Final Report to Goddard Space Flight Center on

Tele-Autonomous Control Involving Contact

R. A. Volz
Lejun Shao
Lynn Conway
M. M. Walker

Contract Number NAG5-1024

Robotics Research Laboratory
Electrical Engineering and Computer Science Dept.
College of Engineering
The University of Michigan
Ann Arbor, Michigan 48109-2110

Texas A&M University
Computer Science Department
College Station, Texas 77843

March 1, 1990

ABSTRACT

OBJECT LOCALIZATION AND ITS APPLICATIONS

IN TELE-AUTONOMOUS SYSTEMS

by

Lejun Shao

Chairman: Richard A. Volz, Michael W. Walker

Object localization and its applications in tele-autonomous systems are studied in this thesis.

The thesis first gives a thorough investigation of the object localization problem and then presents two object localization algorithms together with the methods of extracting several important types of object features. The first algorithm is based on *line-segment to line-segment* matching. Line range sensors are used to extract line-segment features from an object, the features may be boundary edges of planar surfaces, the axes of cylindrical surfaces, conic surfaces, or other types of surfaces of revolution. The extracted features are matched to corresponding model features to compute the location of the object. The second algorithm is more general. The inputs of the algorithm are not limited only to

the line features. Featured points (*point-to-point* matching) and featured unit direction vectors (*vector-to-vector* matching) can also be used as the inputs of the algorithm, and there is no upper limit on the number of the features inputed. The algorithm will allow the use of redundant features to find a better solution. The algorithm uses dual number quaternions to represent the position and orientation of an object and uses the least squares optimization method to find an optimal solution for the object's location. The advantage of using this representation is that the method solves for the location estimation by minimizing a single cost function associated with the sum of the orientation and position errors and thus has a better performance on the estimation, both in accuracy and in speed, than that of other similar algorithms.

The thesis discusses the difficulties when an operator is controlling a remote robot to perform manipulation tasks. The main problems facing the operator are time-delays on the signal transmission and the uncertainties of the remote environment. How object localization techniques can be used together with other techniques such as predictor display and time desynchronization to help to overcome these difficulties are then discussed. The thesis discusses two cases where object localization can help: 1) the case where direct manual control is used to perform a tele-manipulation task; 2) the case where the remote system has certain degree of automation ability.

ACKNOWLEDGEMENTS

First I would like to give my special thanks to Kurt Lloyd who helped me to find my advisor Volz, a wonderful choice. In four years of association with Professor Volz, he always puts his students first and tries every opportunity to let his students contribute more to the society. He was always very patient, willing to listen to my ideas and help direct my thoughts. He was always full of energy, knew how to deal with different people and treated them well. I have a great deal of respect for him and would like to express my sincere thanks to him for his invaluable help.

I was so fortunate to have selected an excellent thesis committee. All of them are outstanding in their research fields. The committee members are Richard Volz, Lynn Conway, Michael Walker, Robert Howe and Terry Weymouth. I also was very fortunate to have worked with some of the members in the committee and learned a lot from them. I would like to give special thanks to Michael Walker for his expert help with dual quaternion concept and other precious ideas and suggestions.

I would like to thank to my wife Yurang for her sacrifice and encouragement. I owe a lot to my children who gave me great enjoyment for the past several years. I do not know how to find a proper word to express my thanks to my parents who devoted their whole love to me and shaped me into the person I am today.

I also wish to thank Al Dobryden for his support in settling down my experiments and

in solving for my many software problems. Finally I would like to thank the support of the research grants from NASA Goddard No. NAG5-1024 and support from CICE graduate fellowships and other industrial gifts.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

Appendix

CHAPTER 1

Introduction

1.1 Background

Until now, *direct manual control* or *teleoperation* has been the dominant technique to telecontrol the operation of a remote robot, in which all the intellectual inputs are from the human operator in a control center distant from the workspace. The techniques have been mainly applied to underwater operations and nuclear industry in circumstances hazardous to human beings. There are many remote manipulators being used in nuclear industry today [41].

Recently, research scientists have put great efforts to develop telerobotic technology for space applications. NASA, in particular, has carried out considerable work in developing telerobotic systems over recent years [4] [55], [65], [67], [81]. The direct motivations have been the Congress's legislation on the construction of a permanent Space Station which is scheduled to be put into orbit in the middle of the 1990's; and the legislation that no less than ten percent of the total Space Station costs must be used in developing automation and robotics technology [27]. For this purpose, [84] has described an early view of NASA's automation and robotics technology development program. The objective of the program is to improve the productivity, capability, autonomy, and safety of future space missions.

Other factors also contributed to the increasing research activity in this area such as the strong desire to replace humans for the performance of hazardous, strenuous, or repetitive tasks and so on. Several important types of tasks have been identified as the candidates for telerobot applications, including: Space Station construction, maintenance on the Space Station, remote satellite servicing *etc.* [84]. To achieve success in all the work just mentioned, however, the present technology must be evolved from the current level to a much higher level. According to Iyengar and Kashyap [60] the evolution of autonomy can be described as the following four different levels with increasing intelligence and difficulty:

- teleoperation – direct manual control of the remote manipulator;
- telesensing, or telepresence – human operator will feel physically present at the remote site by receiving remote sensing information.
- telerobotics, or supervisory control – the human operator acts merely as a supervisor, intermittently communicating to the remote system, giving suggestions, changing orders and *etc.*
- intelligent autonomous robots – the human operator supplies a single high-level command, in response to which the robot does all the necessary tasks.

The technologies needed to reach the goal of intelligent autonomous robots have been identified. Among other things, such as real-time path planning, the design of dexterous manipulator, adaptive control, *etc.*, the vision capability is definitely a very important one. The research under this category includes the recognition, localization and the inspection of objects, optimal sensor configuration, e.g., sensor mounting requirements, hierarchical sensing with varying degree of coarseness, integration of sensing data, the development of high-speed and high-quality sensing hardware and software, *etc.*. The object localization problem is the topic of this thesis.

Object localization refers to the problem of determining the location (position and orientation) of a known object in three dimensional space. Before a manipulator can do any operation on an object, either grasping the object, or inserting the object into

another one, the manipulator must know the location of the object(s). In an industrial environment, especially in large scale batch manufacturing environment, the manipulators are pre-programmed. A set of fixture and jigs are provided to guarantee the exact location of the object or to limit the uncertainty on the object's location to a tolerable degree so that the manipulator's operations would not fail. In space applications, however, the situation is totally different. Most of the tasks are small scale and thus require the flexibility of the remote manipulator. One of the flexibilities the remote manipulator should have is the ability to locate an object and adjust its position accordingly.

1.2 Research Goal

Automatic object localization has attracted increasing attention recently in both computer vision and robotics communities because researchers and scientist in these fields have found the important role this technique will play in developing autonomous intelligent machines. On the other hand, as an independent research topic, only few research work on object localization can be found in literature. Often the object localization problem is combined into object recognition research and the emphasis of the research in most cases is on object recognition. But object localization has many issues of its own and the recognition-localization is only one strategy toward this problem as will be seen in Chapter 3.

The objective of the thesis is to investigate methods of object localization, i.e., given a known object, find its position and orientation. The methods explored are intended to be used in tele-autonomous applications, especially in space programs such as Space Station construction, maintenance and repairing. But, the results obtained should be applicable to other related areas such as flexible robotic assembly, inspection and *etc.*. The methods

explored in this thesis should meet the following requirements:

1. They should be fast. A complete localization process should be finished in seconds;
2. They are accurate so the results of localization can be used in following telerobotic operations reliably;
3. They can locate most types of the industrial parts so the methods developed in this thesis will have real practical meaning.

Based on these requirements, the thesis addresses the following problems:

1. What degree of accuracy is required in tele-manipulation tasks? Furthermore, if the accuracy requirement for a particular tele-manipulation task is given and other resources of potential position errors during the execution of the task can also be identified, what degree of object localization accuracy should be provided by the sensing system?
2. What types of range sensing techniques are good for fast localization?
3. Are there better algorithms or mathematical methods which would further increase the speed and efficiency of the computations needed in localization process? Can the same computation be carried out by using closed form formula instead of iterative methods without losing accuracy?
4. Is the algorithm sensitive to noise?

Two localization algorithms are proposed in the thesis. The extraction of certain types of features such as line-segment features is also described. All the algorithms and methods explored are directed toward the goal of fast, accurate and robust object localization.

1.3 The Overall Organization of the Thesis

This thesis discusses the issues of object localization and its applications in a tele-autonomous robot.

Chapter 2 gives a detailed discussion of the applications of object localization technique to tele-autonomous systems. It begins by a brief review of the development of tele-autonomous system concepts and implementation. The difficulties in performing tele-manipulation tasks are then described, which include time-delay problem and the various uncertainties in the remote environment. How object localization techniques can be used to help the successful completion of these tele-manipulation tasks are outlined. Two situations are discussed: 1) the situations where the tele-manipulation tasks are performed manually, e.g., local operator directly controls all the operations of a remote manipulator; 2) the situations of limited autonomous control where the remote system has a certain degree of automation ability.

A thorough investigation of object localization problems is presented in Chapter 3. This includes a formulation of the object localization problem; a description of the overall organization of object localization systems; a classification of object localization strategies and a discussion of important issues related to the quality of object location determination such as ranging techniques, speed and accuracy consideration, types of locatable objects, mathematical formulation methods and *etc.* Arguments for using a line-based range sensing system for a fast and accurate object localization are presented.

Chapters 4 and 5 present a detailed discussion of object localization techniques. Chapter 4 deals with the line-segment matching based localization method. It consists of three parts. The first part discusses the process of using line range data to extract certain types

of features. This includes the extraction of edge parameters from a planar surface, the extraction of axis parameters from a surface of revolution and the extraction of the center point from a sphere. The performance of the extraction algorithms with data corrupted by noise are discussed. The second part presents the mathematical formulation of the line-segment matching localization algorithm, which includes the cases that the two line-segments are in the same plane and they are in different planes, and a discussion of the sensitivity analysis of the algorithm.

Chapter 5 discusses a more general case where, in addition to line-segment features, other features such as featured points, surface normals, *etc.* are also available and the number of available features might be more than the minimum required. An optimal localization algorithm using dual quaternions is presented. The advantages of using this algorithm, compared with other similar algorithms, are its high accuracy, fast execution speed and flexibility.

Experimental results both from computer simulation and from real range data are provided in the last part of each of the two chapters, which confirmed these algorithms' high-speed and high-accuracy performance.

Chapter 6 describes techniques for computing the location of a line sensor relative to its mounting position, particularly when the sensor is mounted somewhere on a joint of a tele-robot. This refers to the sensor calibration problem. Both internal calibration and external calibration are discussed. For the internal calibration, we only deal with the calibration method of the line sensor which is used in our experiments. The external calibration methods are more general and the solutions are given for three different situations: 1) an object used for the calibration purpose can be accurately placed in a pre-determined position and the sensor can locate the object from a single frame of measurement; 2) the

object's position is unknown but the sensor is able to locate it by using a single frame of measurement; 3) the object's location is unknown and the sensor can only locate certain features from a single frame of measurement.

Chapter 7 contains a discussion of what additional work has to be done in order for the technique discussed in previous chapters to be used in practical tele-autonomous systems. The potential areas of future research are identified with a conclusion being given in the last.

Final Report to Goddard Space Flight Center on

Tele-Autonomous Control Involving Contact

R. A. Volz
Lejun Shao
Lynn Conway
M. M. Walker

Contract Number NAG5-1024

Robotics Research Laboratory
Electrical Engineering and Computer Science Dept.
College of Engineering
The University of Michigan
Ann Arbor, Michigan 48109-2110

Texas A&M University
Computer Science Department
College Station, Texas 77843

March 1, 1990

CHAPTER 2

APPLICATIONS OF OBJECT LOCALIZATION IN TELE-AUTONOMOUS SYSTEMS

In this chapter, we present a discussion of the problems of tele-controlling the movement of a remote robot when the movement involves contact with objects and when the control involves a significant time delay. We also examine how object localization techniques can be used to overcome some of these difficulties.

2.1 Time Delay and the Tele-Autonomous System Concept

2.1.1 Time delay problem and predictor display

As described in the previous chapter, the growing demand for applying tele-manipulation technology in space applications has resulted in considerable work in telerobotic research. While pure teleoperation techniques have been successfully used in undersea operations and nuclear industry for years, researchers have found them difficult to apply to the space environment due to transmission or telemetry time delays. The signal transmission delay affects the operator's use of the remote manipulator. When direct telecontrol is employed, the operator's strategy of performing a given task usually consists of a sequence of open-loop moves, e.g., each move is followed by a wait of one roundtrip delay time to check the

feedback signal when it arrives. In a space environment, when the control signals are from earth, the time delay can be quite significant, making it very difficult, if not impossible, for the operator to control the remote robot accurately and efficiently.

The time-delay problem exists not only in tele-manipulation tasks where an operator controls a remote mechanical device or a telerobot to perform a manipulation task, but also in many other remote control tasks, such as the tasks of guiding and controlling certain kind of remote vehicles – spacecraft, aircraft, ships or cars. Researchers and scientists have approached this problem in many ways most of which are based on some forms of prediction with respect to the time-delayed signal.

An early attempt to deal with the time-delay problem in tele-manipulation was made by Ferrell and Sheridan [40]. A controller was built at the local site, which had a structure similar to the remote manipulator arm. The controller was worn by the operator on his shoulder so that he could move it to maintain its orientation relative to the remote arm. The operator drove the controller with no time delays present, with the control signals being sent to the remote site as well. The local controller might be more properly called the predictive manipulator because the local controller was used to control the future movements of the remote manipulator.

Bernotat and Widlok described the use of prediction display to guide and control remote vehicles [11]. In their work, the term "prediction display" was defined as

"A display which gives information about the future status of a value."

They found that predictive display of one or more system variables such as position, speed and acceleration in the form of points, curves, *etc.* was very useful in permitting the pilot to achieve smooth, stable control of the remote vehicles. Kelley used a similar approach in submarine operation and found substantial improvement in the performance

through prediction display [71].

Another use of predictive display was for the optimal control of launch vehicles during boost [44]. A manual guidance system which combines the use of predictive displays and optimal control theory enabled the operator to generate continually a predicted fuel-optimal trajectory.

A significant progress on the time-delay problem has been achieved recently by Noyes and Sheridan through a method called *"predictor display"* [89] or *"forward simulation"* [28]. In this scheme the operator directly drives a local simulator of the telerobot rather than the real telerobot with the control signals being sent, in parallel, to the simulator and the remote robot. The graphical display of the undelayed telerobot simulator is then overlaid onto the video pictures returning from the remote site. Obviously, the simulation at the local site operates in a predictive fashion, or in a forward simulation mode. That is, the simulation the operator controls is what the remote manipulator will be doing several seconds in the future. This method allows an operator to move the manipulator and immediately see the result of the action without waiting for the return video signals. Experiments have shown that if high quality models are available, repositioning tasks can be done much faster than without using forward simulation [54] [28]. An example of the predictive display system concept is presented in Fig.2.1. The wire frame telerobot is the forward simulation of the remote robot, which directly responds to operator control, and the solid frame represents the time-delayed image of the real telerobot.

From above description, we know that the "predictor display" is an extension of the predictive manipulator idea. In predictive manipulator control, the operator controls a local manipulator which has a structure similar to the remote manipulator; in "predictor display" the operator controls a graphical simulation of the real manipulator. Thus, the

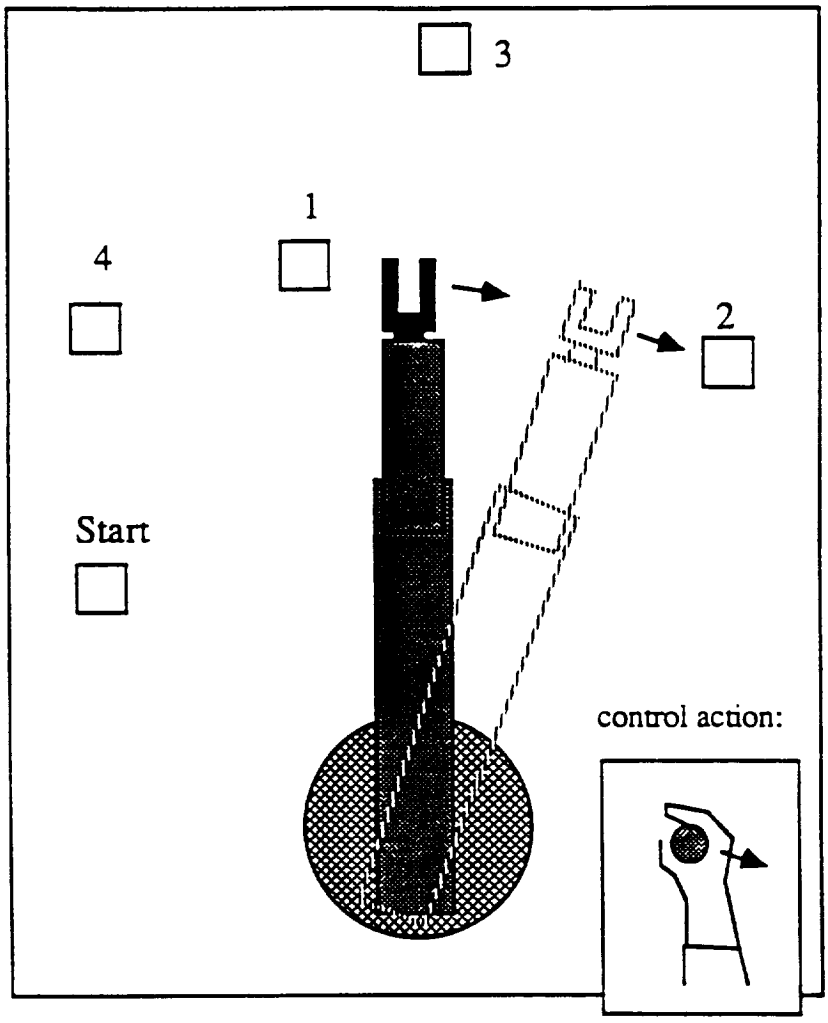


Figure 2.1. An example of prediction display. The human operator is controlling a simulated telerobot (dotted robot) while the position of the real telerobot is displayed in shaded color.

"predictor display" not only provides a method of dealing with the time-delay problem, but also take the full advantage of computer graphics, geometrical modeling, computer simulation and other related techniques to greatly increase the operator's control ability. When discussing the "predictor display" method, we should note the difference between the concepts of "predictor display" and "prediction display". While "predictor display" presents a simulation of the tele-manipulator (a 3-D object) on a graphical screen and the operator is able to control the movement of the simulator in real time, the "prediction display" only displays current and predicted values of variables in the form of curves or pointers.

2.1.2 Time and position desynchronization

Conway, Volz and Walker [28], [29], [30] have further developed the concept of predictor display by breaking the synchronization between the simulated time frame and that of the remote robot. They introduced *time clutch* and *position clutch* control modes to allow the operator to desynchronize the time and position frames, respectively, of the simulation and the remote robot.

Time desynchronization control enables the operator to disengage the "direct-gearing" of simulated-time and real-time and move the simulator as fast as skill and judgement will allow. Their hypothesis is based on the assumption that the human operator can often think of and generate a path segment more quickly than the telerobot can follow it. This assumption is particularly valid for large space telerobots such as the Remote Manipulator System [87]. As a result, such a generated path segment can be followed by the telerobot at nearly its maximum speed. Experiments have shown that when the time clutch is used to perform the task of touching a series of boxes, the performance is

better than the performance with no time clutch even in the case of no time delay [31]. In the position desynchronization mode the operator moves the simulated robot without sending any control signal to the remote site. This allows the operator to carry out difficult maneuvers on the simulator and allows the real robot to move directly to the end result (when the position clutch is released) without following all of the operator's false moves. Thus, these control modes provide much greater control capability than previous methods.

2.1.3 Basic tele-autonomous architecture

The control modes mentioned above together with other control mechanisms such as the time brake, time ratio logic, task handoff and rendezvous constitute the basic frame of a tele-autonomous system. One such an experimental system has been implemented at University of Michigan. A detailed description of its implementation can be found in [30]. The basic system architecture is given in Fig.2.2.

At the local control site a force-sensing joystick is used as an input device for the human operator to control the simulation of the telerobot. The input signal to the joystick is sampled and sent to an input transducer, where the positional signal is generated based on this sample. The input is then sent to a path history buffer (PHB) which is controlled by the PHB controller. A manipulator geometrical model, models of the remote environment and a path history animator (MGA) are built into the system in order to simulate the remote site. Other control mechanisms such as a time clutch, position clutch, time brake, *etc.* can also be found in the diagram.

The system at the remote site consists of a remote manipulation controller (RMC) which can faithfully follow new position commands, a remote control queue (RCQ) that can hold command values sent from the local site and its associated queue controller (QC).

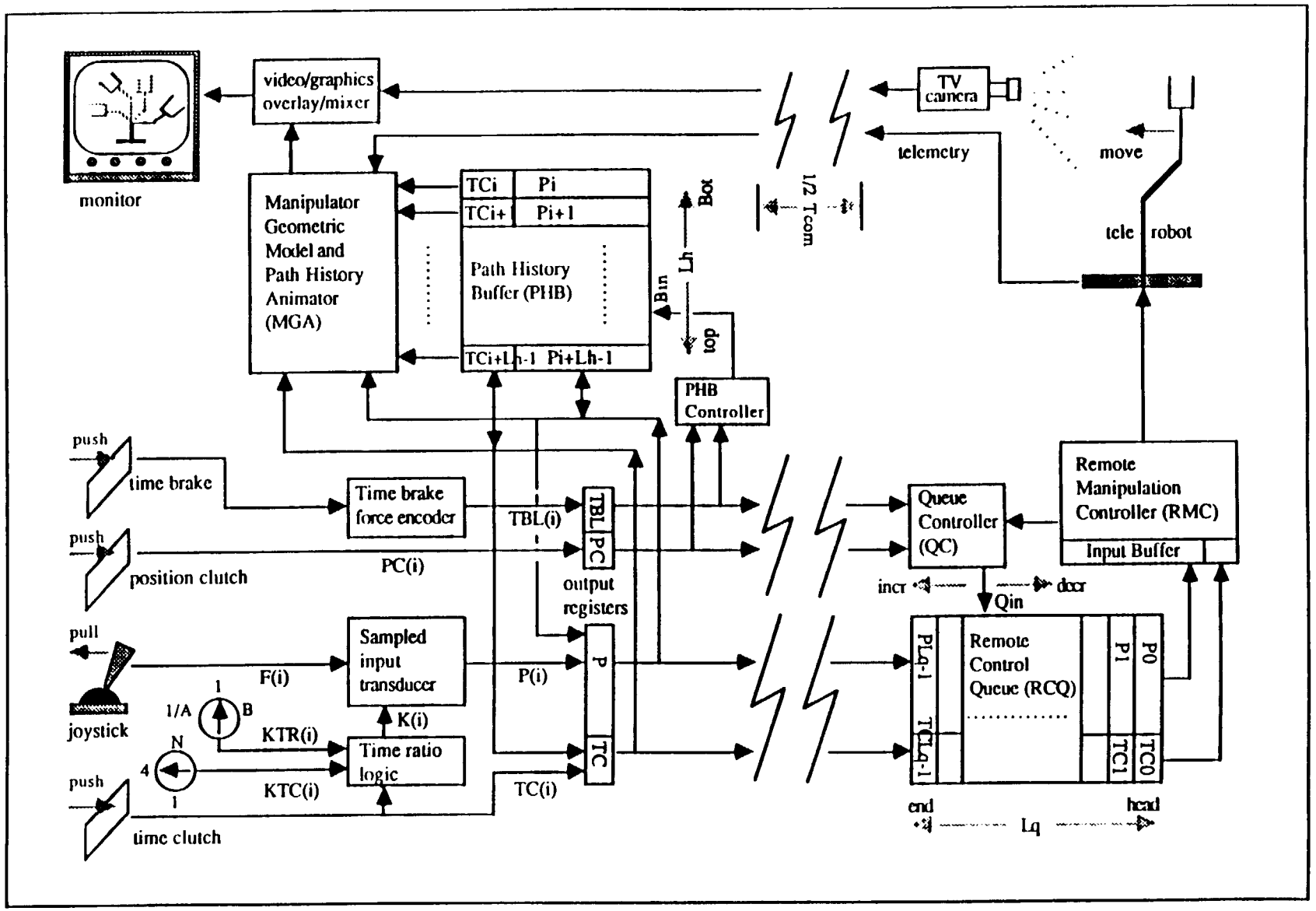


Figure 2.2. Details of Basic Tele-autonomous System Architecture.

The detailed structure and functions are described in [30].

2.2 Imperfect Simulation Problems

As can be seen above, predictor display and other control modes have been introduced to defeat time delay problems. The fundamental principle behind these methods is the direct control of a simulation of the telerobot rather than direct control of the actual robot. Because simulation is used, the quality of the model will be an important factor in determining the quality of results produced. One problem will arise: Can we build a high quality model? The building of a rather precise geometrical model for each individual component and even the remote environment is not totally impossible. In most cases, the telerobot and key components of a space station, shuttle, satellite, *etc.*, are designed and built on earth. The environment in which the telerobot will work is also rather structured. All of these will allow us to build a quite good computer model. But in practice various uncertainties still exist and these uncertainties will result in imperfect simulation. These uncertainties include the following:

- Imperfect simulation of the remote robot — The human operator moves the simulated robot and assumes that the remote robot will reach the same position and orientation several seconds later. The remote robot, however, will in general reach a different position even though the error may be small.
- Imperfect simulation of the remote environment — The object's location is not exactly the same as anticipated even if an accurate object model is known.
- Imperfect part model — Usually the part tolerances are not built into the model, which will result in geometric uncertainties.

What are the possible problems the imperfect simulation will cause during the implementation of a tele-manipulation task?

Consider now, in a more detailed way, what is involved in a tele-manipulation task. The process of implementing a tele-manipulation task, no matter how complex, involves repeated performance of the following sequence of activities:

1. **move** the end-effecter of the tele-robot from the current position to a new one;
2. **grasp** an object;
3. **move** the object to another location;
4. **manipulate** the object.

We call steps 1 and 3 *global motion* activities and steps 2 and 4 *fine motion* activity. Table 2.1 gives a list of tele-manipulation tasks specified by RATS (Robotic Assessment Task Set). The operations of all the tasks follow this pattern.

During global motion activities, one of the main concerns is collision avoidance. It is obvious that contact could be avoided in this stage. Also, as long as the collision avoidance requirement can be met, the path along which the robot's end-effecter moves from the initial position to the goal position usually need not be very accurate. Therefore, the existence of imperfect simulation is not a crucial factor to the success of global motion. We anticipate that in the near future the implementation of global motions will still be under direct manual control. The state of the art is not yet good enough to do global motion planning automatically in real time. Yet, as has been demonstrated [29], tele-autonomous control is capable of doing all of the global motion tasks. During the fine motion stage (including grasping), however, contact is inevitable and will cause difficulties during the execution of the tele-manipulation tasks.

Consider now what might happen in the following scenarios, where the remote ma-

• Assembly
– Truss Assembly
– Utility Line Connection
– Station Interface Adapter (SIA) to Truss Connection
– Solar Dynamic Array Assembly
• Maintenance & Servicing
– Solar Power Converter ORU Changeout
– HRSO File Canister Changeout
– SMM MEB Replacement
– Electrical Connector Inspection
– GRO Refueling

Table 2.1. A Partial List of Robotic Assessment Test Set (RATS)

nipulator's movement involves contact with other objects. If the above described tele-autonomous system structure is used but only straightforward teleoperations are applied and the simulation is not quite perfect (as will always be the case), then the following scenarios occur:

Scenario 1: Move and Place

A path that places an object on a surface in the simulation might actually extend into the surface in the corresponding real situation. Thus, when the real robot reaches the surface, it will continue to try to move through the surface, creating large forces unless there is compliance in the system. Large forces could either damage the parts or the robot or cause misalignments that would prevent success in further robot operations.

Scenario 2: Peg-in-Hole

Let the remote robot be compliant so that large forces do not build up. Now, suppose that the operator successfully places a peg in a hole in the simulation and the real robot tries to follow this action. Because of various uncertainties, the part held in the hand of the real robot might hit the edge of the hole. While large forces will not build up, the operator does not know that the operation was not successfully completed, and the remote robot does not know what path to follow to successfully complete the motion. Indeed, the remote robot does not have a concept of what success would mean for the operation, except to have matched the commanded position, which it cannot do.

In both cases, the imperfect simulation has resulted in the failure of the operation.

In summary, the fundamental problem facing telemanipulation is the time-delay problem. Predictor display and forward simulation provide one means to cope with this problem. In this case the local operator is driving a simulator of the tele-manipulator instead of the real one, with the control signal being sent to the remote site in parallel. But imperfect simulation and uncertainty of the remote environment makes simple direct teleoperation difficult to apply in situations where the movement of the remote manipulator involves contact with objects.

One solution is to develop a highly intelligent remote manipulator so the local operator needs only to give a high level instruction to the remote site and let the remote manipulator decide how to act. While a highly intelligent remote manipulator is what the people have hoped for and recent advances in artificial intelligence, human cognitive modeling, and computer vision indeed have provided us a greater opportunity than ever to build very powerful remote manipulation systems, they are still not powerful enough for full automation.

As an alternative, human-guided control can offer an attractive possibility. That is,

the remote site has only limited intelligence and is still able to finish quite sophisticated tasks with the help from the human operator. As the technology evolves, the remote site will gradually take more responsibility until full automation can be achieved. This, in fact, is the core idea behind the tele-autonomous system concept.

The uncertainty problem usually can be approached from two different objectives: 1) develop techniques that help **avoid** the uncertainty; and 2) techniques that help the system **adapt** to the problems. Several possible approaches have been suggested in [130], such as simultaneous force and compliance control [80] [95] [126] [128] [136], force/torque and contact constraint reflection [35] [83], dynamic replanning [7] [34] [131] [132] [133], *etc.*, and there has been some success with these approaches. Their applications, however, are very limited. This is because either their methods can be only applied under very special conditions, or the experimental results are not very persuasive [56] [74].

In the next section we will describe how object localization technique can provide the necessary intelligence in the remote site to overcome some of the uncertainty problems.

2.3 Applications of Object Localization

The tele-autonomous system is designed to allow many different levels of control to be carried out in order to achieve maximum flexibility and efficiency. Although there is no general agreement on how to divide these levels and how to name them, one thing is certain: the classification of control modes is based on the degrees of operator involvement in telemanipulation, the ways in which the operator interacts with the remote system and the levels of intelligence in the remote system.

In this section, we discuss two situations where a low level of intelligence introduced through object localization could help the execution of telemanipulation tasks.

Situation 1: Direct manual control is used to perform a telemanipulation task. That is, all the remote manipulator's operations such as positioning, grasping and manipulating follow the local operator's instructions.

Situation 2: Limited autonomous control where the remote system has a certain degree of automation ability. The operator needs only to send high level commands to the remote site and lets the remote site perform a task (or subtask) automatically.

The tele-manipulation tasks that fit into the first situation include the tasks which will be performed only "once in a life time" or performed infrequently. For these tasks, automation is very expensive; or very difficult to implement. Many operations such as infrequent maintenance operations or actions reacting to unforeseen events are examples of the first type of tasks.

The tele-manipulation tasks that fit into the second situation include the tasks which will be performed repeatedly with no unforeseen events likely to happen during execution, or tasks which are very easy to automate even they may only be executed infrequently. Many Space Station Construction tasks such as "Truss Assembly" and "Solar Dynamic Array Assembly", as listed in Table 2.1, are examples of the second category of tasks.

2.3.1 Basic assumptions and the concept of relative move

Our approach to solving the uncertainty problems under direct manual control is to combine the use of object localization's technique in the remote site and the use of *relative move* mode of operation. This approach is based upon several reasonable assumptions:

- The local simulation contains correct models of all the objects in the workspace and the locations of these objects is approximately known before any telemanipulation

can be applied on the objects. The approximate location of an object can either be modeled beforehand or be established through remote sensors. In the later case, we assume that TV pictures of the remote site can be viewed by the local operator. Whenever the operator finds an object from the TV pictures in which he/she is interested and the location of which is not modeled in the local world modeling system, he/she can send a command to ask the remote sensors to provide this information to the local site.

- Remote sensors can accurately locate objects in the workspace *in real-time*; they just cannot send the information to the local controller in a timely manner.
- The objects to be manipulated and the simulations of the objects in the actual and simulated workspaces, respectively, are sufficiently close to each other that the differences will not lead to a singularity or violation of workspace problems.

With these assumptions the basic idea of this approach is for the operator to control the simulated robot exactly as described earlier for tele-autonomous operation. However, instead of sending the absolute position commands to the telerobot, the positions are first converted to being *relative* to the object being manipulated, and the relative position transmitted. The remote robot, by assumption, is capable of sensing the position of the object to be manipulated in real-time. It then uses the sensed position of the object to transform the received relative commands into absolute position commands and then proceeds to follow the commanded positions. If the system is sufficiently accurate, as described above, the commands should succeed.

- s[et] < *obj* >: Set reference frame to *obj* frame.

  By default, the reference frame is the world frame or robot base frame. The reference frame could be set into any other object frames which must have been modeled in the world modeling system.

- m[ove] < *pos* >: Move the telerobot's end-effecter to next position *pos*.

  The position has six parameters: three for translation, three for rotation. The position specification is relative to the current reference frame set by the latest s < *obj* > command.

- l[ocate] < *pos* >: Locate object.

  Determine the location of the object set by the latest s < *obj* > command. In order to provide assistance to the remote system and speed up the localization process, the command has an optional parameter *pos* that tells the remote system which part of the object will be sensed by the remote sensors.

- g[rasp]: Grasp the object.

- r[un] < *prog* >: Begin to run a program named *prog*.

Table 2.2. A list of the Basic Commands to Implement Relative Move

2.3.2 Implementation of relative move

To implement the new control mode, in addition to simply sending a sequence of servoing positions to the remote site, the local site needs also to send other necessary commands to the remote site to guide the operations of the remote manipulator. A list of the basic commands are described in Table 2.2.

Let us take a closer look at how to use the relative mode and object localization technique to control the telemanipulator. For purposes of illustration, a simple **move-**

then-grasp operation is to be performed by using the direct continuous teleoperation control method.

Assume that the operator will use a force-sensing joystick to directly control the movement of the simulation of the telemanipulator. The following symbols will be used to express position input and timing of the operations in the local site (see [31]):

- T_s: The simulation sample period, at each interval of which the input signal from the joystick will be sampled.
- $\mathbf{F(i)}$: The input sampled at time $i * T_s$.
- $\mathbf{P(i)}$: The position of the tele-manipulator's end-effecter calculated from the sample $\mathbf{F(i)}$.

$\mathbf{P(i)}$ and $\mathbf{F(i)}$ are each six-dimentional vectors where $\mathbf{F(i)}$ is a vector of the force and torques measured at the joystick and $\mathbf{P(i)}$ is a vector of three-translational and three-rotational coordinates. $\mathbf{P(i)}$ is calculated as:

$$\mathbf{P(i)} = \mathbf{P(i-1)} + \Delta\mathbf{P(i)}, \tag{2.1}$$

where $\Delta\mathbf{P(i)}$ is a function of $\mathbf{F}(()i)$ and other related variables. In a simplified case, $\Delta\mathbf{P(i)}$ can be expressed as

$$\Delta\mathbf{P(i)} = K(i) * \mathbf{F(i)} \tag{2.2}$$

and $K(i)$ is the gain parameter.

By default, the world frame is initially set to the reference frame and the calculated position $\mathbf{P(i)}$ will be with respect to the world frame. If the reference frame is set to an object frame, as is needed to implement the relative move, the calculated $\mathbf{P(i)}$ has to be transformed into the position vector which specifies the position of the tele-manipulator's

end-effecter with respect to the object frame before its value can be stored into its dedicated local output buffer. There is no problem for the transformation, because by assumption the relative location between the world frame and the simulated reference object frame must have been established in the world modeling system. The formula of the transformation is

$$\mathbf{P(i)}_o = \boldsymbol{T}_w^o \mathbf{P(i)}_w \tag{2.3}$$

where

$\mathbf{P(i)}_w$ is the position value of the simulated end-effecter calculated from the joystick's reading with respect to the world frame;

$\mathbf{P(i)}_o$ is the transformed position value of the simulated end-effecter with respect to the reference object frame;

$\boldsymbol{T}_w^o$ is the simulated transformation matrix between the world frame and the reference object frame.

The position value $\mathbf{P(i)}_o$ will then be put into the output buffer each time it is calculated.

During each sample period T_s, the $\mathbf{P(i)}$ value in the output buffer, no matter which reference frame it uses, is transmitted to the queue in the remote system, and then the new value of the position $\mathbf{P(i+1)}$ is put into a local output buffer to be sent during the next sample circle.

The control sequence in the local site is first to "*move*" the simulated telerobot close to the simulated object, then to "*set*" a new reference object frame, followed by sending a "*locate*" command; and then continuously "*move*" the simulated telerobot, still in absolute mode, with the position values being constantly transformed into the ones relative to the reference object frame and then executed; and finally *grasp* the object.

If the operator feels that it is more convenient to graphically control the movement of the simulation of the telerobot relative to the reference object frame, he can do so without

any difficulty. The sampled data from the joystick can be set to represent position and orientation values relative to the reference frame instead absolute values (in this case, the transformation process of using Eq.(2.3) is no longer needed). The computer graphics techniques enable the operator to view the simulation with great flexibility – in any angle, in any size, focus on any part of the environment, *etc.*. All these abilities make the *relative move* mode technically feasible.

Now, let us look at how the remote system will execute the commands from the local site. These commands are held by a remote control queue (RCQ). Because in tele-autonomous system the time can be de-synchronized, there is no fixed relationship between the position of any entry in the queue and the time at which the entry will be processed by the remote manipulation controller, though the relative time ordering between entries will be preserved.

When the reference frame is the world frame, the position value in the queue can be used directly to control the manipulator. If it is not, the position value which specifies the relative position between the end-effecter and the reference frame has to be transformed into the value which specifies the relative position between the end-effecter and the world frame. The task facing the RMC when it finds a *locate* command is to find the relative transformation between the referenced object frame and the world frame in real-time. Once the transformation relationship has been found, the remote controller will continue executing position commands in the RCQ. But the position value has to be transformed so that it is the value of the end-effecter with respect to world frame

$$\mathbf{P(i)}_w = \boldsymbol{T}_o^w \mathbf{P(i)}_o \tag{2.4}$$

and the derived position value will be used by the RMC to move the telerobot.

Table 2.3 shows the timing of the operations on both the local site and remote site.

Local Site	Commands	Remote Site
	s world	
T_s	**m** p_1	T_1
$2 * T_s$	**m** p_2	T_2
...	...	...
$j * T_s$	**m** p_j	T_j
	s object i	
	l k	T_l
$(j+1) * T_s$	**m** p_{j+1}	T_{j+1}
$(j+2) * T_s$	**m** p_{j+2}	T_{j+2}
...	...	...
$(j+n) * T_s$	**m** p_{j+n}	T_{j+n}
	g	

Table 2.3. Timing of Move-Grasp Operations on Both Sites

The key to the success of the *relative move* mode is the concept of time-desynchronization such that the timing in the remote site does not have to follow the timing in the local site. Thus, the position commands generated at a constant rate (1 command per T_s unit) at the local site are performed by the remote controller at varying times: $T_1, T_2, \cdots, T_{j+n}$.

2.3.2.1 Signal feedback – a method of improving the perfection of local model

We have mentioned that imperfect simulation, especially imperfect simulation of object locations, will cause many problems during a tele-manipulation task and that object localization can be used to cope with this problem. In this section we will show that the use of object localization can improve the results of imperfect simulation of object locations.

Usually, only position values of the remote manipulator's joints are sent back to update the local model each time immediately after the tele-manipulator is moved to a new position. Now, if after each "*locate*" operation, the position values of the referenced object frame are also sent back simultaneously with the telerobot's position values to update the modeled location of that object frame, the updated model object location in the local site will become much more accurate than before, provided that the following conditions are met:

1. The sensing system in the remote site can provide accurate measurement;
2. The telerobot position values provided by the RMC are quite accurate;
3. The sensing system in the remote system is well calibrated such that the relative location between the telerobot and the sensor is known accurately.
4. The model object has not been moved by the operator at the time when the updated object location values are sent back;

Let us assume that the sensor is mounted somewhere on the telerobot's gripper. Let T_b^a represent a transformation matrix between coordinate frame a and coordinate frame b, symbols g, o, s represent, respectively, the gripper frame, the referenced object frame and the sensing frame in the remote site, and mg, mo, w represent, respectively, the modeled gripper frame, modeled referenced object frame and the world frame in the local model.

After a "*locate*" operation, the sensing system can provide an accurate measurement of T_o^s (condition 1). From condition 3, a transformation matrix T_o^g will also be available accurately from the formula:

$$T_o^g = T_s^g T_o^s \tag{2.5}$$

The values of the transformation matrix together with the position values of the telerobot's gripper will be sent back to the local site simultaneously. The same values will be treated as the position values of corresponding modeled coordinate frames, e.g., the position values of gripper frame will be thought as of T_{mg}^w and T_o^g will be treated as T_{mo}^{mg}. Condition 2 guarantees the accuracy of matrix T_{mg}^w. Therefore, the transformation matrix

$$T_{mo}^w = T_{mg}^w T_{mo}^{mg} \tag{2.6}$$

will become quite reliable and can be used to update the modeled location of the reference object. The time-delay has no influence on the result of updating because the object's position is fixed during this period of time from condition 4.

Once the object location model has been updated with high accuracy, the operator will have greater confidence in controlling the simulation of the telemanipulator. If the modeled object location is not very distant from the true location, the operator might even feel that nothing has been happened during his continuous tele-manipulation.

2.3.3 Object localization used in autonomous operations

The Object localization technique can also be used in situations where the remote system operates in autonomous mode. The operations usually go through the following sequence: The operator moves the simulated tele-manipulator to a pre-specified position and then sends a command to the remote system to activate a pre-programmed routine

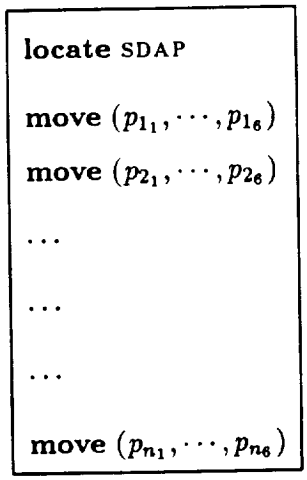


Figure 2.3. A Simple Routine With Relative Position Specifications

for the tele-manipulator to execute. The routine is coded in such a way that the position path is specified relative to a reference object frame instead of the absolute value. Before the execution of the routine, the remote system has to first *"locate"* the reference object and then convert the specified positions path to the position path of the tele-manipulator's end-effecter.

For example, if a task of cleaning the surface of a Solar Dynamic Array Plate (SDAP) is to be performed by the tele-manipulator, the cleaning tool which is attached on the tele-manipulator's end-effecter will move along the surface of SDAP following a certain path. The path of the cleaning tool is specified in a routine by a series of positions, all are specified with respect to the SDAP coordinate frame – another form of relative move (see Figure 2.3).

Each time, before the telerobot executes the routine, it first *locates* the SDAP. The real path of the end-effecter with respect to the world frame can then be calculated by using the formula shown in Eq.2.4. Thus, the routine outlined above gives the telerobot the flexibility to adapt to various locations of the SDAP.

CHAPTER 3

METHODS AND STRATEGIES OF OBJECT LOCALIZATION

In the last chapter we have discussed the importance of object localization techniques in tele-autonomous systems applications. Indeed, the availability of efficient means of locating objects is one of the key factors to the success of developing such systems. In addition, object localization techniques have also found many applications in other areas of technology, specially in robotic manufacturing such as automatic assembly, part inspection and so on.

Object localization has long been defined as a part of the object recognition process in computer vision research [12]. But in most instances the emphasis of the research has been on object recognition. Object localization is only a by-product. In robotic applications, however, object localization usually is the ultimate goal. It has many of its own problems to be solved, such as real-time considerations, accuracy issues, types of locatable objects, working conditions, *etc.*, which object recognition research generally does not address. In some telerobot systems "**locate**" has been defined as one of the basic independent operations the system is to perform [96]. As a result, the object localization problem, as an independent research topic, has attracted increasing attention recently.

This chapter will give an overview of the three-dimensional object localization problem. First, it provides a closer examination of the problem and then describes general object localization system structure. This is followed by a discussion of some important issues

of object localization. Current available object localization algorithms and systems are surveyed in the last section. These algorithms are characterized by their feature and matching strategies, the range-finding methods, the types of locatable objects and the mathematical formulating methods.

3.1 The Object Localization Problem

Object localization is the determination of the location of stationary and moving objects. What must be solved when a robot vision system is trying to locate an object? As Gunnarsson pointed out [49]:

> Localizing a part means being able to answer the following question: " In a given frame of reference, what are the Cartesian coordinates of any specified point on the part's surface."

In practice, it is both impossible and unnecessary to take real measurements on every point of the object's surface in order to locate it. Instead, in a model-based system where the shapes of the objects in the scene are known and a complete geometric database description of each object with respect to that object's coordinate frame is available, the localization problem is solved by identifying the location relationship between two coordinate frames: the reference frame and the object coordinate frame. Once the location relationship between these two coordinate frames is determined, the Cartesian coordinates of any point on the object surface with respect to the reference frame can be obtained by a simple transformation process.

To locate an unknown object, the best way is to first explore the object and try to establish a model description for that object before any localization process is carried out. Recently, researchers have begun working on this object exploration problem and have achieved some preliminary results [33] [104].

Thus, a necessary component of every object localization system is the world modeling system which stores, among other things, the descriptions of all the object geometrical shapes and a definition of the sensing coordinate system. Each object must have a corresponding coordinate frame associated with it in order for the object to be described in the world modeling system; the coordinate frame might be specified either implicitly or explicitly.

The relative location between two coordinate frames can be specified in any one of the following ways:

1. *Position and orientation:*

   The position is usually specified by a 3×1 position vector $\mathbf{p} = (p_x, p_y, p_z)$. There are three different representations of orientation:

   - Represented by three rotation angles: this could be Eular angles α, β, γ, or the rotation angles about the coordinate axes.
   - Represented by a unit vector $\mathbf{r}$ and an angle θ.
   - Represented by a quaternion or its variation [64] [70] [94] .

   In practice, the terms of "rotation" and "translation" are also frequently used to represent the relationship between two coordinate systems. They have the same meanings as "position" and "orientation". Both terms are used in our discussion.

2. *A 4×4 homogeneous transformation matrix.*

3. *Dual number quaternion:* [121]

   This is an extension of quaternion representation in which each quantity is changed to a dual quantity [26]. The dual quaternion has a similar interpretation as the real

quaternion:

$$\hat{q} = \begin{bmatrix} \sin(\hat{\theta}/2)\hat{n} \\ \cos(\hat{\theta}/2) \end{bmatrix}$$

where the vector $\hat{n}$ is a unit line vector about which the coordinate system has been rotated and translated and $\hat{\theta}$ is the dual angle of rotation and translation.

