1 should be taken. If we set a confidence interval of 95%, for example, there is 5% chance that we would reject H_0 when their means are equal. The numbers shown as *confidence* in the following Tables mean that we would allow *confidence* out of 1 where we reject H_0 when H_0 is true. For example, 0.05 in Confidence indicates that we can reject H_0 with 95% $((1 - 0.05) \times 100\%)$, meaning

[2]If the two variances are equal, the *pooled variance* s_p^2 is used:

$$s_p^2 = \frac{(n-1)s_x^2 + (m-1)s_y^2}{n + m - 2}.$$

that there *is* a significant difference between the two means.

D.2 Data from Experiment 1

Subjects	Environment Models	Avoidable Collisions at wrist	Avoidable Collisions st elbow	Unavoidable Collisions	Singularities*	Travel Distance
A	I	5	0	0	0	2336
	II	3	0	0	0	2238
	III	2	0	1	1	2644
	IV	1	1	0	0	2437
	V	0	2	0	0	2383
	VI	0	2	1	0	2569
B	I	4	0	0	0	3358
	II	3	0	0	0	3305
	III	1	0	1	0	3422
	IV	7	1	0	1	3568
	V	4	0	0	0	3535
	VI	6	0	0	0	3683
C	I	0	0	0	0	2816
	II	1	2	2	0	2601
	III	0	0	0	0	3247
	IV	1	0	0	0	3155
	V	0	0	0	1	3230
	VI	1	0	0	1	3099
D	I	0	0	0	0	2859
	II	0	0	0	0	3180
	III	0	0	0	0	3081
	IV	0	0	0	0	2995
	V	0	0	0	0	3045
	VI	4	0	1	1	3244
E	I	1	0	0	0	2410
	II	3	0	0	0	2395
	III	5	0	1	0	2643
	IV	1	0	0	0	2533
	V	3	1	0	2	2681
	VI	1	1	0	0	2597
F	I	0	0	0	1	2589
	II	0	0	0	0	2674
	III	0	0	0	0	2677
	IV	1	0	0	0	2715
	V	4	0	0	0	2783
	VI	0	0	0	0	3281

* (Singularity at the workspace boundaries.)

Table D.1: Data from Experiment 1.

D.3 Data from Experiment 2

Variables	Statistics	Field Of View		
		10°	20°	30°
$\Delta X/Z_0$	mean($\times 10^3$)	0.108	-0.166	0.050
	std. dev.($\times 10^3$)	0.184	0.329	0.502
	observations	20	20	20
$\Delta Y/Z_0$	mean($\times 10^3$)	-0.031	-0.147	0.208
	std. dev.($\times 10^3$)	0.373	0.530	0.646
	observations	20	20	20
$\Delta Z/Z_0$	mean($\times 10^3$)	-0.555	1.365	-5.777
	std. dev.($\times 10^3$)	6.444	9.205	12.036
	observations	20	20	20
$\Delta\Theta$	mean(degrees)	0.885	1.195	1.940
	std. dev.	0.598	0.620	0.986
	observations	20	20	20

Table D.2: Effects of Field-of-View for the Rectangular Solid.

2 Fields-Of-View on Comparision	Variables	*T*-Test		
		t-static	d.o.f.	significance
10° vs. 20°	$\Delta X/Z_0$	3.246	38	0.002
	$\Delta Y/Z_0$	0.800	38	0.429
	$\Delta Z/Z_0$	-0.764	38	0.449
	$\Delta\Theta$	-1.610	38	0.116
20° vs. 30°	$\Delta X/Z_0$	-1.612	38	0.115
	$\Delta Y/Z_0$	-1.901	38	0.065
	$\Delta Z/Z_0$	2.108	38	0.042
	$\Delta\Theta$	-2.860	38	0.007
10° vs. 30°	$\Delta X/Z_0$	0.479	38	0.635
	$\Delta Y/Z_0$	-1.434	38	0.160
	$\Delta Z/Z_0$	1.711	38	0.095
	$\Delta\Theta$	-4.092	38	0.000

Table D.3: *T*-Test for Different Fields-of-View.