The objective of the object localization algorithm is to determine the parameters which specify the corresponding representation.

Which representation method is the best one to choose in real application? It depends on many factors. But among other things, how this localization problem is formulated in mathematical terms and which mathematical tool is used to carry out the computation are definitely the main factors in the selection.

For instance, if geometrical analysis is used to derive the location relationship between two coordinate frames, vector algebra is the right tool during the derivation. The relative location between these two frames is best expressed by the vector representation method – a position vector and a rotation direction vector together with a rotation angle. In many cases, the localization problem can be formulated as an optimization problem using a least-squares minimization to solve the problem,. The quaternion representation is a better choice here because it does not involve any trigonometric computation and the number of unknowns to be optimized are nearly the minimum.

In some applications the localization problem can be simplified due to extra constraints imposed on the object. For example, if an object is so constrained that a planar surface of the object is always lying on another planar surface, there are only three degrees of freedom for the object: one degree of rotation and two degrees of translation.

How does one solve for these position/orientation parameters if sensor data and object

models are given? Usually the computation is carried out by a matching process. That is, the object localization algorithm will try to find a "best" transformation which will put sensed features into its corresponding model features. The matching feature pairs may be of the same type or may be from different types; they may be 2-D features or 3-D feature. But all the features are of geometric types, e.g., those that specify position and orientation, such as points, edges, surfaces and *etc.*. Table 3.1 shows some known feature matchings which have been used in the literature to derive the location of an object. Sometimes, a combination of feature matchings is necessary to completely specify a rigid transformation. The geometric features to be extracted and matched can be classified as low-level features and high-level features. Possible low-level features include points, vectors, line segments, axes, surface patches, edges, boundaries, *etc.*. Possible high-level features include straight dihedrals, circular dihedrals [18], principle directions of surface curves, minimum, maximum and mean curvatures of surfaces, Gaussian curvatures, *etc.*. Usually the lower the level of features, the greater the number of features to be extracted.

From the above description, it is not difficult to imagine that a general object localization system should contain the following components: (1) **sensing system**: to provide necessary measurements; (2) **world model**: to give a geometrical description of all the objects in the environment including robot, sensors and their relationships; (3) **feature extraction**: to retrieve geometrical features which are to be used in the matching process; (4) **matching**: to try to pair the sensed features with corresponding model features; and (5) **computing**: to calculate the transformation parameters.

A general configuration is shown in Fig. 3.1. However, many variations could exist in real applications.

Take the sensing system as an example. The task of the sensing system is to provide

Measured features	Matched to
Point	Point
	Planar surface
	Surface patch
Surface normal	Surface normal
Line segment	Line segment
	Planar surface
	Surface patch
Edge	Edge
Planar surface	Planar surface
Quadric surface	Quadric surface
Gaussian curvature	Gaussian curvature

Table 3.1. Known Matching Strategies in Object Localization

enough data for the feature extraction unit. If a single measurement taken by the sensing system can provide enough sensed data, the connection between these two units is a one way relation. Sometimes, the sensing system has to make a series of measurements in order to meet the needs of the feature extraction unit, such as the case where a spot-range sensor is used. In this situation, the whole feature extraction might go through a repeated sensing-extraction process, and a scheduling algorithm may be necessary to guide the sensing-extraction process. The function of the scheduling algorithm is to find an optimal measurement path in order to reduce the measurement times. [102], [103] have described such an algorithm.

Another example is the matching unit. The matching process is the process of finding

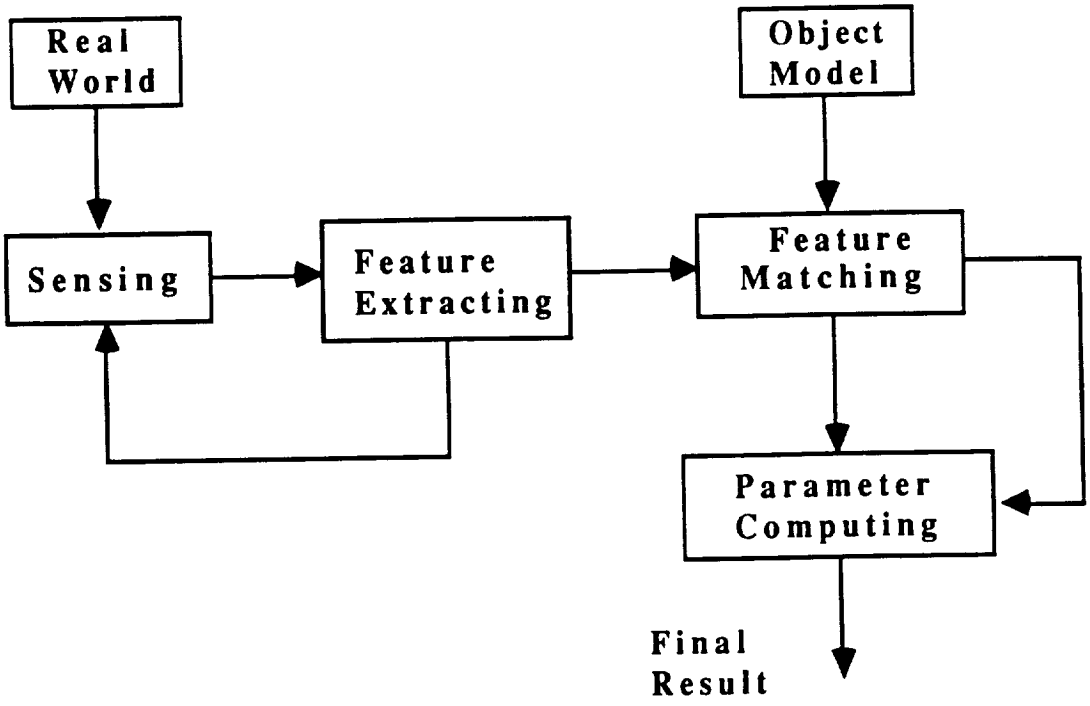


Figure 3.1. Object localization system organization

the pairings of sensed features and model features. Depending on the intelligent level of the system, the matching could be performed differently. On the lowest level, the system has no matching ability at all. That is, there is no matching unit in the system. Such systems can often find applications in highly-structured environment in which the relative locations of all the objects are approximately known *a prior*. Thus, the system knows where to make necessary measurements in order to locate the objects, what features are expected to be extracted from the measurements and what matching modeled features correspond to the sensed feature. No matching process is needed in this situation. The matching process could also be provided manually. On telerobotics systems, for example, the teleoperator might interactively assist the model matching by indicating with a light pen which features in the image (e.g. edges, corners) correspond to those in a stored model

[3]. If none of these conditions are met, the system has to have its own matching unit to pair the features automatically. This requires systems with higher levels of intelligence.

There is no common solution for the implementation of the object localization mechanism we have just shown. System structure may have different configurations. Each component in the system could be implemented in various ways, too.

3.2 Some Issues

We have just shown that a general structure for object localization systems. In practice, there are some important issues which must be considered when a real localization system is to be designed. Among them, the important ones are the speed and accuracy requirements of localization, the types of locatable objects, sensing methods, sensor installation, *etc.*.

3.2.1 Real-time execution

A **Hierarchical** control structure has been defined as a standard telerobot control architecture [3] and has been adopted by researchers to develop individual telerobot systems, such as systems developed at Goddard [96], the University of Michigan [120], *etc.*. As the functions of vision systems are different at each level, so are the requirements for the object localization algorithms. Usually the higher the level, the slower the completion rate. See Table 3.2 for typical completion rates at each level of telerobot control.

At the object task planning level, for example, one of the functions of the vision system is to recognize the environment. The object localization system, as a part of the vision system, is used to give approximate measurements of the locations of the objects in the

	Average rate of change in output	Average replanning interval	Planning horizon
Servo	1 KHz	1 millisec.	15 msec.
Primitive	62Hz	16 millisec.	300 msec.
E-Move	8Hz	128 millisec.	2 sec.
Object/task	1 Hz	1 second	30 sec.
Service Bay	.1 Hz	10 second	> 10 min.
Mission	0.01 Hz	1.7 minutes	> 1 hour

Table 3.2. The rate of subtask completion at each level of hierarchy. ([3])

environment. The execution time is in the minute range. At the E-move level, however, the rate of completion is in the range of seconds. If a visual-feedback control strategy is used here, the localization system has to generate updated measurements for the control system to adjust the robot's movement in the same time frame. Real-time issues will become important. Based on different timing requirements, the strategies of localization might be also different.

3.2.2 Accuracy

Accuracy is another important issue in object localization. There are two concepts about accuracy, e.g., **absolute accuracy** ϵ and **relative accuracy** $\Delta\epsilon$.

- *Absolute accuracy* is defined as the difference between a measured value m and the "true" value s. That is, $\epsilon = m - s$.

- *Relative accuracy* is defined as the difference between a measured difference Δp and it's actual difference $\Delta p'$. That is $\Delta\epsilon = \Delta p - \Delta p'$. For example, if a point is moved Δx units along x axis and we measure the change of the movement as Δm, the relative accuracy of the measurement is $\Delta\epsilon = \Delta m - \Delta x$.

Different types of telerobotics tasks require different accuracy. High accuracy requirement occurs primarily in assembly types of tasks. According to our survey of eight telerobotics tasks [96], for move-pickup-put types of tasks the accuracy requirement is $0.1 - 1$ *inches* $(2.54 - 25.4mm)$ in translation and $2 - 10$ degrees in rotation; for assembly types of tasks, especially for insertion, the accuracy is much higher: $0.03 - 0.0625$ *inches* $(0.762 - 1.587mm)$ in translation and about 1 degree in rotation.

Another study on the accuracy issue of sensor-driven robotic assembly tasks is done by [47]. According to the author's analysis on a typical peg-in-hole assembly task – the most frequent assembly operation [72, 88], if a 1.75 *inches* $(44.5mm)$ square $\times 10$ *inches* $(254mm)$ length oblong peg is to be inserted in a hole with a clearance of 0.004 *inches* $(0.1mm)$ on each side, and we do not want the insertion to fail because of various misalignments (translational, rotational etc.) on the position between the peg and hole, the maximum allowable misalignments are 0.02 *inches* $(0.5mm)$ in translation and 0.5 degree in rotation by using [127]'s formula. These two figures are quite consistent.

[47] has further analyzed the accuracy requirement for the sensor. He concludes that if a 98.8 percent chance of successful assembly is assumed and the errors from other resources such as robot positioning error, robot kinematics error, sensor-robot coordinate alignment error *etc.* are also considered, the sensor should provide localization accuracy up to 0.0055 *inches* $(0.14mm)$ in translation and 0.14 degree in rotation.

The accuracy requirement in an assembly task is in fact based on the relative accuracy

most of the time, not the absolute accuracy. This is because during an assembly the system only needs to know the relative location between the two relevant objects.

Absolute accuracy to a large extent depends on the accuracy of the sensing system. The achievement of high relative accuracy, on the other hand, does not necessarily mean the use of highly accurate sensors. The achievement of high relative accuracy usually requires a good design for the object localization algorithm.

Therefore, when designing an algorithm, one must evaluate its performance according to both it's absolute accuracy and relative accuracy and the emphasis should be on the relative accuracy. This is because: 1) the tasks which require high accuracy are that of assembly types in most cases, and 2) performing an assembly task mainly requires relative accuracy.

3.2.3 The types of locatable objects

It is best if the system can locate arbitrary-shaped objects, as long as the shapes are describable. For an arbitrary-shaped surface, the surface curvature properties such as Gaussian curvature, minimum, maximum and mean curvatures are often used as the features to match in order to locate the object. The problem of using these features is the high-sensitivity associated with either measurement errors or slight distortions of the object surfaces [49].

In industrial environments a more practical requirement for the sensing system is the location of man-made objects. The shapes of most of these industrial parts are not very complex. Their surfaces contain only one or a combination of a few simple and well-known types of surfaces, e.g., planar surfaces and quadric surfaces (cylinders, cones, spheres, etc.) [14]. If a localization system is able to locate objects having such types of surfaces

reliably, it will work for most of the time. How to deal with complicated objects? Instead of developing a universal localization algorithm, a simple alternative solution is to make special marks on the object so that the localization system has no problem detecting these marks and uses these detected marks as the matching features to compute the localization parameters of that object.

Therefore, among other things, localizability should also be a consideration during the design stage of an industrial part. Some guidelines should be given to meet the localizability requirement. Sometimes, very simple modifications made on the part design can greatly ease the part-localization process.

3.2.4 Sensing system

What types of sensing techniques should be used in a localization system? Where should the sensors be installed? How should the basic sensor requirements be determined in specific applications? These are just some of the issues in the design of a sensing system.

3.2.4.1 Ranging methods

Jarvis [63] has presented an early overview of range finding techniques. The range-finding methods can be classified, based on the types of illumination, into two categories: passive methods and active methods.

Passive methods use normal, unstructured or natural illumination to acquire simple images from 2-D cameras, then process these images to obtain distance information. Occlusion cues [97], texture gradients [125], shape from shading [93], depth from focusing [58] [62], stereo disparity [24] [57] [61] [129], range from motion [119] [123], moiré fringe

contours[59] are all examples of passive techniques. Because range information is extracted from intensity images, it requires extensive effort to process images and is thus difficult to be used to solve the real-time object localization problem.

Active methods use controlled energy beams, such as ultrasonic, radio, white light, or laser beams, to acquire range information through the detection of the reflected energy. Because of the obvious advantages of laser beams over other energy sources, most sensors use lasers as the energy source. Based on their range finding principles, the active range measuring techniques can be divided into the time-of-flight method, triangulation method and striped lighting method.

The principle of the time-of-flight method is very simple. The distance of an object is determined by measuring the time it takes for the controlled laser beam to travel from the source to the object and back. In time-of-flight methods, the lasers can be used as pulsed-mode or modulated continuous-wave mode. In the pulsed-mode method, the time is measured by counting the number of pulses a laser beam takes to go from the source, bounce off a target point and return coaxially to a detector. To achieve high resolution, the electronic circuit must have fast response and high time resolution in order to detect and process returning signals. Modulated continuous-wave laser range-finders determine distance by measuring the phase between the received wave and a reference signal. For this type of laser range sensor, the depth resolution will depend on the waveform frequencies used by the sensors. The higher the frequency, the better the depth resolution will be. A range sensor built at the Toshiba Corporation, Heavy Apparatus Engineering Laboratory, using the phase-shift measurement technique has been reported [82]. It has a field of view of 28 x 28 degrees and has a dynamic measurement range from $0.2m - 1.4m$ $(7.87 - 55.12 inches)$. By using a laser beam modulated at dual frequency of 1 GHz, the

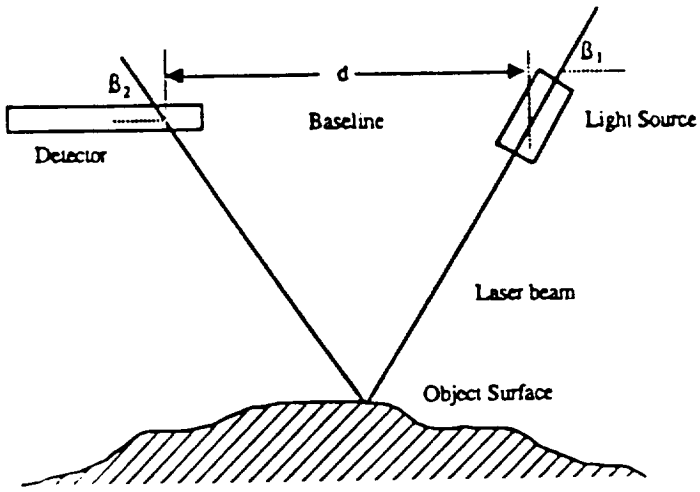


Figure 3.2. Triangulation-based laser range sensor.

sensor can achieve depth resolution of $0.075mm$ $(0.00295inches)$.

Triangulation-based laser range-finders use geometric principles to obtain range information. The sensors project a beam of light with a known shape (point, line) onto the object to be measured. The reflected light is picked up by detector array. Trigonometry is used to compute the distance of the projected spot (or line) from the sensor head. No image analysis is required during range acquisition (see Fig. 3.2). This type of sensor is mostly suitable for short range measurement. This is because longer measurement distance requires a larger baseline distance between the laser source and the detector array, and, as a result, a larger sensor . The advantages of this type of sensor are fast response time and accurate measurement. The disadvantage is the unavoidable "missing part" problem caused by occlusion. The sensors can be made in different shapes. A spot range sensor can only measure the distance of one point at a time; line range sensor can measure distance along a line on the surface of an object; a multi-spot sensor can measure many points in a special pattern at a time (see Fig. 3.3) [66].

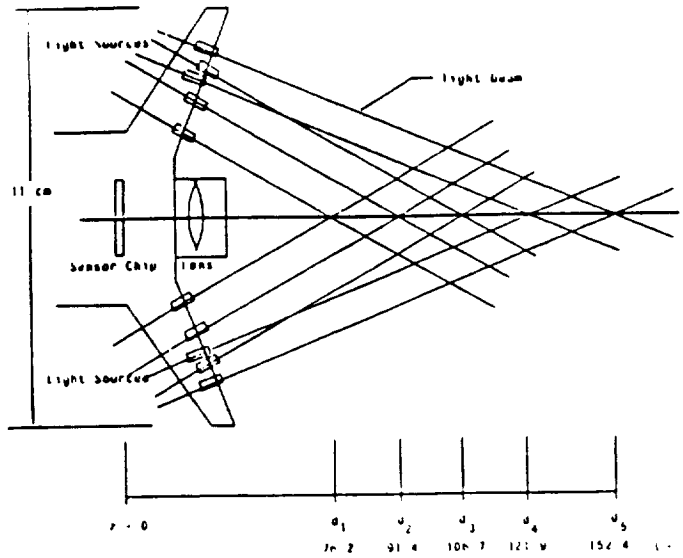


Figure 3.3. Configuration of the Multi-Spot Source Proximity Sensor [66].

The striped lighting method is perhaps the most frequently used range finding method for locating object distance. In its simplest form, a single slit of light is projected onto the scene and the scene is viewed by an offset camera. If the light strikes an object, the reflected light will be imaged on the camera's two-dimensional image plane. A one-to-one correspondence exists between all the points shown on the image and the points lying along the slit of light. The 3-D coordinates of each point which the slit of light projects on the object surfaces can be obtained from the corresponding 2-D coordinates in the image plane by optical triangulation computation. The striped lighting method in fact also belongs to the triangulation range sensing category. But it needs to process the image in order to extract the reflected light features. Because an image reflected from the surfaces of an object by a slit of light consists of straight lines and curves, it takes much less effort to extract these line or curve features than the effort in processing a general range image. The latter usually has to go through several processing steps such as segmentation, feature

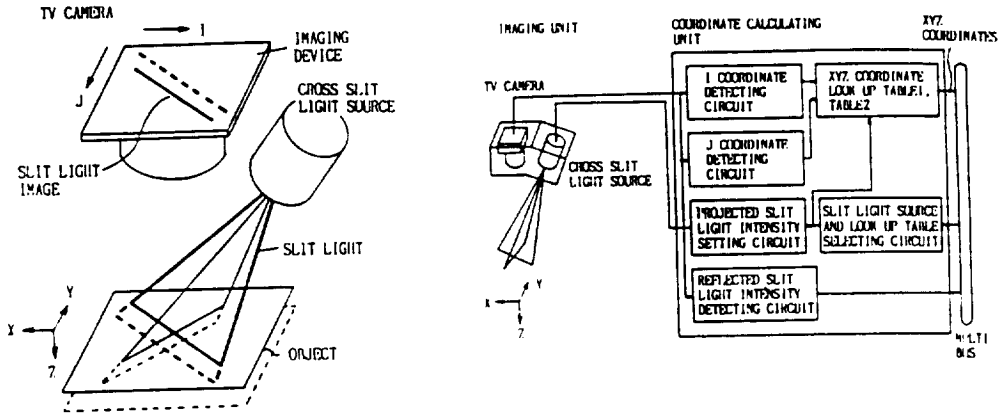


Figure 3.4. A striped light range sensor

grouping and etc., before the needed features can be extracted. Fig. 3.4 shows an example of this kind of sensor and its characteristics developed by Toyota Central Research and Development Labs [91].

Based on the same principle, many advanced striped lighting laser range sensing systems have been developed recently, which exhibit greatly improved speed, accuracy, size and other sensor performance measures. [92] has reported a cross-slit lighting sensor. It is very compact: $140mm \times 110mm \times 47mm$ $(5.51 \times 4.33 \times 1.85inches)$ in size and $800g$ in weight. It has a standoff distance of $100mm(Z)$ $(3.93inches)$ with $70mm(X) \times 70mm(Y) \times 50mm(Z)$ $(2.755 \times 2.755 \times 1.968inches)$ of measuring area and $0.25mm$ $(0.0098inches)$ of accuracy. It can measure 242 points on each slit and 6 frames per second.

To speed up the ranging process, multiple beams of light can be projected onto the area of a scene. The key to the success in using the multi-slit method is the determination of the matching between a detected stripe and its original position in the projection grid. Once the mapping is established, the range of each projected light stripe imaged by the

camera can be calculated by triangulation. The matching problem can be solved by either sequential scan projection [90], or simultaneous projection with special pattern [124] such as color-encoded projection [22], random dotted pattern projection [79], gray-code pattern projection [100] [101] and so on. Because each slit of light can be matched, the slits can be processed independently, which leads to the potential of parallel processing and thus real-time sensing.

3.2.4.2 Sensor Installation

Range sensor installation is also an important issue. The task of a sensing system is to provide object location information timely and accurately during the execution of a telerobotics task. For accuracy consideration, the sensors should be placed close to the object, such as on the robot's end-effecter. Here the problem is how to determine the exact position between the object and the gripper after the gripper picked up that object. There are several possible solutions:

1. Have a special fixture attached on the telerobot's gripper so that whenever the gripper picks up an object, the object will always be "locked" in a pre-determined position. This method is feasible from a technology point of view, but not very flexible in practice.

2. If the telerobot has two arms, each end-effecter can install one range sensor. The object's position with respect to one gripper after the object is grasped by that gripper can be accurately measured by the range sensor on another arm. If needed, that arm can move toward this arm closely enough to make accurate measurement.

3. In addition to the range sensor placed at the robot's end-effecter, a global range

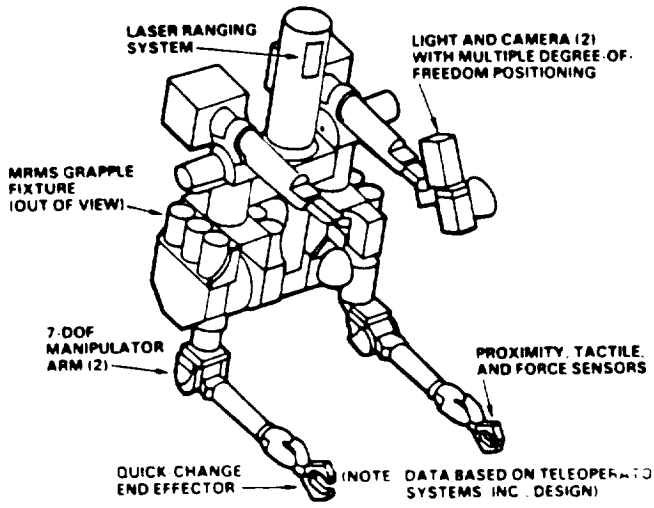


Figure 3.5. Rockwell Telepresence kit. [23]

sensor can be installed at a global position, like the position of human eyes as we have seen in the design of some space telerobots (see Fig. 3.5).

3.3 Feature Extraction & Matching Strategies

As we have said, the object localization process basically is a feature matching process, that is, finding a best estimate of transformation parameters which will align some modeled object features with certain (perhaps different types of) measured object features. Based on how feature matching is realized, the object localization algorithms can be broadly divided into two categories: the algorithms which do not involve any recognition process, and those which have more or less a recognition process involved. We call these two types of algorithms direct-localization algorithms and recognition-localization algorithms, respectively.

Obviously, the second type of algorithm has a higher intelligence level than the first. Even within the second type of algorithms, the intelligence levels can be different. Some

of them can establish matches within one object, some of them can do it within a group of the same type of objects, others can match the features within a group of different types of objects. At the highest level, the algorithm could locate unmodeled objects. To do this, a set of primitive features should be specified in a database, which will form the basic frames of any object to be constructed. Before localizing the unknown object, the algorithm must explore the object and establish a model for the object using the set of primitives.

Each type of algorithm can be further classified according to its sensing methods, the types of features used for matching, mathematical formulation methods, the types of locatable objects and so on.

3.3.1 Direct-Localization

Direct-localization algorithms are primarily used in the situations where either the working environment is highly-structured, the position relationships among the objects in the environment have previously been established approximately, or human beings can provide assistance in defining the required measurements. The telerobotics applications in most space programs meet these requirements.

Because no recognition is involved, the localization process is quite simple. The extracted features and model features can be used as inputs for direct computation. The time of localization depends on the time spent on measurements and feature extraction computation.

One method proposed by Gunnarsson and Prinz [49, 50] is based on their observation that if a set of points are measured and these measurements are distributed on the object surfaces, the best transformation is the one which will make the sum of distances between

each measured point and it's corresponding transformed surface minimal. Their idea leads to a POINT-TO-SURFACE matching strategy. Their algorithm, when formulated in mathematical terms, becomes a least squares minimization problem and can be used to locate arbitrarily-shaped objects. Usually an iterative numerical procedure is needed to solve for the problem. The numerical procedure they used is a modified Lagrange multiplier and Newton-Raphson method. Because a good initial guess can be provided due to the fact that the object's approximate location is assumed to be known, the convergence of the algorithm is guaranteed in most cases.

Gordon and Seering [48] developed a system which uses striped-light and camera sensing to gather necessary range data. The system can only locate planar objects. LINE-TO-SURFACE matching is used in their algorithm. The striped-light, when projected on the planar surfaces of the object, generates straight-line segments. The scene is then viewed by a camera. The equation of each line segment can be obtained by analyzing the corresponding image of that line segment as viewed by the camera. Three independent line segments are needed to compute the rotation and translation parameters. The fact that the line vector is perpendicular to the rotated model surface normal vector can be used to derive the rotation. The algorithm uses quaternions to represent rotation and uses a numerical method to compute it. They also give a closed-form solution for the rotation when three mutually perpendicular surfaces are sensed. The calculated rotation is then used to compute the translation.

The same striped-light and camera sensing system is also used by Rutkowski, Benton *et al.* [10, 98]. But their matching strategy is POINT-TO-SURFACE matching. In their algorithm, the measured points are from extracted line-segments, either straight or curved. Their method imposes no particular constraints on the shapes of the object surfaces, as

long as the object surfaces can be partitioned into a collection of primitive surfaces, such as planes, cylinders, or spheres. The computation is carried out by a repeated location adjustment. The location adjustment is expressed by three quantities: the rotation center, rotation axis and translation vector. To guarantee a fast convergence of their algorithm, the center of mass data point is chosen as the rotation center instead of the origin of the model's coordinate system.

When comparing these methods, we find that all of them have very high measurement accuracy and fast execution speed. For example, Gordon's system can locate an object in about 2.5 seconds, including the times spent on image acquisition, image processing and computation, and can achieve a relative accuracy of 0.002 inches in translation and 0.1 degrees in rotation when a two-inch cube is being located. It is capable of reliably assembling components with little clearance without using force controlled motion. In Gunnarsson's algorithm the measurement error is the same order of magnitude as the sensor error. These algorithms also have some problems. The problem associated with stripped-light sensing is that it requires an extra light source with a special pattern, which sometimes is inconvenient. The use of a spot sensor or line sensor has the problem of multi-measurement, e.g., the sensor has to be installed on the robot's moving part and be moved together with the robot in order to take multi-measurement. This will slow down the localization process.

High-level features can also be used to locate objects. For example, Thorne and *et al.* [115] describe an algorithm which uses features such as the radii or curvatures of a space curve along the surface to locate an object. The curvatures k or radii ρ of a space curve can be expressed as a function of the length s of the curve, e.g., $k = \kappa(s)$ (or $\rho = p(s)$), which is independent of the coordinates of the curve and is thus invariant under

rotation and translation. The algorithm assumes that there exists a particular feature line or *fingerprint* for each object. The feature line could be a certain portion of the curved edge(s) of the object or a curve on the object surface. A curvature plot along the feature line can be drawn. In the database, the feature line is specified by a set of discrete points. Associated with each point is information about its coordinate (x, y, z), radius of curvature, curvature, delta length, and total length. The total length is zero for the first point. The localization is proceeded through POINT-TO-POINT matching. The method first measures a set of discrete points along the feature line and then finds a corresponding point for each measured point. A least squares optimization algorithm is used to find the location parameters. A similar algorithm which uses Iso-Gaussian (a curve connecting points of constant Gaussian curvature) matching to localize an object has been described by Gunnarsson [49].

3.3.2 Recognition-Localization

In many applications the objects can be placed anywhere in the environment. Therefore, if a measurement is made and some sensed features are extracted, the localization system has no prior knowledge about which object or which part of the object the sensed features belong to. In this case, in order to compute the location of an object, a recognition process is needed, which will establish the matches between a set of sensed features and the model features.

There are two popular matching strategies: *tree searching* and *clustering.*

In the tree searching strategy, if there are k sensed features $S_i, i = 1 \cdots k$ and l_m model features $M_j, j = 1 \cdots l_m$ for object $O_m, m = 1 \cdots w$, a searching tree can be constructed for each known object O_m such that the tree has l_m levels, and each intermediate node

has k branches. Each path from root to leaf represents a potential matching. The total number of possible matchings, or the searching space for object O_m is l_m^k, which is very huge. To reduce the searching space, several methods have been proposed.

One algorithm proposed by Grimson, Lozano-Perez *et al.* [43] [45] is to use local geometric constraints such as distance constraint, angle constraint, direction constraint, triple-product constraint and so on to reduce the searching space. Beginning from the root of the tree down, at each node, a local constraint test is made to see if the sensed features up to that level are consistent with these constraints. If not, the entire subtree is discarded from consideration.

A similar tree searching method is used in Faugeras and Hebert's work [37, 38, 39]. Instead of local constraints, rigidity is used as the basic constraint during the tree search process. Every path from the root to an intermediate node (level k for instance) represents a partial matching. The algorithm computes a best rigid transformation $\mathbf{T}_k$ up to that level (k). Then $\mathbf{T}_k$ is applied to the next unmatched model primitive M_{k+1} and only those sensed primitives that are sufficiently close to $\mathbf{T}_k M_{k+1}$ are considered. The computations are carried out by least squares optimization techniques. As each new pair of primitives adds to the partial matching list, the new estimation of transformation has to be started over again. The algorithm's underlying paradigm is "*locating while recognizing*" which is different from the paradigm of "*locating after recognizing*" used in the Grimson *et. al.* algorithm.

Reducing the number of sensed and model features is another important method to speed up the tree searching process. The use of higher level features can effectively reduce the size of the searching tree because fewer features are usually adequate. The system developed by Bolles, Horaud *et. al.* [18, 19] is such an example. Three different types of

edges are used as the primitive features. They are: straight dihedrals, circular dihedrals, and straight tangentials. They are higher level features: one pair of matched features can determine all but one of the object's six degrees of freedom.

Clustering is another technique used in recognition-localization algorithms. The principle of clustering is very simple:

For each element in the sensed feature list

for each element in the model feature list

if they are compatible, compute a transformation candidate

put it into cluster space.

The cells with the largest counts are expected to represent the location.

While the principle is simple, the implementation is not so easy. The high dimensions (six) and huge space of clustering are just two difficulties. Different methods have been proposed to accommodate these problems. Three dimensional clustering, the use of proper size of cells and hierarchical clustering are some of them [6]. Several systems have been proposed using the clustering technique. Linnainmaa *et. al.* [76, 77, 78], Silberberg, Harwood, *et. al.* [110], and Stockman *et. al.* [112, 113, 114] are typical examples. One property of clustering is the algorithm's parallel structure, which will have an important impact on the future development of object localization algorithms.

In most algorithms, the least squares optimization is the mathematical tool to estimate the best transformation if many feature-pairs are found. During the optimization process, bad data can influence the accuracy of computation. Therefore, a filtering process usually is needed to detect and remote bad data points. Another mathematical tool is based on the use of probability theory. Bolle and Cooper [13] [15, 16, 17] have presented a statistics approach of combining pieces of information to estimate 3-D complex-object position. They

formulate the optimal object localization as a Bayesian probability estimation problem. The objective is to find the most likely transformation T that maps the model primitives onto the measured range data. The likelihood $p(Y|\mathbf{T})$ should be maximized with respect to T, where Y is the measurement data. If κ primitives have been extracted from range data and matched to model primitives, then $p(Y|\mathbf{T}) = \prod_{k=1}^{\kappa} p(Y_k|\mathbf{T})$. That means that to arrive at a global optimal solution, the maximum likelihood estimation has to be applied locally. Based on this analysis, they derived a different formula for minimizing the estimation error from the traditional least squares optimization formula. To arrive at an optimal solution, a thorough analysis of measurement errors and a good error model are needed.

3.4 Summary of the Chapter

We have discussed general object localization problems and localization strategies. Different levels of telerobot control have different requirements on the localization system, such as speed, accuracy, the level of intelligence, *etc.*. At the low level, the consideration of real-time execution and high accuracy is important. At the high level, the use of AI (artificial intelligence) technology becomes crucial.

CHAPTER 4

3-D OBJECT LOCALIZATION USING LINE-SEGMENT MATCHING

4.1 Introduction

Two important factors which will influence the applicability of object localization techniques in tele-manipulation tasks involving contact with objects are speed and accuracy. Gordon and Seering have demonstrated that the successful applications of high-speed and accurate localization system in certain automatic assembly tasks can avoid using traditional force-controlled motion or precise part fixturing assembly method and therefore will simplify robotic operations and improve the flexibility and reliability of performing these tasks [47]. Many recognition-localization based object localization algorithms such as [39] [45] [76] [112] described in Chapter 3, though having higher levels of intelligence, are not suitable for these applications. To achieve high-speed localization, the efforts should focus on the following aspects:

1. Develop fast sensing techniques.
2. Reduce the overhead for processing sensed data.
3. Develop fast algorithms to compute the location parameters. Use closed-form formulas whenever possible.

For the purpose of high-speed and accurate object localization, laser range sensing systems are the natural choice over other types of range sensors and intensity-image based sensors.

Bastuscheck [8] has discussed the techniques which can lead to the generation of range data in real-time. Assuming that a range image is the result of sensor measurement, the image consists of 500 × 500 elements and the sensor's standoff (working area) is about 40 – 80 inches (1 – 2 m) with 20 inches (0.5 m) depth (working) range, he concluded that time-of-flight laser range sensors might be able to generate range images at video rate (30 frames of a second). He also found that ratio image range sensors and slit-of-light (or multi-slit of light) range sensors, both of which are based on triangulation techniques, have the potential to produce 5 to 10 frames of range images per second. Range sensors which are based upon either the time-of-flight or triangulation principles and are able to generate 5-10 frames of range data per second have been reported recently [82, 91, 92], although the number of range readings per frame are all less than 500 × 500.

The real-time range data generation is only the first step toward high-speed localization. It has to be followed by a fast data pre-processing step and a fast location computation process in order to achieve real-time localization. The task of range data processing is to provide necessary parameters for location computation. If the range data is from a range image, to extract these parameters the system has to go through a series of pre-processing steps which include image-segmentation, region-grouping, feature extraction and feature matching. The pre-processing is usually time-consuming. To deal with this problem, we can take the following measures:

- *Using a priori knowledges about object location*

  Because the high-speed and high-accuracy requirements of locating an object are

needed in most cases only when the robot is quite close to that object and is going to perform manipulations on it, it is reasonable to assume that a priori knowledge about the object position is known approximately before the sensing system takes precise measurements on the object. Thus, the direct-localization strategy can be employed and the feature matching process is no longer necessary.

- *Using suitable sensing strategy*

  Usually only a few features need to be extracted in order to locate an object. For example, the extraction of any one of the following groups of matching features is enough to derive location (transformation) parameters:

  - Three pairs of independent matching points;
  - Two pairs of independent matching line-segments;
  - One pair of matching line-segment and one pair of matching point;
  - Three line-segments matched on three independent planer surfaces;
  - Six points matched on three independent planar surfaces;
  - Three pairs of matching planar surfaces;

  As a consequence, we do not have to use general range image(s) to obtain needed features; certain forms of sparse range data can also be used to provide us enough information. Take axis extraction as an example. If the axis of a cylinder surface is to be extracted from a range image, the usual steps are first to segment the range image into different regions each of which contains only one type of surface, find the region corresponding to the cylinder surface, then extract the cylinder's surface parameters from the region, and finally compute the axis of the cylinder surface from the surface parameters.

  The same task can also be completed by using a line range sensor. The sensor first

projects a plane of a laser beam on the cylinder surface and then takes range readings along the intersection of the laser beam and the surface. Finally, the system uses the information to compute the cylinder axis directly from these range data. Obviously, the second method is much faster than the first one because image-segmentation, region-grouping processes are not necessary in the second method.

Up to now, only limited research directed at fast and accurate object location determination has been reported in the literature.

The algorithm presented by Gunnarsson and Prinz [49, 50] uses spot range data which is obtained by using spot sensors to make measurements on the surfaces of an object to locate that object. In their algorithm for locating a planar object the measured points have to be distributed on at least three surfaces; the measured points on each surface should be distributed over as much of the surface area as possible in order to get accurate estimation. If a single spot range sensor is used, the sensor has to be moved many times to gather necessary data. Even if a multi-spot range sensor is used, it is most likely that the sensor will still need to move several times to face different surfaces in order to complete the required measurements. While feature extraction and computation are fast, the overall localization process is slowed down by the time needed to move the sensor.

The system developed by Gordon and Seering [48] uses a striped-light and camera sensing technique to locate an object. They only studied the case when the object is polygon. Again, at least three independent surfaces need to be accessed to obtain enough range data.

In this chapter, a fast and efficient localization algorithm is presented, which is based on LINE-TO-LINE matching. Any sensors which are able to make range measurements along a line or lines on the object's surfaces can be used as the sensing device in the

algorithm. Such sensors include a line range sensor, a plane of light and camera ranging system, or a multi-plane of light and camera ranging system. The measured range data will be used to extract line features, which could be boundary edges for planar surfaces or axes for surfaces of revolution. The line features are matched to corresponding modeled line features. Closed form formulas are used to carry out most computations throughout the localization process to speed up the whole localization process. The algorithm requires a highly constrained environment. That is, either the environment is a highly-structured, or the position relationships among the objects in the environment have previously been established approximately. With this assumption, fairly fast and reliable measurements can be taken, which in turn will lead to a fast and accurate localization.

In the next chapter an optimal localization algorithm will be presented. The inputs of the algorithm are not limited only to the line features. Featured points (POINT-TO-POINT matching), featured unit direction vectors (VECTOR-TO-VECTOR matching), *etc.* can also be used as the inputs to the algorithm, and there is no upper limit on the number of the features inputed. The algorithm will allow the use of redundant features to find a better solution. As will be seen, the optimal algorithm is both fast and accurate compared with current available algorithms.

Measurement rate:	5 frames/second
The number of pixels:	63
Depth resolution:	10 micrometer
Lateral resolution:	30 micrometer
Standoff:	30 mm
Depth range:	4mm
Line length:	2 mm

Table 4.1. A triangulation based line range sensor specifications

4.2 Sensing System

4.2.1 Sensor requirements

What characteristics should an ideal range sensing system have? In our previous analysis, we have discussed certain important requirements a range sensor should possess in order to perform tele-manipulation tasks. We have mentioned that at E-move level the localization system should be able to generate an updated measurement in about a second. We have investigated the accuracy requirement of performing candidate tele-manipulation tasks and concluded that the ranging system should be able to provide measurement accuracy up to about 0.01 inches (0.254 mm) in translation and 0.2 degree in rotation [96]. We have also listed the advantages of using a line range sensor or a plane of light and camera ranging system in reducing the overhead of extracting needed geometrical features from sensed data. These discussions should give a good reference for design of a range sensing systems. In addition, other factors should also be considered in

selecting sensor design parameters, such as ease of use, *etc.*. Based on the above discussion, a list of important performance parameters for an ideal range sensor is given as follows:

- Measurement dynamic range: 8 to 80 inches (0.2 to 2.0 m)
- Measurement rate: 5-10 frames per second.
- Measurement depth resolution: 0.005 inches (0.01 mm).
- Varying field of view: 35 to 10 degree.
- Use multi-line range sensor or multi-plane of light and camera ranging system.
- Compact size and light weight.

Though we have not found any range sensor in the market which meets all the above requirements, some sensors with specifications which are quite close to these requirements have been reported recently .

A line-range sensor manufactured by CyberOptics Corp. was selected as a prototype sensor in our experiments to test the performance of the algorithm. This sensor is a triangulation-based laser range sensor and its specifications are given in Table 4.1: [32]

4.2.2 3-D range sensing fundamentals

The triangulation based range-finding technique uses a known geometric structure between the source of laser beam and the detectors to determine the distance. Figure 4.1 shows the geometry of such sensors when making a spot measurement. Any position along the laser source beam or range can be determined from the position image on the detector by using the relationship:

$$Z_r = h \cdot \tan[\tan^{-1}(\frac{Z_0}{h}) + \tan^{-1}(\frac{p}{d\cos B})] \quad \text{(see[69])} \tag{4.1}$$

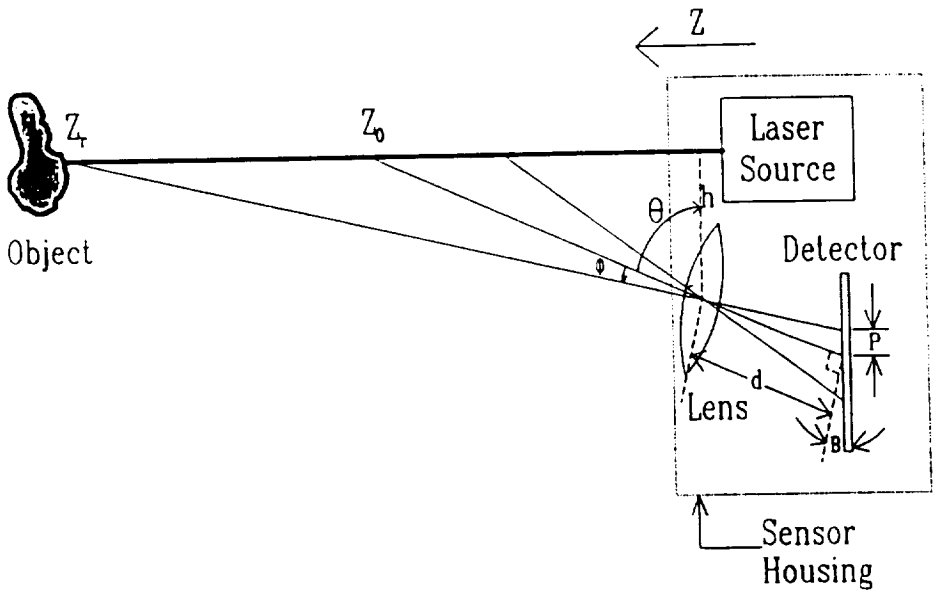


Figure 4.1. Geometry of a Laser Based Triangulation Sensor. ([69])

The line-range sensor uses the same principle. If we assign a right-hand coordinate system to the sensor (see Fig.4.2), the coordinates of points of the object in 3-D space lying on the projected line may be derived as a function of the readings of the sensor detector array. Let θ be the angle of field of view, k the number of pixels of the detector array, and e_i the range reading of pixel i, where $1 \leq i \leq k$. Then the coordinates of the point $\mathbf{v}_i$ in space corresponding to pixel i are given by

$$\mathbf{v}_i = \begin{bmatrix} \frac{1}{2}(\tan\theta)(\frac{2}{1-k}i - \frac{1+k}{1-k})e_i \\ 0 \\ e_i \end{bmatrix} \tag{4.2}$$

This information will be used in the next section as the input to extract line-segment parameters.

4.3 Line-Segment Feature Extraction

The object localization process basically consists of two steps: 1) line-segment parameter

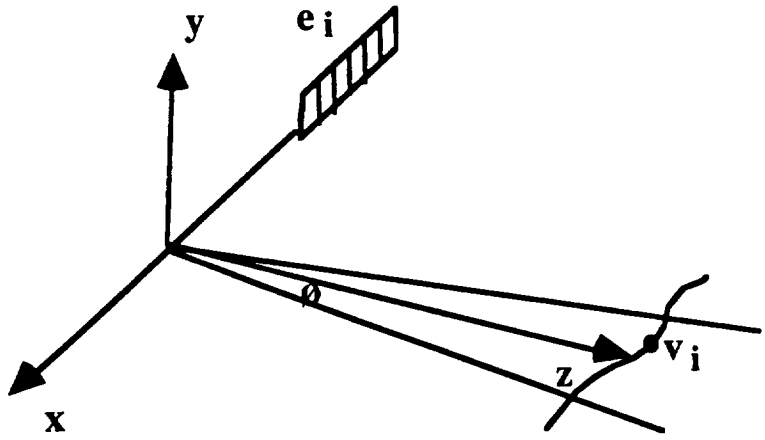


Figure 4.2. Line range sensor configuration

extraction from sensed range data and 2) location parameter determination. Two types of line-segment are considered in the feature extraction step:

1. The line-segment to be extracted is a boundary edge of a planar surface;
2. The line-segment to be extracted is an axis of a surface of revolution.

Before describing the extraction process, we give a brief discussion on the geometrical 3-D feature representation.

4.3.1 Geometrical representation of 3-D features

3-D feature representation is a basic problem in object localization. Usually one feature can be represented in several different, though equivalent, ways. But usually we will select the one which is best for certain applications in terms of efficiency, less calculation error, easy computation as the representation of that feature. In the following, we will describe the representation schemes of those 3-D features that will be used in our object localization

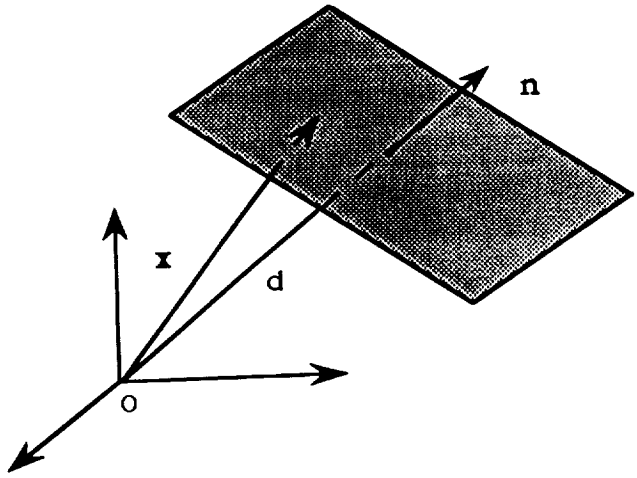


Figure 4.3. The representation of a planar surface

algorithm.

Unit direction vector: A unit direction vector is a 3×1 vector with its magnitude equals to one. That is, if N is a unit direction vector, we have

$$N = \begin{bmatrix} N_x \\ N_y \\ N_z \end{bmatrix} \tag{4.3}$$

and

$$N^T N = 1 \tag{4.4}$$

The unit direction vectors can be used to represent surface normals and the directions of line-segments such as the direction of an axis, the direction of a boundary edge and so on.

Planar surface: A planar surface in our applications is represented in either one of the following ways:

- A plane can be thought of as a set of points that satisfies

$$\mathbf{n} \cdot \mathbf{x} = d \tag{4.5}$$

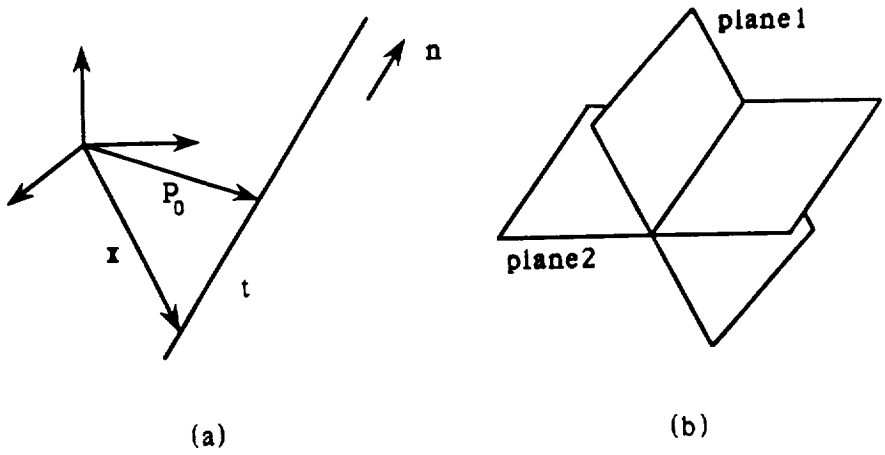


Figure 4.4. Representation of a line: (a). vector form; (b). the intersection of two planes

where $\mathbf{x} = (x, y, z)$ is any point on a surface, $\mathbf{n}$ is the surface normal and d is the distance of the plane to the origin.

- An equivalent form of Eq.(4.5) is

$$Ax + By + Cz + D = 0 \tag{4.6}$$

where A, B, C are directional parameters and at least one of them is a non-zero number.

Line: A line in our applications is represented in one of the following two ways:

- A line is represented by a pair $(\mathbf{l}_0, \mathbf{n})$, e.g., the line has the characteristics that it passes through a point $\mathbf{l}_0$ and is parallel to $\mathbf{n}$. The formula to represent the line is

$$\mathbf{l} = \mathbf{l}_0 + \nu\mathbf{n} \qquad -\infty < \nu < \infty \tag{4.7}$$

where $\mathbf{l}$ is any point on the line.

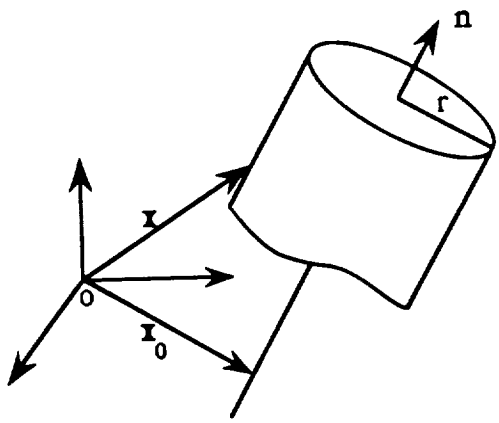


Figure 4.5. The representation of a cylindrical surface

- A line is thought of as the intersection of two planes:

$$L: \quad \begin{cases} A_1x + B_1y + C_1z + D_1 = 0 \\ A_2x + B_2y + C_2z + D_2 = 0 \end{cases} \tag{4.8}$$

Cylindrical surface: A circular cylinder is a cylinder with a circular cross section and can be specified by a triple $(\mathbf{n}, \mathbf{x}_0, r)$, where $\mathbf{n}$ specifies the direction of the axis, $\mathbf{x}_0$ is the translation vector of the axis from the origin and r represents the radius of the cylinder. There are other types of cylinders such as elliptic cylinders, hyperbolic cylinders, *etc.*. In our study, we mainly deal with circular cylinders and will call cylinders. A point $\mathbf{x} = (x, y, z)$ is on the surface of a cylinder if and only if the distance between the point and the axis of the cylinder is equal to r. Therefore, the general form of a cylindrical surface can be expressed as

$$|\mathbf{x} - \mathbf{x}_0 - ((\mathbf{x} - \mathbf{x}_0) \cdot \mathbf{n})\mathbf{n}| = r \tag{4.9}$$

Conic surface: To define a cone, we need to specify a point $\mathbf{v}$ to be the vertex of the cone, a unit vector $\mathbf{n}$ to define the direction of the axis of the cone, and an angle θ to be the half

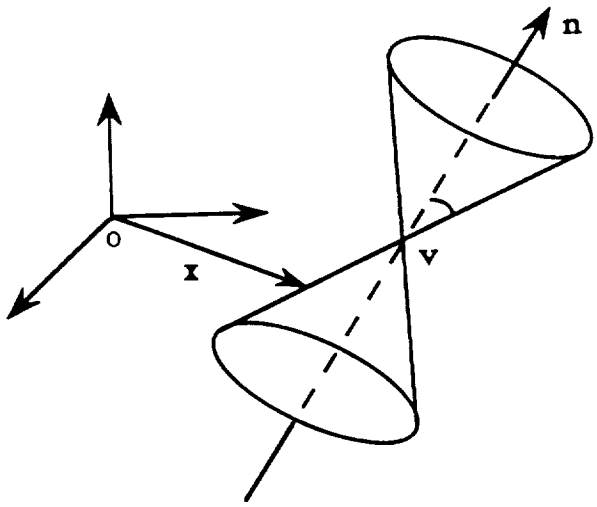


Figure 4.6. A conic surface is specified by a triple $(\mathbf{v}, \mathbf{n}, \lambda)$

angle of the cone. A point $\mathbf{x} = (x, y, z)$ is on the cone if and only if the vector $\mathbf{v} \cdot \mathbf{x}$ makes the angle θ with $\mathbf{n}$ or $-\mathbf{n}$. The general form of a circular cone can therefore be expressed as:

$$|(\mathbf{x} - \mathbf{v}) \cdot \mathbf{n}| = \lambda |\mathbf{x} - \mathbf{v}| \cdot |\mathbf{n}| \tag{4.10}$$

where

$\lambda = \cos\theta$, and θ is the half angle of the cone;

$\mathbf{v} = (v_1, v_2, v_3)$ is the vertex of the cone;

$\mathbf{n} = (n_1, n_2, n_3)$ is the unit direction vector of the axis;

$\mathbf{x}$ is any point on the cone.

The absolute value on $(\mathbf{x} - \mathbf{v}) \cdot \mathbf{n}$ is needed to give us both sides of the cone. That is, any circular cone can be specified by a triple $(\mathbf{v}, \mathbf{n}, \lambda)$.

Spherical surface: A spherical surface can be determined by the center $\mathbf{c}$ of the sphere and its radius r. That is, if a point $\mathbf{x}$ is on the surface, it will meet the equation:

$$|\mathbf{x} - \mathbf{c}| = r \tag{4.11}$$

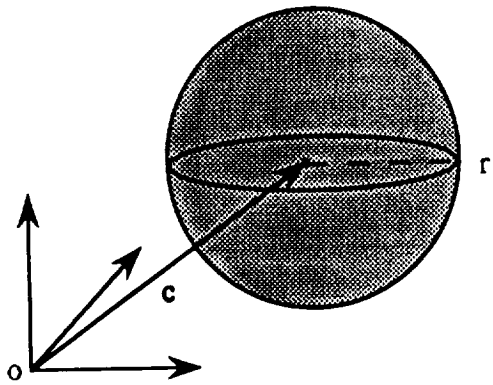


Figure 4.7. The representation of a spherical surface

4.3.2 Boundary edge parameter extraction

4.3.2.1 Boundary edge representation

A boundary edge of a polygon is usually considered as an intersection of two planar surfaces. If the shape of two planar surfaces in the polygon is concave, the boundary edge is an internal boundary edge. If the shape of the two planar surfaces is convex, it is an external boundary edge. In real applications such as in industry, the boundary edges of most parts are rounded. That is, the two planar surfaces are not directly connected. Instead, they are connected through a piece of cylindrical surface having certain radius. Therefore, we will deal with two types of boundary edges: rounded edges and sharp edges and they are modeled in computer as follows:

```
type_of_edge  =  (rounded, sharp);
boundary_edge  =  record of
        plane1,
        plane2 : plane;
        edge: type_of_edge;
        radius: real;
        end;
```

In most cases, when the object is to be displayed on screen in either wire frame form or in shaded colors, the rounded-edge feature is not displayed. It is a very useful feature, however, in our object localization process. Fig.4.8 gives an example of two different types of boundary edges. Fig. 4.8.(a) shows an object with a sharp edge. Fig. 4.8.(b) is its range image when the line range sensor is toward the edge of the object. Fig. 4.8.(c) and (d) shows an object with rounded edges and a slice of its corresponding range image taken toward the edge of the object.

4.3.2.2 Boundary edge feature extraction

The objective of boundary edge extraction is to find the parameters of the intersection of the two planar surfaces which compose the boundary edge. The process of boundary edge feature extraction is first to make a multi-slice of measurements using the line range sensor along the boundary edge and then to compute the parameters from these measured data. Based upon the sensor's accessibility on the object's surfaces, there exist two sensing patterns: 1) the sensor can access only one surface of the boundary edge (see pattern (a) in Fig.4.9), and 2) the sensor can only access both surfaces of the boundary edge (see pattern (b) and (c) in Fig.4.9).

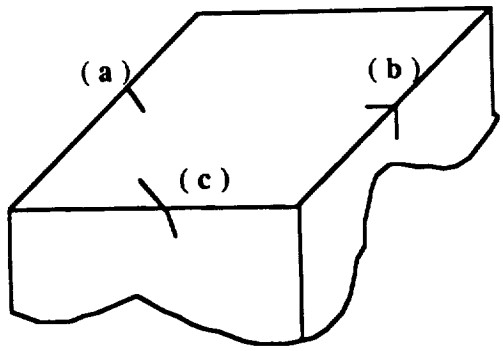


Figure 4.9. The boundary edge sensing patterns

For the sharp-type edge, the boundary edge extraction is quite easy. If the sensing pattern is (a), e.g., the sensor can only access one surface of the boundary edge, the system needs only to detect the boundary edge point from each slice of range readings. This can be done by examining the sensor detector's readings and distinguishing the part of sensor cells which has no output from another part of sensor cells which do exist readings. Usually the separation point is the corner point and the error is about one pixel.

If both surfaces are accessible, the sensing system will try to divide each slice of range data into two parts based on the changes in slope from their readings. The two parts of range data from all slices will then be used to derive two planar equations. The line-segment equation is the combination of the two equations. The derivation is a two-pass process. During the first pass, all the range data will be used to derive an approximate plane equation. A least squares fitting can be used for the purpose.

Suppose the equation of the planar surface to be determined is $ax + by + z = d$. We need to find the a, b and d from a set of k readings (x_i, y_i, z_i), where $i = 1, \cdots, k$ and

$k > 3$. The following equation can be obtained:

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ & \vdots & \\ x_k & y_k & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ -d \end{bmatrix} = \begin{bmatrix} -z_1 \\ -z_2 \\ \vdots \\ -z_k \end{bmatrix} \tag{4.12}$$

The equation can be solved by a closed form formula and have the following solution:

$$\begin{bmatrix} a \\ b \\ -r \end{bmatrix} = W \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \\ \sum z_i \end{bmatrix} \tag{4.13}$$

where $d = r/\sqrt{a^2 + b^2 + 1}$ and

$$W = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & k \end{bmatrix}^{-1} . \tag{4.14}$$

See Appendix A for the derivation of the solution.

The approximate planar equation will then be used to reject bad or unreliable data. The system will compute the distance between each measured point and the hypothesized plane. A maximum acceptable threshold value must be provided. Data having distance value exceeding the threshold will be discarded. The remained range data will be used to carry out the planar surface parameter computation again using the same formula.

For a rounded edge, if both surfaces are accessible, the basic strategy is to try to separate measured points on each slice which belong to either one of the planar surfaces from those points which belong to the rounded part of the boundary edge. The separation is still based on the change of slopes. Fig.4.11 shows a part of a slice range data measured on a rounded edge when both sides of the surfaces are accessible.

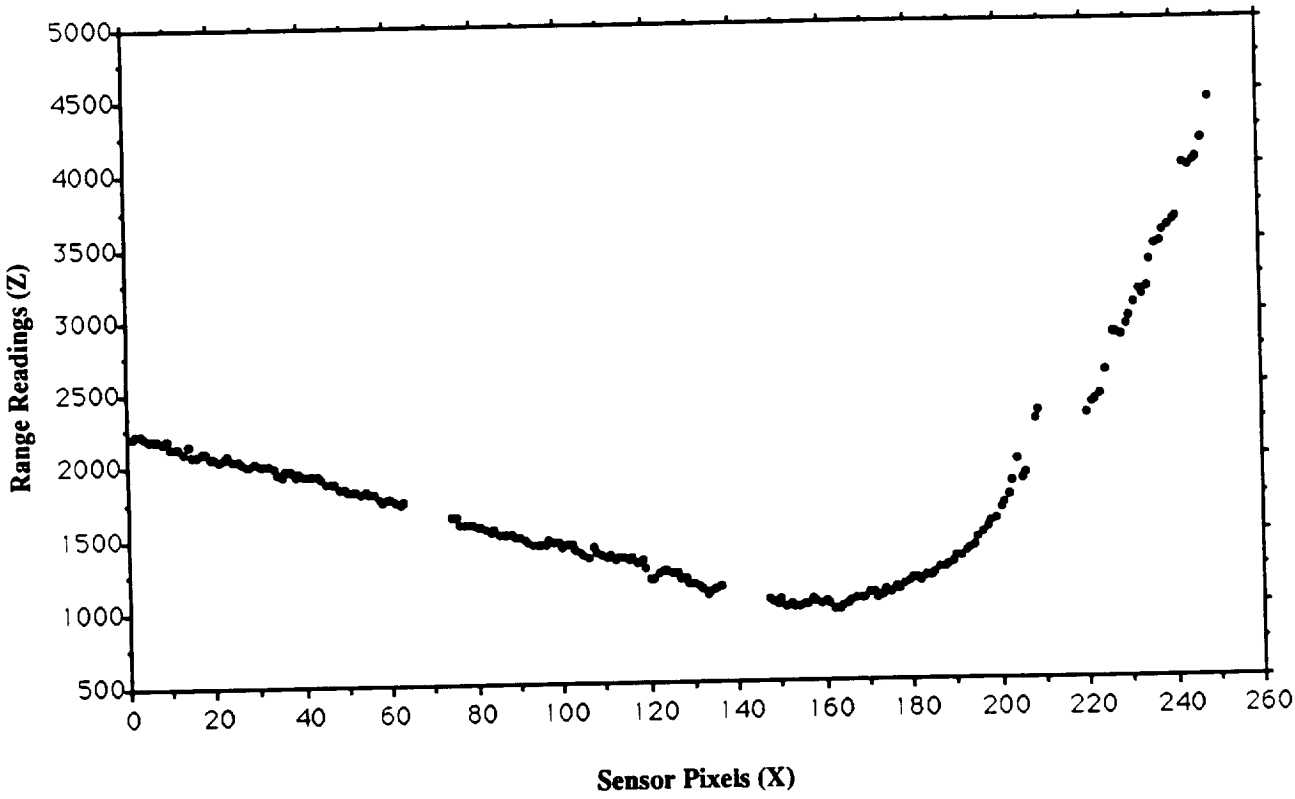


Figure 4.10. A part of a slice range data measured on a rounded edge.

If only one surface is accessible, the derivation process is not as easy as in the sharp-type edge case. But by using the geometric property of the boundary edge, this task can still be accomplished. The procedure is first to divide the range data of each slice into two groups, one belongs to the surface, another belongs to the rounded edge and then to compute the surface parameters from the two groups of range data. A planar equation can be found from the first group of range data. The second group of range data can be used to find the axis parameters of the rounded edge (cylindrical surface). Usually, the intersection of the plane of a laser beam and a cylindrical surface is an ellipse. The axis parameter determination can be done by using one of the methods given in Section 4.3.3.

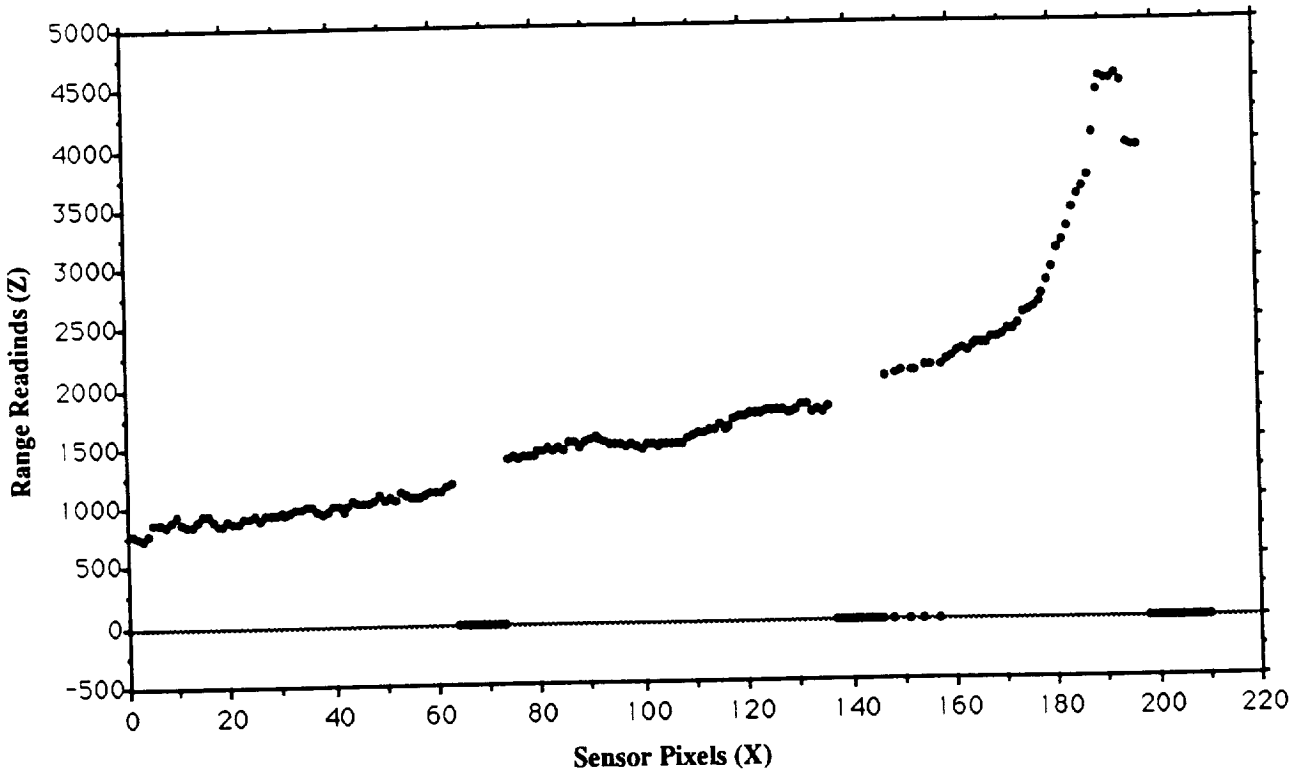


Figure 4.11. The range data where only one planar surface is accessible.

Now, we have computed one surface equation and the location of the axis of the rounded edge. From the modeled description of the boundary edge, we know the radius of the rounded edge and the angle between the two surfaces. These conditions are sufficient to derive the location of the boundary edge.

Fig.4.10 gives an example of the range image where only one planar surface of the boundary edge is accessible. By using the prior knowledge about the boundary edge, we are still able to extract the corner point from the data.

4.3.2.3 Plane parameter error estimation

The estimation is made by deriving the standard deviations of the parameters of the planar

surface. The standard deviation for a set of measured range data $z_i, i = 1 \cdots k$ is

$$\sigma_z = \sqrt{\frac{\sum(\delta z_i)^2}{k-1}} \tag{4.15}$$

where $\delta z_i = z_i - \hat{z}$ and $\hat{z}$ is the most probable value of z_i's and is equal to

$$\hat{z} = \frac{1}{k}\sum_{i=1}^{k} z_i \tag{4.16}$$

According to the error theory [9], if a variable v is derived from a set of variables $u_i \;\; i = 1, \cdots, k$ and has the function relationship

$$v = V(u_1, \cdots, u_k) \tag{4.17}$$

then, the propagation of error from $u_i \;\; i = 1, \cdots, k$ to v is expressed by the following formula:

$$\sigma_v^2 = \sum_{i=1}^{k}(\frac{\partial V}{\partial u_i})^2\sigma_{u_i}^2 + 2\sum_{i=1}^{k}\sum_{j=1\neq i}^{k}(\frac{\partial V}{\partial u_i})(\frac{\partial V}{\partial u_j})\rho_{u_iu_j}\sigma_{u_i}\sigma_{u_j} \tag{4.18}$$

where $\rho_{u_iu_j}$ is the correlation coefficient of variables u_i and u_j and is determined by the formula:

$$\rho_{u_iu_j} = \frac{1}{(k-1)\sigma_{u_i}\sigma_{u_j}}\sum_{i=1}^{k}(\delta u_i \delta u_j) \tag{4.19}$$

If all the variables are independent and uncorrelated, the equation (4.18) will be simplified as

$$\sigma_v^2 = \sum_{i=1}^{k}(\frac{\partial V}{\partial u_i})^2\sigma_{u_i}^2 \tag{4.20}$$

In our case, because variables a, b and d are dependent on $z_i, i = 1, \cdots, k$ and their relationships are determined by the equation (A.7), the standard deviation for parameters a, b and r can be explicitly expressed. Using (A.7), the variable a is expressed as

$$a = f(z_1, \cdots, z_k) \tag{4.21}$$

$$= \sum_{i=1}^{k}(W_{11}x_iz_i + W_{12}y_iz_i + W_{13}z_i) \tag{4.22}$$

If z_i's are independent, e.g., $\rho_{z_i z_j} = 0$, then

$$\sigma_a^2 = \sum_{i=1}^{k} (\frac{\partial f}{\partial z_i})^2 \sigma_{z_i}^2 \tag{4.23}$$

$$= \sum_{i=1}^{k} (W_{11}x_i + W_{12}y_i + W_{13})^2 \sigma_{z_i}^2 \tag{4.24}$$

Furthermore, if all z_i's have the same distribution property, we have

$$\sigma_a^2 = \sigma_z^2 (\sum_{i=1}^{k} (W_{11}x_i + W_{12}y_i + W_{13})^2) \tag{4.25}$$

The expression (4.25) can be further simplified.

$$\sigma_a^2 = \sigma_z^2 \sum_{i=1}^{k} (W_{11}x_i + W_{12}y_i + W_{13}) \sum_{i=1}^{k} (W_{11}x_i + W_{12}y_i + W_{13}) \tag{4.26}$$

$$= \sigma_z^2 \sum_{i=1}^{k} (W_{11}^2 x_i^2 + W_{12}^2 y_i^2 + W_{13}^2 + \tag{4.27}$$

$$2W_{11}W_{12}x_i y_i + 2W_{11}W_{13}x_i + 2W_{12}W_{13}y_i) \tag{4.28}$$

$$= \sigma_z^2 \sum_{i=1}^{3} W_{1i} (\sum_{j=1}^{3} W_{1j} M_{ji}) \tag{4.29}$$

Because the matrix W is the inverse of M, e.g., $MW = I$. Using the property of inverse matrix, we have $WM = I$. That is

$$\sum_{j=1}^{3} W_{1j} M_{ji} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{4.30}$$

Therefore

$$\sigma_a^2 = \sigma_z^2 W_{11} \tag{4.31}$$

Similarly, $\sigma_b^2, \sigma_{-d}^2$ can be derived:

$$\sigma_b^2 = \sigma_z^2 W_{22} \tag{4.32}$$

$$\sigma_{-r}^2 = \sigma_z^2 W_{33} \tag{4.33}$$

It should be noted that during the above derivation process, we do not make any assumption about the types of the distribution of the range data. That means, the result can be used no matter how the measured range data are distributed, Gaussian or non-Gaussian, as long as all the measured points have the same distribution.

The standard deviation of surface normal n and distance d can also be derived from equations (A.8) to (A.11) and (4.20). That is,

$$\begin{aligned}\sigma_{n_x}^2 &= \frac{1}{s^2}\sigma_a^2 + \frac{a^2}{s^4}\sigma_s^2 \\ &= \frac{\sigma_a^2 + n_x^2\sigma_s^2}{s^2}\end{aligned}$$

Similarly,

$$\sigma_{n_y}^2 = \frac{\sigma_b^2 + n_y^2\sigma_s^2}{s^2} \tag{4.34}$$

$$\sigma_{n_z}^2 = \frac{n_z^2\sigma_s^2}{s^2} \tag{4.35}$$

$$\sigma_d^2 = \frac{\sigma_r^2 + d^2\sigma_s^2}{s^2} \tag{4.36}$$

In matrix form, they can be expresses as

$$\begin{bmatrix} \sigma_{\mathbf{n}}^2 & \sigma_d^2 \end{bmatrix} = \frac{1}{s^2}\begin{bmatrix} 1 & 0 & 0 & \\ 0 & 1 & 0 & \mathbf{n}^2 \\ 0 & 0 & 0 & \\ 0 & 0 & 1 & d^2 \end{bmatrix}\begin{bmatrix} \sigma_a^2 \\ \sigma_b^2 \\ \sigma_r^2 \\ \sigma_s^2 \end{bmatrix} \tag{4.37}$$

And finally, from $s = \sqrt{a^2 + b^2 + 1}$, we have

$$\sigma_s^2 = \frac{a^2\sigma_a^2 + b^2\sigma_b^2}{a^2 + b^2 + 1} \tag{4.38}$$

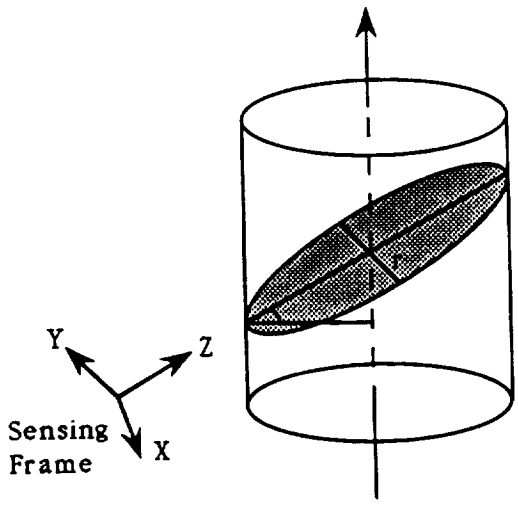


Figure 4.12. The intersection of a plane and a cylinder is an ellipse

4.3.3 Axis parameter extraction of a cylindrical surface

The most frequently seen surfaces of revolution in industry include cylindrical surfaces, conic surfaces and spherical surfaces. The extraction of cylinder parameters is the one most studied by researchers. The cylinder parameters can be extracted from range images [53], intensity images [25], or line-range finders [20, 46]. Basically, there are two methods of extracting cylinder parameters by using line range sensors, which are called one-scan and two-scan methods, respectively, depending on how many line(s) of range data are used to do the extraction.

4.3.3.1 One scan method

When a laser line beam is projected on the surface of a cylinder, the intersection of the projection plane and the cylindrical surface is an ellipse. The range data obtained from the projection can be used to derive the ellipse parameters. There are several algorithms

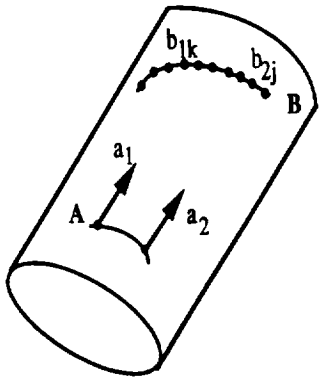


Figure 4.13. The extraction of axis from two scans ([46])

which can do ellipse fitting [21, 86, 85, 99]. Once the ellipse is fitted to a set of range data, the cylinder axis can be found very easily because of the well known characteristics of the ellipse (see Fig. 4.12): 1) Its center lies on the axis of the cylinder; 2) Its conjugate dimension is equal to the radius of the cylinder; and 3) The ratio of its principal dimension to its conjugate dimension is the cosine of the angle between the axis of the cylinder and the projection plane.

Thus, only one scan of the line range sensor on the cylinder surface is enough to solve for cylinder axis parameters. The accuracy will be improved if multiple scans are used. In this case, the final result is obtained by averaging all the results calculated from each scan. This method also has its problem. Most algorithms which have been used to derive ellipse parameters are iterative or recursive and therefore not efficient and sometimes converge very slowly [2, 21, 85, 86, 99, 118].

4.3.3.2 Two scan method

In the two-scan method [1, 46], two scans on the cylindrical surface are needed to extract axis parameters. The extraction consists of two steps. The first step is to use the two scans to compute the direction of the axis of the cylinder; the second step is to compute the position of the axis.

Step 1. Find the Direction of the Axis:

This method uses the geometric properties of a cylinder. That is, given two scans A and B on a cylinder, select the pairs a_1, b_1, a_2, b_2 so that line segments a_1b_1 and a_2b_2 are parallel (see Fig. 4.13). Then the line segments might either be parallel to the cylinder axis, or they might not. If they are not parallel to the axis, the four points a_1, b_1, a_2, b_2 are the only points that the two line segments a_1b_1, a_2b_2 intersect with the surface of the cylinder. Therefore a third scan C can be used to test the parallelism of the two lines and the cylindrical axis. This can be done by extending one of the two parallel lines until it intersects with the plane of a the third scan and testing whether the intersection point is on the ellipse defined by that scan. If it is, the two lines are also parallel to the cylindrical axis. Otherwise, they are not. [1, 46] have given detailed discussions about the validity of the argument. [46] has described an algorithm. The basic idea of the algorithm is as follows:

> Pick two points on one scan, $a_i (i = 1, 2)$; the points should be widely separated on the scan. Try to select two points $b_i (i = 1, 2)$ on the second scan so that the dot product of the two unit vectors $v_{i,i}, (i = 1, 2)$ from a_i pointing to b_i is close to 1. Store the two pair of points.
> Find several pairs of points by repeated executing the procedure.

The above procedure will be performed on two other scans. If all the vectors give

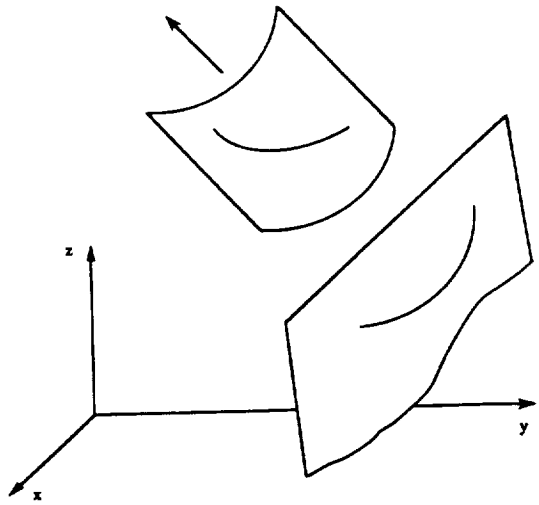


Figure 4.14. Projection of a circular cylindrical surface

roughly the same direction and are parallel to the axis of the cylinder, then the axis direction will be the average values over all the vectors.

Step 2. Find the Axis Displacement:

The axis displacement can be obtained by using a projection method. Once the axis direction is known, all the range data will be projected to a plane which is perpendicular to the axis. The projection matrix is made in such a way that the z axis in the coordinate system is rotated to the direction of the normal vector $\mathbf{n}$ of the plane. All the range data will then be projected on the plane by multiplying M with these data points and taking only x and y components of the multiplication. If the axis direction of the cylindrical surface is correct, the projected data to the plane will form an arc (see Fig 4.14). The rotation matrix has the property that it is an orthonormal matrix and that $\mathbf{Mn} = (0 \;\; 0 \;\; 1)^T$. The

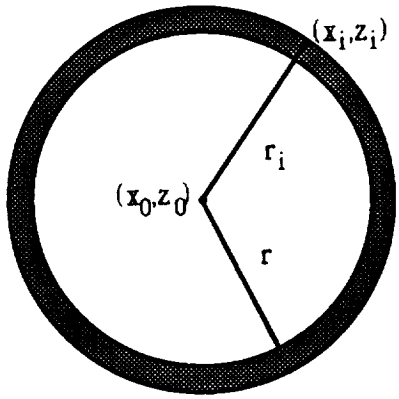


Figure 4.15. The error between the area of a circle and the measured area of the circle

matrix is not unique. One of them has the form:

$$\mathbf{M} = \begin{bmatrix} \frac{n_y}{g} & -\frac{n_x}{g} & 0 \\ \frac{n_x n_z}{g} & \frac{-n_y n_z}{g} & -g \\ n_x & n_y & n_z \end{bmatrix} \tag{4.39}$$

where $g = \sqrt{n_x^2 + n_y^2}$.

The arc will then be fitted in a circle. During the process, the circle's center and radius will be calculated. In fact, we are only interested in the circle's center.

Traditional methods use either iterative method [75] or nonlinear optimization methods [53] to derive the circle parameters. While the accuracy of these methods is promising, the execution speed will be slowed down quite markedly as the number of measured points increases. To speed up the circle parameter calculation, a closed form formula is used. The formula is based on least squares optimization, which tries to minimize the squares of the errors between the area of the circle and the area of the circle defined by the center

of the circle and the measured point.

Suppose that the projected range data are expressed as

$$\{S = (x_i, y_i) | i = 1, \cdots, k, k > 2\}$$

and the desired circle is located at $c = (x_0, y_0)$ with radius r, The measured area from the measured point (x_i, y_i) is $\pi((x_i - x_0)^2 + (y_i - y_0)^2)$ and the area of the circle will be πr^2.

The error between the two areas is (see Fig. 4.15)

$$e_i = \pi((x_i - x_0)^2 + (y_i - y_0)^2) - \pi r^2 \tag{4.40}$$

The traditional least squares optimization can be used to minimize the error function:

$$E = \sum_{i=1}^{k} e_i^2 \tag{4.41}$$

and the circle parameters can be solved from it by a closed form formula. See Appendix B for the detailed derivation.

The two dimensional coordinates of the center will then be back-projected in the original 3-D coordinate system and be used as the displacement of the axis.

The efficiency of the closed form formula can be clearly seen in Fig. 4.16, where the execution speed of the closed form algorithm for different number of sampled points is plotted against the execution speed of the iterative algorithm described in [75]. The comparison of the two algorithms was carried out on a SUN-3/50 computer without using math-coprocessor. The chart shows the following facts:

1. The closed formula algorithm is about 10 times faster than the iterative algorithm. When the number of sample points are 100, their execution times are 0.2 and 2.1 seconds, respectively, when the iterative error = 0.001. Their execution times increase to 0.84 and 8.01 seconds, respectively, as the number of sample points reach 400.

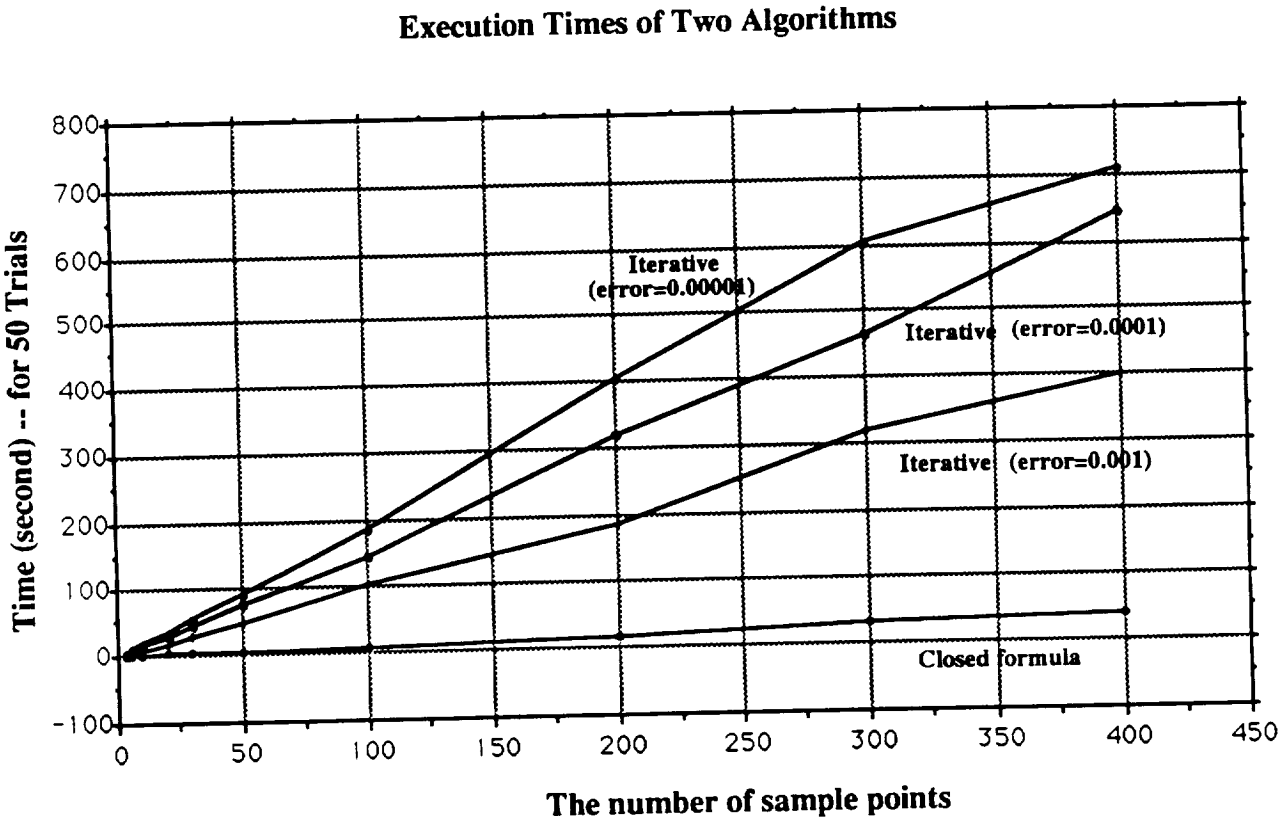


Figure 4.16. A comparison of the execution speeds of the two algorithms

2. The execution time for the iterative algorithm will increase considerably when higher accuracy is required. But the execution speed of the closed formula algorithm is less influenced by the accuracy requirement. This is because higher-accuracy means more loops for the iterative algorithm. The average execution time will increase about 20% when the error bound changes from 0.001 to 0.0001 and increase another 10% when the error bound becomes 0.00001.

Experiments have shown that the two algorithms basically have the same level of localization accuracy. See section 4.5 of this chapter for more detailed demonstration.

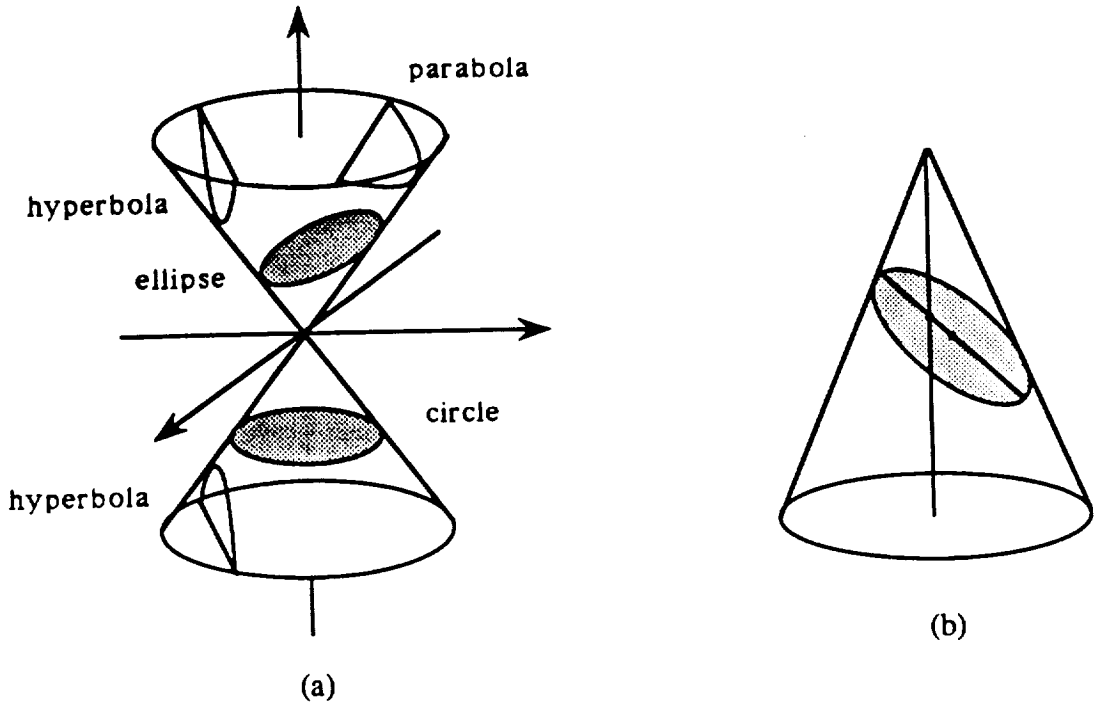


Figure 4.17. The intersection of a plane and a cone: (a) general cases; (b) ellipse and its axis

4.3.4 Axis parameter extraction of a conic surface

In contrast to the study on cylinder parameter extraction, very little research work on conic parameter extraction could be found in literature. In the following, we will discuss how the two methods described in the last section can be extended to the case of extracting conic parameters.

4.3.4.1 One scan method

In general, the intersection of a conic surface and a line of laser beam produces a quadratic curve. The shape of the curve could be a hyperbola, a parabola, an ellipse, a

circle, or a straight line depending on the relative position between the light plane and the cone (see Fig. 4.17(a)). In most cases, the shape of the curve is an ellipse. If the intersection curve is an ellipse, it has the following important property:

> The intersection point between the axis of a cone and the light plane is always on the principal axis of the ellipse (see Fig. 4.17(b)).

Section C.1 of Appendix C gives the detailed discussion about this property.

Now, suppose a conic surface is sensed by a laser line range sensor. Because the approximate position of the cone is known, it should be no difficulty in controlling the sensor's position so that the intersection of the laser beam plane and the conic section is an ellipse curve. The ellipse parameters can be extracted from the range readings along the curve by using the algorithms discussed in the previous section. Now we have the following question:

> *Can we extract the conic axis parameters by measuring only one slice of line-segment along the conic surface?*

The answer is YES, if we know the half angle θ of the cone and know that the intersection curve is an ellipse. Section C.2 of Appendix C gives the formula of carrying out the required computation and describes the derivation of the formula. Again, multiple measurements will improve the accuracy of the axis extraction.

4.3.4.2 Three scan method

The three-scan method we use for axis parameter extraction is based on the following observation:

> Given three scans on a conic surfaces, which are obtained as the intersections of three different laser beam planes with the conic surface, we select any two scans from them. If a pair of points on these two scans have the property

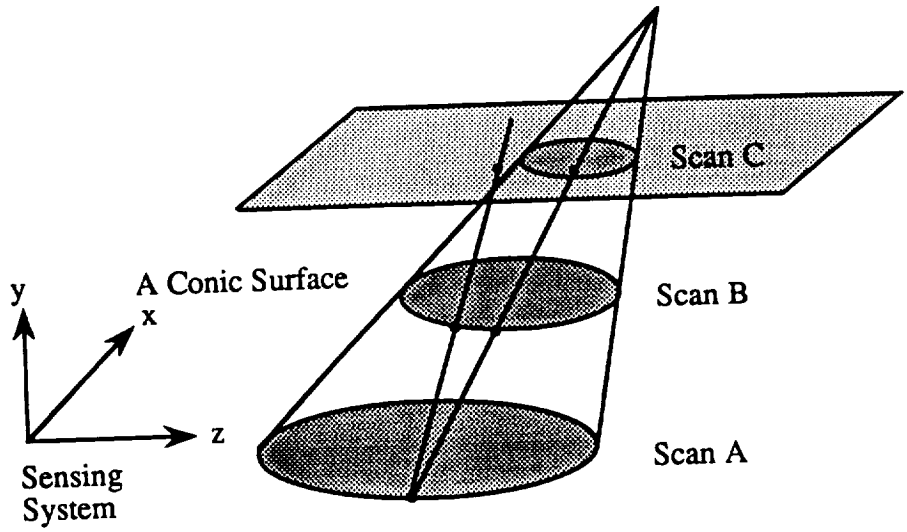


Figure 4.18. Three scan method to extract the axis of a conic surface.

that the line formed by connecting the two points passes through the vertex of the cone, then the line must intersect with the third scan. We call this line a generator of the cone.

The extraction process, like the two-scan method for cylinder axis extraction, again consists of two steps: axis direction extraction and axis position extraction.

Step 1. Find the Direction of the Axis:

The extraction of axis direction uses the geometric property of a cone that the angle between the axis of the cone and any generator of the cone is always the same. That is, suppose $\mathbf{n}_1$ and $\mathbf{n}_2$ represent the unit direction vectors of two generators of the cone and $\mathbf{n}$ represent the the unit direction vector of the axis of the cone, we have

$$\mathbf{n}_1 \cdot \mathbf{n} = \mathbf{n}_2 \cdot \mathbf{n} \tag{4.42}$$

e.g.,

$$\mathbf{n} \cdot (\mathbf{n}_1 - \mathbf{n}_2) = 0 \tag{4.43}$$

Input: Sets of measured points taken from three scans on a conic surface

$A = \{\mathbf{a}_i = (a_{i_x}, a_{i_y}, a_{i_z}) | i = 1, \cdots, m_1\}$;

$B = \{\mathbf{b}_i = (b_{i_x}, b_{i_y}, b_{i_z}) | i = 1, \cdots, m_2\}$;

$C = \{\mathbf{c}_i = (c_{i_x}, c_{i_y}, c_{i_z}) | i = 1, \cdots, m_3\}$;

Output: A pair of points $(\mathbf{a}_1, \mathbf{b}_k)$.

Begin

01 Find a best-fit plane ζ from the set of measured points $\mathbf{c}_i$;

02 Select an arbitrary point on scan A, $\mathbf{a}_1$;

03 For i := 1 to m_2 do

04 Compute the intersection point $\mathbf{p}_i$ between the line $\overline{a_1 b_i}$ and the plane ζ;

05 Find the distance d_i between $\mathbf{p}_i$ and scan C and store the value;

{* After steps 03-06, a distance curve about d_i is obtained *}

06 Smooth the distance curve ;

07 Find the point $\mathbf{b}_k$ on the the curve with minimum the distance to Scan C;

End

Figure 4.19. Three scan algorithm to find a generator of a cone.