Variables	Normal View		50% Occluded View		T-static	D.O.F.	Significance
	mean	std. dev.	mean	std. dev.			
$\Delta X/Z_0$	0.108*	0.184*	0.039*	0.443*	0.641	38	0.525
$\Delta Y/Z_0$	-0.031*	0.373*	0.175*	0.921*	-0.928	38	0.359
$\Delta Z/Z_0$	-0.555*	6.444*	-2.834*	12.720*	0.715	38	0.479
$\Delta\Theta$	0.885	0.598	1.950	1.223	-3.498	38	0.001

* (Premultiplied by 1000.)

Table D.4: Effects of Occlusion for the Rectangular Solid.

Variables	Normal View		50% Occluded View		T-static	D.O.F.	Significance
	mean	std. dev.	mean	std. dev.			
$\Delta X/Z_0$	0.197*	0.466*	-0.302*	1.004*	2.127	38	0.040
$\Delta Y/Z_0$	0.033*	0.414*	0.541*	1.423*	-1.857	38	0.071
$\Delta Z/Z_0$	6.254*	9.213*	8.791*	13.423*	-0.774	38	0.444
$\Delta\Theta$	2.195	1.146	3.900	2.107	-3.179	38	0.003

* (Premultiplied by 1000.)

Table D.5: Effects of Occlusion for the Cylindrical Solid.

Variables	Rectangular Solid		Cylindrical Solid		T-static	D.O.F.	Significance
	mean	std. dev.	mean	std. dev.			
$\Delta X/Z_0$	0.108*	0.184*	0.197*	0.466*	-0.802	38	0.428
$\Delta Y/Z_0$	-0.031*	0.373*	0.033*	0.414*	-0.518	38	0.608
$\Delta Z/Z_0$	-0.555*	6.444*	6.254*	9.213*	-2.709	38	0.010
$\Delta\Theta$	0.885	0.598	2.195	1.146	-4.532	38	0.000

* (Premultiplied by 1000.)

Table D.6: Difference Betweem the Rectangular Solid and the Cylindrical Solid.

Variables	Rectangular Solid		Cylindrical Solid		T-static	D.O.F.	Significance
	mean	std. dev.	mean	std. dev.			
$\Delta X/Z_0$	0.039*	0.443*	-0.302*	1.004*	1.388	38	0.173
$\Delta Y/Z_0$	0.175*	0.921*	0.541*	1.423*	-0.966	38	0.340
$\Delta Z/Z_0$	-2.834*	12.720*	8.791*	13.423*	-2.811	38	0.008
$\Delta\Theta$	3.920	2.089	3.900	2.1076	0.030	38	0.976

* (Premultiplied by 1000.)

Table D.7: Difference Betweem the 50% Occluded Rectangular Solid and the 50% Occluded Cylindrical Solid.

D.4 Data from Experiment 3

Subjects	Time Spent for Introducing Objects(seconds)			
	Environment Model I		Environment Model II	
	Precise Model	Crude Model	Precise Model	Crude Model
A	2347	1401	2798	1080
B	3667	1253	3688	1306
C	2886	1789	2785	1346
D	1145	889	985	687

Table D.8: Data for Introducing 6 Objects into a World Model.

Modes	Subjects	Number of Intermediate Points	Number of Impending Collisions	Total Travel Distance	Time Spent for Operation (seconds)
Collision Detection Only with a Perfect World Model	A	38	4	3233.5	1347
	B	61	6	3283.6	1339
	C	34	2	3089.8	1083
	D	33	2	2739.1	831
Collision Avoidance with a Perfect World Model	A	25	2	2704.2	866
	B	37	4	3258.5	1032
	C	26	6	3106.2	898
	D	27	7	2603.5	732
Collision Avoidance with a Crude World Model	A	39	5	3507.0	1539
	B	43	4	3411.6	1136
	C	51	6	3352.1	1324
	D	31	7	2792.8	1065

Table D.9: Task Performance of Stacking Objects in Environment Model I.

Modes	Subjects	Number of Intermediate Points	Number of Impending Collisions	Total Travel Distance	Time Spent for Operation(sec)
Collision Detection Only with a Perfect World Model	A	31	2	2690.0	969
	B	33	1	2547.7	1036
	C	38	3	2719.3	1102
	D	25	5	2065.4	593
Collision Avoidance with a Perfect World Model	A	25	0	2685.4	888
	B	31	2	2102.4	691
	C	19	1	2305.0	701
	D	23	1	2036.5	493
Collision Avoidance with a Crude World Model	A	23	6	2436.0	860
	B	30	2	2277.0	975
	C	23	2	2328.3	878
	D	16	1	2054.0	850

Table D.10: Task Performance of Stacking Objects in Environment Model II.