This means that the vector $\mathbf{n}$ is perpendicular to vector $\mathbf{n}_1 - \mathbf{n}_2$. Given three different generators $\mathbf{n}_i, i = 1, 2, 3$, we can derive the the axis direction vector by the formula:

$$\mathbf{n} = \frac{(\mathbf{n}_1 - \mathbf{n}_2) \times (\mathbf{n}_2 \times \mathbf{n}_3)}{|(\mathbf{n}_1 - \mathbf{n}_2) \times (\mathbf{n}_2 \times \mathbf{n}_3)|} \tag{4.44}$$

and it is unique.

The algorithm for finding a generator of a cone can be described as follows:

Three consecutive scans are taken on a conic surface and they are named A, B

and C from the bottom scan to the top scan respectively (see Fig. 4.18). Pick a point on scan A, $\mathbf{a}_1$. Try to select a point $\mathbf{b}_i$ on scan B so that the line formed by $(\mathbf{a}_1, \mathbf{b}_i)$ is close to scan C (see Fig4.19 and 4.20 for algorithms).

One potential problem in using the above algorithm is that the existence of noise can lead to a false local minimum distance value between the line and the scan. The real implementation of the algorithm actually computes more points. The distance values should tend to approach minimum and then start increasing. The algorithm will smooth the distance curve and find the minimum point from the smoothed curve and take this point as the required $\mathbf{b}_i$.

An implementation of the algorithm is described in Fig. 4.19. In that algorithm, the step 05 – "find the distance d_i between $\mathbf{p}_i$ and scan C" may require significant time to compute. We can do better if the three scan lines are parallel. In real applications, when a multiple-scan sensor is used, the scan lines are usually parallel. Even if only a single scanner is used, to make parallel scans is not very difficult. In this case, if we use same sensor coordinate frame described in Fig.4.2, then the three scan planes have the equations $y = w_i, i = 1,2,3$. For convenience, the scan plane A can be assigned to $y = 0$. After such an assignment, the step 01 of the algorithm in Fig.4.19 is unnecessary. In addition, if the x values are used to represent the sensor cell's positions, their values are integers which will further simplify the implementation of the algorithm. A revised version of the algorithm is shown in Fig. 4.20.

Among these three scans, we can arbitrarily assign any scan as scan A, scan B or scan C. But in practice, we usually select the scan which is closest to the bottom of the cone as scan A, the middle scan as scan B and the scan which is closest to the vertex of the cone as scan C. There are two obvious advantages of doing such kind of assignment:

Input: Sets of measured points taken from three parallel scans on a conic surface

$A = \{\mathbf{a}_i = (a_{i_x}, 0, a_{i_z}) | i = 1, \cdots, m_1\}$;

$B = \{\mathbf{b}_i = (b_{i_x}, w_2, b_{i_z}) | i = 1, \cdots, m_2\}$;

$C = \{\mathbf{c}_i = (c_{i_x}, w_3, c_{i_z}) | i = 1, \cdots, m_3\}$;

Output: A pair of point $(\mathbf{a}_1, \mathbf{b}_k)$.

Begin

01 Select a point on scan A, $\mathbf{a}_1$;

02 For i := 1 to m_2 do

03 Compute the intersection point $\mathbf{p}_i$ between the line $\overline{a_1 b_i}$ and the plane $y = w_3$;

04 If a measured point $\mathbf{c}_i$ on C scan can be found so that c_{i_x} is equal to round(p_{i_x}), go step 06;

05 Go to next loop;

06 Find the difference Δz_i between p_{i_x} and c_{i_x} and store the value;

07 Go to next loop;

08 Smooth the Δz_i's data;

09 Find a point $(x_1, \Delta z_k)$ on the the smoothed curve, which is the extreme point of the curve;

10 Find the point $\mathbf{b}_o$ corresponding to that point;

End

Figure 4.20. Revised three-scan algorithm: all scans are parallel.

1. By using such an assignment, a little change on b_i will result in big change in Δz_i. As a result, the distance curve will have a sharper shape and it is easier to find the required point;

2. Relatively few intersection points $\mathbf{p}_k$ computed from step 03 have corresponding $\mathbf{c}$ points. As a result, relatively few Δz_k values will be stored and less effort will be spent on processing it.

A computer simulation was performed to compare the performance of two different types of assignments. A cone having a half angle of 30^o is placed at a position with respect to the sensor frame such that the vertex's position is $(15, 15, 15)$ and the axis direction vector of the cone is $(2, 3, 4)$. Three scan planes are $y = 0$, $y = 2$ and $y = 5$, respectively. All the units are in centimeters. Each scan has 255 measurement points, which have covered about one third of the intersection ellipse. Fig.4.21 shows the distance curve using our preferred scan assignment. Fig.4.22 shows the resulting curve of using different assignment, where the scan B is assigned to the scan closest to the vertex of the cone and the scan C is assigned to the middle scan. The measurement noise has a normal distribution with a standard deviation of 0.3. In Fig.4.21, only about one half of the intersection points have their correspondence. On the contrary, all the intersection points can find their correspondence in Fig.4.22. The difference in the sharpness of the resulting curves is also obvious.

Two methods have been tested for Δz_i data smoothing. One method is called curve fitting. In this method, all the measured data are used to fit into a single quadric curve in the form of $z = a + bx + cx^2$. In the second method we do not try to use only one curve to fit all the data. Instead, each portion of data will be fitted into a quadric curve, with each quadric curve having different parameters. The new curve as a whole is smoothed.

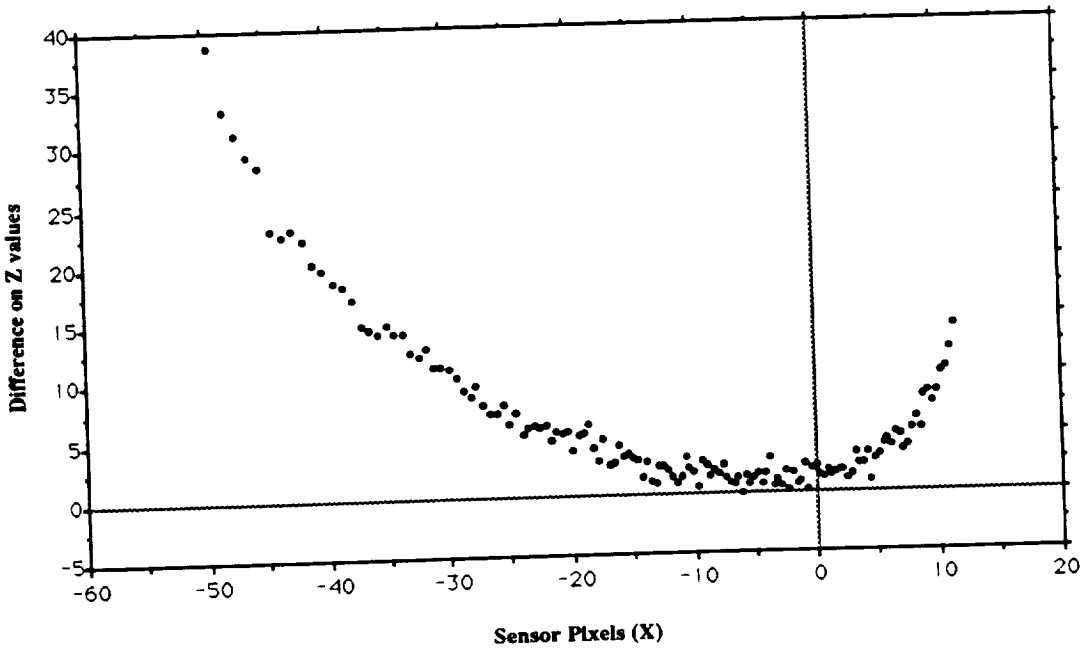


Figure 4.21. Distance curve using the preferred scan assignment.

We will take the extreme point of the smoothed curve as the required point. Appendix D gives the detailed description of the two methods for smoothing. The two methods give similar results. No matter which method is used, one thing should keep in mind, namely, the objective of data smoothing is to determine the point which will let us find a generator of the cone. Therefore, it is important to observe the shape of the curve as whole through smoothing, not to find the exact value at each point.

The algorithm just described is quite insensitive to sensor measurement noise and roughness of the object surface. This is because the point which will lead us to find a generator of the cone is obtained by observing the tendency of the distance curve instead of simply computing a minimum or maximum point from the data set.

To derive the axis direction, at least three generators are needed. In real applications,

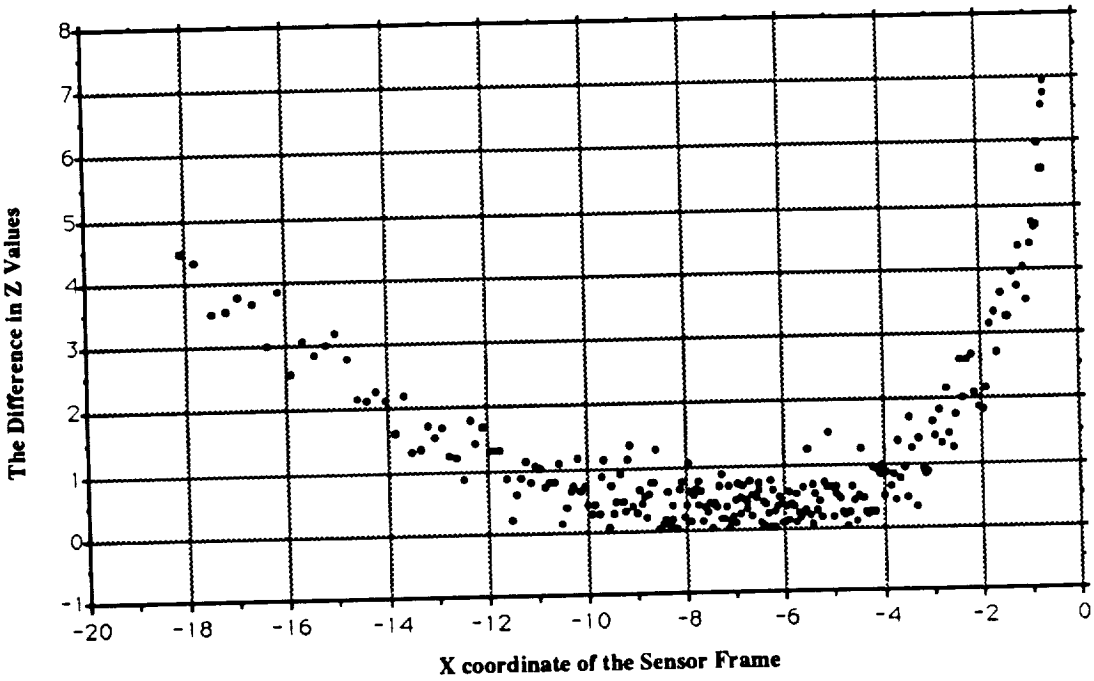


Figure 4.22. Distance curve using a different scan assignment.

we usually compute more than that and average the results to obtain a better approximation.

Step 2. Find the Axis Displacement:

The axis displacement computation procedure for a conic surface is the same as the procedure for computing the axis displacement of a cylindrical surface. Once the conic axis direction is derived, the projection plane and projection matrix can be found. Instead of projecting range data, all the generators will be projected onto that plane. The result of the projections is a set of two-dimensional straight lines. Usually, these lines are not parallel. Thus, each pair of lines will have a cross point. If $n(n > 3)$ lines are projected, the number of cross points is $\frac{n(n-1)}{2}$. These cross points are distributed on a very small circle area. The center of the area can be derived from these cross points by using the

closed-formula circle parameter estimation algorithm described in last section.

4.3.5 Axis parameter extraction of other surface of revolution

The three-scan method can in fact be used for any cylindrical type of surfaces, such as a circular cylinder, elliptic cylinder, hyperbolic cylinder and so on. The axis direction vector can be determined in the manner described in Fig. 4.20. All the measured points will then be projected onto a projection plane in order to find the axis displacement. Except for the case of a circular cylinder where a closed form formula is available to find the center of the projected arc, an optimal quadric curve fitting is needed to find the quadric curve parameters. For other types of surfaces of revolution, we have to analyze the geometrical properties in order to apply proper methods.

4.4 Location Determination

Suppose two non-parallel line-segments, $\tilde{l}_i$, $i = 1,2$, either boundary edges or axes or a combination of them, have been extracted, and are expressed as

$$\tilde{l}_i : \ \tilde{\mathbf{l}}_i = \tilde{\mathbf{l}}_{0_i} + \nu \tilde{\mathbf{n}}_i \quad i = 1,2 \quad -\infty < \nu < +\infty \tag{4.45}$$

Here $\tilde{\mathbf{l}}_{0_i}$ are the vectors from the origin of the sensing coordinate system perpendicular to and intersecting the lines the line-segments lie on, and $\tilde{\mathbf{n}}_i$ are the unit direction vectors of the line-segments viewed from the sensing coordinate system. Their corresponding line-segments l_i in object model are expressed as

$$l_i : \ \mathbf{l}_i = \mathbf{l}_{0_i} + \nu \mathbf{n}_i \quad i = 1,2 \quad -\infty < \nu < +\infty \tag{4.46}$$

where $\mathbf{l}_{0_i}$ are the vectors from the origin of the object coordinate system perpendicular to and intersecting the lines on which the line-segments lie, and $\mathbf{n}_i$ are the unit direction vector of the line-segments. If the two pairs of measured line-segments $(\tilde{\mathbf{l}}_1, \tilde{\mathbf{l}}_2)$ and modeled line-segments $(\mathbf{l}_1, \mathbf{l}_2)$ have the same angle and the same distance, a unique transformation matrix can be derived from them, which will transform the modeled line-segments to their corresponding measured line-segments. That is, the location determination is to find a transformation matrix $\mathbf{T}$ such that

$$\tilde{\mathbf{l}}_i = \mathbf{T}\mathbf{l}_i \quad i = 1,2 \tag{4.47}$$

In the case where they are not well matched, more matching pairs must be found in order to derive an optimal solution, which is the task of Chapter 5.

4.4.0.1 Determining the Rotation Parameters

The rotation can be represented in several ways. In our case, because two pairs of matching vectors are given, it is natural to use the properties of vector algebra to derive the rotation parameters. Thus the best way of representing the rotation is to use the axis-angle representation method. The rotation axis $\mathbf{r}$ can be easily derived by using the fact that the rotation axis should be perpendicular to both $(\mathbf{n}_1\text{-}\tilde{\mathbf{n}}_1)$ and $(\mathbf{n}_2\text{-}\tilde{\mathbf{n}}_2)$ and has the following form:

$$\mathbf{r} = \frac{(\mathbf{n}_1 - \tilde{\mathbf{n}}_1) \times (\mathbf{n}_2 - \tilde{\mathbf{n}}_2)}{\|(\mathbf{n}_1 - \tilde{\mathbf{n}}_1) \times (\mathbf{n}_2 - \tilde{\mathbf{n}}_2)\|} \tag{4.48}$$

to within an ambiguity of π radian.

Once the rotation axis $\mathbf{r}$ has been computed, the rotation angle can be determined by using the following formula:

$$\cos\alpha = \frac{\mathbf{n}_i \cdot \tilde{\mathbf{n}}_i - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2}{1 - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2} \tag{4.49}$$

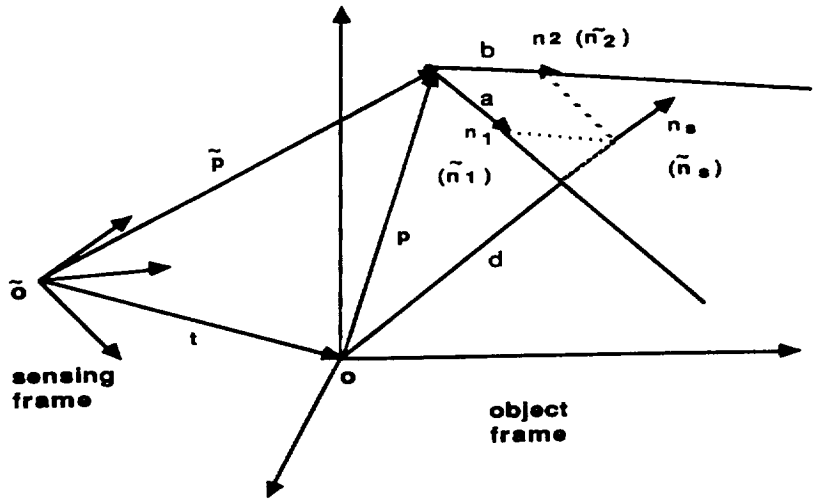


Figure 4.23. Two line-segments are in the same plane

Given values for $\mathbf{r}$ and α, the rotation submatrix of the transformation matrix can be determined. For a detailed derivation, see Appendix E.

4.4.0.2 Determining the Translation

When considering the determination of the displacement, we have to deal with two possible cases: 1) the two line-segments are located in the same plane; and 2) they are in different planes.

In the first case (see Fig 4.23), the intrinsic geometric properties of the line-segments are used to derive the formulas. From the given conditions, the intersection point $\mathbf{p}$ of the two lines emanating from the line-segments and the parameters $(\mathbf{n}_s, d)$ of the plane determined by the two line-segments, all expressed in the object coordinate system, can be calculated. Also from Fig.4.23, we have the following equation:

$$\mathbf{o} = \mathbf{p} + a\mathbf{n}_1 + b\mathbf{n}_2 - d\mathbf{n}_s \tag{4.50}$$

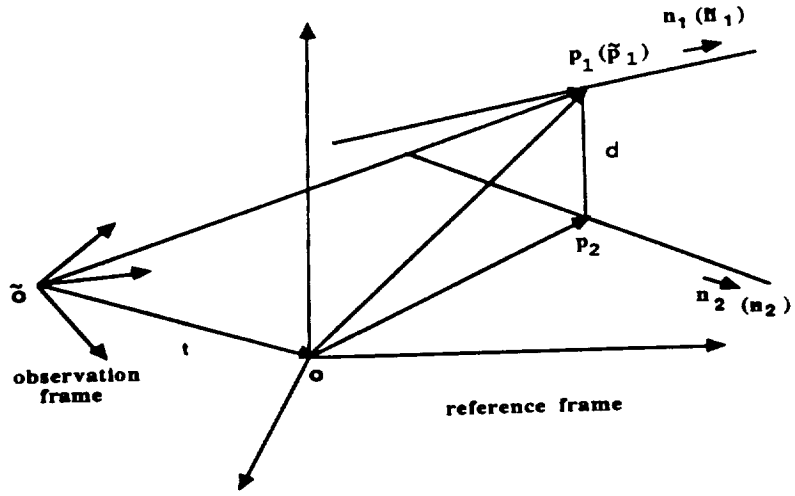


Figure 4.24. The cross point of two line-segments

The parameters a and b can be derived by using the properties of vector algebra:

$$\begin{aligned} a &= ((d\mathbf{n}_s - \mathbf{p}) - b\mathbf{n}_2) \cdot \mathbf{n}_1 \\ b &= \frac{(d\mathbf{n}_s - \mathbf{p}) \cdot \mathbf{n}_2 - ((d\mathbf{n}_s - \mathbf{p}) \cdot \mathbf{n}_1)(\mathbf{n}_1 \cdot \mathbf{n}_2)}{1 - (\mathbf{n}_1 \cdot \mathbf{n}_2)^2} \end{aligned}$$

Using the constants d, a, b and the extracted line-segment parameters, the translation vector $\mathbf{t}$ is given by

$$\mathbf{t} = \tilde{\mathbf{p}} + a\tilde{\mathbf{n}}_1 + b\tilde{\mathbf{n}}_2 - d\tilde{\mathbf{n}}_s \tag{4.51}$$

where the parameters $\tilde{\mathbf{n}}_1$, $\tilde{\mathbf{n}}_2$, $\tilde{\mathbf{n}}_s$, and $\tilde{\mathbf{p}}$ are the same parameters corresponding to Eq.(4.50) but are observed from sensing coordinate frame, and can be computed from the the extracted line-segment. The correctness of the Eq.(4.51) is based on the fact that during any transformation, the relative position and orientation of vectors $\mathbf{p}$, $\mathbf{n}_1$, $\mathbf{n}_2$ and $\mathbf{n}_s$ are unchanged.

For the second case where two line-segments are in different planes, we can first find the two cross points of a line which is perpendicular to both of the two line-segments and

has the shortest distance. See Fig.4.24. Once the cross points have been found, we can use the properties of vector algebra to compute the displacement vector, going through the same procedure as described in the the first case.

If the modeled line-segments are given as Eq. (4.46), the coordinates of cross points $\mathbf{p}_1$ and $\mathbf{p}_2$ can be derived by computing the corresponding ν values of each of the line-segments represented by Eq. (4.46). Thus,

$$\nu_1 = \frac{(\mathbf{l}_{0_1} - \mathbf{l}_{0_2}) \cdot \mathbf{n}_1 - (\mathbf{n}_1 \cdot \mathbf{n}_2)(\mathbf{l}_{0_1} - lo2) \cdot \mathbf{n}_2}{(\mathbf{n}_1 \cdot \mathbf{n}_2)^2 - 1} \tag{4.52}$$

and

$$\nu_2 = \frac{-(\mathbf{l}_{0_1} - lo2) \cdot \mathbf{n}_2 + (\mathbf{n}_1 \cdot \mathbf{n}_2)(\mathbf{l}_{0_1} - \mathbf{l}_{0_2}) \cdot \mathbf{n}_1}{(\mathbf{n}_1 \cdot \mathbf{n}_2)^2 - 1} \tag{4.53}$$

Because the two line-segments are non-parallel, $(\mathbf{n}_1 \cdot \mathbf{n}_2)^2 - 1$ will not be equal to zero and therefore we do not need to consider the overflow problem here.

The coordinates of the same two cross points which, however, are observed from the sensing system $\tilde{\mathbf{l}}$ and $\tilde{\mathbf{p}}_2$ can be derived by using the same forms of Eq.(4.52) and (4.53). What we need to do is to change all the vector notation from modeled notation into tilde notation.

4.5 Experiment Results

4.5.1 Accuracy Performance on the Circular Arc Parameter Estimation

Computer simulations have been carried out on a SUN computer to compare the accuracy performance of the circular arc parameter estimation between the closed formula method and the iterative method [75]. After each simulation process, the estimated circular center

and radius are calculated. Because we are only interested in the the circular center, the results of circular radius estimation are not shown here. In fact, only the estimations of center's x-coordinate are demonstrated in the following figures.

The simulation results are presented in Fig. 4.25 and Fig. 4.26. In Fig. 4.25, all the data points are evenly taken from an arc of 180° and radius 20.0 units, and the center of the arc is placed at the origin (0.0, 0.0) of the 2-D coordinate system. The data points are corrupted with a normal distribution along both X and Z axes. Three different levels of corruptions are used in the simulation, e.g., with a standard deviation of 0.1, 1.0 and 2.0 respectively. The number of measured data points range from a minimum of 3 points to a maximum of 400 points. For each given set of data points and each corruption level setting, 25 trials are performed for both of the two algorithms. The results are then used to compute the standard deviations on the center's location. The computed standard deviations are drawn on the figure. Fig. 4.26 shows the simulation results performed on an arc of 60°.

From the two figures, we have the following observations:

- When the input noise is small, the iterative method provides a better estimation.
- As the input noise increases, the difference on accuracy performance of the two algorithms becomes smaller. They have the same level of accuracy performance when the input noise reaches to the level of standard deviation 1.0.
- If the input noise increases further, the closed formula gives a better result.
- In any case, the output error level is much less than the input noise level. For example, when the input has a noise level of standard deviation 1.0 and contains a total number of 100 measured data points on an arc of 180°, the output error only has a standard deviation of 0.07 for both of the algorithms.

In practice, the selection of a best algorithm is depended on many factors such as the required execution speed, the roughness of object surfaces, sensor quality, the number of measured points *etc.*.

4.5.2 The Accuracy Issues of the Revised Three-Scan Algorithm

The key step of the cone parameter extraction is the extraction of a cone's generator. We have proposed a three-scan algorithm in Section 4.3.4.2 for this purpose. A revised three-scan algorithm in which all three scans are parallel was illustrated in Fig. 4.20. In this section, computer simulations are set to further demonstrate the performance of the algorithm under different conditions. The cone we used in the simulations has the same parameters: a half angle of 30^{o}, a vertex position of $(15, 15, 15)$. The scan A and C are in planes $Y = 0$ and $Y = 5$, respectively. The simulation process is as follows:

> For each scan B position and each simulated measurement noise level, 30 trials are performed. During each trial, the simulation program first produces a set of normal-distributed measurement data, and then runs the revised three-scan algorithm. The output of the algorithm is a point $\mathbf{b}_i$ on scan B such that the line $\overline{\mathbf{a}_1\mathbf{b}_i}$ represents a generator of the cone. Suppose the ideal point is $\mathbf{b}_o$, the erroron pixel will be $\delta e_i = |i - o|$. After the 30 trials, the average error on pixel position is computed and stored in the table.

Table 4.2 shows the accuracy performance of the algorithm with different measurement noise levels and different scan B positions. The cone direction vector is $(2, 3, 4)$. Each scan covers about one third of the intersection ellipse and consists of 250 pixels. The first column of the Table 4.2 lists the simulated measurement noise levels; the next three columns give the average computed pixel position errors with the scan B being positioned at planes $Y = 1$, $Y = 2$ and $Y = 3$, respectively. The table clearly tells us that the noise levels

have no influence on the output accuracy, e.g., the algorithm is noise-insensitive. In fact, the table even shows a slight improvement on the output accuracy when the measurement noise is increased from $Sd = 2.5$ level to $Sd = 5.0$ level. The table also shows that the pixel position errors become smaller as the middle scan B moves closer to the bottom scan A. If the pixel position errors are converted into corresponding generator's direction errors, the actual average angular errors on the generator's direction are 0.369°, 0.386° and 0.513° for scan B at planes $Y = 1$, $Y = 2$ and $Y = 3$, respectively.

Table 4.3 shows the accuracy performance when all the scans are perpendicular to the axis of the cone, e.g., the cone direction vector equals to $(0, 1, 0)$. The algorithm's noise-insensitive property is also demonstrated in the table. The table also shows that the accuracy becomes worse as the middle scan B moves closer to the bottom scan A.

Table 4.4 and Fig. 4.27 present the detailed simulation results taken from a set of randomly produced data points. Table 4.3 lists the cone and scan parameter settings and gives a partial listing of the X values corresponding to the pixels on scan B and C, their corresponding theoretical Z values and the corrupted Z values. A pixel on scan A (ellipse 1) is selected as the $\mathbf{a}_1$ in the three-scan algorithm. Fig. 4.27(a) shows the theoretical distance curve and the distance points computed from corrupted Z values; and (b) shows the smoothed distance curve after the three-scan algorithm is implemented. In this simulation, the final selected pixel is coincident with the ideal pixel, e.g., no error.

Fig. 4.28 presents the simulation result taken from another random set of measured data when the scans are perpendicular to the axis of the cone. After the execution of the algorithm, the output (the final selected pixel position) is three pixels away from the ideal pixel position. In the figure, the pixel positions are converted into corresponding x coordinates. Thus, it is hard to see the exact pixel position error directly from the figure.

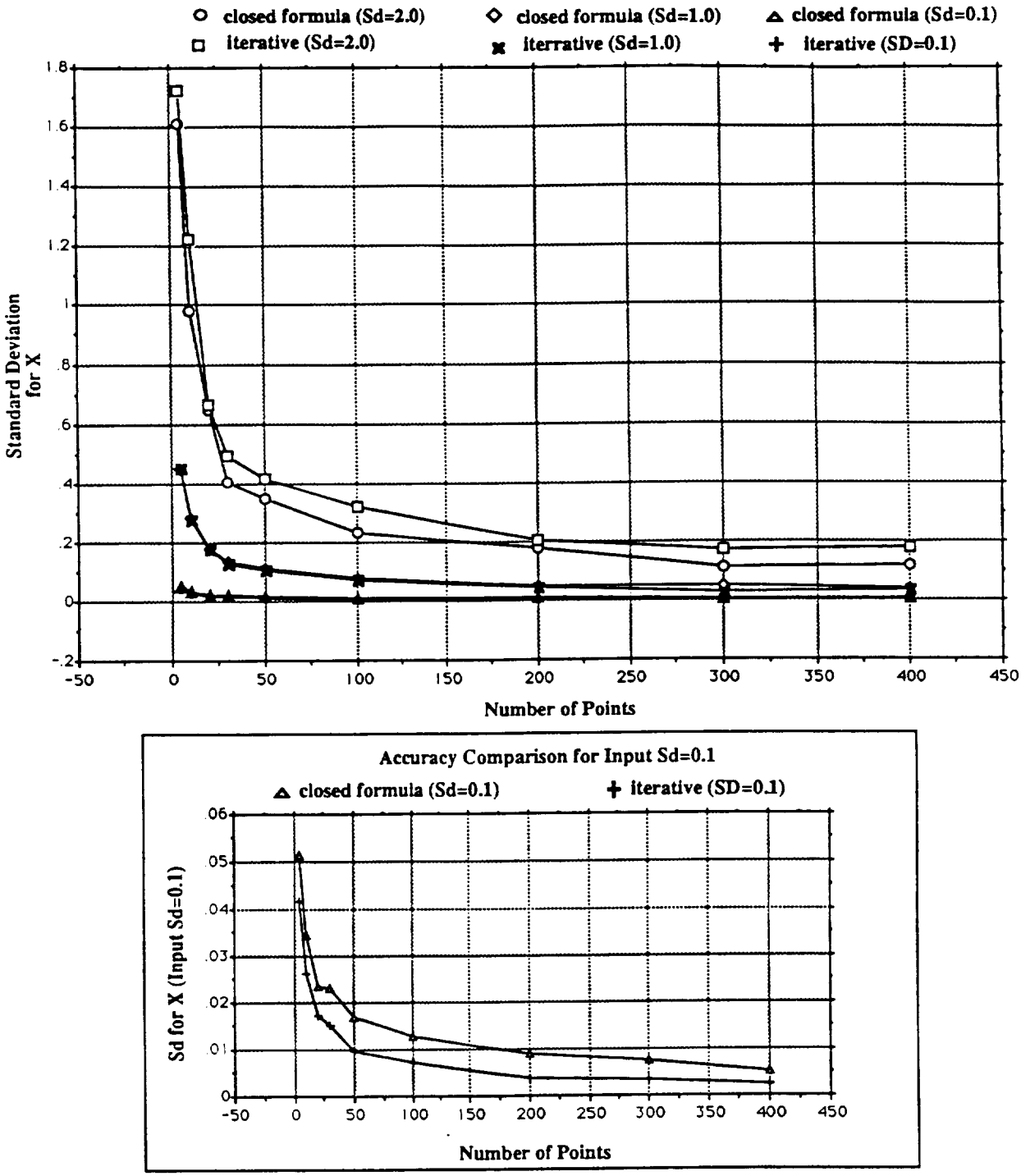
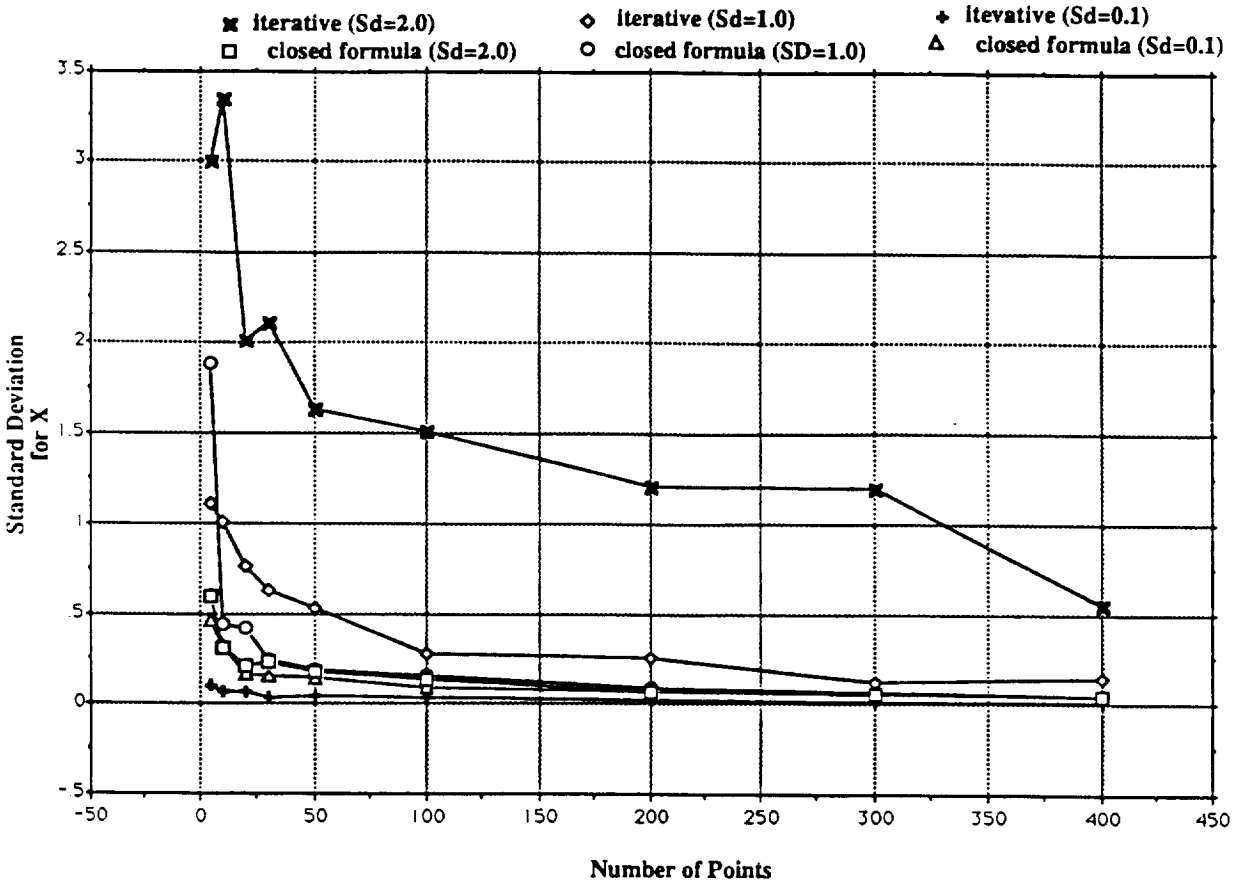


Figure 4.25. (a). Accuracy Comparison of Two Algorithms (Arc=180 degree); (b). Enlargement of (a) for the Input Sd=0.1 Case.



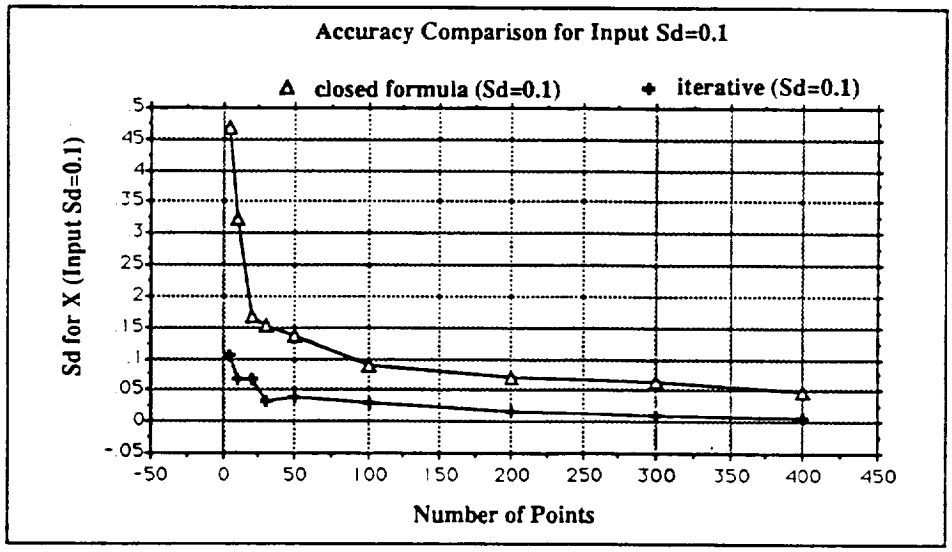


Figure 4.26. (a). Accuracy Comparison of Two Algorithms (Arc=60 degree); (b). Enlargement of (a) for the Input Sd=0.1 Case.

Cone direction vector: (2, 3, 4) Scan A : Y=0, Scan C: Y=5			
Input Sd	Average Errors on Pixel Position		
	Y=1	Y=2	Y=3
0.1	3.0000	6.0000	10.4000
0.5	2.7667	5.9333	10.5000
1.0	2.8667	5.8333	10.5333
2.5	3.2000	5.4667	10.4333
5.0	1.3667	4.7667	9.9333

Table 4.2. Accuracy performance of the three-scan algorithm with cone direction vector (2,3,4)

Cone direction vector: (0, 1, 0) Scan A : Y=0, Scan C: Y=5			
Input Sd	Average Errors on Pixel Position		
	Y=1	Y=2	Y=3
0.1	2.7333	5.9667	5.7000
0.5	7.9667	4.6667	5.8333
1.0	4.5000	5.2000	2.6667
2.5	5.3000	5.6667	4.9667
5.0	5.9000	5.7333	4.6667

Table 4.3. Accuracy performance of the three-scan algorithm with cone direction vector = (0, 1, 0)

standard deviation: 1.0
noise factor: 0.1
ellipse 2 plane: Y=1
ellipse 3 plane: Y=5
number of points: 250
start angle :35
ending angle: 160
cone:
vertex: 15 15 15
direction vector: 2 3 4
half angle: 30

ellipse 1 data:

x	model z	measured z
-20.302	-8.617	-6.491

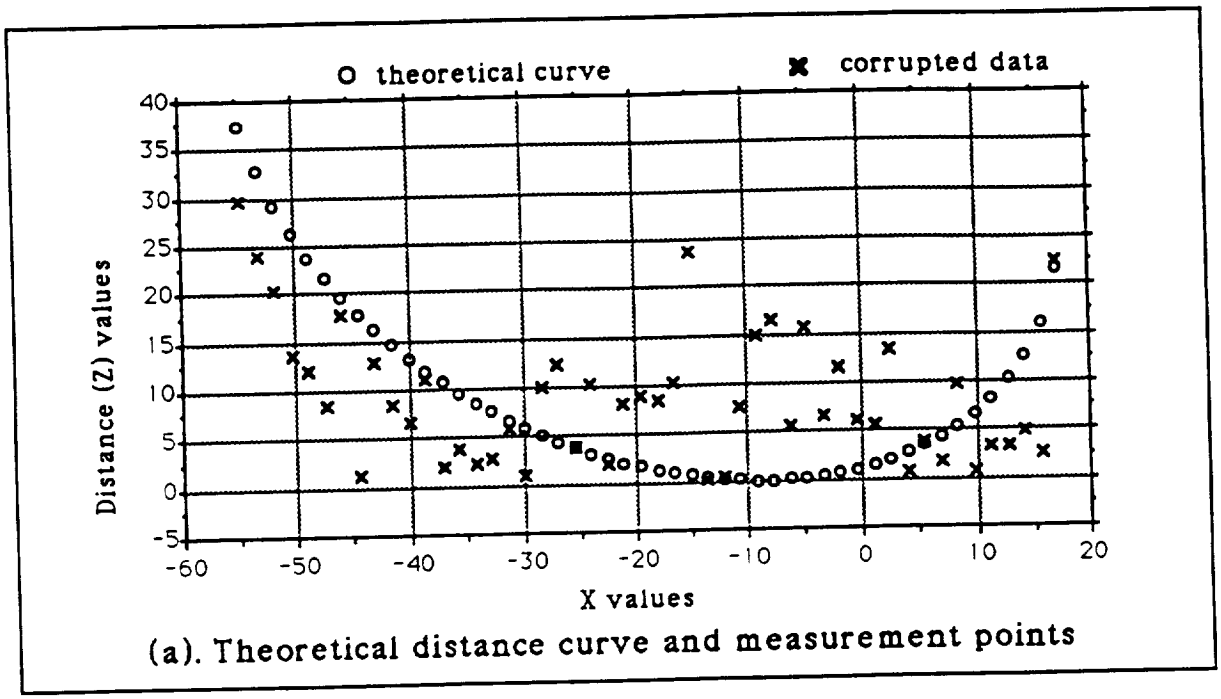
ellipse 2 data: (pixel 108-157)

	x	model z	measured z
108	-9.577	0.984	1.895
109	-9.870	0.734	0.885
110	-10.164	0.482	-0.344
111	-10.458	0.228	0.789
112	-10.752	-0.030	-0.197
113	-11.045	-0.289	0.669
114	-11.339	-0.551	-1.194
115	-11.633	-0.815	-0.190
116	-11.927	-1.082	-2.471
117	-12.220	-1.351	0.766
118	-12.514	-1.622	-0.932
119	-12.808	-1.896	-0.064
120	-13.101	-2.172	-3.174
121	-13.395	-2.450	-2.309
122	-13.689	-2.731	-2.413
123	-13.983	-3.013	-2.312
124	-14.276	-3.298	-3.069
125	-14.570	-3.586	-4.905
126	-14.864	-3.875	-3.365
127	-15.158	-4.167	-3.888
128	-15.451	-4.460	-3.410
129	-15.745	-4.756	-6.137
130	-16.039	-5.054	-4.768
131	-16.333	-5.354	-5.084
132	-16.626	-5.657	-6.384
133	-16.920	-5.961	-5.691
134	-17.214	-6.268	-7.767
135	-17.508	-6.576	-6.029
136	-17.801	-6.887	-8.497
137	-18.095	-7.199	-8.500
138	-18.389	-7.514	-7.346
139	-18.682	-7.831	-6.269
140	-18.976	-8.150	-6.549
141	-19.270	-8.470	-11.643
142	-19.564	-8.793	-9.322
143	-19.857	-9.118	-9.334
144	-20.151	-9.445	-9.840
145	-20.445	-9.773	-10.132
146	-20.739	-10.104	-9.314
147	-21.032	-10.437	-11.406
148	-21.326	-10.771	-10.565
149	-21.620	-11.108	-12.740
150	-21.914	-11.446	-12.740
151	-22.207	-11.787	-11.421
152	-22.501	-12.129	-12.939
153	-22.795	-12.473	-12.848
154	-23.089	-12.819	-12.366
155	-23.382	-13.167	-14.125
156	-23.676	-13.517	-13.553
157	-23.970	-13.869	-12.334

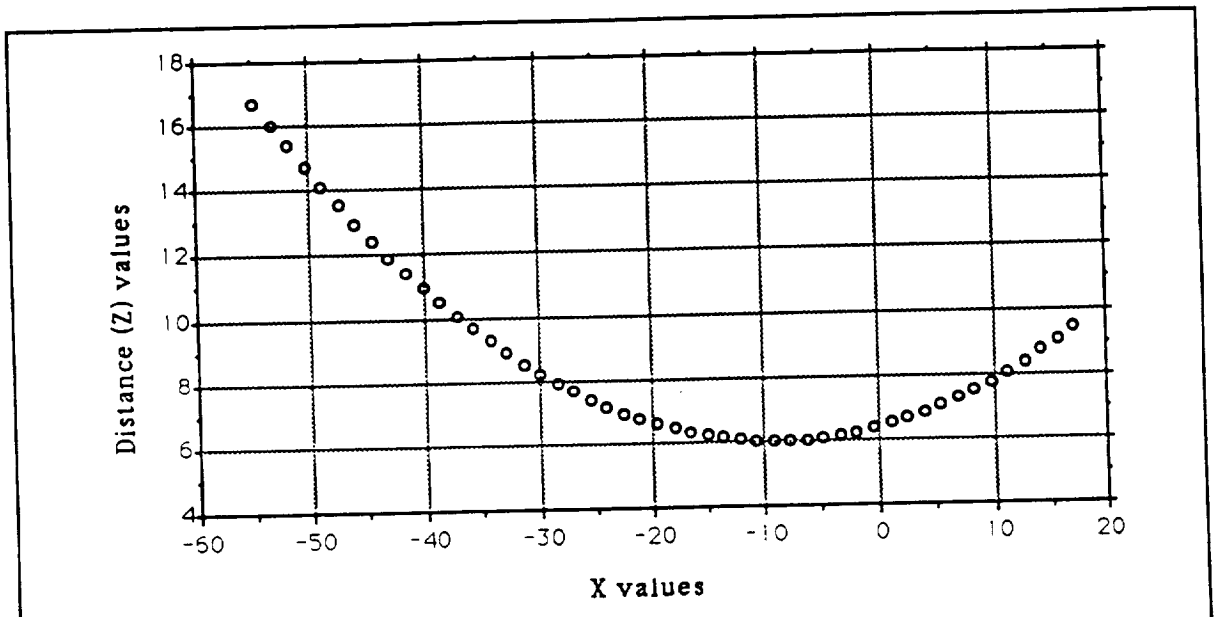
ellipse 3 data: (pixel 108-157)

x	model z	measured z
-4.863	2.932	1.372
-5.072	2.734	5.160
-5.282	2.536	2.465
-5.492	2.335	1.543
-5.702	2.133	2.537
-5.912	1.930	3.019
-6.122	1.725	2.882
-6.331	1.518	1.111
-6.541	1.310	-0.004
-6.751	1.100	1.205
-6.961	0.888	1.500
-7.171	0.676	1.920
-7.380	0.461	1.870
-7.590	0.245	1.283
-7.800	0.028	0.221
-8.010	-0.191	-0.444
-8.220	-0.412	0.391
-8.429	-0.633	-0.807
-8.639	-0.857	-0.964
-8.849	-1.082	0.218
-9.059	-1.308	-1.868
-9.269	-1.535	-1.539
-9.479	-1.765	-2.378
-9.688	-1.995	-3.032
-9.898	-2.227	-2.364
-10.108	-2.461	-1.190
-10.318	-2.695	-3.601
-10.528	-2.932	-2.719
-10.737	-3.169	-3.826
-10.947	-3.408	-3.755
-11.157	-3.649	-3.614
-11.367	-3.890	-4.978
-11.577	-4.133	-2.961
-11.786	-4.378	-5.399
-11.996	-4.624	-5.273
-12.206	-4.871	-3.103
-12.416	-5.120	-5.650
-12.626	-5.369	-5.609
-12.836	-5.621	-3.914
-13.045	-5.873	-6.682
-13.255	-6.127	-6.652
-13.465	-6.383	-4.972
-13.675	-6.639	-6.847
-13.885	-6.897	-5.504
-14.094	-7.156	-6.539
-14.304	-7.417	-6.676
-14.514	-7.679	-6.620
-14.724	-7.942	-6.748
-14.934	-8.207	-10.205
-15.143	-8.472	-7.180

Table 4.4. A listing of simulation data

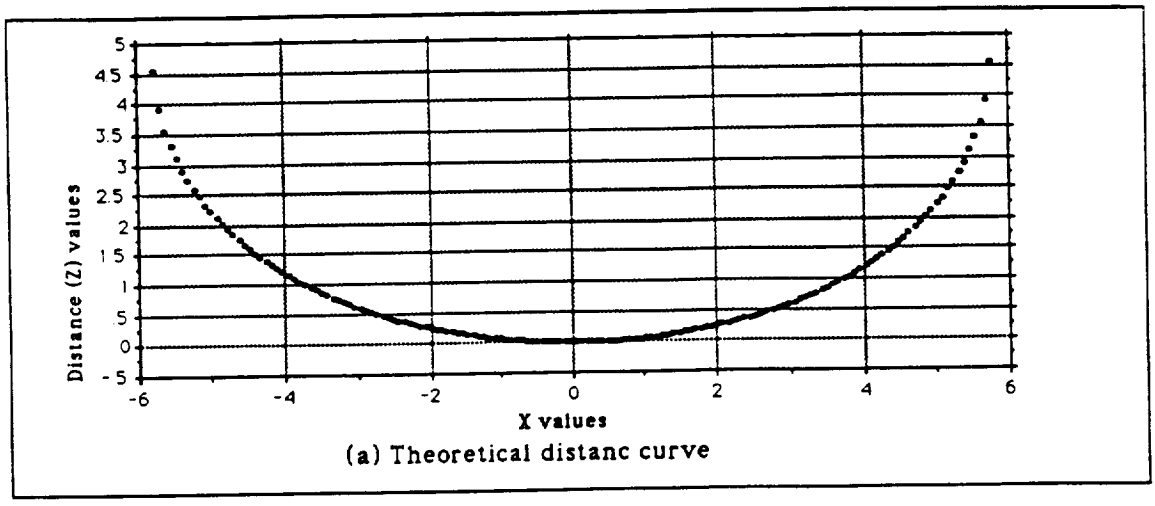


(a). Theoretical distance curve and measurement points

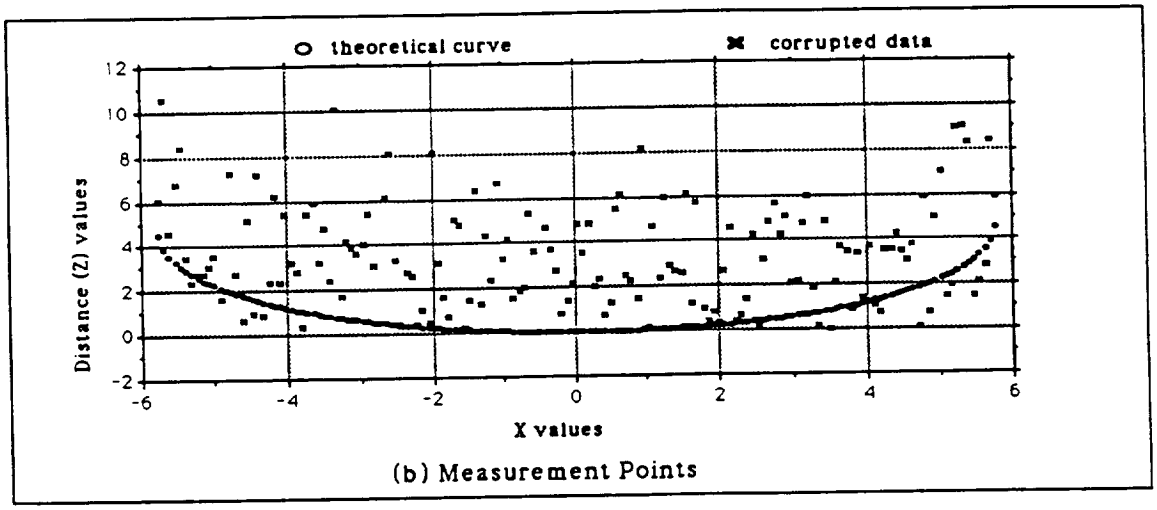


(b). Smoothed distance curve

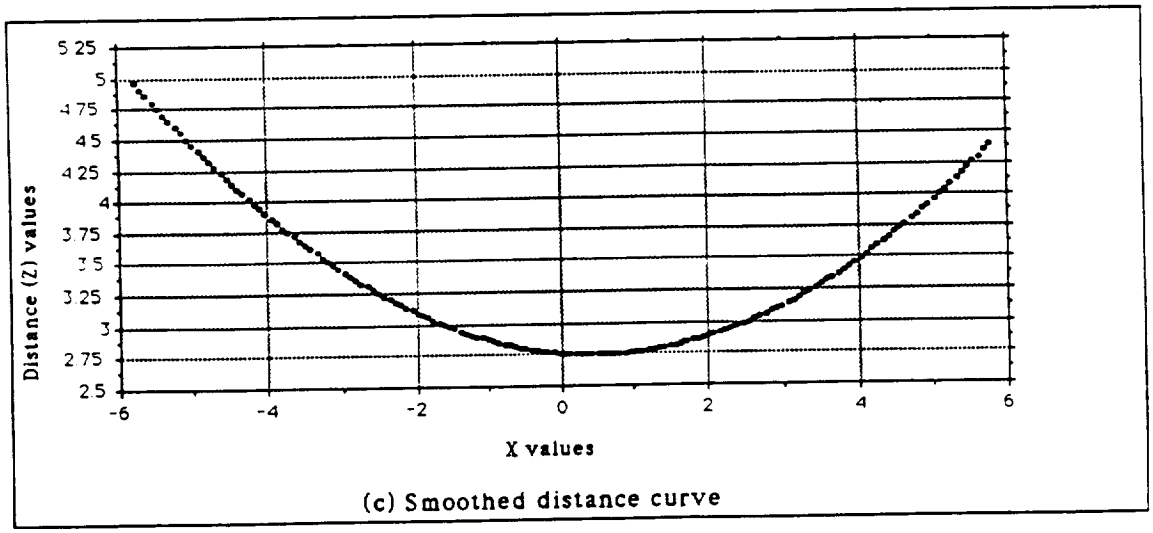
Figure 4.27. Distance curves: (a). theoretical distance curve and corrupted data points; (b) distance curve after smoothing. (cone direction vector = (2,3,4))



(a) Theoretical distanc curve



(b) Measurement Points



(c) Smoothed distance curve

Figure 4.28. Distance curves: (a). theoretical distance curve; (b). distance curve resulted from measurement errors; (c) distance curve after smoothing. (cone direction vector = (0,1,0))

CHAPTER 5

AN OPTIMAL SOLUTION FOR OBJECT LOCALIZATION

As we have mentioned, two pairs of non-parallel matching line-segments are sufficient to completely specify a transformation matrix. In practice, however, it is often the case that more than the minimum number of required features can be extracted from sensed range data. For example, three or more line-segments may be extracted from sensed range data either by using multiple line sensors or by using multiple measurements with each measurement displaced slightly. Point features such as the corner points (vertices) of an object, masked points, or the center of a sphere may also be extracted from sensed range data.

Techniques which combine redundant sensed features, whether or not they are of the same type, to determine the object location can improve the accuracy of localization. Least squares optimization techniques are frequently used to find the best estimate of the transformation matrix from those redundant features. Two approaches for using optimization techniques to find the best estimate of the transformation matrix have been introduced [5, 49, 37]. In one method, an optimal orientation of the object is determined first, which is then used as a basis to find the position of the object. That is, the translation vector is a function of an optimal rotation matrix $\boldsymbol{R}$ and other measured quantities. Many object localization algorithms use this method [5, 45, 48, 49, 78, 98]. The problem with this approach is the possibility of the existence of the accumulated errors in calculating

the translation vector due to the errors from previous calculations and measurements. For example, in [5]'s SVD algorithm the translation vector $\mathbf{t}$ is computed from $\boldsymbol{R}, \tilde{\mathbf{p}}_i, \mathbf{p}_i$, e.g., $\mathbf{t} = f(\boldsymbol{R}, \tilde{\mathbf{p}}_i, \mathbf{p}_i)$ where $\tilde{\mathbf{p}}_i$ and $\mathbf{p}_i$ are sets of measured points and corresponding modeled points in 3-D space respectively. Because both $\tilde{\mathbf{p}}_i$ and $\boldsymbol{R}$ have errors, the error of the resulting translation vector will be compounded due to error propagation. The second approach is to compute two optimal solutions separately, one for the orientation and the another for the position [37], which is not very efficient. The common characteristics of these two methods are that both approach the problem of determination of orientation and position separately. That is not surprising, because the transformation matrix itself can be easily decomposed into two parts: a rotation submatrix and a position vector. The difference between these two approaches lies in the way the optimization is done: the first only optimizes the rotational part of the homogeneous transformation matrix and the translational part is then derived from it, while the second method optimizes both rotational and translational parts separately.

In this chapter, we present an efficient algorithm which is based on the use of dual number quaternions [26]. The method solves for the orientation and the position of an object by minimizing a single cost function associated with the sum of the orientation and position errors. The performance, both in accuracy and in speed, compared with that of the previous methods will be discussed. The required input data for the algorithm is a combination of measured points on the surfaces of an object, measured unit direction vectors from that object and their corresponding modeled features. Examples of point features might be any combinations of those which we have mentioned at the beginning of this chapter. Examples of unit vector features include the surface normals, edge direction vectors, or axis direction vectors.

A brief description of the concept and properties of dual numbers is given in Appendix F. More detailed discussion can be found in [105] and [134]. In the following sections, we begin with the introduction of the definition of dual number quaternions. We show how they are used to represent object location, why they are a valid representation of location and their correspondence with the more familiar homogeneous transforms. Then we give a brief description of the important properties of the dual number quaternions. Next, we formulate the object localization problem as a dual number quaternion optimization problem and an algorithm is derived to solve the problem. Simulation results are shown in section 5.3.

5.1 Dual Number Quaternions

This section begins with the definition of dual number quaternions, their properties, and their physical interpretation. It concludes by showing how to convert back and forth from the dual number quaternion representation of location to the homogeneous transformation representation.

5.1.1 Properties of dual number quaternions

Quaternions are four element vectors, which are thought of as consisting of a 3×1 vector component and a scalar component. For example the quaternion q is:

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \mathbf{q} \\ q_4 \end{bmatrix} \tag{5.1}$$

Symbol(s)	*Description*
Quaternion: q, e, a, b, r, s, t,n, p, n_i, p_i, n_i^0, p_i^0, $\tilde{n}_i$, $\tilde{p}_i$	
e	unit quaternion
t	translation quaternion
r	real part of a dual quaternion
s	dual part of a dual quaternion
n	direction quaternion
p	position quaternion
n_i^0	modeled direction quaternion
p_i^0	modeled position quaternion
n_i	transformed model direction quaternion
p_i	transformed model position quaternion
$\tilde{n}_i$	measured direction quaternion
$\tilde{p}_i$	measured position quaternion
Vector: $\mathbf{t}$, $\mathbf{q}$, $\mathbf{a}$, $\mathbf{n}$, $\mathbf{p}$, $\mathbf{r}$, $\mathbf{p}_i$, $\mathbf{p}_i^0$, $\tilde{\mathbf{p}}_i$, $\mathbf{n}_i^0$	
$\mathbf{t}$	translation vector
$\mathbf{n}$	rotation axis unit direction vector
$\mathbf{p}$	position vector
$\mathbf{p}_i^0$	modeled position vector
$\tilde{\mathbf{p}}_i$	measured position vector
$\mathbf{p}_i$	transformed model position vector
$\mathbf{n}_i^0$	modeled direction vector
4×4 matrix: T, I, A, C_1, C_2, C_3, Q, W	
T	homogeneous transformation matrix
Q W	quaternion matrices
3×3 matrix: R, K	
R	rotation matrix
K	skew-symmetric matrix
Scalar: θ, d, e, λ_1, λ_2, α_i, β_i	
θ	rotation angle
d	distance between two vectors
e	errors from least squares optimization
λ_1, λ_2	Lagrange multipliers
α_i, β_i	weighting factors
Dual quantities: $\hat{q}$, $\hat{\mathbf{q}}$, $\hat{\mathbf{n}}$, $\hat{\theta}$	
$\hat{q}$	dual quaternion
$\hat{\mathbf{n}}$	dual vector of rotation
$\hat{\theta}$	dual angle of rotation

Table 5.1. A list of symbols appeared in this chapter

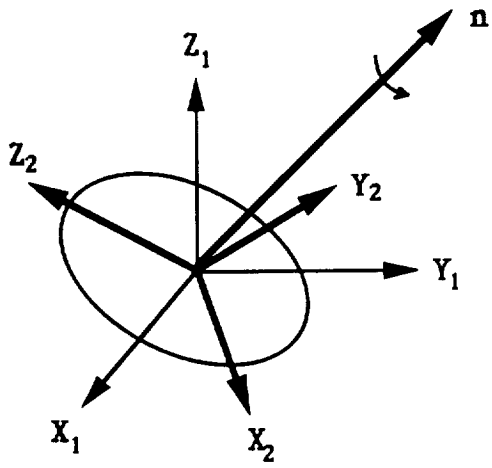


Figure 5.1. Illustration of rotation for a real quaternion

In our notation, each quaternion is represented with a boldface script character, such as $\boldsymbol{q}$; and each 3×1 vector is represented with a boldface Roman character, such as $\mathbf{q}$.

Quaternions have been used extensively as a method of parameterizing orientation [52, 68, 94].

In this application, the components of the quaternion have the following interpretation.

$$\boldsymbol{q} = \begin{bmatrix} sin(\theta/2)\mathbf{n} \\ cos(\theta/2) \end{bmatrix} \tag{5.2}$$

The components of this quaternion are called the Euler Symmetric Parameters. As illustrated in Figure (5.1), the vector $\mathbf{n}$ is the unit vector about which the coordinate system has rotated and θ is the amount of rotation about $\mathbf{n}$. The corresponding rotation matrix $\boldsymbol{R}$ can be expressed as

$$\boldsymbol{R} = (q_4^2 - \mathbf{q}^T\mathbf{q})I + 2\mathbf{q}\mathbf{q}^T + 2q_4\boldsymbol{K}(\mathbf{q}) \tag{5.3}$$

where $\boldsymbol{K}$ is the skew-symmetric matrix

$$\boldsymbol{K}(\mathbf{q}) = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{5.4}$$

The extension of this equation to include the representation of position and orientation is made by simply changing all of the quantities in the equation to dual quantities [26].

$$\hat{q} = \begin{bmatrix} \hat{q}_1 \\ \hat{q}_2 \\ \hat{q}_3 \\ \hat{q}_4 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{q}} \\ \hat{q}_4 \end{bmatrix} \tag{5.5}$$

Note that whether an item is a 3×1 vector or a quaternion, if it's components are dual numbers, it is signified by placing a hat over it as in the above example. A brief description of the concept of dual numbers and their important properties can be found in Section $F.1$ of Appendix F.

There are two parts of a dual quaternion.

$$\hat{q} = \boldsymbol{r} + \epsilon \boldsymbol{s} \tag{5.6}$$

where $\boldsymbol{r}$ and $\boldsymbol{s}$ are both real quaternions and are called the real part and dual part, respectively.

The dual quaternions have a similar interpretation as the real quaternion.

$$\hat{q} = \begin{bmatrix} \sin(\hat{\theta}/2)\hat{\mathbf{n}} \\ \cos(\hat{\theta}/2) \end{bmatrix} \tag{5.7}$$

where the dual vector $\hat{\mathbf{n}}$ represents a line in 3-D space about which the coordinate system has rotated and translated and $\hat{\theta}$ is the dual angle of rotation and translation. The dual

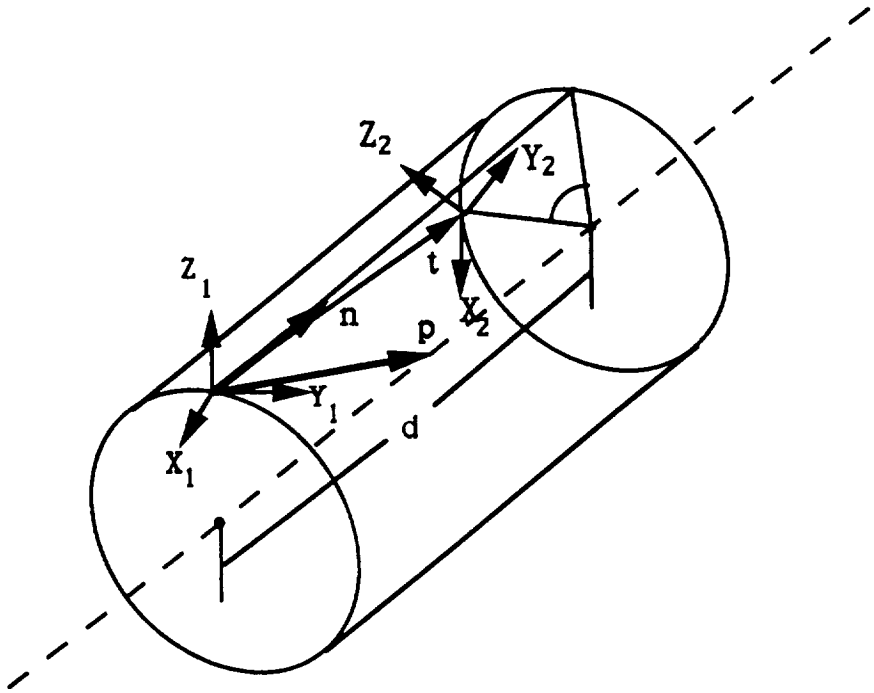


Figure 5.2. Illustration of rotation and translation for a dual quaternion

vector $\hat{\mathbf{n}}$ and dual angle $\hat{\theta}$ are

$$\hat{\mathbf{n}} = \mathbf{n} + \epsilon \mathbf{p} \times \mathbf{n} \tag{5.8}$$

and

$$\hat{\theta} = \theta + \epsilon d \tag{5.9}$$

where $\mathbf{n}$ is a unit vector which specifies the direction of the rotation axis and also the direction of translation; the rotation is about the line having direction $\mathbf{n}$ passing through the point $\mathbf{p}$ with a rotation angle of θ; and d is the distance of translation along the direction specified by $\mathbf{n}$ (see Appendix F for the discussion of $\mathbf{n}$ and $\mathbf{p}$). The geometrical interpretation of the representation can be explained as follows:

> Traditionally, the transformation of a coordinate frame is specified by a translation vector $\mathbf{t}$, a rotation axis $\mathbf{n}$ and a rotation angle θ. A new coordinate

frame is formed by first translating the original coordinate frame along $\mathbf{t}$ and then rotating it with respect to $\mathbf{n}$ by an angle θ. Of course, the sequence of translation and rotation can be reversed.

With dual quaternion representation, the same transformation can be formed by first translating the original coordinate frame along the direction of $\mathbf{n}$ by a distance of d and then rotating it by an angle of θ with respect to a line having a unit vector $\mathbf{n}$ as its direction and passing through a point $\mathbf{p}$.

See Figure 5.2 for an illustration of the interpretation. It can be proven that for each $(\mathbf{n}, \mathbf{p}, d, \theta)$ transformation representation, we can always find a unique corresponding $(\mathbf{t}, \mathbf{n}, \theta)$. On the other hand, for each $(\mathbf{t}, \mathbf{n}, \theta)$ transformation representation, there exists a set of corresponding $(\mathbf{n}, \mathbf{p}, d, \theta)$'s. (See section 5.1.2 and appendix F for a detailed description).

If we place Eq.(5.8) and (5.9), e.g, expressions for $\hat{\mathbf{n}}$ and $\hat{\theta}$, into Eq.(5.7), expand and simplify that equation by using the properties of dual numbers, and compare the results with Eq.(5.6), we have the following equations:

$$r = \begin{bmatrix} \sin(\theta/2)\mathbf{n} \\ \cos(\theta/2) \end{bmatrix} \tag{5.10}$$

and

$$s = \begin{bmatrix} d/2\cos(\theta/2)\mathbf{n} + \sin(\theta/2)(\mathbf{p} \times \mathbf{n}) \\ -d/2\sin(\theta/2) \end{bmatrix} \tag{5.11}$$

A dual quaternion has eight elements whereas the minimum number of independent variables to represent a 3-D object transformation is six, which means that two of the eight elements in dual quaternion representation are not independent. In fact, it can be shown from Eq. (5.10) and (5.11) that the components of any dual quaternion, if they are

$\boldsymbol{Q}(\boldsymbol{a})^T\boldsymbol{Q}(\boldsymbol{a}) = \boldsymbol{Q}(\boldsymbol{a})\boldsymbol{Q}(\boldsymbol{a})^T = \boldsymbol{a}^T\boldsymbol{a}\boldsymbol{I}$
$\boldsymbol{W}(\boldsymbol{a})^T\boldsymbol{W}(\boldsymbol{a}) = \boldsymbol{W}(\boldsymbol{a})\boldsymbol{W}(\boldsymbol{a})^T = \boldsymbol{a}^T\boldsymbol{a}\boldsymbol{I}$
$\boldsymbol{Q}(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{W}(\boldsymbol{b})\boldsymbol{a}$
$\boldsymbol{Q}(\boldsymbol{a})^T\boldsymbol{a} = \boldsymbol{W}(\boldsymbol{a})^T\boldsymbol{a} = (\boldsymbol{a}^T\boldsymbol{a})\boldsymbol{e}$
$\boldsymbol{Q}(\boldsymbol{Q}(\boldsymbol{a})\boldsymbol{b}) = \boldsymbol{Q}(\boldsymbol{a})\boldsymbol{Q}(\boldsymbol{b})$
$\boldsymbol{W}(\boldsymbol{W}(\boldsymbol{a})\boldsymbol{b}) = \boldsymbol{W}(\boldsymbol{a})\boldsymbol{W}(\boldsymbol{b})$
$\boldsymbol{Q}(\boldsymbol{a})\boldsymbol{W}(\boldsymbol{b})^T = \boldsymbol{W}(\boldsymbol{b})^T\boldsymbol{Q}(\boldsymbol{a})$
$\boldsymbol{a}$ and $\boldsymbol{b}$ are arbitrary quaternions
$\boldsymbol{e}$ is the unit quaternion $= [0001]^T$

Table 5.2. Properties of quaternion matrices

defined by Eq. (5.7)–(5.9), satisfy the following two constraints

$$\boldsymbol{r}^T\boldsymbol{r} = 1 \tag{5.12}$$

$$\boldsymbol{r}^T\boldsymbol{s} = 0 \tag{5.13}$$

Two important matrix functions of quaternions are the matrices $\boldsymbol{Q}(\boldsymbol{r})$ and $\boldsymbol{W}(\boldsymbol{r})$ which are defined as:

$$\boldsymbol{Q}(\boldsymbol{r}) = \begin{bmatrix} r_4\boldsymbol{I} + \boldsymbol{K}(\mathbf{r}) & \mathbf{r} \\ -\mathbf{r}^T & r_4 \end{bmatrix} \tag{5.14}$$

$$\boldsymbol{W}(\boldsymbol{r}) = \begin{bmatrix} r_4\boldsymbol{I} - \boldsymbol{K}(\mathbf{r}) & \mathbf{r} \\ -\mathbf{r}^T & r_4 \end{bmatrix} \tag{5.15}$$

where $\boldsymbol{K}(\mathbf{r})$ is the skew-symmetric matrix as was defined in Eg. (5.4).

Useful properties of the $\boldsymbol{Q}$ and $\boldsymbol{W}$ matrices which are utilized in the derivation of the localization algorithm are given in Table (5.2). All these properties can easily be verified by direct substitutions.

5.1.2 Relation to homogeneous transforms

A common method of representing the position and orientation of a coordinate system is with homogeneous transforms. Since homogeneous transforms are more common in use than dual number quaternions, the following is provided as a reference to show how to convert from one to the other.

5.1.2.1 Computing the homogeneous transform given the dual quaternion

Eq.(5.10) shows that the real part $\boldsymbol{r}$ of the dual quaternion has exactly the same form as that defined in Eq.(5.2). As a result, the rotation matrix $\boldsymbol{R}$ can be written in terms of the components of the dual quaternion in the following familiar way.

$$\boldsymbol{R} = (r_4^2 - \mathbf{r}^T\mathbf{r})\boldsymbol{I} + 2\mathbf{r}\mathbf{r}^T + 2r_4\boldsymbol{K}(\mathbf{r}) \tag{5.16}$$

or

$$\begin{bmatrix} \boldsymbol{R} & 0 \\ 0^T & 1 \end{bmatrix} = \boldsymbol{W}(\boldsymbol{r})^T\boldsymbol{Q}(\boldsymbol{r}) \tag{5.17}$$

The position vector can be written in terms of the components of the dual quaternion in the following way (see section $F.2$ of Appendix F for the detailed derivation).

$$\boldsymbol{t} = \boldsymbol{W}(\boldsymbol{r})^T\boldsymbol{s} \tag{5.18}$$

where t is the translation quaternion for the translation vector $\mathbf{t}$ and is defined as

$$t = 1/2 \begin{bmatrix} \mathbf{t} \\ 0 \end{bmatrix} \tag{5.19}$$

5.1.2.2 Computing the dual quaternion given the homogeneous transform

Given a homogeneous transform T specified by a rotation matrix R and a translation vector $\mathbf{t}$, one can compute the corresponding r and s as follows:

$$r_4 = 1/2\sqrt{R_{11} + R_{22} + R_{33} + 1} \tag{5.20}$$

where R_{ij} denotes the $ij - th$ element of the matrix R. A value of r_4 equal to zero represents a rotation of 180 degrees. If r_4 is not zero, then:

$$\mathbf{r} = \frac{1}{4r_4} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix} \tag{5.21}$$

If r_4 is zero, then:

$$\mathbf{r}\mathbf{r}^T = 1/2(R + I) \tag{5.22}$$

So r can be determined from any nonzero column of $1/2(R + I)$, call it $\mathbf{a}$. Thus,

$$\mathbf{r} = \pm\frac{\mathbf{a}}{||\mathbf{a}||} \tag{5.23}$$

Either sign will work since the rotation angle is 180 degrees,

Having determined r, the value of s can be computed from Eq. (5.18)

$$s = W(r)t \tag{5.24}$$

Thus a homogeneous transformation matrix and a corresponding dual quaternion can be converted from one to another.

5.2 Problem Formulation and Solution

This section begins with the formulation of the problem as an optimization problem. The following section presents the solution to the problem.

5.2.1 Problem formulation

As we have mentioned, two types of sensor measurements are considered: the position of points on an object and the unit vector on the object such as the unit normal, edge direction vector, *etc.*. To facilitate the analysis we define quaternion representations of these quantities. Let $\mathbf{p}$ be the position vector of a point on the object surface. We define the position quaternion as

$$\boldsymbol{p} = 1/2 \begin{bmatrix} \mathbf{p} \\ 0 \end{bmatrix} \tag{5.25}$$

Let $\mathbf{n}$ be a unit vector extracted from the object. We define the direction quaternion as

$$\boldsymbol{n} = \begin{bmatrix} \mathbf{n} \\ 0 \end{bmatrix} \tag{5.26}$$

To determine the position and orientation of an object, we make measurements of k unit vectors for which we have a correspondence to the models and store them in the direction quaternions, $\tilde{\boldsymbol{n}}_i$. The tilde denotes measured values. Similarly, we make measurements of l points on the object and store them in the position quaternions, $\tilde{\boldsymbol{p}}_i$.

Corresponding to each measured point $\tilde{\boldsymbol{p}}_i$, there is a database description of that point $\mathbf{p}_i^0$ which is described with respect to the object coordinate system. If $\mathbf{t}$ and $\boldsymbol{R}$ are the translational and rotational parts of the transformation matrix which is to be determined, the modeled point will be transformed into position $\mathbf{p}_i$

$$\mathbf{p}_i = \mathbf{t} + \boldsymbol{R}\mathbf{p}_i^0 \tag{5.27}$$

If these modeled points are represented by position quaternions $\boldsymbol{p}_i^0$ and $\boldsymbol{p}_i$ and a dual quaternion is used to represent the transformation parameters, from Eq. (5.17) and (5.18), Eq. (5.27) will become the following:

$$\boldsymbol{p}_i = \boldsymbol{W}(\boldsymbol{r})^T \boldsymbol{s} + \boldsymbol{W}(\boldsymbol{r})^T \boldsymbol{Q}(\boldsymbol{r}) \boldsymbol{p}_i^0 \tag{5.28}$$

For the same reason, for the modeled direction quaternion $\boldsymbol{n}_i$ and $\boldsymbol{n}_i^0$, we have the relation:

$$\boldsymbol{n}_i = \boldsymbol{W}(\boldsymbol{r})^T \boldsymbol{Q}(\boldsymbol{r}) \boldsymbol{n}_i^0 \tag{5.29}$$

The approach for computing the position and orientation of the object is to determine $\boldsymbol{r}$ and $\boldsymbol{s}$ which minimizes the error between the $\tilde{\boldsymbol{p}}_i$ and $\boldsymbol{p}_i$ and the $\tilde{\boldsymbol{n}}_i$ and $\boldsymbol{n}_i$. That is, we select the $\boldsymbol{r}$ and $\boldsymbol{s}$ to minimize the following error function:

$$E = \sum_{i=1}^{k} \alpha_i (\boldsymbol{n}_i - \tilde{\boldsymbol{n}}_i)^2 + \sum_{i=1}^{l} \beta_i (\boldsymbol{p}_i - \tilde{\boldsymbol{p}}_i)^2 \tag{5.30}$$

where the α_i and β_i are constant positive weighting factors.

We consider each of these terms individually.

$$(\boldsymbol{n}_i - \tilde{\boldsymbol{n}}_i)^2 = 2(1 - \boldsymbol{r}^T \boldsymbol{Q}(\tilde{\boldsymbol{n}}_i)^T \boldsymbol{W}(\boldsymbol{n}_i^0) \boldsymbol{r}) \tag{5.31}$$

$$\begin{aligned} (\boldsymbol{p}_i - \tilde{\boldsymbol{p}}_i)^2 &= \boldsymbol{s}^T \boldsymbol{s} + 2\boldsymbol{s}^T (\boldsymbol{W}(\boldsymbol{p}_i^0) - \boldsymbol{Q}(\tilde{\boldsymbol{p}}_i)) \boldsymbol{r} \\ &\quad -2\boldsymbol{r}^T \boldsymbol{Q}(\tilde{\boldsymbol{p}}_i)^T \boldsymbol{W}(\boldsymbol{p}_i^0) \boldsymbol{r} + ((\boldsymbol{p}_i^0)^T \boldsymbol{p}_i^0 + \tilde{\boldsymbol{p}}_i^T \tilde{\boldsymbol{p}}_i) \end{aligned} \tag{5.32}$$

Thus, the error function can be written as a quadratic function of $\boldsymbol{r}$ and $\boldsymbol{s}$.

$$E = \boldsymbol{r}^T \boldsymbol{C}_1 \boldsymbol{r} + \boldsymbol{s}^T \boldsymbol{C}_2 \boldsymbol{s} + \boldsymbol{s}^T \boldsymbol{C}_3 \boldsymbol{r} + constant \tag{5.33}$$

where

$$\boldsymbol{C}_1 = -2 \sum_{i=1}^{k} \alpha_i \boldsymbol{Q}(\tilde{\boldsymbol{n}}_i)^T \boldsymbol{W}(\boldsymbol{n}_i^0) - 2 \sum_{i=1}^{l} \beta_i \boldsymbol{Q}(\tilde{\boldsymbol{p}}_i)^T \boldsymbol{W}(\boldsymbol{p}_i^0) \tag{5.34}$$

$$\boldsymbol{C}_2 = (\sum_{i=1}^{l} \beta_i) \boldsymbol{I} \tag{5.35}$$

$$C_3 = 2\sum_{i=1}^{l}\beta_i(W(p_i^0) - Q(\tilde{p}_i)) \tag{5.36}$$

$$constant = 2\sum_{i=1}^{k}\alpha_i + \sum_{i=1}^{l}\beta_i((p_i^0)^T p_i^0 + \tilde{p}_i^T \tilde{p}_i) \tag{5.37}$$

We compute r and s to minimize this error function subject to the constraints:

$$r^T r = 1 \tag{5.38}$$

$$s^T r = 0 \tag{5.39}$$

5.2.2 Problem solution

The optimal dual number location quaternion is obtained by adjoining the constraint equations to the error equation and then minimizing the resulting function without constraints.

$$\begin{aligned}\tilde{E} &= r^T C_1 r + s^T C_2 s + s^T C_3 r + constant \\ &+ \lambda_1(r^T r - 1) + \lambda_2(s^T r)\end{aligned}$$

where λ_1 and λ_2 are Lagrange multipliers. Taking the partial derivatives gives:

$$\frac{\partial \tilde{E}}{\partial r} = (C_1 + C_1^T)r + C_3^T s + 2\lambda_1 r + \lambda_2 s = 0 \tag{5.40}$$

$$\frac{\partial \tilde{E}}{\partial s} = (C_2 + C_2^T)s + C_3 r + \lambda_2 r = 0 \tag{5.41}$$

Thus, the solution of equations (5.38), (5.39), (5.40), and (5.41), for r and s gives the optimal solution for the position and orientation of the object.

To solve these equations, we begin by solving for λ_2. Multiplying equation (5.41) by r and solving for λ_2 gives:

$$\lambda_2 = -r^T C_3 r \tag{5.42}$$

Since C_3 is skew symmetric:

$$\lambda_2 = 0 \tag{5.43}$$

We can now solve for s as a function of r from Eg. (5.41),

$$s = -(C_2 + C_2^T)^{-1} C_3 r \tag{5.44}$$

Substituting equations (5.43) and (5.44) into Eg. (5.40) gives:

$$Ar = \lambda_1 r \tag{5.45}$$

where

$$A = \frac{1}{2}(C_3^T (C_2 + C_2^T)^{-1} C_3 - C_1 - C_1^T) \tag{5.46}$$

Thus, the quaternion r is an eigenvector of the matrix A and λ_1 is the corresponding eigenvalue. In general there will be four solutions to this equation. Since A is real and symmetric, all of the eigenvalues and eigenvectors are real and the eigenvectors will be orthogonal. The desired solution is identified by referring back to the original error equation (5.33).

Multiplying Eg. (5.40) by r^T gives:

$$r^T C_1 r = 1/2 r^T (C_1 + C_1^T) r = -1/2 s^T C_3 r - \lambda_1 \tag{5.47}$$

Multiplying Eg. (5.41) by s^T gives:

$$s^T C_2 s = 1/2 s^T (C_2 + C_2^T) s = -1/2 s^T C_3 r \tag{5.48}$$

Substituting these into Eg. (5.33) gives:

$$E = constant - \lambda_1 \tag{5.49}$$

Thus, the error is minimized if we select the eigenvector corresponding to the largest positive eigenvalue.

Having computed $\boldsymbol{r}$, we can now substitute back into Eg. (5.44) to obtain $\boldsymbol{s}$ to complete the solution for the position and orientation of the object.

To give a clearer picture of the above derivation process, in the following the optimal dual number quaternion localization algorithm, (DQ algorithm), will be summarized.

From the algorithm we can see that the execution times for steps $2 - 4$ are basically constant and the execution time for step 1 has a linear relationship to the number of measured vectors. Therefore, the algorithm is an $O(n)$ algorithm in time complexity.

DQ Localization Algorithm:

Inputs: a set of k measured points $\tilde{\boldsymbol{p}}_i^0$, l measured unit vectors $\tilde{\boldsymbol{n}}_i^0$; the corresponding modeled points $\boldsymbol{p}_i$ and vectors $\boldsymbol{n}_i$; as well as weighting factors α_i and β_i chosen heuristically to reflect the reliability of the data points.

Output: an estimation of transformation matrix T.

Step 1. Compute matrices $\boldsymbol{C}_1, \boldsymbol{C}_2$ and $\boldsymbol{C}_3$:

$$\boldsymbol{C}_1 = -2\sum_{i=1}^{k} \alpha_i \boldsymbol{Q}(\tilde{\boldsymbol{n}}_i)^T \boldsymbol{W}(\boldsymbol{n}_i^0) - 2\sum_{i=1}^{l} \beta_i \boldsymbol{Q}(\tilde{\boldsymbol{p}}_i)^T \boldsymbol{W}(\boldsymbol{p}_i^0)$$

$$\boldsymbol{C}_2 = (\sum_{i=1}^{l} \beta_i)\boldsymbol{I}$$

$$\boldsymbol{C}_3 = 2\sum_{i=1}^{l} \beta_i (\boldsymbol{W}(\boldsymbol{p}_i^0) - \boldsymbol{Q}(\tilde{\boldsymbol{p}}_i))$$

Step 2. Compute the 4×4 symmetric matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = 1/2(\boldsymbol{C}_3^T(\boldsymbol{C}_2 + \boldsymbol{C}_2^T)^{-1}\boldsymbol{C}_3 - \boldsymbol{C}_1 - \boldsymbol{C}_1^T)$$

Step 3. Compute the eigenvector $\boldsymbol{r}$ corresponding to the largest positive eigenvalue of matrix $\boldsymbol{A}$ and derive $\boldsymbol{s}$ from $\boldsymbol{r}$.

Step 4. Compute the matrix T from $\boldsymbol{s}$ and $\boldsymbol{r}$.

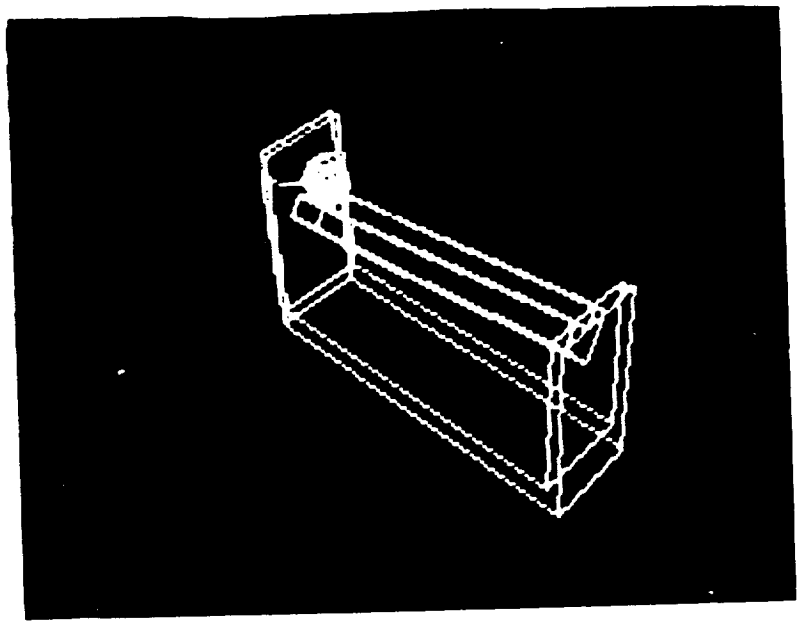


Figure 5.3. The object "Feeder".

5.3 Simulation Results

To examine the performance of the DQ algorithm – the accuracy and the speed, computer simulations have been carried out on a Compaq 386/20 with an 80387-20 math-processor. The simulation data are from a modeled "Feeder" (see Fig. 5.3) which has 32 vertices, 50 surfaces and 48 edges. The size of the "Feeder" is $322 \times 84 \times 151$ units. The SVD algorithm is selected as a sample algorithm to compare the accuracy and performance with our DQ algorithm.

Because the SVD algorithm only accepts 3-D points as its inputs, the sole inputs for the two algorithms are the sampled points in order to have a fair comparison of their accuracy. The algorithms were tested using 5, 10, 20 and 30 points as input data. Each of these tests was repeated for 25 different choices of points, e.g., 25 different sets of 5, 10, 20 and 30 points were run, and each of these was run 20 times with random errors (see discussion below) added to the sample values. In our simulation, the required number

Number of Point Correspondences	Method Used							
	SVD				Dual Number			
	x	y	z	θ	x	y	z	θ
5	1.434	3.013	1.190	0.147	0.461	0.277	0.509	0.147
10	1.133	2.373	0.843	0.046	0.133	0.215	0.169	0.046
20	0.296	0.607	0.254	0.040	0.102	0.187	0.108	0.040
30	0.171	0.246	0.125	0.037	0.115	0.115	0.087	0.037

Table 5.3. Comparison of Standard Deviation of Transformation Parameters

of 3-D points $\boldsymbol{p}_i^0$ are randomly selected from a "Feeder"'s vertices at the beginning of each trial. The corresponding measured points $\tilde{\boldsymbol{p}}_i$ are then generated by first rotating an angle of 36° around an axis through the origin with direction vector (3.0,4.0,6.0) followed by a translation of (7,8,13), and finally by adding to each coordinates of the resulting points Gaussian random noise with mean zero and standard deviation of 0.5. These measured data and modeled data are used to compute the estimated orientation and translation parameters. To simplify the simulation, all the weighting factors α_i and β_i are set to 1. The standard deviations for the resulting orientation and translation parameters are calculated from these twenty trials. Table (5.3) lists the simulation results. All the algorithms are written in Turbo Pascal. Mathpak 87 subroutine package from Precision Plus Software was used to carry out all the matrix computation, as well as SVD and eigenvalue calculations.

From Table (5.3) we see that the two algorithms produce the same rotation errors no matter how many points are used during the simulations, which is expected. For the

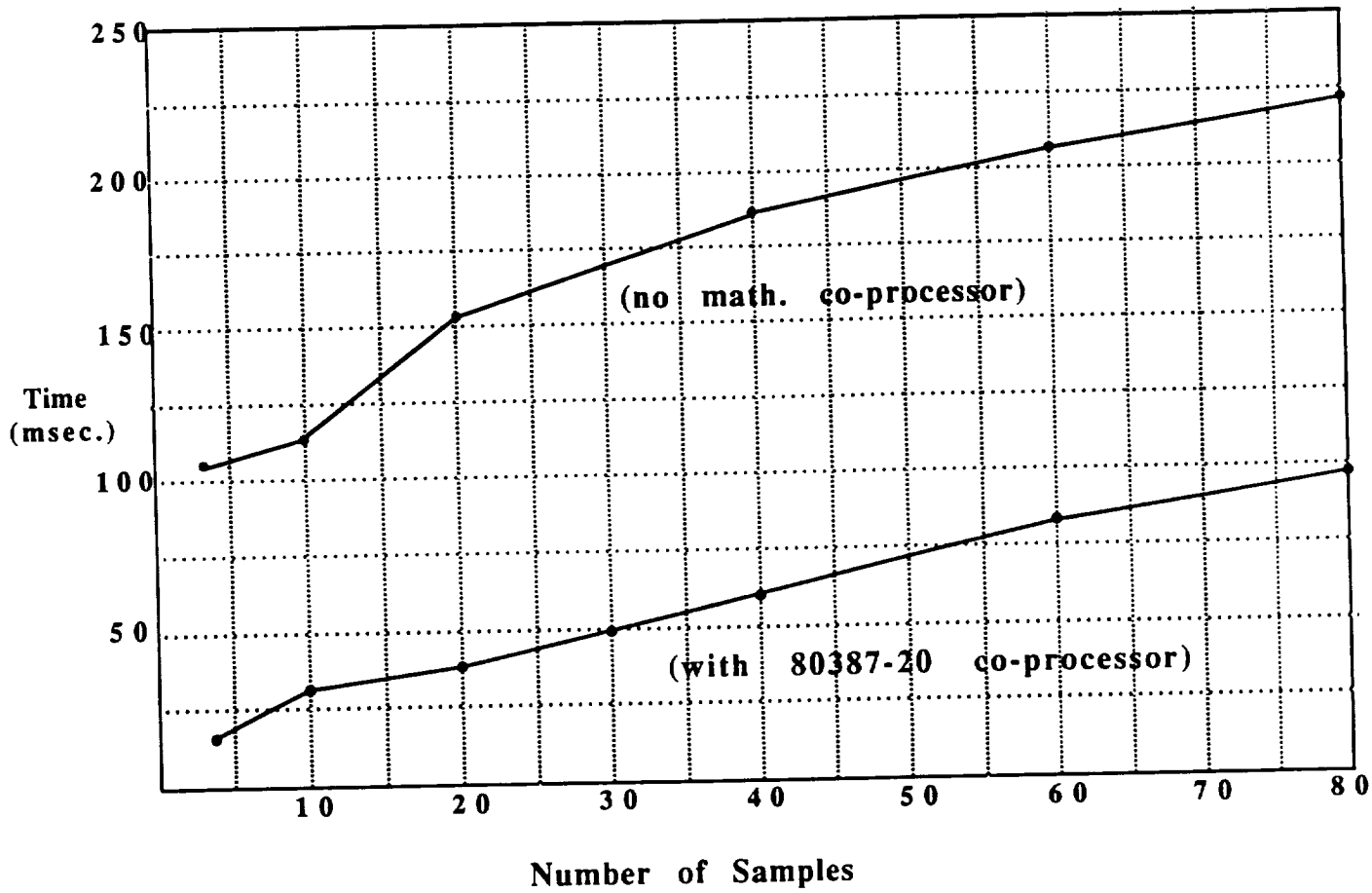


Figure 5.4. Solution times for the DQ algorithm on a Compaq 386/20 computer.

translation errors, the DQ algorithm exhibits better performance than the SVD algorithm in all the cases. Even in the case of 30 points, which is supposed to provide a good estimation, the DQ algorithm provides average accuracy improvement of 40% for the translation parameter calculation compared with the SVD algorithm.

Figure (5.4) shows the solution times. Except for the case of three samples, which uses three points – the minimum number of required points, all the samples consist of an equal number of points and vectors. From this figure we see that the computation time is

approximating linear in the number of samples taken, which confirmed our analysis about the algorithm's time complexity. The computation time increases at about a rate of 1.0 millisecond/sample when a math-processer is used.

During simulation, we also find that the SVD algorithm has many crashes. Table 5.4 provides a partial listing of data sets taken from the vertex set of the "Feeder", in which the SVD algorithm crashes. We have never encountered a crash from DQ algorithm.