System Comparison	Variables	Model I		Model II		Degree Of Freedom
		t-static	significance	t-static	significance	
PD vs. PA	int. points	1.783	0.125	1.975	0.096	6
	time	1.952	0.099	1.660	0.148	6
PA vs. CA	int. points	-2.447	0.050	0.395	0.706	6
	time	-3.129	0.020	-2.307	0.061	6

Table D.11: T-Test for Different Environments.

Bibliography

Ambler, A. P. and Popplestone, R. J. (1975). Inferring the positions of bodies from specified spatial relationships. *Artificial Intelligence*, pages 157–174.

Asada, H. and Slotine, J. (1986). *Robot Analysis and Control*. J. Wiley.

Bejczy, A. K. (1975). Distribution of control decisions in remote manipulation. In *IEEE Conference on Decision Control*, Houston.

Besl, P. J. and Jain, R. C. (1985). Three-dimensional object recognition. *Computing Surveys*, 17(1).

Bonner, S. and Shin, K. G. (1982). A comparative study of robot languages. *Computer*.

Bowyer, A. and Woodwark, J. (1983). *A Programmer's Geometry*. Butterworths.

Boyce, J. (1979). Interference detection among solids and surfaces. *Communications of the ACM*, 22(1).

Brady, J. M., Hollerbach, J. M., and Johnson, T. M. (1983). *Robot Motion: Palnning and Control*. MIT Press, Cambridge, MA.

Brady, M. (1984). Artificial intelligence and robotics. AI Memo 756, MIT AI Lab.

Brooks, R. A. (1982a). Symbolic error analysis and robot programming. *International Journal of Robotics Research*, 1(4).

Brooks, R. and Lozano-Pérez, T. (1982b). A subdivision algorithm in configuration space for findpath with rotation. AI Memo 684, MIT AI Lab.

Brooks, R. (1983a). Planning collision free motions for pick and place operations. AI Memo 725, MIT AI Lab.

Brooks, R. A. (1983b). Solving the find-path problem by good representation of free space. *IEEE Transactions on Systems, Man, and Cybernatics*, SMC-13(3).

Brooks, T. L. (1979). Superman: A system for supervisory manipulation and the study of human/computer interactions. Master's thesis, MIT. Superman.

Canny, J. F. (1987). *The Complexity of Robot Motion Planning.* PhD thesis, Department of Electrical Engineering and Computer Science, MIT.

Chapanis, A. (1959). *Research Techniques in Human Engineering.* Johns Hopkins Press, Baltimore.

Craig, J. J. (1986). *Introduction to Robotics: Mechanics & Control.* Addison-Wesley.

Das, H. (1989). *Kinematic Control and Visual Display of Redundant Teleoperators.* PhD thesis, Department of Mechanical Engineering, MIT.

Donald, B. R. (1984). Motion planning with six degrees of freedom. AI Memo 791, MIT AI Lab.

Durrant-Whyte, H. (1987). Uncertain geometry in robotics. *IEEE International Conference on Robotics and Automation,* pages 851–856.

Faugeras, O. D. and Hebert, M. (1983). A 3-d recognition and positioning algorithm using geometrical matching between primitive surfaces. In *8th International Conference on Artificial Intelligence,* pages 996–1002.

Faverjon, B. (1984). Obstacle avoidance using an octree in the configuration space of a manipulator. In *IEEE Conference on Robotics and Automation,* Atlanta, Georgia.

Faverjon, B. (1989). Hierarchical object models for efficient anti-collision algorithms. In *IEEE Conference on Robotics and Automation.*

Ferrell, W. (1965). Remote manipulation with transmission delay. *IEEE Transactions of Human Factors in Electronics,* HFE-6:24–32.

Ferrell, W and Sheridan, T. B. (1967). Supervisory control of remote manipulation. *IEEE Spectrum,* 4(10), 81–88.

Fikes, R. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence,* 2:189–208.

Fisher, R. B. (1983). Using surfaces and object models to recognize partially obscured objects. In *8th International Conference on Artificial Intelligence,* pages 989–995.

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

Grimson, W. E. L. and Lozano-Pérez, T. (1984). Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research,* 3(3).

Haralick, R., Joo, H., Lee, C., Zhuang, X., Vaidya, V., and Kim, M. (1989). Pose estimation from corresponding point data. In Freeman, H., editor, *Machine Vision for Inspection and Measurement,* pages 1–84. Academic Press.

Harrington, S. (1983). *Computer Graphics: A programming Approach.* McGraw-Hill Book Inc.

Hayward, V. and Paul, R. P. (1983). Robot manipulator control using the 'c' language under unix. *IEEE Workshop on Languages for Automation.*

Herskovits, A. (1986). *Language and Spatial Cognition.* Cambridge University Press, Cambridge.

Hoffman, R. M. and Hebert, M. (1986). Applications of high-performance graphics workstations in robotic simulation. In *16th International Symposium on Industrial Robots,* Brussels, Belgium.

Horn, B. K. (1987). *Robot Vision.* MIT Press, Cambridge, MA.

Kak, A. C. (1985). Depth perception for robots. In Nof, S., editor, *Handbook of Industrial Robotics,* pages 272–319. J. Wiley.

Kalay, Y. (1989). *Modeling Objects and Environments.* John Wiley & Sons, New York.

Khatib, O. (1986). Real-time obstacle avoidance for manipulator and mobile robots. *International Journal of Robotics Research,* 5(1).

Khosla, P. and Volpe, R. (1988). Superquadric artificial potentials for obstacle avoidance and approach. In *IEEE Conference on Robotics and Automation,* Philadelphia, Pennsylvania.

Kim, W. S., Tendick, F., and Stark, L. (1987). Spatial displays and spatial instruments. In *NASA Conference,* number Publication 10032, Asilomar, California.

Larsen, R. J. and Marx, M. L. (1986). *An Introduction to Mathematical Statistics and Its Applications.* Prentice Hall.

Latombe, J. and Mazer, E. (1981). LM: A high-level programming language for controlling assembly robots. In *11th International Symposium on Industrial Robots,* pages 683–690, Tokyo.

Latombe, J.-C. (1991). *Robot Motion Planning.* Kluwer Academic Publishers.

Lieberman, L. I. and Wesley, M. A. (1977). Autopass: An automatic programming system for computer controlled mechanical assembly. *IBM Journal Res. Dev,* 21(4):321–333.

Lozano-Pérez, T. and Winston, P. H. (1977). LAMA: A language for automatic mechanical assembly. In *5th International Joint Conference on Artificial Intelligence,* pages 710–716, MIT.

Lozano-Pérez, T. and Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM,* 22(10).

Lozano-Pérez, T. (1981). Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-11(10):681–689.

Lozano-Pérez, T. (1982). Robot programming. AI Memo 698a, MIT AI Lab.

Lozano-Pérez, T. and Brooks, R. (1985). Task-level manipulator programming. In Nof, S., editor, *Handbook of Industrial Robotics*, pages 404–418. J. Wiley.

Lozano-Pérez, T. (1986). A simple motion planning algorithm for general robot manipulator. AI Memo 896, MIT AI Lab.

Cheung, E. and Lumelsky, V. (1988). Motion Planning for Robot Arm Manipulators with Proximity Sensing. In *IEEE Conference on Robotics and Automation*, Philadelphia, Pennsylvania.

Cheung, E. and Lumelsky, V. (1990). Motion Planning for a Whole-Sensitive Robot Arm Manipulator. In *IEEE Conference on Robotics and Automation*, Cincinnati, Ohio.

Markley, F. L. and Lerner, G. M. (1985). Three-axis attitude determination methods. In Wertz, J. R., editor, *Spacecraft Attitude Determination and Control*, pages 410–435. Reidel.

McCarthy, J. M. (1990). *An Introduction to Theoretical Kinematics*. MIT Press, Cambridge, Mass.

Myers, J. and Agin, G. (1982). A supervisory collision avoidance system for robot controllers. *Robotics Research and Advanced Applications*.