Fig. 5.5, 5.6 and Fig. 5.7 give more complete simulation results for the two algorithms. All the simulation processes are exactly the same as that we have described in the begining of this section except more noise levels are tested (seven levels in fact), from Sd=0.1 to Sd=5.0. In Fig. 5.5, only points are used as the input of the DQ algorithm. Fig. 5.6 shows the simulation results when the input consists of equal numbers of sample points and direction vectors. The purpose of this simulation is to see if there is any improvement on the localization accuracy when direction vectors are added as the input to the DQ algorithm. Comparing these two figures, we find that when the number of sample points is small (below 10 points), adding direction vectors as additional input to the DQ algorithm indeed improves the output accuracy; if more that 10 points are used as the input, adding direction vectors has no influence on the accuracy of the computation. Fig. 5.7 shows the simulation results of the SVD algorithm. Compared with Fig. 5.5, we find that, in most cases, DQ algorithm gives better estimation than the SVD algorithm.

```
Begin the execusion of the algorithm:   input data SD = 1.5

SVD algorithm crashed on the following data set: (loop 4)
i: 1  -152.400  -42.862    9.525
i: 2  -152.400  -11.225  127.475
i: 3  -152.400    0.000   89.237

 SVD algorithm crashed on the following data set: (loop 5)
i: 1  -161.925   42.862  127.475
i: 2   161.925  -42.862  142.875
i: 3   161.925  -42.862   -9.525

 SVD algorithm crashed on the following data set: (loop 7)
i: 1   152.400   42.862  142.875
i: 2   161.925  -42.862  142.875
i: 3  -152.400   42.862   -9.525

 SVD algorithm crashed on the following data set: (loop 8)
i: 1   152.400    0.000   87.568
i: 2   152.400  -24.696   85.318
i: 3  -152.400   42.862   -9.525

 SVD algorithm crashed on the following data set: (loop 9)
i: 1   152.400   42.862   -9.525
i: 2  -161.925   42.862   -9.525
i: 3  -152.400    0.000  116.229

SVD algorithm crashed on the following data set:  (loop 12)
i: 1   152.400   42.862   -9.525
i: 2   152.400   24.696   85.318
i: 3  -152.400  -42.862   -9.525

SVD algorithm crashed on the following data set:  (loop 15)
i: 1  -152.400   42.862    9.525
i: 2  -152.400   42.862  127.475
i: 3  -152.400   11.225  127.475

SVD algorithm crashed on the following data set:  (loop 19)
i: 1   152.400   24.696   85.318
i: 2   152.400  -42.862   -9.525
i: 3   161.925   42.862  142.875

SVD algorithm crashed on the following data set:  (loop 20)
i: 1   152.400   42.862  142.875
i: 2  -152.400  -42.862   -9.525
i: 3   152.400    0.000   87.568

SVD algorithm crashed on the following data set:  (loop 21)
i: 1   161.925  -42.862   -9.525
i: 2  -152.400  -42.862   -9.525
i: 3   152.400   11.225   98.814

SVD algorithm crashed on the following data set:  (loop 23)
i: 1  -152.400    0.000  116.229
i: 2  -152.400  -11.225  127.475
i: 3   152.400  -42.862   -9.525

SVD algorithm crashed on the following data set:  (loop 24)
i: 1   152.400  -11.225   98.814
i: 2  -152.400  -11.225  127.475
i: 3  -152.400  -24.696  113.979


SVD algorithm crashed on the following data set:  (loop 25)
i: 1  -152.400  -42.862   -9.525
i: 2   152.400  -42.862    9.525
i: 3   152.400   42.862    9.525

Printing the results:

                DQ algorithm                          SVD algorithm
loop     ox       oy       oz      otheta     |  ox1     oy1     oz1     othetal

  4   -21.136    3.421  -11.936     5.752     |  0.000   0.000   0.000   0.000
  5    -7.199   -4.964  -13.358     0.421     |  0.000   0.000   0.000   0.000
  7    -7.547   -8.093  -13.406     0.122     |  0.000   0.000   0.000   0.000
  8    -5.363  -11.756  -11.740    -2.014     |  0.000   0.000   0.000   0.000
  9    -6.722   -9.025  -12.312    -0.129     |  0.000   0.000   0.000   0.000
 12    -8.418   -7.235  -11.562    -0.207     |  0.000   0.000   0.000   0.000
 15    -8.144   -0.614  -10.902     1.700     |  0.000   0.000   0.000   0.000
 19    -7.042   -9.142  -11.996     0.360     |  0.000   0.000   0.000   0.000
 20    -5.523   -6.858  -13.750     0.195     |  0.000   0.000   0.000   0.000
 21    -6.173   -8.294  -13.271    -0.048     |  0.000   0.000   0.000   0.000
 23    -6.231  -13.926  -15.750     0.855     |  0.000   0.000   0.000   0.000
 24    -4.194  -11.968  -13.139    -0.477     |  0.000   0.000   0.000   0.000
 25    -7.770   -7.691  -13.198    -0.135     |  0.000   0.000   0.000   0.000
```

```
Begin the execusion of the algorithm

SVD algorithm crashed on the following

data set (sd = 1.5):


 i: 1 -152.400  -42.862  127.475
 i: 2  152.400   11.225   98.814
 i: 3  161.925  -42.862   -9.525
 i: 4 -161.925  -42.862  127.475
 i: 5  152.400   11.225   98.814

 Printing the results:

                DQ algorithm
 loop     ox       oy       oz      otheta
   2   -4.487   -11.891  -17.168    -1.090


                SVD algorithm
          |  ox1     oy1     oz1     othet
          |  0.000   0.000   0.000
```

Table 5.4. Data sets taken from the vertex set of the "Feeder" in which SVD algorithm crashes

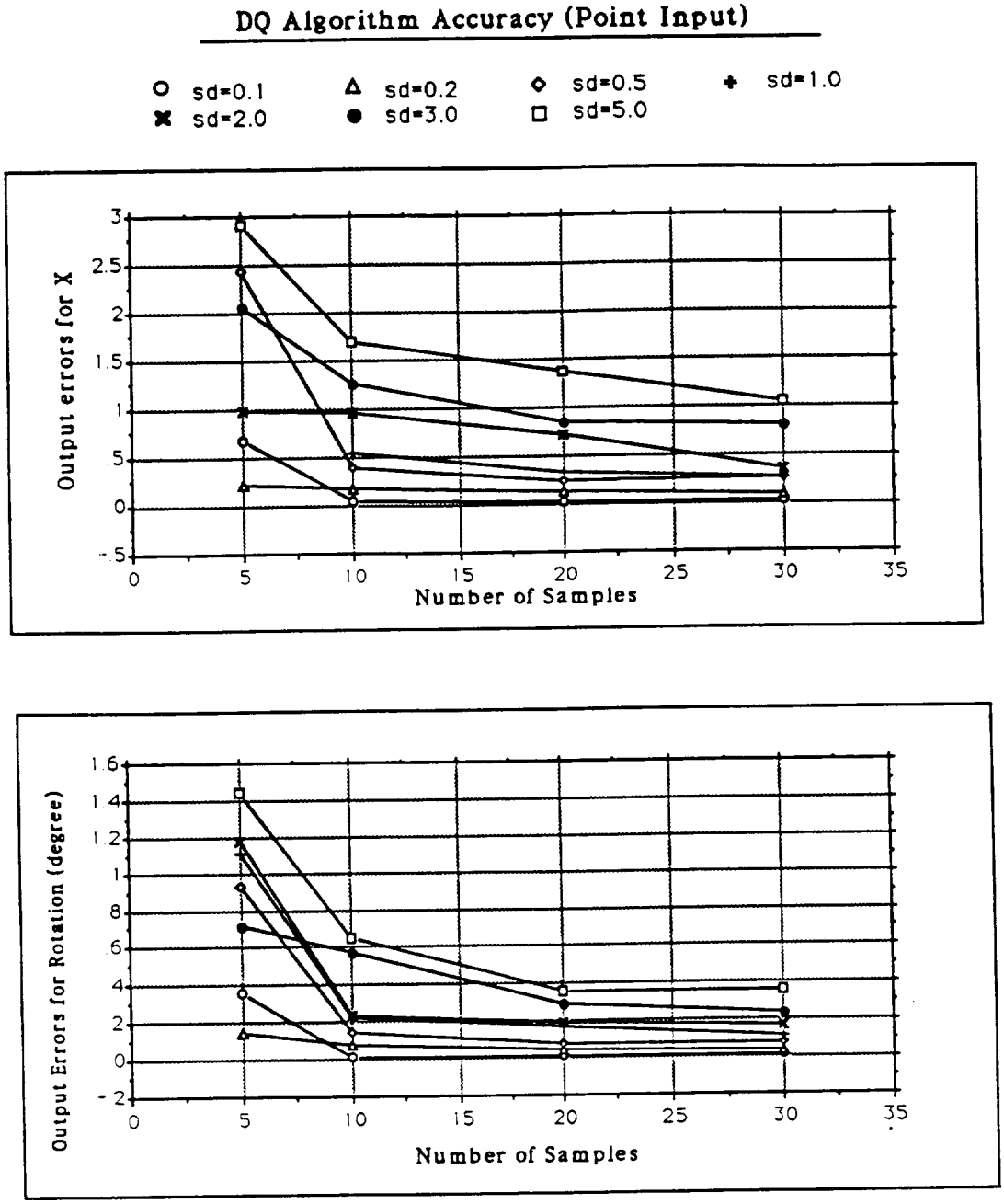


Figure 5.5. DQ Algorithm Accuracy Testing Result – Point Input

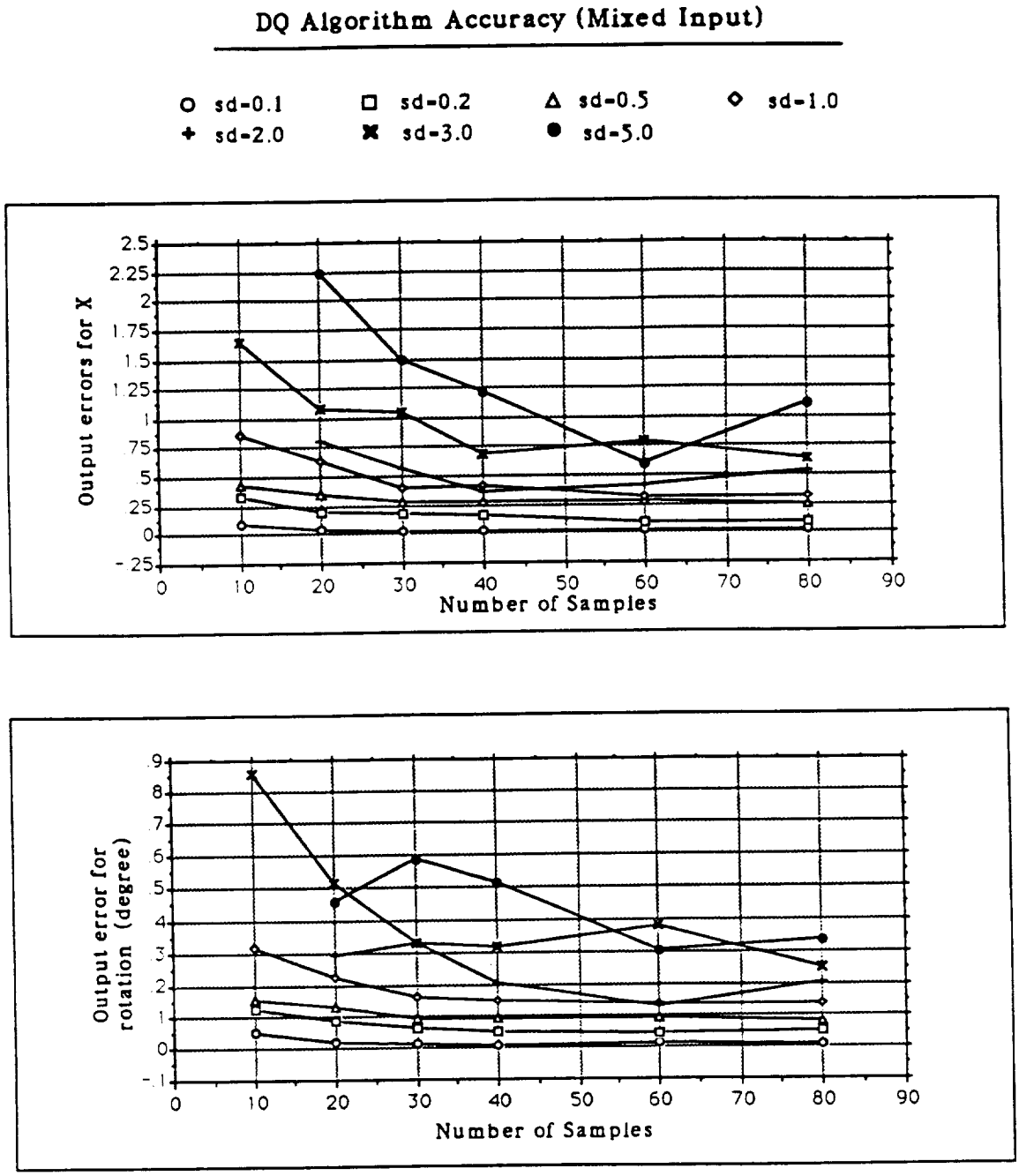


Figure 5.6. DQ Algorithm Accuracy Testing Result – Mixed Input

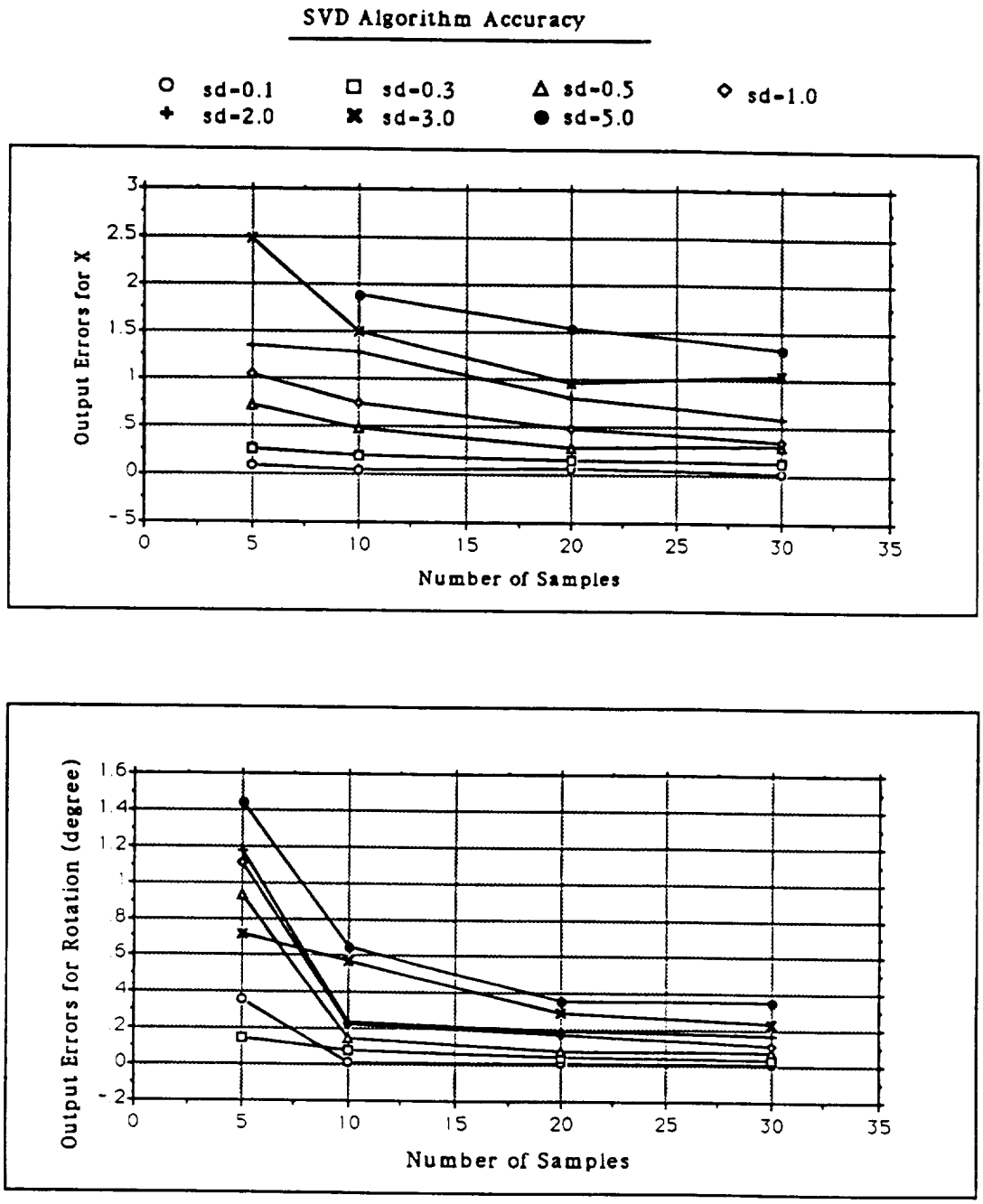


Figure 5.7. SVD Algorithm Accuracy Testing Result

CHAPTER 6

LINE RANGE SENSOR CALIBRATION

6.1 Internal parameter calibration

Sensor internal parameter calibration is the process of identifying parameter values associated with sensor internal structure. More specifically, the internal calibration is to determine the transformation matrix which describes the relationship between the 3-D sensor coordinate frame and the 2-D coordinate frame of the image plane. That is, for each readings in a sensor array, different sensor structures require different internal parameter specifications.

The triangulation-based point range sensor which is described in Chapter 4 uses Eq.(4.1) to calculate the depth value at each position. Therefore, the focal distance d, the source receiver separation value h and the twist angle B between the lens and the detector array are the internal parameters that must be carefully calibrated in order to get an accurate measurement. For the line range sensor, because Eq.(4.2) is used to establish the relationship between the 3-D coordinates in the sensor frame and their corresponding sensor array readings, the angle θ of field of view and the correct e_i value for each pixel i are the parameters which need to be calibrated. In our experiments, because a commercial product is used, it is reasonable to assume that the internal calibration has been completed

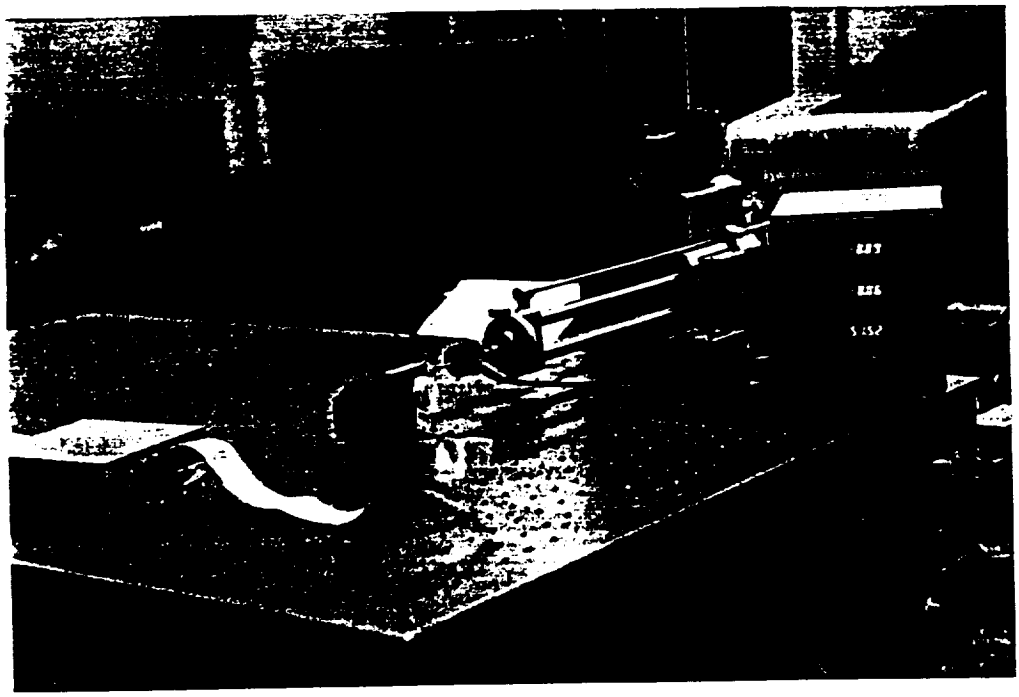


Figure 6.1. X-Y-Z Positioner used for sensor calibration test

before the sensor is delivered and the sensor specifications can be used as the reference for meaningful range measurements. During the experiments, however, we noticed that the sensor readings do not agree with what the sensor specifications. Therefore, a simple test was arranged to test the internal sensor parameters, e.g., the field of view (or the line length) and the coefficients of the linear relationship between the sensor detector array readings and true distances.

6.1.1 Test equipment

The equipment used to perform the tests included an X-Y-Z manual positioner which provide translational movement in three degrees of freedom with a position accuracy of $0.01mm$ along each direction (see Figure 6.1) and a special-shaped test object. The object

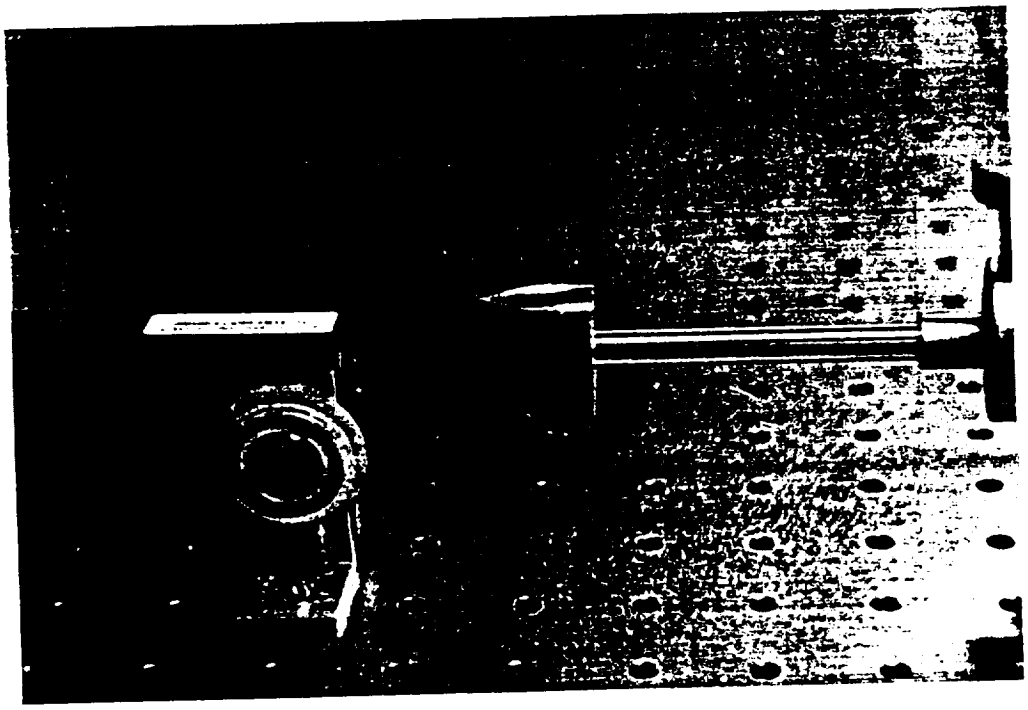


Figure 6.2. Calibrate sensor's angle of field of view and e_i readings

is made of a prism and a cylinder. The cylinder portion of the calibration object is tightly mounted on the movable head of the positioner so that the axis of the object is coincident with the Z axis of the positioner. The position of the object is so adjusted that the sharp edge of the prism of the object is parallel to Y axis of the positioner

6.1.2 The e_i readings

Experiments have shown that the relationship between the sensor readings and the real distance is a linear one. That is, if e_{i_k} represents the kth readings from the detector cell i, z_{i_k} is its corresponding true depth data and a total number of n readings are taken from

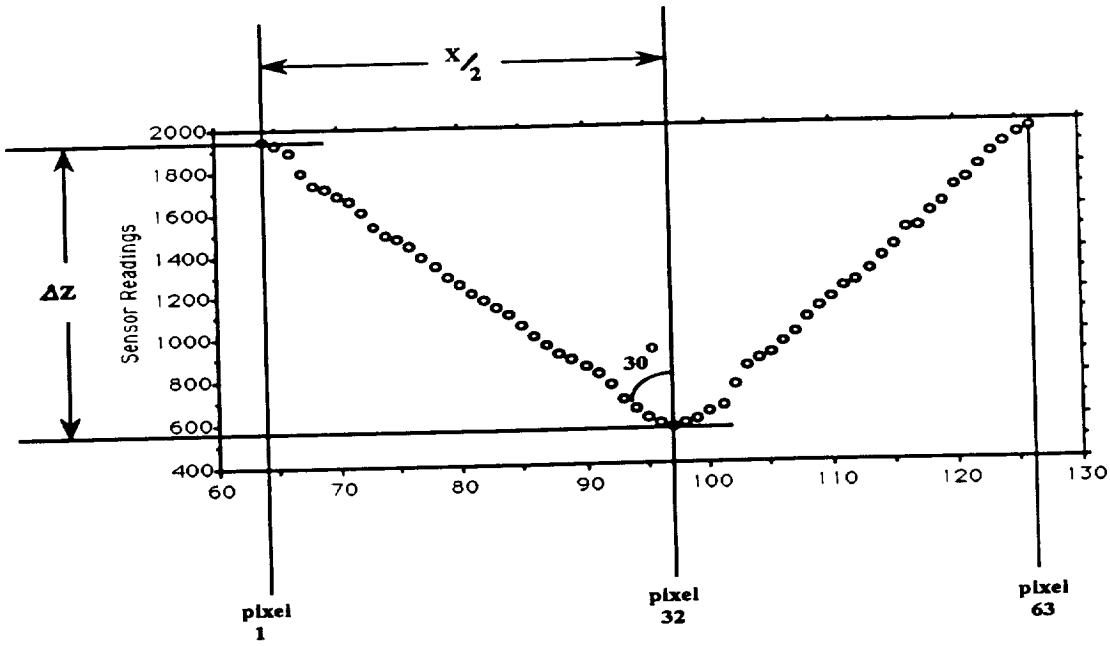


Figure 6.3. A frame of range image to compute the angle of field of view.

different distances, we have the following set of linear equations for each $i \in [1, 63]$

$$z_{i_k} = a_i e_{i_k} + b_i \quad 1 \leq k \leq n. \tag{6.1}$$

A least squares fit can be used to compute each a_i and b_i.

In real experiments, the sensor is placed in such a way that its yz plane is coincident with the yz plane of the positioner and the x axes of the two coordinate systems are parallel. The prism mounted on the positioner is moved along the z direction (see Fig.6.2). As the prism moves, the readings and its corresponding distance values are recorded and Eq.(6.1) is used to compute the coefficients.

6.1.3 The angle of field of view

The same experimental set-up can also be used to estimate the line length. The line length can be calculated by using the properties of trigonometric functions. Fig.6.3 is a

frame of range image obtained during the measurement process as described in section 6.1.2. The line length x_i can be derived by using the following formula:

$$x_i = 2 * \tan 30^O * \Delta z_i \tag{6.2}$$

where i represents the ith frame of measurement and

$$\Delta z_i = z_{1_i} - z_{32_i} = (a_1 e_{1_i} + b_1) - (a_{32} e_{32_i} + b_{32}) \tag{6.3}$$

The final value of line length can be computed by averaging all the x_i's. Using the simple test procedure, the computed line length is $1.722mm$, which is quite different from the specification ($2.0mm$). The corresponding angles of field of view are $3°17'$ and $3°49'$. This modified value has been used in feature extraction experiments and the correctness of the result has been confirmed from these experiments.

6.2 External Parameter Calibration

Usually range sensors are mounted somewhere on the robot to make measurements of the surrounding objects. To make sensor information useful for the robot, the relative position between the sensor coordinate frame and the coordinate frame of the robot must be calibrated, which is called the external calibration problem.

6.2.1 Related coordinate frames and transformation matrices

In order to describe various calibration procedures, in the following a list of definitions of useful coordinate frames and transformation matrices is given:

W : The world coordinate frame. Usually it is assumed to be attached to the base of the robot, and is always fixed no matter how the robot arm moves.

S : The line sensor coordinate frame. By convention, the z axis coincides with the optical axis of the sensor and the x axis is parallel to the detector array.

O : The calibration object coordinate frame. This coordinate frame is used to give a complete geometric specification of the object so that the coordinates of every point on the object surface are known a priori relative to this frame.

G : The Gripper coordinate frame. Usually the gripper is installed on the last link of the robot arm.

$\boldsymbol{T}_O^W$: The transformation matrix from O to W. Its rotation submatrix and translation vector are $\boldsymbol{R}_O^W$ and $\mathbf{p}_O^W$ respectively.

$\boldsymbol{T}_S^G$: The transformation matrix from S to G. Its rotation submatrix and translation vector are $\boldsymbol{R}_S^G$ and $\mathbf{p}_S^G$ respectively.

$\boldsymbol{T}_{G_i}^W$: The transformation matrix from gripper frame G_i to W. Its rotation submatrix and translation vector are $\boldsymbol{R}_{G_i}^W$ and $\mathbf{p}_{G_i}^W$ respectively.

$\boldsymbol{T}_O^{S_i}$: The transformation matrix from O to sensor frame S_i. Its rotation submatrix and translation vector are $\boldsymbol{R}_O^{S_i}$ and $\mathbf{p}_O^{S_i}$ respectively.

$\boldsymbol{T}_{G_j}^{G_i}$: The transformation matrix of the gripper frame G_j to gripper frame G_i. Its rotation submatrix and translation vector are $\boldsymbol{R}_{G_j}^{G_i}$ and $\mathbf{p}_{G_j}^{G_i}$ respectively.

$\boldsymbol{T}_{S_j}^{S_i}$: The transformation matrix of the sensor frame S_j to sensor frame S_i. Its rotation submatrix and translation vector are $\boldsymbol{R}_{S_j}^{S_i}$ and $\mathbf{p}_{S_j}^{S_i}$ respectively.

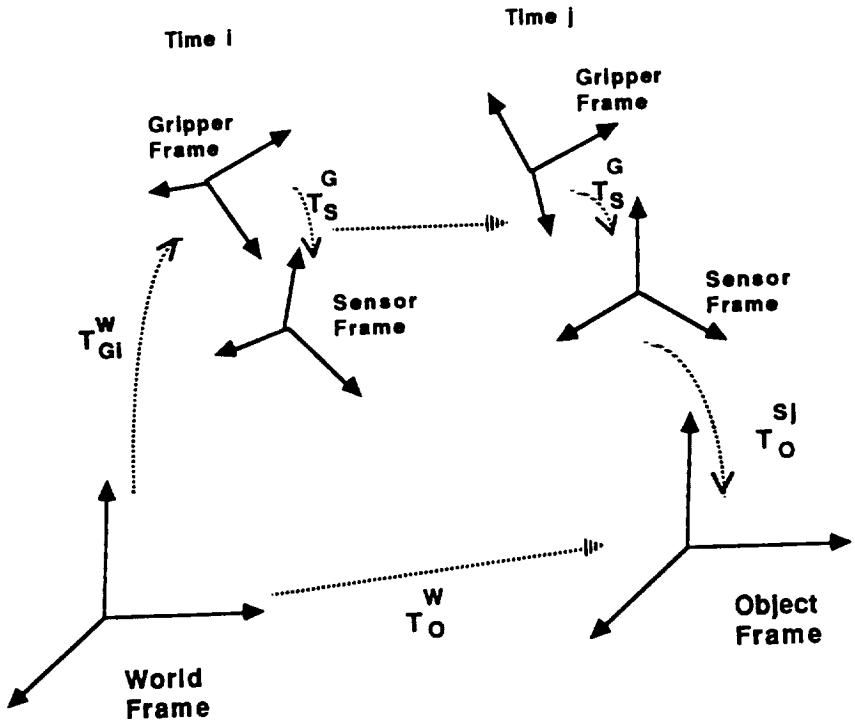


Figure 6.4. Illustration of the relationships among coordinate frames during calibration

6.2.2 Basic relationships among coordinate frames

As illustrated in Fig.(6.4), the W, G, S and O frames have the following relationships:

1. *Fixed transformation matrices*:

   The relative positions between the pair W and O as well as the pair G and S are fixed during calibration process. Therefore, the transformation matrices $\boldsymbol{T}_O^W$ and $\boldsymbol{T}_S^G$ do not change. The objective of the external calibration is to determine $\boldsymbol{T}_S^G$, e.g., the transformation matrix of the sensor coordinate frame with respect to the gripper coordinate frame.

2. *Computable transformation matrices*:

The transformation matrices $(T_{G_i}^{W})$'s are computable matrices during calibration. The transformation matrix $T_{G_i}^{W}$ can be obtained by the robot controller from the readings of the joint encoder values and the computation of a chain of forward kinematic transformations along these joints. The matrices of $(T_{O}^{S_i})$'s will also be available, if the line sensor can determine the location of the object with respect to the sensor frame by using only one frame of measurement, such as is the case when the line sensor uses a grid of lines, or multiple lines to make a measurement and two line-segments features can be extracted from a single frame of line range image.

3. *Derivable transformation matrices:*

   The transformation matrices $T_{G_j}^{G_i}$ and $T_{S_j}^{S_i}$ can be directly derived from the observations at two different locations

$$T_{G_j}^{G_i} = [T_{G_i}^{W}]^{-1} T_{G_j}^{W} \tag{6.4}$$

   and

$$T_{S_j}^{S_i} = T_{O}^{S_i} [T_{O}^{S_j}]^{-1} \tag{6.5}$$

4. *The basic transformation equation:*

   As the robot arm moves from one position to another, the following transformation chain holds and is the basis to compute T_{S}^{G}:

$$T_{O}^{W} = T_{G_i}^{W} T_{S}^{G} T_{O}^{S_i} \tag{6.6}$$

As we will see, to accomplish external calibration, the sensor has to take necessary measurements on a calibration object and make corresponding computation. The degree of difficulty in solving this problem will be different if external conditions are varying. In the following, we will discuss calibration procedures for three different situations:

1. The object is placed in a pre-determined position and the sensor can locate the object from a single frame of measurement;
2. The object location is unknown and the sensor is still able to locate the object by using a single frame of measurement;
3. The object location is unknown and the sensor can only locate certain features from a single frame of measurement, for example locate a position vector from that object

.

The calibration is quite trivial in the first case in which the relative location between the world coordinate frame and the object frame, T_O^W, is precisely known. This condition can be met if it is possible to place the object accurately in a pre-determined location. In this case, the transformation matrix can be easily derived from Eq.(6.6)[1]

$$T_S^G = [T_{G_i}^W]^{-1} T_O^W [T_O^{S_i}]^{-1} \tag{6.7}$$

Although this method is very straightforward and requires little computation, it is rarely seen in real application because finding the accurate location between the world frame W and the object frame O is not easy.

6.2.3 The calibration algorithms with T_O^W unknown

In this section, we describe the calibration procedures for a more general case in which (1) there are no constraints on the relative location between the object frame and the world frame, and (2) the line range sensor is able to locate the calibration object each

[1] An implicit assumption has been made here that the matrices $T_{G_i}^W$ are available, as it is the case in general when the matrix can be obtained from the readings of robot controller's decoder

time the robot moves to a particular position, e.g, $T_O^{S_i}$ is known at location i. In this case the calibration of the sensor's position with respect to robot gripper frame is computed by displacing the robot and observing the changes in the sensor frame using the sensing system.

Suppose the robot arm moves from position 1 to position 2 (see Figure 6.4). We will have two transformation equations, e.g., Eq.(6.6) for $i = 1, 2$. Because the object is fixed during the calibration, the following equation can be obtained

$$T_{G_1}^W T_S^G T_O^{S_1} = T_{G_2}^W T_S^G T_O^{S_2} \tag{6.8}$$

Except for singular points, the transformation matrices are invertible and Eq.(6.8) can be changed into the following form

$$AT_S^G = T_S^G B \tag{6.9}$$

from Eq.(6.8), where

$$A = [T_{G_2}^W]^{-1} T_{G_1}^W = T_{G_1}^{G_2} \tag{6.10}$$

$$B = T_O^{S_2} [T_O^{S_1}]^{-1} = T_{S_1}^{S_2} \tag{6.11}$$

Because A and B are known matrices, the calibration problem can be thought of as a problem of solving a homogeneous transformation equation of the form $AX = XB$ [109].

Finding the general solutions for the matrix equations of the form $AX = XB$ is not an unsolvable problem and it has long been discussed in linear algebra theory [51]. [109] also has given a solution for this equation. However, in our application, because rotation is involved in the computation which enables us to use many important properties of rotation, we can expect to derive a much simpler solution.

In order to explore the properties of the rotation matrix, we decompose Eq.(6.9) into

the following form

$$\begin{bmatrix} \boldsymbol{R}_A & \mathrm{p}_A \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_S^G & \mathrm{p}_S^G \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_S^G & \mathrm{p}_S^G \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_B & \mathrm{p}_B \\ 0 & 1 \end{bmatrix} \tag{6.12}$$

This equation can further be expressed as the following two equations:

$$\boldsymbol{R}_A \boldsymbol{R}_S^G = \boldsymbol{R}_S^G \boldsymbol{R}_B \tag{6.13}$$

$$[\boldsymbol{R}_A - \boldsymbol{I}]\mathrm{p}_S^G = \boldsymbol{R}_S^G \mathrm{p}_B - \mathrm{p}_A \tag{6.14}$$

where

$\boldsymbol{I}$ is the 3×3 identity matrix;

$\boldsymbol{R}_A = [\boldsymbol{R}_{G_2}^W]^{-1} \boldsymbol{R}_{G_1}^W = \boldsymbol{R}_{G_1}^{G_2}$is the rotation submatrix of $\boldsymbol{A}$;

$\boldsymbol{R}_B = \boldsymbol{R}_O^{S_2}[\boldsymbol{R}_O^{S_1}]^{-1} = \boldsymbol{R}_{S_1}^{S_2}$ is the rotation submatrix of $\boldsymbol{B}$;

p_A and p_B are the translation parts of $\boldsymbol{A}$ and $\boldsymbol{B}$.

From Eq.(6.13) and (6.14), we can observe the following

1. $\boldsymbol{R}_S^G$ is dependent only on rotational matrices $\boldsymbol{R}_A$ and $\boldsymbol{R}_B$. That is, it can be solved independently without the knowledge of the translational components of those homogeneous transformation matrices.

2. p_S^G is dependent on $\boldsymbol{R}_S^G$. That is, p_S^G can be solved only after $\boldsymbol{R}_S^G$ is solved.

3. The key for the sensor calibration is to solve for Eq.(6.13).

In next section, two algorithms are given to solve for the rotation matrix $\boldsymbol{R}_S^G$. Both algorithms use the properties of the rotation matrix to make the computation both simple and accurate.

6.2.3.1 Solving for the rotation matrix R_S^G

Before solving for the rotation matrix R_S^G, we first explore the geometrical interpretations of Eq.(6.13). To make our analysis clearer, we rewrite the Eq.(6.13) in a more general form $R_A R = R R_B$.

Lemma 6.1 *The eigenvalues of a rotation matrix are* 1 *and* $\cos\delta \pm i\sin\delta$ *where* δ *is as gives below. Let* $\lambda_1 = \cos\delta + i\sin\delta$ *and* $\lambda_2 = \cos\delta - i\sin\delta$*; then* δ *can be calculated by*

$$\delta = \arctan\left(|Re(\lambda_1 - \lambda_2)|, \lambda_1 + \lambda_2\right) \tag{6.15}$$

where $Re(\lambda_1 - \lambda_2)$ *is the real part of* $\lambda_1 - \lambda_2$.

Proof: See page 412 of [73].

Lemma 6.2 *Rotation matrices* R_A *and* R_B *have the same angle of rotation.*

Proof: Because a rotation matrix is always invertible, we have $R_A = R R_B R^{-1}$. So R_A and R_B are similar. Similar matrices have the same eigenvalues and thus have the same angle of rotation from Lemma 6.1. #

Lemma 6.3 *Let* n_A *be the rotation axis of* R_A *and* n_B *the rotation axis of* R_B*. For any matrix* R*, if* $R_A R = R R_B$*, then* $\mathrm{n}_A = R\mathrm{n}_B$.

Proof: From given condition that $R_A R = R R_B$, we have

$$R_A R\ \mathrm{n}_B = R R_B \mathrm{n}_B \tag{6.16}$$

Because a rotation matrix always rotates its axis into itself, the right side of Eq.(6.16) can be simplified

$$R_A R\ \mathrm{n}_B = R\ \mathrm{n}_B \tag{6.17}$$

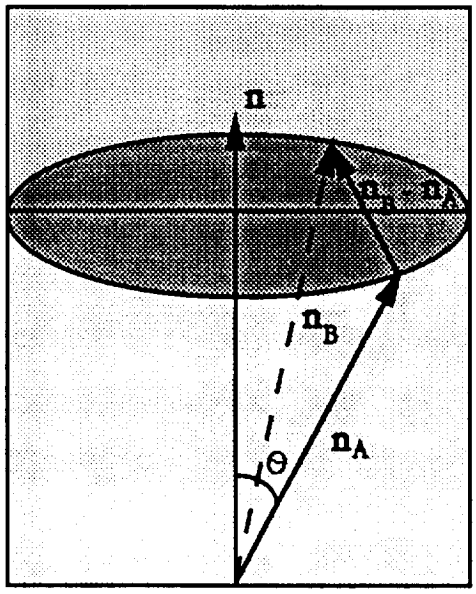


Figure 6.5. Geometrical interpretation of the rotation axis

Now suppose $\boldsymbol{R}\ \mathbf{n}_B = \mathbf{n}_k$, then Eq.(6.17) becomes

$$\boldsymbol{R}_A \mathbf{n}_k = \mathbf{n}_k \tag{6.18}$$

Eq.(6.18) means that the vector $\mathbf{n}_k$ is an eigenvector of $\boldsymbol{R}_A$ with the corresponding eigenvalue equal to 1. But only the rotation axis of that rotation matrix has this property. Therefore $\mathbf{n}_k = \mathbf{n}_A$. That is

$$\boldsymbol{R}_A \boldsymbol{R}\ \mathbf{n}_B = \boldsymbol{R}_A \mathbf{n}_k = \mathbf{n}_A \tag{6.19}$$

Substituting the result of Eq.(6.19) into the left side of Eq.(6.17), we have the required equation:

$$\mathbf{n}_A = \boldsymbol{R}\ \mathbf{n}_B \tag{6.20}$$

#

The geometrical interpretation of Lemma 6.3 is that the rotation axis of rotation matrix $\boldsymbol{R}$ will rotate $\mathbf{n}_B$ into $\mathbf{n}_A$. Because a rotation preserves the angle between the transformed vector and the rotation axis, and if we define $\mathbf{n}$ as the rotation axis of $\boldsymbol{R}$, we will have the following relationship:

$$\mathbf{n} \cdot \mathbf{n}_B = \mathbf{n} \cdot \mathbf{n}_A \tag{6.21}$$

That is, $\mathbf{n} \cdot (\mathbf{n}_B - \mathbf{n}_A) = 0$, which means that the rotation axis of $\boldsymbol{R}$ is perpendicular to $(\mathbf{n}_B - \mathbf{n}_A)$. In fact, we have the following Lemma:

Lemma 6.4 *If* $\mathbf{n} \perp (\mathbf{n}_B - \mathbf{n}_A)$ *then vector* $\mathbf{n} \times (\mathbf{n}_B + \mathbf{n}_A)$ *is parallel to* $(\mathbf{n}_B - \mathbf{n}_A)$.

Proof: Because

$$\begin{aligned} (\mathbf{n}_B - \mathbf{n}_A) \cdot (\mathbf{n}_B + \mathbf{n}_A) &= \mathbf{n}_B \cdot \mathbf{n}_B + \mathbf{n}_B \cdot \mathbf{n}_A - \mathbf{n}_A \cdot \mathbf{n}_B - \mathbf{n}_A \mathbf{n}_A \\ &= 0 \end{aligned}$$

We can conclude that the vector $\mathbf{n}_B - \mathbf{n}_A$ is perpendicular to both the rotation axis $\mathbf{n}$ and the vector $\mathbf{n}_B + \mathbf{n}_A$. On the other hand, $\mathbf{n} \times (\mathbf{n}_B + \mathbf{n}_A)$ is also perpendicular to both of these two vectors. Therefore, $\mathbf{n} \times (\mathbf{n}_B + \mathbf{n}_A)$ is parallel to $(\mathbf{n}_B - \mathbf{n}_A)$. #

Based on the above geometrical analysis, it is not difficult to show that the rotation axis always lies in a plane which

1. is perpendicular to the vector $(\mathbf{n}_B - \mathbf{n}_A)$, and

2. intersects $(\mathbf{n}_B - \mathbf{n}_A)$ at its midpoint.

E.g., the rotation axis has one degree of rotation freedom ($\mathbf{n}_B$ can be rotated into $\mathbf{n}_A$ about any line in that plane that also goes through the intersection of $\mathbf{n}_A$ and $\mathbf{n}_B$ – see Figure6.5). That is, one pair of $\boldsymbol{R}_A$ and $\boldsymbol{R}_B$ defines a plane in which the rotation axis

must lie in. As a consequence, at least two distinct pairs of $\boldsymbol{R}_A$ and $\boldsymbol{R}_B$ matrices are needed to determine $\boldsymbol{R}$; and Eq.(4.48), (4.49) and (E.18) will be used to compute the rotation matrix.

From above analysis, two pairs of $\boldsymbol{R}_A$'s and $\boldsymbol{R}_B$'s are the minimum requirement to compute the rotation matrix (in the next section, we will show that this is also the minimum requirement to compute the translation vector). That is, the robot arm has to move at least twice from position w_0 to w_1 and then to w_2 to compute the calibration parameters. Between w_0 and w_1, matrices $\boldsymbol{T}_{A_1}$ and $\boldsymbol{T}_{B_1}$ are obtained; so are $\boldsymbol{T}_{A_2}$ and $\boldsymbol{T}_{B_2}$ between w_1 and w_2.

To get more accurate calibration results, we usually move the robot arm though a series of positions. The final results of the rotation axis and the rotation angle are the average values of these observations [45].

Suppose the robot moves m times, e.g., from position w_0 to w_1, then from w_1 to w_2 *etc.*, until from w_{m-1} to w_m. The observed matrices are $\boldsymbol{T}_{A_i}$'s and $\boldsymbol{T}_{B_i}$'s where $i = 1, \cdots m$. Because two pairs of $\boldsymbol{R}'_A s$ and $\boldsymbol{R}_B$ are needed to derive one rotation parameters, a total number of of $m-1$ rotation parameters can be derived from m pairs of $\boldsymbol{R}_A$s and $\boldsymbol{R}_B$'s. The final rotation parameters will be

$$\mathbf{n} = \frac{1}{m-1} \sum_{i=1}^{m-1} \mathbf{n}_i \tag{6.22}$$

and

$$\theta = \sum_{i=1}^{M-1} \frac{\theta_i}{M-1} \tag{6.23}$$

where $\mathbf{n}_i$ and θ_i are the rotation axis and rotation angle which are derived from the pairs of $\boldsymbol{R}_{A_i}$, $\boldsymbol{R}_{B_i}$ and $\boldsymbol{R}_{A_{i+1}}$, $\boldsymbol{R}_{B_{i+1}}$ matrices. Let $\mathbf{n}_i = [n_{x_i}, n_{y_i}, n_{z_i}]^T$, the standard deviations

Input: matrices $\boldsymbol{R}_{G_i}^{W}$ and $\boldsymbol{R}_{O}^{S_i}$, $i := 1 \cdots m$;

Output: calibration rotation matrix $\boldsymbol{R}_{S}^{G}$.

Begin

For i := 1 to m to do

Compute matrices $\boldsymbol{R}_{A_i}$ and $\boldsymbol{R}_{B_i}$

where $\boldsymbol{R}_{A_i} = [\boldsymbol{R}_{G_{i-1}}^{W}]^{-1}\boldsymbol{R}_{G_i}^{W}$;

$\boldsymbol{R}_{B_i} = \boldsymbol{R}_{O}^{S_i}[\boldsymbol{R}_{O}^{S_{i-1}}]^{-1}$;

Extract $(\mathbf{n}_{A_i}, \theta_{A_i})$ and $(\mathbf{n}_{B_i}, \theta_{B_i})$ from $\boldsymbol{R}_{A_i}$ and $\boldsymbol{R}_{B_i}$;

For i := 1 to m-1 do

Use Eq. (4.48) (4.49) and (E.18) to compute $\mathbf{n}_i$ and θ_i;

Compute final $\mathbf{n}$ and θ by using Eq. (6.23) and (6.22).