Park, J. H. (1991a). Collision avoidance of robot manipulator using heuristics and computer graphics. In *Center for Intelligent Machine Systems Mini-Conference*, MIT, Cambridge, MA.

Park, J. H. and Sheridan, T. B. (1991b). Supervisory teleoperation control system using computer graphics. In *IEEE Conference on Robotics and Automation*, Sacramento, California.

Paul, R. (1981). *Robot Manipulators: Mathematics, Programming, and Control*. MIT Press, Cambridge, MA.

Pletincks, D. (1988). The use of quaternions for animation, modelling and rendering. In Magnenat-Thalmann, N. and Thalmann, D., editors, *New Trends in Computer Graphics: Proc. of CG Internation '88*. Springer-Verlag, New York.

Popplestone, R. J., Ambler, A. P., and Bellos, I. (1978). RAPT: A language for describing assemblies. *Industrial Robot 5*.

Popplestone, R. J., Ambler, A. P., and Bellos, I. (1980). An interpreter for a language for describing assemblies. *Artificial Intelligence*, pages 79–106.

Reif, J. H. (1987). Complexity of the generalized mover's problem. In Schwartz et al., editor, *Planning, Geometry, and Complexity of Robot Motion*, pages 267–281. Ablex Publishing Corp.

Requicha, A. (1980). Representations for rigid solids: Theory, methods, and systems. *Computing Surveys*, 12(4).

Roberts, L. (1965). Machine perception of three-dimensional solids. In Tippett, J., editor, *Optical and Electro-optical Information Processing*, pages 159–197. MIT Press, Cambridge, MA.

Sacerdoti, E. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135.

Sacerdoti, E. (1975). The nonlinear nature of plans. *Advanced Papers of the 4th IJCAI*, pages 206–214.

Sato, T. and Hirai, S. (1985). Language-aided robotics teleoperation system for advanced teleoperation. In *International Conference on Advanced Robotics*, Tokyo, Japan.

Schwartz, J. and Sharir, M. (1982). On the piano movers problem, II: General techniques for computing topological properties of real algebraic manifolds. report 41, Courant Institute of Mathematical Sciences.

Sheridan, T. B. and Ferrell, W. R. (1974). *Man-Machine Systems: Information, Control and Decision Models of Human Performance.* MIT Press, Cambridge, MA.

Sheridan, T. (1976). Toward a general model of supervisory control. In Sheridan, T. and Johannsen, G., editors, *Monitoring Behavior and Supervisory Control.* Plenum Press, New York.

Sheridan, T. and Verplank, W. (1978). Human and computer control of undersea teleoperators. Prepared for office of naval research, Man-Machine Systems Lab.

Sheridan, T. (1983). Supervisory control of remote manipulators, vehicles and dynamic processes: Experiments in command and display aiding. Technical report, MIT Man-Machine Systems Lab.

Sheridan, T. B. (1986). Human supervisory control of robot systems. In *IEEE Conference on Robotics and Automation*, San Francisco, California.

Sheridan, T. B. (1987). Supervisory control. In Salvendy, G., editor, *Handbook of Human Factors*, pages 1243–1268. J. Wiley.

SiliconGraphics (1989). *Graphics Library Programming Guide.* SiliconGraphics.

Smith, R. (1985). Fast robot collision detection using graphic hardware. *The 1st IFAC Symposium on Robot Control.*

Smith, R. (1987). On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68.

Smith, R., Self, M., and Cheeseman, P. (1987). A stochastic map for uncertain spatial relationship. *The 4th International Symposium on Robotics Research*, pages 467–474.

Sun, K. and Lumelsky, V. (1989). Motion planning with uncertainty for a 3d cartesian robot arm. *The 5th International Symposium on Robotics Research*, pages 417–424.

Takase, K. and Paul, R. (1981). A structed approach to robot programming and teaching. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(4).

Udupa, S. M. (1977). Collision detection and avoidance in computer controlled manipulators. In *5th International Joint Conference on Artificial Intelligence.*

Whitesides, S. H. (1985). Computational Geometry and Motion Planning. In Toussaint G. T., editor, *Computational Geometry*, pages 377–427. North-Holland.

Yoerger, D. R. (1982). *Supervisory Control of Underwater Telemanipulators: Design and Experiment*. PhD thesis, MIT.