End.

Figure 6.6. The calibration of rotation parameters: Algorithm 1.

are

$$\sigma_{\mathbf{n}_\alpha} = \sqrt{\sum_{i=1}^{m-1} \frac{(\mathbf{n}_{\alpha_i} - \mathbf{n}_\alpha)^2}{m-1}} \qquad \alpha = x, y, z. \tag{6.24}$$

and

$$\sigma_\theta = \sqrt{\sum_{i=1}^{m-1} \frac{(\theta_i - \theta)^2}{m-1}} \tag{6.25}$$

Figure 6.2.3.1 shows the algorithm to derive the rotation parameters of the matrix $\boldsymbol{R}_{S}^{G}$ using rotation axis-angle representation $(\mathbf{n}, \theta)$.

Another algorithm which uses quaternions to represent rotation to compute rotation parameters is described in [116] [117]. In addition to the above-mentioned rotation properties, one more theorem is derived and enables the algorithm to use the least squares

optimization method to compute required parameters.

Lemma 6.5 *If $\mathbf{q}$ is the quaternion representation of a rotation, then $\mathbf{q}$ has the following relationship with $\mathbf{q}_A$ and $\mathbf{q}_B$*

$$(\mathbf{q}_B + \mathbf{q}_A) \times \frac{1}{\sqrt{1 - |\mathbf{q}|^2}}\mathbf{q} = \mathbf{q}_B - \mathbf{q}_A \tag{6.26}$$

Proof: See [117].

The corresponding algorithm is described in Figure 6.7.

The first algorithm has the advantage of "recursive" computation. That is, each time if we take one more observation, the only computation we need is to compute one more estimated rotation parameters and average the new estimated results with the old ones. All the old results are useful and can be saved for later use. But, this method involves trigonometric calculation, which is not very efficient.

The second algorithm, on the contrary, uses quaternion representation and as a consequence, trigonometric computation is avoided during the derivation process. Thus, the algorithm is very simple and accurate. Least Squares Optimization can be used to solve for a system of linear equations as shown in Figure 6.7. The disadvantage of the algorithm is its non-"recursive" nature. At each time when a new observation is made, the whole computation has to start over again. The old results have no use here.

6.2.3.2 Solving for the translation vector $\mathbf{p}_S^G$

Once the rotation matrix is obtained, Eq.(6.14) can be used to solve for the translation vector $\mathbf{p}_S^G$. Again, at least two pairs of equations are needed in order to solve uniquely for $\mathbf{p}_S^G$. This fact is based on the following Lemma.

Input: matrices $\boldsymbol{R}_{G_i}^{W}$ and $\boldsymbol{R}_{O}^{S_i}$, $i := 1 \cdots m$;

Output: calibration rotation matrix $\boldsymbol{R}_{S}^{G}$.

Begin

01 For i := 1 to m to do

02 Compute matrices $\boldsymbol{R}_{A_i}$ and $\boldsymbol{R}_{B_i}$

03 where $\boldsymbol{R}_{A_i} = [\boldsymbol{R}_{G_{i-1}}^{W}]^{-1}\boldsymbol{R}_{G_i}^{W}$;

04 $\boldsymbol{R}_{B_i} = \boldsymbol{R}_{O}^{S_i}[\boldsymbol{R}_{O}^{S_{i-1}}]^{-1}$;

05 Extract $\mathbf{q}_{A_i}$ and $\mathbf{q}_{B_i}$ from $\boldsymbol{R}_{A_i}$ and $\boldsymbol{R}_{B_i}$;

06 Solve for a system of linear equations $\boldsymbol{K}(\mathbf{q}_{B_i} + \mathbf{q}_{A_i}) \cdot \mathbf{q}' = \mathbf{q}_{B_i} - \mathbf{q}_{A_i}$

by using linear least squares technique

07 Compute θ: $\theta = 2\tan^{-1}|\mathbf{q}'|$;

08 Compute $\mathbf{q}$: $\mathbf{q} = \mathbf{q}'/\sqrt{1 + |\mathbf{q}'|^2}$;

09 Compute $\boldsymbol{q}$ from $\mathbf{q}$ and θ;

10 Derive rotation matrix $\boldsymbol{R}$ from quaternion $\boldsymbol{q}$.

End.

notice:

1. $\boldsymbol{K}(\mathbf{r})$ in line 06 is the screw-symmetric matrix

2. The formula in line 06 uses the fact that $\boldsymbol{K}(\mathbf{r}) \cdot \mathbf{n} = \mathbf{r} \times \mathbf{n}$

Figure 6.7. The calibration of rotation parameters: Algorithm 2. [116]

Lemma 6.6 *The rank of matrix* $[\boldsymbol{R} - \boldsymbol{I}]$ *is 2.*

Proof: Because $\boldsymbol{R}$ has three different eigenvalues (see Lemma 6.1), it is diagonalizable. From the theory of linear algebra [42], there exists a 3×3 matrix $\boldsymbol{E}$ such that

$$\boldsymbol{R} = \boldsymbol{E}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_2 \end{bmatrix} \boldsymbol{E} \tag{6.27}$$

where the jth column of the matrix $\boldsymbol{E}$ is the eigenvector $\mathbf{e}_j (j = 1, 2, 3)$ of corresponding eigenvalue. Therefore,

$$\boldsymbol{R} - \boldsymbol{I} = \boldsymbol{E}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_2 \end{bmatrix} \boldsymbol{E}^{-1} - \boldsymbol{E}\boldsymbol{I}\boldsymbol{E} \tag{6.28}$$

$$= \boldsymbol{E}^{-1} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \lambda_1 - 1 & 0 \\ 0 & 0 & \lambda_2 - 1 \end{bmatrix} \boldsymbol{E} \tag{6.29}$$

E.g, $\boldsymbol{R} - \boldsymbol{I}$ is similar to $\begin{bmatrix} 0 & 0 & 0 \\ 0 & \lambda_1 - 1 & 0 \\ 0 & 0 & \lambda_2 - 1 \end{bmatrix}$.

Matrix $\begin{bmatrix} 0 & 0 & 0 \\ 0 & \lambda_1 - 1 & 0 \\ 0 & 0 & \lambda_2 - 1 \end{bmatrix}$, however, has a rank of 2 and similar matrices have the same rank. Therefore, the rank of matrix $\boldsymbol{R} - \boldsymbol{I}$ is 2. #

The number of unknowns of Eq.(6.14) is 3 whereas the rank of matrix $[\boldsymbol{R}_A - \boldsymbol{I}]$ is 2. Thus, at least two pairs of independent observations are needed to solve for $\mathbf{p}_S^G$.

If more observations are available, a linear least squares optimization equations can be set up.

6.2.4 External calibration of a line range sensor

Up to now all the calibration algorithms are under the assumption that the sensor is able to determine the location of the object by using only one frame of measurement. In real applications, this is not always the case. For example, a spot sensor can only give a depth reading of a single point at each position. To locate a polyhedral object, at least six points on three different surfaces have to be measured in order to determine the location of the object [49]. A line range sensor emitting a single striped laser light can only give the depth information along a line on the object surface. To locate an object, two line-segments have to be extracted. Because such kind of sensors provides less information than those we have described earlier, more constraints on the shape of the calibration object and on the movement of the robot are expected in order to calibrate such sensors. In this section we will introduce an algorithm to calibrate a line range sensor. As we described in the previous chapter, a line sensor can extract the parameters of a cone with only one measurement along a line on the surface of the cone if the half angle of the cone is known and the intersection curve between the line and the surface is an ellipse. It means that the position of the cone vertex can be located from a single frame of measurement. This important property will be used in the calibration process.

If we make a series of measurements on the cone and the parameters of the cone can be extracted after each measurement, we will have the following equation

$$\boldsymbol{p}_O^W = \boldsymbol{T}_{G_i}^W \boldsymbol{T}_S^G \boldsymbol{p}_O^{S_i} \tag{6.30}$$

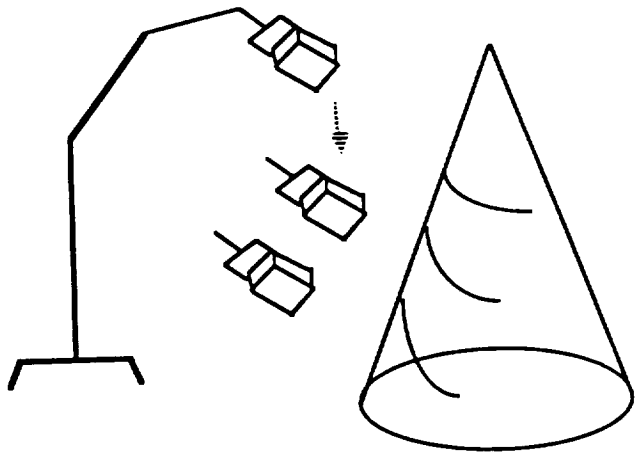


Figure 6.8. Calibration of a line range sensor

where $\boldsymbol{p}_O^W$ is the quaternion representation of the position of the cone vertex with respect to the world frame, and $\boldsymbol{p}_O^{S_i}$ is the quaternion representation of the position of the cone vertex with respect to the sensor frame.

Because the $\boldsymbol{p}_O^W$ is a constant during calibration, we have

$$\boldsymbol{T}_{G_i}^W \boldsymbol{T}_S^G \boldsymbol{p}_O^{S_i} = \boldsymbol{T}_{G_j}^W \boldsymbol{T}_S^G \boldsymbol{p}_O^{S_j} \tag{6.31}$$

Again, Eq.(6.31) can be expressed as

$$\begin{bmatrix} \boldsymbol{R}_{G_i}^W & \mathrm{p}_{G_i}^W \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_S^G & \mathrm{p}_S^G \\ 0 & 1 \end{bmatrix} \boldsymbol{p}_O^{S_i} = \begin{bmatrix} \boldsymbol{R}_{G_j}^W & \mathrm{p}_{G_i}^W \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_S^G & \mathrm{p}_S^G \\ 0 & 1 \end{bmatrix} \boldsymbol{p}_O^{S_j} \tag{6.32}$$

and has the following form

$$\boldsymbol{R}_{G_i}^W \boldsymbol{R}_S^G \mathrm{p}_O^{S_i} + \boldsymbol{R}_{G_i}^W \mathrm{p}_S^G + \mathrm{p}_{G_i}^W = \boldsymbol{R}_{G_j}^W \boldsymbol{R}_S^G \mathrm{p}_O^{S_j} + \boldsymbol{R}_{G_j}^W \mathrm{p}_S^G + \mathrm{p}_{G_j}^W \tag{6.33}$$

If we restrict the robot arm to translational movement during calibration, the rotation matrix $\boldsymbol{R}_{G_i}^W$ is constant

$$\boldsymbol{R}_{G_i}^W = \boldsymbol{R}_{G_j}^W = \boldsymbol{R}_G^W \tag{6.34}$$

Eq.(6.33) can be simplified

$$\boldsymbol{R}_G^W \boldsymbol{R}_S^G \mathbf{p}_O^{S_i} + \mathbf{p}_{G_i}^W = \boldsymbol{R}_G^W \boldsymbol{R}_S^G \mathbf{p}_O^{S_j} + \mathbf{p}_{G_j}^W \tag{6.35}$$

where $\boldsymbol{R}_G^W$ is a constant rotation matrix.

Eq.(6.35) can be further rewritten in the form

$$\boldsymbol{R}_S^G(\mathbf{p}_O^{S_i} - \mathbf{p}_O^{S_j}) = (\boldsymbol{R}_G^W)^{-1}(\mathbf{p}_{G_j}^W - \mathbf{p}_{G_i}^W) \tag{6.36}$$

Let $\mathbf{p}_{S_j}^{S_i} = \mathbf{p}_O^{S_i} - \mathbf{p}_O^{S_j}$ and $\mathbf{p}_{G_j}^{G_i} = (\boldsymbol{R}_G^W)^{-1}(\mathbf{p}_{G_j}^W - \mathbf{p}_{G_i}^W)$, we have

$$\boldsymbol{R}_S^G \mathbf{p}_{S_j}^{S_i} = \mathbf{p}_{G_j}^{G_i} \tag{6.37}$$

The Eq.(6.37) can be interpreted as the following:

The rotation matrix $\boldsymbol{R}_S^G$ rotates the vector $\mathbf{p}_{S_j}^{S_i}$ into the vvector $\mathbf{p}_{G_j}^{G_i}$.

Therefore, all the analysis from last section can be applied here and can be used to derive the desired transformation matrix. Again, at least two pairs of observations are needed to specify the transformation matrix. Thus, the calibration procedure to calibrate a line range sensor is basically the same as the general calibration procedures with two more constraints (see Figure 6.8):

1. Constraint on the shape of the object: the object is a cone with a known half angle;
2. Constraint on the movement of the robot: only translational movement is allowed during calibration.

CHAPTER 7

SUMMARY AND FUTURE WORK

7.1 Summary

7.1.1 Object location determination

Real-time and high-accuracy are the two primary concerns in many object localization applications. To achieve real-time localization, the efforts should focus on the following:

1) Developing fast sensing techniques.

2) Reducing the overhead for processing sensed data.

3) Developing fast algorithms to compute the location parameters.

To achieve high-accuracy, in addition to developing high quality range sensors, developing robust or noise-insensitive localization algorithms is equally important.

The thesis has discussed different methods and strategies for object localization and important issues for fast and accurate localization. The thesis has presented two object localization algorithms and examined the applications of these algorithms in telemanipulation tasks and other manufacturing operations. Both algorithms are fast and noise-insensitive.

The first algorithm is based on *line-segment to line-segment matching*, e.g, matching modeled line-segment features to the line-segment features extracted from the range data which are taken on the surfaces of an object. The line-segment features may be boundary edges of a polygon or the intersection of two planar surfaces of an object; or an axis of the surface of revolution of an object. The thesis has provided a detailed description of the whole localization process including line-segment feature extraction and location computation. The emphasis is on the extraction of two types of boundary edges: sharp-type edges and rounded-type edges, and the extraction of the axis of a cylindrical surface or a conic surface. The feature extraction was performed both on real line range data and the simulation of noisy range data. Two sources of noise hve been considered, e.g., the sensor measurement noise and the roughness of the object surfaces. During simulation, the sensor noise is assumed to have a normal distribution and the roughness of the object surfaces is assumed to be uniformly distributed. The feature extraction, particularly axis extraction, performed equally well with different noise levels. To reduce the overhead for processing sensed range data, the thesis has made every effort to try to develop non-iterative methods to solve the computation problems needed during the localization process and has achieved some success. Closed form solutions are used to carry out most of computations during the localization process.

The second algorithm is more general. The inputs to the algorithm are not limited only to line features. Featured points (*point-to-point* matching) and featured unit direction vectors (*vector-to-vector* matching) can also be used as the inputs of the algorithm and there is no upper limit on the number of the features inputed. The algorithm will allow the use of redundant features to find a better solution. The algorithm uses dual number quaternions to represent the position and orientation of an object and uses the least

squares optimization method to find an optimal solution for the object location. The advantage of using this representation is that the method solves for the location estimation by minimizing a single cost function associated with the sum of the orientation and position errors. Thus it provides better estimation performance, both in accuracy and in speed, compared with other similar algorithms. Because the thesis does not deal with the problem of how point features and vector features are extracted from range data, only computer simulation is used to test the performance of the algorithm. The experimental results have indeed displayed a significant improvement in localization accuracy over other similar algorithms.

7.1.2 Object localization technique in tele-autonomous system

When controlling a remote robot to perform manipulation tasks, an operator has to face two problems: time-delays on the signal transmission and the uncertainties of the remote environment.

Prediction display and forward simulation constitute a good method to overcome the time-delay problem. In this approach the operator controls a simulation of the remote robot without time-delay, with the control signals sent in parallel to the remote system. The introduction of time and position desynchronization has further developed the concept of predictor display, which allows the operator to desynchronize the time and position frames, respectively, of the simulation and the remote robot and further enhance the operator's control ability and flexibility.

The uncertainty problem usually can be approached from two different objectives: 1) developing techniques that help **avoid** the uncertainty; and 2) developing techniques that help the system **adapt** to the uncertainty. The use of object localization techniques in

a remote system to overcome the uncertainty problem is based on the measurement and error-removal strategy and thus belongs to the first category.

The thesis has discussed two cases where the object localization could help. The first case is the situation where direct manual control is used to perform a tele-manipulation task. The second case is the situation where the remote system has certain degree of automation ability.

To handle both cases, a relative move mode is proposed in order for the remote system to use the localization ability to perform tele-manipulation tasks. The basic idea of this approach is that the operator will control the simulated robot as usual, but the position signals, before being sent to the remote site, are converted to positions relative to the object to be manipulated and then transmitted. The remote system is capable of sensing the relative position of the object to be manipulated in real-time. It uses this sensed information to transform the received relative commands into absolute position commands and then proceed to follow the commanded positions. One of the essential prerequisites to implement the above control sequence is the separation of the time frames between the local operations and remote system commands. The time-desynchronization has played an important rule again.

7.2 Future Work

7.2.1 Feature Extraction and Localization Algorithms

This thesis addressed only limited types of feature extraction problems. Even these problems have not been solved completely. The followings are some of the interesting topics which are worthy of further research:

1. Development of efficient and reliable algorithms to do either elliptical curve or general quadric curve fitting from a set of measured range data. Investigation of how the shapes of the resulting curve are influenced by the patterns of measurement noises.

2. Extending the multi-scan axis extraction method to the axis extraction of other types of surfaces of revolution or a generalized cylinder.

3. Study of the method of extracting other types of features, such as point features, vector features, curvature features or features which can be used to describe irregular surfaces. One example is to find an efficient algorithm to do extraction of sphere parameters (the radius and the center of the sphere).

The two object localization algorithms need to be further studied. In the line-segment matching algorithm, a closed form formula is used to compute the rotation and translation. In deriving these formulas, it is assumed that the two pairs of line-segments are perfectly aligned. In fact, because of various errors, they are not. Therefore, studying the effects of their mis-alignment is another research topic.

In the DQ algorithm, the thesis has not discussed the selection of the weighting factors. Intuitively, the weighting factors are closely related to the reliability of corresponding matching features. Let us take the extraction of two surface normals as an example. The two surface normals extracted from two planar surfaces are used as part of the algorithm's inputs. If one surface is toward the sensor, another surface is near parallel to the sensor's optical axis, it is obvious that the extracted surface normal obtained from the first one is more reliable than that of the second one. A quantitative analysis is needed and will certainly further improve the accuracy of the algorithm.

7.2.2 Object localization in tele-autonomous systems

The thesis has discussed two situations in which the object localization technique can help the completion of tele-manipulation tasks. Because of the lack of adequate sensing devices, only a small part of the testing is performed on real data. Simulation testing, however, has shown that the technique performs well. The sensor the author used is indeed one with a very high resolution, but its line length is too short to collect enough line range data in just one frame of measurement. We have found from the literature that some recently developed range sensors have characteristics which are very close to our requirements, but we have not had time to acquire and use such sensors.

The thesis has explored only a small number of tele-autonomous system problems. Many problems in areas such as part design, part grasping and multi-sensor feedback have to be further investigated and resolved, and their solutions have to be applied to tele-autonomous systems together with the object localization technique before an efficient and workable strategy can be used to implement tele-manipulation tasks.

APPENDICES

APPENDIX A

THE FORMULAS USED TO FIND THE PLANAR EQUATIONS

Suppose the equation of the planar surface to be determined is $ax + by + z = r$. We need to find the a, b and r from a set of k readings (x_i, y_i, z_i), where $i = 1, \cdots, k$ and $k > 3$. We have the following equation:

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ & \vdots & \\ x_k & y_k & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ -r \end{bmatrix} = \begin{bmatrix} -z_1 \\ -z_2 \\ \vdots \\ -z_k \end{bmatrix} \tag{A.1}$$

Because the range data are obtained by a line range sensor (or a range image), it is reasonable to assume that x_i and y_i are independent variables with no error and z_i is the dependent variable which does contain error. Then for each pair of (x_i, y_i), the corresponding z value should be

$$z = ax_i + by_i - r \tag{A.2}$$

However, the measured z value is z_i. The error in the ith measurement is then

$$e_i = (ax_i + by_i - r) - z_i \tag{A.3}$$

By using the least squares method, the parameters a, b, and d are to be chosen so that

the following sum of squares is to be minimized:

$$E^2 = \sum_{i=1}^{k} e_i^2 = \sum_{i=1}^{2} (ax_i + by_i - r - z_i)^2 \tag{A.4}$$

Taking derivatives of E^2 with respect to a, b and d and set each of the derivatives equal to zero, we have the following three equations:

$$\begin{aligned}
\frac{\partial E^2}{\partial a} &= 2\sum_{i=1}^{k} (ax_i + by_i - r - z_i)x_i = 0 \\
\frac{\partial E^2}{\partial b} &= 2\sum_{i=1}^{k} (ax_i + by_i - r - z_i)y_i = 0 \\
\frac{\partial E^2}{\partial r} &= -2\sum_{i=1}^{k} (ax_i + by_i - r - z_i) = 0
\end{aligned}$$

In matrix form, they can be expressed as

$$\begin{bmatrix} \sum x_i^2 & \sum x_i y_i & \sum x_i \\ \sum x_i y_i & \sum y_i^2 & \sum y_i \\ \sum x_i & \sum y_i & k \end{bmatrix} \begin{bmatrix} a \\ b \\ -d \end{bmatrix} = \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \\ \sum z_i \end{bmatrix} \tag{A.5}$$

where the summation is from $i = 1$ to k, or can be expressed as

$$M \begin{bmatrix} a \\ b \\ -r \end{bmatrix} = \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \\ \sum z_i \end{bmatrix} \tag{A.6}$$

for simplicity.

If the matrix M has an inverse, say W, the solution can be expressed as

$$\begin{bmatrix} a \\ b \\ -r \end{bmatrix} = W \begin{bmatrix} \sum x_i z_i \\ \sum y_i z_i \\ \sum z_i \end{bmatrix} \tag{A.7}$$

If the surface normal is represented by $\mathbf{n} = (n_x, n_y, n_z)$, then

$$n_x = a/s \tag{A.8}$$

$$n_y = b/s \tag{A.9}$$

$$n_z = 1/s \tag{A.10}$$

$$d = r/s \tag{A.11}$$

where $s = \sqrt{a^2 + b^2 + 1}$.

The plane equation, when expressed by surface normal and distance parameters, will be in the following equivalent form:

$$n_x x + n_y y + n_z z = d \tag{A.12}$$

or

$$\mathbf{n} \cdot \mathbf{x} = d \tag{A.13}$$

APPENDIX B

CIRCLE PARAMETER DETERMINATION USING LEAST SQUARES OPTIMIZATION

Suppose that the 3-D range data are projected on the plane which is perpendicular to the axis of the cylinder and are expressed as $S = \{(x_i, y_i) | i = 1, \cdots, k, k > 2\}$. Assume also that the desired circle is located at $c = (x_0, y_0)$ with radius r, The area of the circle will be πr^2 and the measured area from the measured point (x_i, y_i) is $\pi((x_i - x_0)^2 + (y_i - y_0)^2)$. The error between the two areas is

$$e_i = \pi((x_i - x_0)^2 + (y_i - y_0)^2) - \pi r^2$$

and the error function to be minimized is

$$E = \sum_{i=1}^{k} e_i^2 \tag{B.1}$$

Differentiating Eq.(B.1) with respect to r, x_0 and y_0 and setting the derivatives equal to zero results in

$$\frac{\partial E}{\partial r} = \sum_{i=1}^{k} [((x_i - x_0)^2 + (y_i - y_0)^2) - r^2] r = 0 \tag{B.2}$$

$$\frac{\partial E}{\partial x_0} = \sum_{i=1}^{k} [((x_i - x_0)^2 + (y_i - y_0)^2) - r^2](x_i - x_0) = 0 \tag{B.3}$$

and

$$\frac{\partial E}{\partial y_0} = \sum_{i=1}^{k} [((x_i - x_0)^2 + (y_i - y_0)^2) - r^2](y_i - y_0) = 0 \tag{B.4}$$

respectively. Eq.(B.2) is equivalent to

$$\sum_{i=1}^{k}((x_i - x_0)^2 + (y_i - y_0)^2) = kr^2 \tag{B.5}$$

Replacing Eq.(B.5) into Eq.(B.3), we have

$$\sum_{i=1}^{k}[((x_i - x_0)^2 + (y_i - y_0)^2) - r^2]x_i = 0 \tag{B.6}$$

Therefore,

$$\sum_{i=1}^{k}((x_i - x_0)^2 + (y_i - y_0)^2)x_i = (\sum_{i=1}^{k} x_i)r^2 \tag{B.7}$$

For the same reason, replacing Eq.(B.5) into Eq.(B.4) yields

$$\sum_{i=1}^{k}((x_i - x_0)^2 + (y_i - y_0)^2)y_i = (\sum_{i=1}^{k} y_i)r^2 \tag{B.8}$$

To eliminate the r^2 term from Eq.(B.7), we multiply both side of Eq. (B.5) by $\sum_{i=1}^{k} x_i$ and subtract both side of Eq.(B.7), which is multiplied by k:

$$\sum_{i=1}^{k}((x_i - x_0)^2 + (y_i - y_0)^2)(\sum_{i=1}^{k} x_i) - k\sum_{i=1}^{k}((x_i - x_0)^2 + (y_i - y_0)^2)x_i = 0 \tag{B.9}$$

Similarly, to eliminate the r^2 term from Eq.(B.8), we multiply both side of Eq. (B.5) by $\sum_{i=1}^{k} y_i$ and subtract both side of Eq.(B.8) which is multiplied by k:

$$\sum_{i=1}^{k}((x_i - x_0)^2 + (y_i - y_0)^2)(\sum_{i=1}^{k} y_i) - k\sum_{i=1}^{k}((x_i - x_0)^2 + (y_i - y_0)^2)y_i = 0 \tag{B.10}$$

Next we use the following simplified notation:

$$\sum x = \sum_{i=1}^{k} x_i \quad \sum z = \sum_{i=1}^{k} y_i \quad \sum xz = \sum_{i=1}^{k} x_i y_i$$

$$\sum x^2 = \sum_{i=1}^{k} x_i^2 \quad \sum z^2 = \sum_{i=1}^{k} z_i^2 \quad \sum x^2 z = \sum_{i=1}^{k} x_i^2 y_i$$

$$\sum x^3 = \sum_{i=1}^{k} x_i^3 \quad \sum z^3 = \sum_{i=1}^{k} z_i^3 \quad \sum xz^2 = \sum_{i=1}^{k} x_i z_i^2$$

Expanding Eqs.((B.9), (B.10), we will then have

$$\begin{cases} 2(\sum^2 x - k\sum x^2)x_0 + 2(\sum x \sum z - k\sum xz)y_0 = (\sum x)(\sum x^2 + \sum z^2) - k(\sum x^3 + \sum xz^2) \\ 2(\sum x \sum z - k\sum xz)x_0 + 2(\sum^2 z - k\sum z^2)y_0 = (\sum z)(\sum x^2 + \sum z^2) - k(\sum z^3 + \sum zx^2) \end{cases} \tag{B.11}$$

The x_0 and y_0 can be solved from Eq. (B.11) very easily.

Let

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} 2(\sum^2 x - k\sum x^2) & 2(\sum x \sum z - k\sum xz) \\ 2(\sum x \sum z - k\sum xz) & 2(\sum^2 z - k\sum z^2) \end{pmatrix}$$

$$\boldsymbol{B} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} (\sum x)(\sum x^2 + \sum z^2) - k(\sum x^3 + \sum xz^2) \\ (\sum z)(\sum x^2 + \sum z^2) - k(\sum z^3 + \sum zx^2) \end{pmatrix}$$

Then we have

$$\boldsymbol{A} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \boldsymbol{B} \tag{B.12}$$

and the solution for x_0 and y_0 is

$$x_0 = \frac{b_1 a_{22} - b_2 a_{12}}{|\boldsymbol{A}|} \tag{B.13}$$

$$y_0 = \frac{b_2 a_{11} - b_1 a_{21}}{|\boldsymbol{A}|} \tag{B.14}$$

APPENDIX C

CONIC AXIS PARAMETER EXTRACTION

C.1 The Intersection Point of the Axis of a Cone and a Plane

As we have discussed in Chapter 4, a conic surface can be specified by a triple $(\mathbf{v}, \mathbf{n}, \lambda)$, where $\mathbf{v}$ is the *vertex* of the cone, $\mathbf{n}$ is a unit vector which defines the direction of the *axis* of the cone, and $\lambda = cos\theta$ where θ is the *half angle* of the cone. A point $\mathbf{x} = (x, y, z)$ is on the conic surface if and only if the vector $\mathbf{v} \cdot \mathbf{x}$ makes the angle θ with $\mathbf{n}$ or $-\mathbf{n}$. We have the following general form of a conic equation:

$$|(\mathbf{x} - \mathbf{v}) \cdot \mathbf{n}| = \lambda |\mathbf{x} - \mathbf{v}| \cdot |\mathbf{n}|$$

The above can be written as

$$[n_1(x - v_1) + n_2(y - v_2) + n_3(z - v_3)]^2 = \lambda^2[(x - v_1)^2 + (y - v_2)^2 + (z - v_3)^2] \quad \text{(C.1)}$$

The line equation which is coincident with the axis of the cone can be expressed as

$$\begin{cases} \frac{x - v_1}{y - v2} = \frac{n_1}{n_2} \\ \frac{z - v_3}{y - v2} = \frac{n_3}{n_2} \end{cases} \quad \text{(C.2)}$$

Without losing generality, the plane $y = 0$ is selected as the intersection plane in order to simplify the discussion. The intersection of the cone with the plane will produce the

following quadratic equation:

$$[n_1(x - v_1) - n_2v_2 + n_3(z - v_3)]^2 = \lambda^2[(x - v_1)^2 + v_2^2 + (z - v_3)^2] \tag{C.3}$$

which has a general form

$$Ax^2 + Bxz + Cz^2 + Dx + Ez + F = 0 \tag{C.4}$$

The coordinates of the intersection point of the axis line of the cone with the plane $y = 0$ are

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} v_1 - v_2\frac{n_1}{n_2} \\ 0 \\ v_3 - v_2\frac{n_3}{n_2} \end{pmatrix} \tag{C.5}$$

From Eq.(C.3) and Eq.(C.5) we can make the following observations:

- The changes of v_1 and v_3 merely translate the conic section on the plane.
- The effect of changing n_1 and n_3 on the conic section will be a rotation in the xz-plane.
- As we change v_1, v_3, n_1 and n_3, the position of the intersection point of the cone with the plane will also change.

In summary, the changes of parameters v_1, v_3, n_1 and n_3 will only affect the position and orientation of the intersection curve and will have no affect on the shape of the quadratic curve (conic section) intersected by the plane.

Now let us set $v_3 = 0$ and $n_3 = 0$. This corresponds to assuming that the vertex of the cone is in the yz-plane and the axis of the cone is in the plane also. When these values are substituted into Eq.(C.3), the equation of the conic becomes

$$\begin{aligned}(\lambda^2 - n_1^2)x^2 - 2 \quad & [v_1(\lambda^2 - n_1^2) - n_1n_2v_2]x + \lambda^2z^2 \\ & = 2n_1n_2v_1v_2 - v_1^2(\lambda^2 - n_1^2) - v_2^2(\lambda^2 - n_2^2).\end{aligned} \tag{C.6}$$

and the position of the intersection point of the axis with the plane will then become

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} v_1 - v_2\frac{n_1}{n_2} \\ 0 \\ 0 \end{pmatrix} \tag{C.7}$$

From this equation we can conclude that if the quadratic intersection curve is an ellipse, the intersection point of the axis of a conic surface with the plane is always on the principal axis of the ellipse.

We can further simplify Eq.(C.6) by properly translating the conic section along the z axis. To do that, we can set $\mu^2 = \lambda^2 - n_1^2$ and $v_1 = n_1 n_2 v_2/\mu^2$. The resulting conic equation then reduces to

$$\mu^2 x^2 + \lambda^2 z^2 = k^2 \tag{C.8}$$

in the case $0 < \lambda^2 - n_1^2 < \lambda^2$.

That is, the right-hand side of the Eq.(C.6) reduces to a positive constant k^2. To see that this is correct we only need to observe that the left-hand side of Eq.(C.6) reduces to the left-hand side of Eq.(C.8), which is positive for any (x, z). Hence the right-hand side of Eq.(C.6) must be a positive constant.

After simple arithmetic manipulation, the expression for x in Eq.(C.7) can be simplified as

$$x = v_2 n_1 (1 - \lambda^2) \tag{C.9}$$

which will not be equal to zero because the half angle θ of a cone can never be zero. Eq.(C.9) tells us that if the curve of the intersection between a plane and a cone is an ellipse, the intersection point of the axis of a cone and a plane can never be coincident with the center of the ellipse. Eq.(C.9) also gives us the amount of translation from the center of the ellipse along the principal axis of the ellipse.

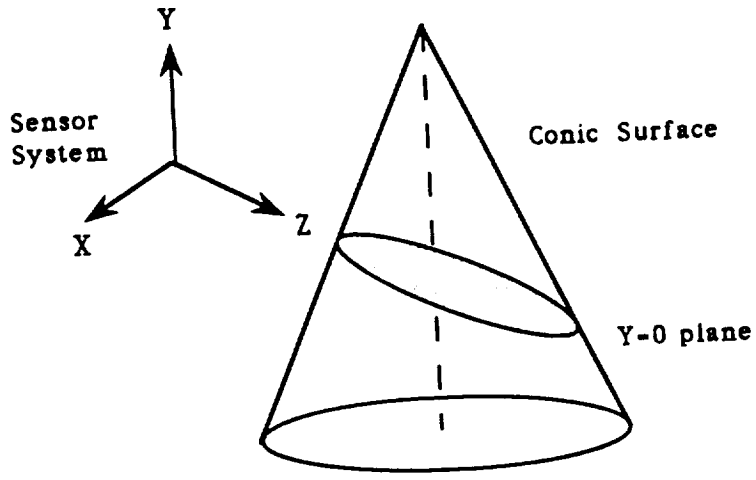


Figure C.1. Cone Axis Parameter Extraction from a Line Sensing System

C.2 The Cone Axis Parameter Extraction

The discussion in last section will be very helpful in solving the axis parameter extraction problem. Suppose a cone is sensed by a laser line range sensor. The sensor coordinate system is shown in Fig. C.1. The measurements are made along the X axis of the coordinate system. The depth data are expressed as Z values. That is, the measurements are taken in the $y = 0$ plane. After the measurements, ellipse parameters can be extracted and the ellipse can be expressed as

$$Ax^2 + Bxz + Cz^2 + Dx + Ez + F = 0 \tag{C.10}$$

On the other hand, the intersection of a cone with the $y = 0$ plane is given in Eq.(C.3). Comparing the corresponding terms of the above two equations, we have a set of six

equations with six unknown:

$$\begin{cases} \lambda^2 - n_1^2 = A \\ -2n_1n_3 = B \\ \lambda^2 - n_3^2 = C \\ 2n_1(n_2v_2 + n_3v_3) - 2v_1A = D \\ 2n_3(n_2v_2 + n_1v_1) - 2v_3C = E \\ \lambda^2(v_1^2 + v_2^2 + v_3^2) - (n_1v_1 + n_2v_2 + n_3v_3)^2 = F \end{cases} \tag{C.11}$$

The **n** can be solved from equations for A, B and C. If $B \neq 0$, both n_1 and n_3 must be non-zero. In this case, we can derive n_1 from the equation for A:

$$n_1 = \pm\sqrt{(A - \lambda^2)} \tag{C.12}$$

and then derive n_3 from the equation for B:

$$n_3 = -\frac{B}{2n_1} \tag{C.13}$$

n_2 can be derived from n_1 and n_3:

$$n_2 = \pm\sqrt{(1 - n_1^2 - n_3^2)} \tag{C.14}$$

Once the cone's axis direction is found, the location of the cone's vertex can be found too from the remained equations for D, E and F. From equations of D and E, we have

$$2n_1n_3v_3 - 2Av_1 = D - 2n_1n_2v_2 \tag{C.15}$$

and

$$-2Cv_3 - 2n_1n_3v_1 = E - 2n_2n_3v_2 \tag{C.16}$$

Therefore

$$v_3 = \frac{\begin{vmatrix} D - 2n_1n_2v_2 & -2A \\ E - 2n_2n_3v_2 & -2n_1n_3 \end{vmatrix}}{\begin{vmatrix} 2n_1n_3 & -2A \\ -2C & -2n_1n_3 \end{vmatrix}} = f_{v_3}(v_2) = a_3v_2 + b_3 \tag{C.17}$$

$$v_1 = \frac{\begin{vmatrix} 2n_1n_3 & D - 2n_1n_2v_2 \\ -2C & E - 2n_2n_3v_2 \end{vmatrix}}{\begin{vmatrix} 2n_1n_3 & -2A \\ -2C & -2n_1n_3 \end{vmatrix}} = f_{v_1}(v_2) = a_1v_2 + b_1 \tag{C.18}$$

where coefficients a_1, a_3 and b_1, b_3 are now known quantities.

Substituting Eqs.(C.17) and (C.18) into the equation for F, we can solve for v_2 with two solutions. The v_1 and v_3 can then be solved by substituting v_2 back to Eq.(C.17) and (C.18).

From above discussion we know that there are usually eight possible solutions for the location of the cone from a given ellipse equation. But only one is correct. The correct one can be identified by two methods:

- Take more measurements. Each additional measurement can give us one more solution set. We can compare these different solution sets and find the one which is common or close to common among all the solutions.
- The approximate location of the cone is supposely known. We can compare the set of solution with the supposed location of the cone's axis and take the one which is closest to the supposed solution as the correct one.

In fact, in practice, the two methods should be combined to find the correct answer.

APPENDIX D

DATA SMOOTHING METHODS

D.1 Quadric Curve Fitting

The measurement data are a set of discrete points on a coordinate plane, which can be expressed as

$$S = \{(x_i, z_i) | \ \ i = 1, \cdots, k\}$$

where in some cases, x_i's are integers. In order to obtain a smoothed curve from these measured data, we first make observations on the data to see what kind of curve is a best fit. We then try to find the parameters which define that curve to give a better approximation. Usually, a least squares optimization is employed, which minimizes the sum of the squares of the difference between each measured value and the corresponding value from the smoothed curve.

If a quadric curve is a suitable approximation to the set of measurement data, the curve can be expressed as

$$z = a + bx + cx^2 \tag{D.1}$$

We will minimize the following error function

$$E = \sum_{i=1}^{k} (z_i - (a + bx_i + cx_i^2))^2 \tag{D.2}$$

Differentiating Eq.(D.2) with respect to a, b and c, and setting them to zero, we obtain a system of three linear equations with three unknown:

$$\begin{cases} \frac{1}{k}\sum_{i=1}^{k} z_i = a + b(\frac{1}{k}\sum_{i}^{k} x_i) + c(\frac{1}{k}\sum_{i=1}^{k} x_i^2) \\ \frac{1}{k-1}\sum_{i=1}^{k-1} z_i = a + b(\frac{1}{k-1}\sum_{i}^{k-1} x_i) + c(\frac{1}{k-1}\sum_{i=1}^{k-1} x_i^2) \\ \frac{1}{k-2}\sum_{i=1}^{k-2} z_i = a + b(\frac{1}{k-2}\sum_{i}^{k-2} x_i) + c(\frac{1}{k-2}\sum_{i=1}^{k-2} x_i^2) \end{cases} \tag{D.3}$$

A closed form formula can be found for the solution of these equations. The extreme value for z and its corresponding position x can be obtained from the resulting quadric equation.

D.2 Data Smoothing

It may be difficult to use only one curve, whether a quadric or a polynomial of higher order, to fit the entire set of measured data. An alternative method is to use piece-wise smoothing. In this method the smoothed value $\bar{z}_i$ at position x_i is dependent only upon its surrounding z values. If the $\bar{z}_i$ value depends on z values at positions $i-2$, $i-1$, i, $i+1$ and $i+2$, it is called five-point smoothing. If the $\bar{z}_i$ value depends on z values at positions from $i-3$ to $i+3$, it is called seven-point smoothing.

Now, suppose we have the following set of measured data:

x	x_0	$x_1 = x_0 + \Delta x$	$\cdots$	$x_i = x_0 + i\Delta x$	$\cdots$	$x_m = x_0 + m\Delta x$
z	y_0	y_1	$\cdots$	y_i	$\cdots$	y_m

If we let $t = \frac{x-x_i}{\Delta x}$, the above table becomes

t	$-i$	$-(i-1)$	$\cdots$	-1	0	1	$\cdots$	$(m-i-1)$	$m-i$
y_{i+t}	y_0	y_1	$\cdots$	y_{i-1}	y_i	y_{i+1}	$\cdots$	y_{m-1}	y_m

If a quadric curve is used to fit each section of piece-wise data, the smoothed z value at position $i+t$ is

$$\bar{z}_{i+t} = a + bt + ct^2 \tag{D.4}$$

where coefficients a, b, c should make

$$\sum_t ((a + bt + ct^2) - z_{i+t})^2 \quad \text{minimum} \tag{D.5}$$

The sum is over $(2n-1)$ of z values with the t's being closest to i within the interval of $[0, m]$. If $n = 2$, it is five-point smoothing; if $n = 3$, it is seven-point smoothing.

When $i - n \geq 0$ and $i + n \leq m$ are satisfied at the same time, the a, b and c can be solved using the following system of equations:

$$\begin{cases} (2n+1)a + 2(\frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6})c = \sum_{t=-n}^{n} z_{i+t} \\ 2(\frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6})b = \sum_{t=-n}^{n} t z_{i+t} \\ 2(\frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6})a + 2(\frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30})c = \sum_{t=-n} n t^2 z_{i+t} \end{cases} \tag{D.6}$$

For $n = 2$ we have the following solutions:

$$\begin{cases} \bar{z}_i = \frac{1}{35}[-3(z_{i-2} + z_{i+2}) + 12(z_{i-1} + z_{i+1}) + 17 z_i] \quad (i = 2, 3, \cdots, m-2) \\ \bar{z}_0 = \frac{1}{35}(31 z_0 + 9 z_1 - 3 z_2 - 5 z_3 + 3 z_4) \\ \bar{z}_1 = \frac{1}{35}(9 z_0 + 13 z_1 + 12 z_2 + 6 z_3 - 5 z_4) \\ \bar{z}_{m-1} = \frac{1}{35}(-5 z_{m-4} + 6 z_{m-3} + 12 z_{m-2} + 13 z_{m-1} + 9 z_m) \\ \bar{z}_{m-2} = \frac{1}{35}(3 z_{m-4} - 5 z_{m-3} - 3 z_{m-2} + 9 z_{m-1} + 31 z_m) \end{cases} \tag{D.7}$$

If $n = 3$, the value $\bar{z}_i$ at position i will depend on the values of the seven positions nearest to i. The formula can be obtained from the same system of equations in Eq.(D.6).

APPENDIX E

ROTATION PARAMETER DETERMINATION

E.1 About Eq. 4.48

The derivation of Eq.(4.48) is based on the fact that any rotation will preserve the angle between the transformed vector and the rotation axis (see Fig. E.1.(a)). That is, if vector $\tilde{\mathbf{n}}_i$ is rotated into vector $\mathbf{n}_i$ and $\mathbf{r}$ is the rotation axis, we have

$$\mathbf{r} \cdot \mathbf{n}_i = \mathbf{r} \cdot \tilde{\mathbf{n}}_i \tag{E.1}$$

or, equivalently,

$$\mathbf{r} \cdot (\mathbf{n}_i - \tilde{\mathbf{n}}_i) = 0 \tag{E.2}$$

That is, $\mathbf{r}$ is perpendicular to $(\mathbf{n}_i - \tilde{\mathbf{n}}_i)$ both for $i = 1$ and $i = 2$. Hence $\mathbf{r}$ is given by

$$\mathbf{r} = \frac{(\mathbf{n}_1 - \tilde{\mathbf{n}}_1) \times (\mathbf{n}_2 - \tilde{\mathbf{n}}_2)}{\|(\mathbf{n}_1 - \tilde{\mathbf{n}}_1) \times (\mathbf{n}_2 - \tilde{\mathbf{n}}_2)\|} \tag{E.3}$$

to within an ambiguity of degree of π.

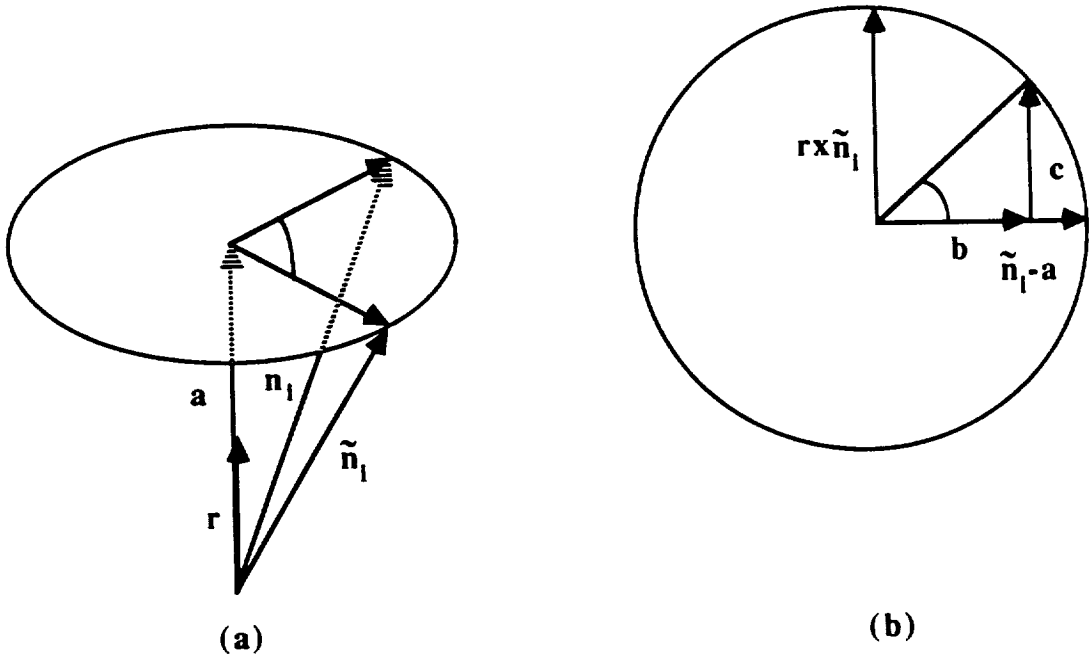


Figure E.1. (a). vector $\tilde{\mathbf{n}}_i$ rotates with respect to vector $\mathbf{r}$ into vector $\mathbf{n}_i$;
(b). the vectors projected on the circle plane.

E.2 About the angle of rotation

Consider the unit vector $\tilde{\mathbf{n}}_i$ rotated by an angle θ into the vector $\mathbf{n}_i$; the rotation takes place along the arc of a circle in the plane perpendicular to $\mathbf{r}$ (see Fig. E.1). Let $\mathbf{a}$ be the projection vector from $\tilde{\mathbf{n}}_i$ onto $\mathbf{r}$, We have

$$\mathbf{a} = (\tilde{\mathbf{n}}_i \cdot \mathbf{r})\mathbf{r} \tag{E.4}$$

with $\mathbf{b}$ and $\mathbf{c}$ as vectors in the circle plane designed in such a way that

$$\mathbf{n}_i = \mathbf{a} + \mathbf{b} + \mathbf{c} \tag{E.5}$$

If α is the angle between $\mathbf{r}$ and $\tilde{\mathbf{n}}_i$ then the radius h of the circle is given by

$$h = \sin\alpha|\tilde{\mathbf{n}}_i| = \frac{|\mathbf{r}\times\tilde{\mathbf{n}}_i|}{|\mathbf{r}|} = |\mathbf{r}\times\tilde{\mathbf{n}}_i|, \quad (|\mathbf{r}| = 1). \tag{E.6}$$

Furthermore

$$\mathbf{b} = (\tilde{\mathbf{n}}_i - \mathbf{r})\cos\theta \quad and \quad \mathbf{c} = (\mathbf{r} \times \tilde{\mathbf{n}}_i)\sin\theta \tag{E.7}$$

We obtain the following

$$\begin{aligned} \mathbf{n}_i &= \mathbf{a} + \cos\theta(\tilde{\mathbf{n}}_i - \mathbf{a}) + \mathbf{c} \\ &= \cos\theta\tilde{\mathbf{n}}_i + (1 - \cos\theta)\mathbf{a} + \sin\theta\mathbf{r} \times \tilde{\mathbf{n}}_i \\ &= \cos\theta\tilde{\mathbf{n}}_i + (1 - \cos\theta)(\mathbf{r} \cdot \tilde{\mathbf{n}}_i)\mathbf{r} + \sin\theta\mathbf{r} \times \tilde{\mathbf{n}}_i \end{aligned} \tag{E.8}$$

Now, "$\cdot\tilde{\mathbf{n}}_i$" the two sides of Eq.(E.8), we have

$$\mathbf{n}_i \cdot \tilde{\mathbf{n}}_i = \cos\theta\tilde{\mathbf{n}}_i \cdot \tilde{\mathbf{n}}_i + (1 - \cos\theta)(\mathbf{r} \cdot \tilde{\mathbf{n}}_i)\mathbf{r} \cdot \tilde{\mathbf{n}}_i + \sin\theta(\mathbf{r} \times \tilde{\mathbf{n}}_i) \cdot \tilde{\mathbf{n}}_i \tag{E.9}$$

Eq.(E.9) can be reduced to

$$\begin{aligned} \mathbf{n}_i \cdot \tilde{\mathbf{n}}_i &= \cos\theta + (1 - \cos\theta)(\mathbf{r} \cdot \tilde{\mathbf{n}}_i)\mathbf{r} \cdot \tilde{\mathbf{n}}_i \\ &= \cos\theta + (1 - \cos\theta)(\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2 \end{aligned} \tag{E.10}$$

Hence

$$\cos\theta = \frac{\mathbf{n}_i \cdot \tilde{\mathbf{n}}_i - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2}{1 - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2} \tag{E.11}$$

From $\cos\theta$, we can derive $\sin\theta$. Using the similar procedure, e.g., first "$\cdot\mathbf{n}_i$" the two sides of Eq.(E.8), we have

$$\mathbf{n}_i \cdot \mathbf{n}_i = \cos\theta\tilde{\mathbf{n}}_i \cdot \mathbf{n}_i + (1 - \cos\theta)(\mathbf{r} \cdot \tilde{\mathbf{n}}_i)\mathbf{r} \cdot \mathbf{n}_i + \sin\theta(\mathbf{r} \times \tilde{\mathbf{n}}_i) \cdot \mathbf{n}_i \tag{E.12}$$

Because $\mathbf{r} \cdot \tilde{\mathbf{n}}_i = \mathbf{r} \cdot \mathbf{n}_i$, Eq. (E.12) becomes

$$1 = \cos\theta\tilde{\mathbf{n}}_i \cdot \mathbf{n}_i + (1 - \cos\theta)(\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2 + \sin\theta(\mathbf{r} \times \tilde{\mathbf{n}}_i) \cdot \mathbf{n}_i \tag{E.13}$$

Combining the terms for $\cos\theta$, we obtain the new equation

$$1 - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2 = \cos\theta(\tilde{\mathbf{n}}_i \cdot \mathbf{n}_i - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2) + \sin\theta(\mathbf{r} \times \tilde{\mathbf{n}}_i) \cdot \mathbf{n}_i \tag{E.14}$$

If $\tilde{\mathbf{n}}_i$ is not parallel to $\mathbf{r}$, then

$$1 = \cos\theta \frac{\tilde{\mathbf{n}}_i \cdot \mathbf{n}_i - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2}{1 - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2} + \sin\theta \frac{(\mathbf{r} \times \tilde{\mathbf{n}}_i) \cdot \mathbf{n}_i}{1 - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2} \tag{E.15}$$

Substituting Eq.(4.49) into Eq.(E.15), we obtain

$$1 = \cos\theta^2 + \sin\theta \frac{(\mathbf{r} \times \tilde{\mathbf{n}}_i) \cdot \mathbf{n}_i}{1 - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2} \tag{E.16}$$

That is

$$\sin^2\theta = \sin\theta \frac{(\mathbf{r} \times \tilde{\mathbf{n}}_i) \cdot \mathbf{n}_i}{1 - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2} \tag{E.17}$$

Therefore

$$\sin\theta = \frac{(\mathbf{r} \times \tilde{\mathbf{n}}_i) \cdot \mathbf{n}_i}{1 - (\mathbf{r} \cdot \tilde{\mathbf{n}}_i)^2} \tag{E.18}$$

E.3 About the rotation matrix

If we define a rotation operator $R(\mathbf{r}, \theta)$, Eq.(E.8) can be rewritten in the following form:

$$\mathbf{n}_i = R(\mathbf{r}, \theta)\tilde{\mathbf{n}}_i \tag{E.19}$$

Because $(\mathbf{r} \cdot \tilde{\mathbf{n}}_i)\mathbf{r} = (\mathbf{r}\mathbf{r}^T)\tilde{\mathbf{n}}_i$ and $\mathbf{r} \times \tilde{\mathbf{n}}_i = \boldsymbol{K}(\mathbf{r})\tilde{\mathbf{n}}_i$, the operator will have the following form:

$$R(\mathbf{r}, \theta) = \cos\theta \boldsymbol{I} + (1 - \cos\theta)\mathbf{r}\mathbf{r}^T + \sin\theta \boldsymbol{K}(\mathbf{r}) \tag{E.20}$$

Let $\mathbf{r} = (r_x, r_y, r_z)$, Then

$$R = \cos\theta \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + (1 - \cos\theta) \begin{bmatrix} r_x^2 & r_x r_y & r_x r_z \\ r_y r_x & r_y^2 & r_y r_z \\ r_z r_x & r_z r_y & r_z^2 \end{bmatrix} + \sin\theta \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}$$

APPENDIX F

DUAL QUATERNIONS AND THEIR APPLICATIONS IN REPRESENTING POSITION AND ORIENTATION

F.1 The concept and properties of dual numbers

A dual number $\hat{a} = a + \epsilon b$ can be defined as a combination of two ordered real numbers a and b with a special multiplication rule for ϵ given by $\epsilon^2 = 0$. The two real numbers a and b can be said to belong to the real part and the dual part of the dual number, respectively.

Addition, subtraction, and multiplication of dual numbers are defined by the formulae:

$$\begin{aligned} (a + \epsilon b) + (c + \epsilon d) &= (a + c) + \epsilon(b + d) \\ (a + \epsilon b) - (c + \epsilon d) &= (a - c) + \epsilon(b - d) \\ (a + \epsilon b)(c + \epsilon d) &= ac + \epsilon(ad + bc) \end{aligned} \tag{F.1}$$

Dual numbers were first considered by the famous German geometer E. Study (1862 - 1930) in the beginning of this century [105]. In his study he used the dual number to represent the dual angle which measures the relative position of two skew lines in space. That is, a dual angle was defined as

$$\hat{\theta} = \theta + \epsilon d \tag{F.2}$$

where the d is a distance between two lines in three-dimensional space and the θ is the angle between their directions.

Some important properties of dual numbers are

1. The product of a dual number $\hat{a}$ and its conjugate $\bar{\hat{a}} = a - \epsilon b$ is

$$\hat{a}\bar{\hat{a}} = a^2 \tag{F.3}$$

2. The modulus of a dual number

$$|\hat{a}| = a \tag{F.4}$$

which can be negative.

3. Due to the fact that $\epsilon^2 = 0$, the dual number function has a very simple form of Taylor series expansion:

$$f(a + \epsilon b) = f(a) + \epsilon b f'(a) \tag{F.5}$$

4. For the dual angle $\hat{\theta}$, we have

$$\sin(\hat{\theta}) = \sin(\theta + \epsilon d) = \sin(\theta) + \epsilon d \cos(\theta) \tag{F.6}$$

and

$$\cos(\hat{\theta}) = \cos(\theta + \epsilon d) = cos(\theta) - \epsilon d \sin(\theta) \tag{F.7}$$

In the above, the properties (F.3) and (F.4) can be derived directly from the definition of multiplication of two dual numbers. To show property (F.5), we only need to know that a function of a dual number $f(a + \epsilon b)$, like the usual functions over the field of complex numbers, can be expanded into a formal Taylor series. That is, according to the theorem of Taylor series expansion, if a function $f(\hat{a})$ is analytic within a circle $|\hat{a} - c| < R \ \ (R > 0)$

where c is the center of the circle, then $f(\hat{a})$ can be expended into a Taylor series within that circle

$$f(\hat{a}) = \sum_{n=0}^{\infty} u_n(\hat{a} - c)^n \tag{F.8}$$

where $u_n = \frac{f^{(n)}(c)}{n!}$ and the series is unique. If we expend $f(\hat{a})$ at point a, the Taylor series will have the form

$$f(a + \epsilon b) = f(a) + \epsilon b f'(a) + \epsilon^2 \frac{b^2}{2} f''(a) + \cdots \tag{F.9}$$

Because $\epsilon^2 = 0$, all the terms with the power of ϵ greater than one in the Taylor series will be zero. To get the last property, we simply expand $\sin(\hat{\theta})$ and $\cos(\hat{\theta})$ into its corresponding Taylor series at θ.

The idea of dual quantities can be extended to define dual vectors, dual quaternions, and dual matrices. These dual quantities enable two different quantities to be combined into one in many ways. For example, a dual vector can be defined to form any line in 3-D space. The direction and position of the line can be specified as follows:

$$\hat{\mathbf{n}} = \mathbf{n} + \epsilon \mathbf{p} \times \mathbf{n} \tag{F.10}$$

where $\mathbf{n}$ is a unit direction vector of the line and $\mathbf{p}$ is a position vector of any point on the line. As an another example, a dual quaternion can be defined to represent any transformation between two coordinate frames which has been shown in Eq.(5.7).

F.2 Position vector expressed in terms of the components of the dual quaternion

The derivation of Eq. (5.18) $\quad t = W(r)^T s$

Consider an object coordinate system which is transformed into a new location, where the transformation is done by first translating the coordinate system in the direction of the unit vector $\mathbf{n}$ by a distance d (Fig.F.1 (a)) and then rotating it by an angle θ with respect to a line having a unit vector $\mathbf{n}$ as its direction and passing through a point $\mathbf{p}$ (Fig.F.1). The position vector $\mathbf{p}$ as shown in Fig.F.1 will be transformed into vector $\mathbf{p}'$. The translation vector $\mathbf{t}$ can thus be expressed as

$$\begin{aligned} \mathbf{t} &= \mathbf{p} + d\mathbf{n} - \mathbf{p}' \\ &= \mathbf{p} + d\mathbf{n} - \boldsymbol{R}\mathbf{p} \\ &= (\boldsymbol{I} - \boldsymbol{R})\mathbf{p} + d\mathbf{n} \end{aligned} \tag{F.11}$$

The rotation matrix $\boldsymbol{R}$, when defined by a rotation angle θ and a rotation axis $\mathbf{n}$, has the form (see Eq.(E.20):

$$\boldsymbol{R} = \cos\theta \boldsymbol{I} + (1 - \cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta \boldsymbol{K}(\mathbf{n}) \tag{F.12}$$

where the $\boldsymbol{K}(\mathbf{n})$ is the often mentioned skew-symmetric matrix.

Because $\mathbf{n}\mathbf{n}^T = \boldsymbol{I} + \boldsymbol{K}(\mathbf{n})\boldsymbol{K}(\mathbf{n})$, Eq. (F.12) can be changed into

$$\begin{aligned} \boldsymbol{R} &= \cos\theta \boldsymbol{I} + (1 - \cos\theta)(\boldsymbol{I} + \boldsymbol{K}(\mathbf{n})\boldsymbol{K}(\mathbf{n})) + \sin\theta \boldsymbol{K}(\mathbf{n}) \\ &= \boldsymbol{I} + (1 - \cos\theta)\boldsymbol{K}(\mathbf{n})\boldsymbol{K}(\mathbf{n}) + \sin\theta \boldsymbol{K}(\mathbf{n}) \\ &= \boldsymbol{I} + 2\sin^2(\theta/2)\boldsymbol{K}(\mathbf{n})\boldsymbol{K}(\mathbf{n}) + \sin\theta \boldsymbol{K}(\mathbf{n}) \end{aligned} \tag{F.13}$$

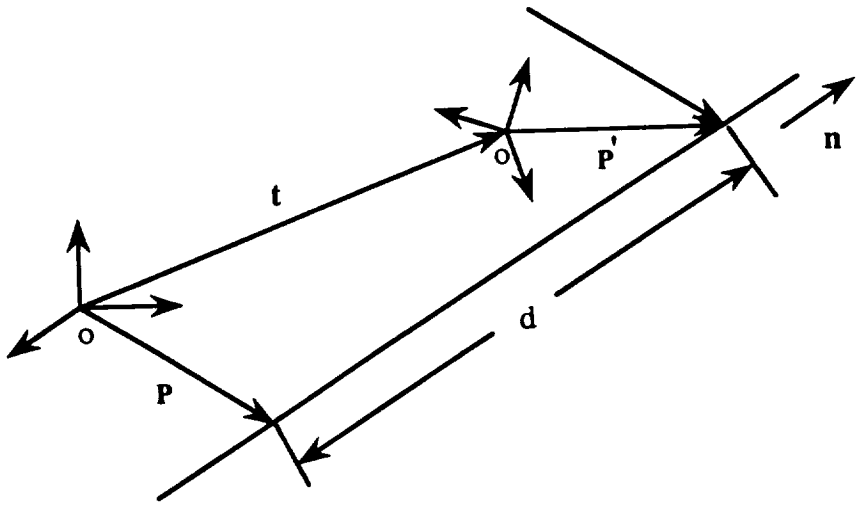


Figure F.1. Illustration of the transformation of an object coordinate system

Replacing Eq. (F.13) into Eq. (F.11), we have

$$\begin{aligned} \mathbf{t} &= (-2\sin^2(\theta/2)\boldsymbol{K}(\mathbf{n})\boldsymbol{K}(\mathbf{n})\mathbf{p} - \sin\theta\boldsymbol{K}(\mathbf{n}))\mathbf{p} + d\mathbf{n} \\ &= 2\sin^2(\theta/2)\mathbf{n}\times(\mathbf{p}\times\mathbf{n}) + \sin\theta(\mathbf{p}\times\mathbf{n}) + d\mathbf{n} \end{aligned} \tag{F.14}$$

On the other hand, from Eq. (5.10) and (5.11) we have

$$\begin{aligned} r_4\mathbf{s} - s_4\mathbf{r} &= (d/2)\mathbf{n} + \sin(\theta/2)\cos(\theta/2)\mathbf{p}\times\mathbf{n} \\ &= (d\mathbf{n} + \sin\theta\mathbf{p}\times\mathbf{n})/2 \end{aligned} \tag{F.15}$$

and

$$\begin{aligned} \mathbf{r}\times\mathbf{s} &= [\sin(\theta/2)\mathbf{n}]\times[d/2\cos(\theta/2)\mathbf{n} + \sin(\theta/2)\mathbf{p}\times\mathbf{n}] \\ &= \sin^2(\theta/2)\mathbf{n}\times(\mathbf{p}\times\mathbf{n}) \end{aligned} \tag{F.16}$$

because $\mathbf{n}\times\mathbf{n} = 0$. Therefore

$$\mathbf{t} = 2(r_4\mathbf{s} - s_4\mathbf{r} + \mathbf{r}\times\mathbf{s}). \tag{F.17}$$

On the other hand,

$$
\begin{aligned}
\boldsymbol{W}(\boldsymbol{r})^T \boldsymbol{s} &= \begin{bmatrix} r_4 \boldsymbol{I} + \boldsymbol{K}(\mathbf{r}) & -\mathbf{r} \\ \mathbf{r}^T & r_4 \end{bmatrix} \begin{bmatrix} \mathbf{s} \\ s_4 \end{bmatrix} \\
&= \begin{bmatrix} (r_4 \boldsymbol{I} + \boldsymbol{K}(\mathbf{r}))\mathbf{s} - s_4 \mathbf{r} \\ \boldsymbol{r}^T \boldsymbol{s} \end{bmatrix} \\
&= \begin{bmatrix} r_4 \mathbf{s} - s_4 \mathbf{r} + \boldsymbol{K}(\mathbf{r})\mathbf{s} \\ 0 \end{bmatrix} = \begin{bmatrix} 1/2\mathbf{t} \\ 0 \end{bmatrix}
\end{aligned}
$$

If we define the translation quaternion $\boldsymbol{t}$ for the translation vector $\mathbf{t}$ as

$$
\boldsymbol{t} = 1/2 \begin{bmatrix} \mathbf{t} \\ 0 \end{bmatrix} \tag{F.18}
$$

then we have the formula

$$
\boldsymbol{t} = \boldsymbol{W}(\boldsymbol{r})^T \boldsymbol{s} \tag{F.19}
$$

F.3 Finding a transformation tuple (n, θ, d, p) from a given dual quaternion

Given a dual quaternion, the rotation submatrix can be derived as previously described, and the rotation axis $\mathbf{n}$ and angle θ can be extracted from the matrix. The translation distance d can be derived from Eq.(5.11)

$$
d = \frac{2 s_4}{\sin(\theta/2)} \tag{F.20}
$$

Eq.(5.11) can also be used to derive $\mathbf{p}$. From Eq.(5.11) we have

$$
\mathbf{s} = d/2 \cos(\theta/2)\mathbf{n} + \sin(\theta/2)(\mathbf{p} \times \mathbf{n}) \tag{F.21}
$$

This equation can be written as

$$
\mathbf{p} \times \mathbf{n} = \frac{\mathbf{s} - d/2 \cos(\theta/2)\mathbf{n}}{\sin(\theta/2)} \tag{F.22}
$$

Because $\mathbf{p} \times \mathbf{n} = \boldsymbol{K}(-\mathbf{n})\mathbf{p}$, we have

$$\boldsymbol{K}(-\mathbf{n})\mathbf{p} = \frac{\mathbf{s} - d/2\cos(\theta/2)\mathbf{n}}{\sin(\theta/2)} \tag{F.23}$$

where the only unknown is $\mathbf{p}$.

The rank of matrix $\boldsymbol{K}(-\mathbf{n})$ is 2 and thus the dimension of null space of the matrix is 1. Eq.(F.23) has the a general solution in the following form:

$$\mathbf{p} = \mathbf{p}_0 + \alpha\mathbf{n} \tag{F.24}$$

where $\mathbf{p}_0$ is any vector in null space.

That is, the solution for $\mathbf{p}$ is not unique. Usually, we select one of the vectors which are perpendicular to $\mathbf{n}$ as the desired solution.

One possible solution is

$$\mathbf{p} = \frac{\alpha}{n_1}\begin{bmatrix} 0 \\ -n_3 \\ n_2 \end{bmatrix} \tag{F.25}$$

where $n_1 \neq 0$.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] G.J.Agin, *" Representation and Description of Curved Objects"*, Stanford AI Memo 173, October 1972.

[2] Antonio Albano, *"Representation of Digitized Contours in Terms of Conic Arcs and Straight-Line Segments"*, Computer Graphics and Image Processing, Vol. 3, 1974, pp.23-33.

[3] Jamas S.Albus, Harry G. Mccain, and Ronald Lumia, *"NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)"*, National Bureau of Standards, Robot System Division, March 13, 1987, pp.12.

[4] J. Andary, D. Hewitt, and S. Hinkal, *" The Flight Telerobotic Servicer Tinman Concept: System Drivers and Task Analysis"*, NASA Conference on Space Telerobotics, Pasadena, CA, Jan. 31-Feb. 2, 1989.

[5] K.S.Arun, T.S.Huaing, and S.D.Blostein, *"Least-Squares Fitting of Two 3-D Point Sets"*, IEEE Trans. on Pattern nalysis and Machine Intelligence, Vol. PAMI-9 No.5, Sept. 1987, pp. 698-700.

[6] D.H. Ballard *"Parameter Networks: Towards a Theory of Low-Level Vision"*, Int'l Joint Conf. on Artificial Intelligence, Vancouver, B.C., August 1981, pp.1068-1078.

[7] J. Barber, R. Desai, R.A.Volz, R. Rubinfeld, B. Schipper, and J. Wolter, *" Automatic Evaluation of Two-Fingered Grips"*, IEEE Journal of Robotics and Automation, August 1987, pp.356-361.

[8] C.M. Bastuscheck, " *Techniques for Real-Time Generation of Range Images*", Proc. of IEEE Int'l Conf. on Robotics and Automation, 1989, pp.262-268.

[9] Yardley Beers, " *Introduction to the theory of error*", Addison-Wesley, Reading, Mass., 1957.

[10] R. Benton and D. Waters, " *Intelligent Task Automation Interim Technical Reports*", AFWAL/MLTC Wright Patterson AFB, Ohio, Reps. 1-7 Oct. 1985.

[11] R. Bernotat and H. Widlok, " *Principles and Applications of Prediction Display*", Institute of Navigation, 19(3):pp.361-370 , July 1966.

[12] Paul Besl and Ramesh Jain, *"An Overview of Three-Dimensional Object Recognition"*, Univ. of Michigan, December 1984, Tech. Rep., RSD-TR-19-84.

[13] Ruud M. Bolle, *"Information Extraction About Complex Three-Dimensional Objects From Visual Data"*, Ph. D. Dissertation, Brown Univ., May 1984, Brown Univ. Tech. Rep., LEMS-6.

[14] Ruud M. Bolle and David B. Cooper, *"Bayesian Recognition of Local 3-D Shape by Approximating, Image Intensity Functions with Quadric Polynomials"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6 No. 4, July 1984, pp.418-429.

[15] Ruud M. Bolle and David B. Cooper, *"On Optimally Combining Pieces of Information, with Application to Estimating 3-D Complex-Object Position from Range Data"*, IBM Tech. Rep. RC 11037, Brown Univ. Tech. Rep. LEMS-8, Feb. 1985.

[16] Ruud M. Bolle and David B. Cooper, *"On Parallel Bayesian Estimation and Recognition for Large Data Sets with Application to Estimating 3-D Complex-Object Po-*

sition from Range Data", Proc. of SPIE Conf. Vision for Robots, Cannes, France, Dec. 1985.

[17] Ruud M. Bolle and David B. Cooper, *"On Optimally Combining Pieces of Information, with Application to Estimating 3-D Complex-Object Position from Range Data"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-8 No.5, Sept. 1986, pp. 619-638.

[18] R. B. Booles, P. Horaud, and M. J. Hannah, *"3DPO: Three Dimensional Part Orientation System"*, Robotics Research: The First Symposium, Cambridge, MA: MIT Press 1984.

[19] P. Horaud, and R. B. Booles, *"3DPO's Strategy for Matching three-Dimensional Objects in Range Data"*, Proc. of IEEE 1984 Int'l Conf. on Robotics, Atlanta, GA, March, 1984, pp. 78-85.

[20] R. C. Bolles, *" Three-Dimensional Locating of Industrial Parts"*, in Robot Sensors Vol. 1, Vision, edited by Aln Pugh, IFS Publications Ltd and Springer Verlag, 1986, pp. 245-253.

[21] Fred L. Boolstein, *" Fitting Conic Sections to Scattered Data"*, Computer Graphics and Image Processing, Vol.9, 1979, pp.56-71.

[22] K.L. Boyer and A.C. Kak, *"Color-Encoded Structured Light for Rapid Active Ranging,"* IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No.1, January, 1986, pp.14-28.

[23] M. A. Bronez, M.M. Clarke, and A. Quinn, *"Requirements Development for a Free-Flying Robot – the "Robin"*, Proc. of IEEE Int'l Conf. on Robotics and Automation, 1986, pp.667-672.

[24] D.J. Burr, *"A fast Filtering Operator for Robot Stereo Vision"*, Proc. of the Seventh Int'l Conf. on Pattern Recognition, July 1984, pp.669-672.

[25] B. Cernusch-Frias, and D.B. Cooper, *" 3-D Space Location and Orientation Parameter Estimation of Lambertian Sphares and Cylinders From a Single 2-D Image By Fitting Lines and Ellipses to Thresholded Data"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No.4, Julu 1984, pp.430-441.

[26] W.K.Cliford, *"Preliminary Sketch of Bi-Quaternions"*, London Math. Soc., 4:381-395, 1873.

[27] *"Public Law 98-371"*, July 18, 1984.

[28] L. Conway, R.A. Volz, and M.W. Walker, *" Tele-Autonomous Systems: Methods and Architectures for Intermingling Autonomous and Telerobotic Technology"*, Proc. of IEEE Int'l Conf. on Robotics and Automation,1987, pp.1121-1130.

[29] L. Conway, R.A. Volz, and M.W. Walker, *" New Concepts in Tele-Autonomous Systems"*, University of Michigan Robotics Research Laboratory Video-Report, February 1987.

[30] L. Conway, R.A. Volz, and M.W. Walker, *" Tele-Autonomous System and Method Employing Time/Position Synchrony/Desynchrony"*, U.S. Patent Application, March 10, 1988.

[31] L. Conway, R.A. Volz, and M.W. Walker, *" Tele-Autonomous Systems: Projecting and Coordinating Intelligent Action at a Distance"*, to appear in IEEE Journal of Robotics and Automation.

[32] CyberOptics Corporation, *"Point/Line Range Sensor Instruction Manual"*, 1988.

[33] P.Dario, M.Bergamasco, *etc.*, *"Tactile Perception in Unstructured Environments: A Case Study fo Rehabilitative Robot Applications"*, Proc. of IEEE Int'l Conf. on Robotics and Automation, April 1987, pp. 2047-2054.

[34] R. Desai, *" On Fine Motion in Mechanical Assembly in Presence of Uncertainty"*, Ph.D. Dissertation, University of Michigan, 1988.

[35] John V. Draper, Joseph N. Herndon and Wendy E. Moore, *"The Implications of Force Reflection for Teleoperation in Space"*, Goddard Conf. on Space Applications of Artificial Intelligence and Robotics, May 1987.

[36] Richard O. Duda, and Peter E. Hart, *"Use of the Hough Transformation to Detect Lines and Curves in Pictures"*, Comm. of ACM, January 1972, pp. 11-15.

[37] Faugeras, O.D. and Herbert, M., *"A 3-D Recognition and Positioning Algorithm Using Geometrical Matching Between Primitive Surfaces"*, Proc. of Eighth Int'l Joint Conf. on Artificial Intell., Aug. 1983, pp.996-1002.

[38] Faugeras, O.D., *"New Steps Toward a Flexible 3-D Vision System for Robotics"*, Proc. 8th Int'l Joint Conf. Pattern Recognition, Montreal, Canada, July-Aug., 1984, pp. 796-805.

[39] Faugeras, O.D. and Herbert, M., *"The Representation, Recognition, and, Locating of 3-D Objects"*, The Int'l Journal of Robotics Research, Fall 1986, pp.27-52.

[40] W.R. Ferell and T.B. Sheridan, *" Supervisory Control of Remote Manipulation"*, IEEE Spectrum, October 1967, pp.81-88.

[41] C.P.Fong, R.S.Dotson, and A.K.Bejczy, *"Distributed Microcomputer Control System for Advanced Teleoperation"*, prof. of IEEE Int'l Conf. on Robotics and Automation 1986, pp. 987-995.

[42] S.H.Friedberg, A.J.Insel, and L.E.Spence, *"Linear Algebra"*, Prentice-Hall, 1979.

[43] Peter C. Gaston and Tomas Lozano-Perez, *"Tactile Recognition and Localization Using Object Models: The Case of Polyhedra on a Plane"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No.3 May 1984.

[44] J.D. Gilchrist and D.E. Soland, *" A Manual Optimal Guidance Scheme Using a Predictive Model"* Tech. Rep. 67-593, 1967, Honeywell Inc.

[45] W.Eric L. Grimson and Tomas Lozano-Perez, *"Model-Based Recognition and Localization from Sparse Range or Tactile Data"*, The Int'l Journal of Robotics Research, Fall 1984 pp.3-35.

[46] Tomas Lozano-Perez, W.Eric L.Grimson, and S.J. White, *"Finding Cylinders in Range Data"*, proce. of IEEE Int'l Conf. on Robotics and Automation, March 1987, pp.202-207.

[47] Steven J. Gordon, *"Automated Assembly Using Feature Localization"*, Ph. D. Dissertation, Dept. of Mechanical Eng., Massachusetts Institute of Technology, Oct. 1986, also Tech. Rep. 932, MIT Artificial Intell. Lab., Cambridge, MA.

[48] Steven J. Gordon and Warren P. Seering, *"Real-Time Part Position Sensing"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI 10, No.3 May 1988.

[49] Kristjan T.Gunnarsson, *"Optimal Part Localization by Database Matching with Sparse and Dense Data"*, Ph.D. dissertation, Dept. of Mechanical Engineering, Carnegie Mellon Univ., May 1987.

[50] Kristjan T. Gunnarsson and Friedrich B. Prinz, *"CAD Model-Based Localization of Parts in Manufacturing"*, Computer, August 1987, pp.66-74.

[51] F.R. Gantmacher, *"Matrix Theory vol. 1"*, Chelsea Publishing Company, New York, NY., 1960.

[52] Wahba Grace, *"A least Squares Estimate of Satellite Attitude"*, problem 65.1, SIAM Review, pp.384-386, July 1966.

[53] Joon Hee Han, *"Range Image Analysis for 3-D Object Recognition"*, Ph. D. dissertation, EECS Dept., University of Michigan, June 1988.

[54] T.Hashimoto, T.B. Sheridan, *Untitled*, Japanese Journal of Ergonomics, 1986.

[55] F. Harrison, *" System Architectures for Telerobotic Research"*, NASA Conference on Space Telerobotics, Pasadena, CA, Jan. 31-Feb. 2, 1989.

[56] J. W. Hill, *"Study of Modeling and Evaluation of Remote Manipulation Tasks with Force Feedback"*, National Aeronautics and Space Administration, Tec. Rep. NASA-CR-158721, July 1979.

[57] K. Homma, and K.S.Fu, *"A Stereo Vision Method Based on Region Recognition"*, Proc. of IEEE Compu. Soc. Workshop on Visual Languages, Dec. 1984, pp. 14-19.

[58] B.K.P.Horn, *"Focusing,"* M.I.T. Project MAC, Ai Memo. 160, May 1968.

[59] M. Idesawa, T. Yatagai, and T. Soma, *"A method for Automatic Measurement of Three-Dimensional Shape by New Type of Moiré Fringe Topography"*, Proc. 3rd Int'l Joint Conf. Artificial Intelligence, November 8-11, 1976, pp.708-712.

[60] S.S.Iyenger and R.L.Kashyap, *" Autonomous Intelligent Machines"*, Computer, June 1989, pp.14-15.

[61] Ramesh Jain, Sandra L. Bartlett, Nancy O'Brien, *"Motion Stereo Using Ego-Motion Complex Logarithmic Mapping"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No.3 May 1987, pp.356-369.

[62] R.A. Jarvis, *"Focus Optimization Criteria for Computer Image Processing"*, Microscope, Vol. 24, 2nd quarter, 1976, pp.163-180.

[63] R. A. Jarvis *"A Perspective on Range Finding Techniques for Computer Vision"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No.2, March 1983, pp. 122-139.

[64] L. J. Jenkins, and J.D. Jurner, *" Optimal Spacecraft Rotational Manervers"*, Elsevier.

[65] E.Kan, P. Landell, S. Oxenberg, and C.Morimoto, *" The JPL Telerobot Operator Control Station: Part II - Software"*, NASA Conference on Space Telerobotics, Pasadena, CA, Jan. 31-Feb. 2, 1989.

[66] Takeo Kanade, *"Three-Dimensional Machine Vision"*, Kluwer Academic Publishers, 1987.

[67] E.Kan, J. Tower, G. Huncka, and G. VanSant, *" The JPL Telerobot Operator Control Station: Part I - Hardware"*, NASA Conference on Space Telerobotics, Pasadena, CA, Jan. 31-Feb. 2, 1989.

[68] J. Keat, *Analysis of least-squares attitude determination routine DOAOP,"* Technical Report, Comp. Sci. Corp., CSC/TM-77/6034, February 1977.

[69] Robert E. Keil, Timothy A. Skunes, and Steven K. Case, *"Achieving High Accuracies with Triangulation-Based Range Sensors"*, Proc. of on SME Conf. Robot 12 and Vision 88, June 1988, pp.5-107 - 5-120.

[70] P. Kelland and P.G. Tait, *"Introduction to Quaternions"*, Macmillan and Co., 1882.

[71] C.R. Kelley, *" Better Control for Complete Manual Systems"*, Control Engineering, August 1967, pp.86-90.

[72] A.S.Kondoleon, *" Application of a Technological-Economic Model of Assembly Techniques to Programmable Assembly Machine Configurations"*, MS Thesis, Dept. of Mechanical Engineering, Mass. Inst. of Technology, 1976.

[73] G. A. Korn, and T. M. Korn, *"Mathematical Handbook for Scientists and Engineers"*, McGraw-Hill, 1961.

[74] D. A. Kugath, *"Experiments Evaluating Compliance and Force Feedback Effect on Manipulator Performance"*, National Aeronautics and Space Administration, Tec. Rep. NASA-CR-128605, August 1972.

[75] U.M. Landau, *" Estimation of a Circular Arc Center and Its Radius"*, Compuer Vision Graphics and Image Processing, Vol. 38, No. 3, 1987, pp. 317-326.

[76] S. Linnainmaa, D. Harwood, and L.S. Davis, *"Pose Determination of a Three-Dimensional Object Using Triangle Pairs"*, Tech. Rep. CAR-TR-95, Center for Automation Research, University of Maryland, 1985.

[77] S. Linnainmaa, D. Harwood, and L.S. Davis, *"Triangle-Based Pose Determination of 3-D Objects"*, Int'l Conf. on Pattern Recognition, 1986, pp. 116-118.

[78] S. Linnainmaa, D. Harwood, and L.S. Davis, *"Pose Determination of a Three-Dimensional Object Using Triangle Pairs"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Volume 10, No. 2, September, 1988, pp. 634-648.

[79] M. Maruyama, and S. Abe, *"Range Sensing by Projecting Multi-Slits with Random Cuts"*, IEEE Int'l Conf. Industrial Application of Machine Vision, Tokyo, Japan, April 1989, pp. 163-168.

[80] M.T.Mason, *"Compliance and Force Control for Computer Controlled Manipulators"*, IEEE Trans. on Syst., Man, Cybern., Vol. SMC-11, June 1981, pp. 418-432.

[81] J. Matijevic, W. Zimmerman, and S. Dolinsky, " *The Architecture of the NASA/OAST Telerobot Testbed*", NASA Conference on Space Telerobotics, Pasadena, CA, Jan. 31-Feb. 2, 1989.

[82] Mitsuo Matsumoto, *"High Accuracy Laser Scanned 3-D Range Sensor"*, IEEE Int'l Conf. Industrial Application of Machine Vision, Tokyo, Japan, April 1989, pp. 157-162.

[83] J. Molino, " *Robotics Development Facility Preliminary Engineering Results*", Flight Telerobotic Servicer In-House Phase B Study, First NASA/Industry Briefing, December 1987.

[84] Melvin D. Montemerlo, *"NASA's Automation and Robotics Technology Development Program"*, Proc. of IEEE Int'l Conf. on Robotics and Automation 1986, pp.977-986.

[85] Tadashi Nagata, Hirotoshi, and Kohichi Ishibashi, " *Detection of an Ellipse by Use of a Recursive Least-Squares Estimator*", Journal of Robotic Systems, 2(2), 1985, pp. 163-177.

[86] Yasuo Nakagawa, and Azriel Rosenfeld, " *A Note on Polygonal and Elliptical Approximation of Mechanical Parts*", Pattern Recognition, Vol. 11, 1979, pp. 133-142.

[87] NASA, " *Shuttle Flight Operations Manual, Payload Development and Retreval Systems*", Vol.16, Flight Operations Directorate, Johnson Space Center, June 1, 1981. April 1987, pp.1282-1291.

[88] J.L. Nevins, and D.E. Whitney, " *Computer Controlled Assembly*", Scientific American, V. 238, No.2, Feb. 1978.

[89] M. Noyes and T.B. Sheridan, " *A Novel Predictor for Telemanipulation through a*

Time Delay", Proc. of the Annual Conf. on Manual Control, NASA Ames Research Center, Moffett Field, CA 1984.

[90] M. Oshima and Y. Shirai, *"Object Recognition Using Three Dimensional Information"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 4, Juy, 1983, pp.353-361.

[91] O. Ozeki, K. Higuchi, and S. Yamamoto, *"Automated Dimension Inspection System for Automotive Plastic Parts with a Laser Probe"*, Proc. of Robot and Vision 12, June 5-9, 1988, Detroit, MI, pp.5-51 – 5-60.

[92] O. Ozeki, and K. Hituchi, *"Recognition of Position and Orientation of Jumbled Parts Using Cross Slit Light 3-D Vision Sensor"*, IEEE Int'l Conf. Industrial Application of Machine Vision, Tokyo, Japan, April 1989, pp. 40-45.

[93] Michael A. Penna, *"A Shape from Shading Analysis for a Single Perspective Image of a Polyhedron"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 11, No. 6, June, 1989, pp.545-554.

[94] Edward Pervin, and Jon A. Webb, *"Quaternions in Computer Vision and Robotics"*, Tech. Rep., Dept. of Computer Science, Carnegie-Mellon University, CMU-CS-82-150, 1982.

[95] M.H.Raibert and J.J.Craig, *"Hybrid Position/ Force Control of Manipulators"*, Trans. ASME, J. DSMC, Vol. 103, June 1981, pp.126-133.

[96] *"Robotic Assessment Test Sets: Level 1, 2, & 3"*, NASA Gorddard, SS-GSFC-0029, 1988.

[97] D. Rosenberg, M.D.Levine and S.W.Zucker, *"Computing relative depth relationships from occlusion cues"*, Proc. of 4th Int'l Joint Conf. Pattern Recognition, Kyoto,

Japen, Nov. 7-10 1978, pp.765-769.

[98] W.S. Rutkowski, R. Benton, and E.W. Kent, *"Model-Driven Determination of Object Pose for a Visually Served Robot"*, Proc. of IEEE Int'l Conf. Robotics and Automation, Raleigh, NC, 1987, pp.1419-1428.

[99] Paul D. Sampson, *"Fitting Conic Sections to "Very Scattered" Data: A Iterative Refinement of the Bookstein Algorithm"*, Computer Graphics and Image Processing, Vol. 18, 1982, pp.97-108.

[100] K. Sato, H. Yamamoto, and S. Inokuchi, *"Tuned Range Finger for High Precision 3D Data"*, Proc. 8th Int'l Conf. on Pattern Recognition, 1986, pp.1168-1171.

[101] K. Sato and S. Inokuchi, *"Range-Imaging System Utilizing Nematic Liquid Crystal Mask"*, Proc. 1st Int'l Conf. on Computer Vision, 1987, pp.657-661.

[102] John L. Schneiter, *"An Objective Tactile Sensing Strategy for Object Recognition and Localization."* Proc. of IEEE Int'l Conf. on Robotics and Automation 1986, pp. 1262-1267.

[103] John L. Schneiter, Ph.D. Thesis, Massachusetts Institute of Technology, Dept. of Mechanical Engineering, 1987.

[104] S.A.Stansfield, *"Visually-Aided Tactile Exploration"*, Proc. of IEEE Proc. of 1987 Int'l Conf. on Robotics and Automation, April 1987, pp. 1487-1491.

[105] E. Study, *"Geometrie der Dynamen"*, Leipzig, Germany, 1903.

[106] L. Shao, and R. A. Volz, *" Methods and Strategies of Object Localization"*, NASA conf. on Space Telerobotics, Pasadina, CA, January 1989.

[107] L. Shao, R.A. Volz, and M.W. Walker, *"3-D Object Location Determination Using Line-Segments Matching"*, IEEE Int'l Conf. Industrial Application of Machine Vision, Tokyo, Japan, April 1989, pp. 145-150.

[108] S. Shekhar, O. Khatib and M. Shimojo, *"Sensor Fusion and Object Localization"*, Proc. of IEEE Int'l Conf. Robotics and Automation, San Francisco, CA, 1986, pp.1623-1628.

[109] Y.C. Shiu, and S. Ahmad, *"Finding the mounting position of a sensor by solving a homogenous transformation equation of the form $AX = XB$"*, Proc. of IEEE Int'l Conf. on Robotics and Automation, 1987, pp.1666-1671.

[110] T.M.Silberberg, D. Harwood, and L.S. Davis, *"Object Recognition Using Oriented Model Points"*, Computer Graphics and Image Processing, Vol. 35, 1986, pp. 47-71.

[111] G. Stockman, S. Kopstein, and S. Benett, *"Matching Images to Models for Registration and Object Detection via Clustering"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-4, No. 3, 1982, pp. 229-241.

[112] G. Stockman and J.C.Esteva, *" Use of Geometrical Constrains to Determine 3-D object Pose"*, Proc. 7th Int'l Conf. on Pattern Recognition, Montreal, Canada, 1984, pp. 742-744.

[113] G. Stockman and J.C.Esteva, *" 3-D Object Pose from Clustering with Multiple Views"*, Pattern Recognition 3, 1985, pp. 279-286.

[114] George Stockman, *"Object Recognition and Localization via Pose Clustering"*, Computer Vision Graphics and Image Processing, Vol. 40, No. 3, 1987, pp.261-387

[115] H. Thorne, F.B. Prinz, and O.K.Kirchner, *"Robotic Inspection by Database Matching"*, Tech. Rep., CMU-RI-TR-85-4, The Robotics Institute, Carnegie Mellon Uni-

versity, March, 1985.

[116] R.Y. Tsai and R. K. Lenz, *"A New Technique for Autonomous and Efficient 3D Robotics Hand/Eye Calibration,"* Proc. of the Fourth Int'l Symposium of Robotics Research, Santa Cruz, CA, August 9-14, 1987.

[117] R.Y. Tsai and R. K. Lenz, *"Real Time Versatile Robotics Hand/Eye Calibration Using 3D Machine Vision,"* Proc. of IEEE Itn'l Conf. on Robotics and Automation, 1988 pp.554-561.

[118] Saburo Tsuji, and Fumio Matsumoto, *"Detection of Ellipses by a Modified Hough Transform"*, IEEE Trans. on Computers, Vol. C-27, No. 8, August 1978, pp. 777-781.

[119] S. Ullman, *"The Interpretation of Visual Motion"*, Cambridge, PA, MIT Press, 1979.

[120] M.M. Walker, J. Dionise *"On the Simulation of Space Based Manipulators with Contact"*, NASA conf. on Space Telerobotics, January 1989.

[121] M.W. Walker, L. Shao and R. Volz *"Optimal Object Localization Using Dual Number Quaternions"*, SPIE Int'l Conf. on Applications of Artificial Intelligence VII, Orlando Fl, March 1989, pp. 603-611.

[122] M.W. Walker, L. Shao and R. Volz *"Estimating 3-D location parameters by using dual number quaternion"*, submitted to Computer Vision, Graphics and Image Processing, June 1989.

[123] Juyang Weng, Thomas S. Huang, and Narendra Ahuja, *Motion and Structure from Two Perspective Views: Algorithms, Error Analysis, and Error Estimation"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Volume 11, No. 5, May, 1989, pp.451-476.

[124] P.M. Will and K.S. Penniongton, *"Grid Coding: A Preprocessing Technique for Robot and Machine Vision"*, Artificial Intelligence, Vol. 2, 1971, pp.319-329.

[125] P. Andrew Witkin, *"Recovering Surface Shape and Orientation from Texture"*, Artificial Intelligence, Vol. 17, August 1981, pp.17-45.

[126] D.E. Whitney, *" Force Feedback Control of Manipulator Fine Motions."* Trans. ASME, J. DSMC, Vol. 99, June 1977, pp. 91-97.

[127] D.E. Whitney, *" Quasi-Static Assembly of Compliantly Supported Rigid Parts"*, Journal of Dynamic Systems, Measurement and Control, Vol. 104, March, 1982.

[128] D.E. Whitney, *" Historical Perspective and State of the Art in Robot Force Control"*, Proc. of IEEE Int'l Conf. on Robotics and Automation, 1985.

[129] C.K.Wu, D.Q.Wang, and R.K. Bajcsy, *"Acquiring 3-D Spatial Data of a Real Object"*, Computer Vision, Graphics and Image Processing, Vol. 28, Oct. 1983, pp.126-133.

[130] R.A. Volz, L. Shao, M.W. Walker, and L.A. Conway, *" Tele-Autonomous Control Involving Contacts"*, Tech. Rep. RSD-TR-08-88, University of Michigan, July 1988.

[131] R.A.Volz, J. Wolter, J. Barber, R. Desai, and A.C. Woo, *" Automatic Determination of Gripping Positions,"* NATO Advanced Research Workshop, September 1986.

[132] Jing Xiao and R.A. Volz, *" A Replanning Approach Towards Uncertainly Handing in Robot Assembly"*, 3rd IEEE Int'l Symposium on Intelligent Control, Aug, 1988.

[133] Jing Xiao and R.A. Volz, *" Design and Motion Constraints for Part-mating Planning in the Presence of Uncertainty"*, Proc. of IEEE Int'l Conf. on Robotics and Automation, 1988.

[134] I.M.Yaglom, *"Complex Numbers in Geometry"*, New York: Academic, 1968.

[135] Qin-Zhong Ye, *" A Preprocessing Method for Hough Transform to Detect Circles"*, Proc. of IEEE Itn'l Conf. on Robotics and Automation, 1986, pp. 651-653.

[136] T. Yoshikawa, *Dynamic Hybrid Position/Force Control of Robot Manipulators - Description of Hand Constraints and Calculation of Joint Driving Force"*, IEEE J. of Robotics and Automation, October 1987, pp. 386-392.