Robotic Measurement and Processing of Arbitrary Oriented Surfaces

by

Wei Tao
B.A., Jilin Institute of Technology, China, 1983

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF APPLIED SCIENCE
in the
Department of Mechanical Engineering

We accept this thesis as conforming
to the required standard

[redacted]

Dr. Yury Stepanenko, Supervisor (Dept. of Mechanical Engineering)

[redacted]

Dr. R. Podhorodeski, Departmental Member (Dept. of Mechanical Engineering)

[redacted]

Dr. J.S. Collins, Outside Member (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. Kin Li, External Examiner (Dept. of Electrical and Computer Engineering)

University of Victoria

Supervisor: Dr. Yury Stepanenko

Abstract

This work addresses the problem of automation of industrial robot operations in an unstructured environment. A novel automatic surface measurement and trajectory planning methodology and software are developed. The methodology includes four steps: (1) object search, (2) measurement of the object surface and collection of data, (3) surface modelling, and (4) the generation of the control program for robot joint actuators which provides the end-effector motion along the required trajectory on the surface with the required force interaction. A computer graphical interface is developed so that the user can generate and modify the task trajectory interactivly. Furthermore, the user can simulate the job task on computer, specify the robot searching path, and control the robot from a remote location.

Examiners:

[redacted]

Dr. Yury Stepanenko, Supervisor (Dept. of Mechanical Engineering)

[redacted]

Dr. R. Podhorodeski, Departmental Member (Dept. of Mechanical Engineering)

[redacted]

Dr. J.S. Collins, Outside Member (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. Kin Li, External Examiner (Dept. of Electrical and Computer Engineering)

Table of Contents

List of Figures

Acknowledgements

I would like to express my appreciation to the many individuals who have made suggestions and aided in the development of this thesis. I would, first of all, like to thank my supervisor Dr. Yury Stepanenko for his constant encouragement, patient advice, and complete moral support throughout the course of this thesis. Dr. Glen Field and Mr. Dave Gawley deserve special thanks for their careful reading of part of the manuscript and their numerous constructive comments, as well as several useful suggestions. Next, I wish to thank Dr. Chunyi Su and Mr. Yong Cao who have aided me in the preparation of this thesis in various ways. Thanks are also due to Mr. Qing Zhang for his encouragement and much help.

Chapter 1

Introduction

1.1 Motivation

Robots have been widely used in industry. In some applications, robots are needed to perform specific tasks on an arbitrary oriented object, such as boat body cleaning, grinding, polishing and curve drawing, etc.. Thus, the location, surface boundary and surface equation of the object need to be known in advance so as to generate force/position trajectories for the required tasks on the object. The basic problems are:

(1). How to measure the object surface by robot without the prior information(i.e. no vision);

(2). How to represent the object surface;

(3). How to visually generate and modify a curve with respect to the surface;

(4). How to maintain the contact force in the desired range while the robot performs the task on the object surface.

1.2 Review of the research

So far, two main kinds of methods have been used for the object identification and surface measurement in robotics. One is to employ computer vision, the other is to use tactile sensing and force control.

A common task in computer vision is to determine the identity, position, and orientation of an unknown object in an image. The unknown object is usually assumed to be identical to one of a set of possible objects that the vision system has been taught to recognize, usually by showing it one or more sample views of each object. The most straightforward way to recognize the objects would be to match models for each possible combination of identity, position, and orientation to the image. Besl and Jain [8] show that such an exhaustive search is hopeless for the case of three-dimensional objects. Because of the complexity of this task, researchers have instead turned to recognizing objects by their features.

Many methods have been proposed to recognize three-dimensional objects with their image. For partially visible objects, most methods operate by selecting local features or combinations of local features, and searching for these features in the image [9],[10], [11]. The recognition time for the above methods increases linearly with the number of possible objects, after a fixed time for preprocessing. This can become a problem as the number of possible object types increases.

Segmentation and model-building are the other important problems in the field of three-dimensional object recognition. The goal of segmentation is to break up a sensed scene into its constituent parts, while model-building aims to organize these parts in a compact and meaningful structure.

Many segmentation methods use two-dimensional grey-level images to extract information from the 3D scene. Research work on stereo vision has led to valuable

results but at a high computational cost [12]. The introduction of true range-finder cameras has brought an interesting solution to the problem of 3D information extraction. This type of camera produces information about the 3D coordinates of the points located on the surface of objects. Jarvis [13] and Poussart [14] have reviewed different range finders. Unfortunately, the calibration process of these cameras is often tedious. The image acquisition is generally slow and important signal processing is necessary to recover the 3D coordinates [15].

The above approaches focus on the identification of the object shape rather than measure the object surface. To track and measure the object surface, it is better to use tactile sensing and force control, which can directly gather the data about the object surface so as to model the surface.

Merlet[2] used force feedback to determine the surface normal and tangent. Blauer[3] and Belanger used an extended Kalman filter to estimate the parameters of the surface (in addition to estimation of the robot state). This is a comprehensive approach to the problem, but it is very computationally demanding and requires significant prior knowledge of the task geometry (e.g. a parametric equation describing the surface). Ish-Shalom[4] presents a more general approach in which the directional information is embodied in desired equations and inequalities, determined from the task, between some given variables in the problem. This relationship is then used to synthesize a specific control law to obtain the desired behavior. The research of Allen [7] is concerned with tactile sensing along with computer vision to recover the shape of an object by using sparse contact data points. Tactile sensing in [7] is directed by prior knowledge of the position of an object. The research of Khatib [6] is concerned with real-time motion and force control of robot systems to accomplish finger sensing and assembly operations. The research of Korzeniowski [5] employs a dual-drive hybrid controller to track along the object's surface in order to gather the surface data. The

results in [5] are based on the assumption that the tracking surface is sufficiently stiff and frictionless.

In this thesis a simple measuring and modelling algorithm for the three dimensional object surfaces is studied. Tactile sensing and force control are employed to enable a robot, with an elastic beam grasped by the gripper, to gather data about the object surface in order to perform a task on the object. The procedure can be divided into three steps. First, since no prior information about the location of the object is assumed, a searching algorithm moves the robot with the beam along a predefined searching path until the object is encountered. Second, a force/position controller is used to move the elastic beam along the object surface, while the touching force and torque are obtained through force feedback. The position and orientation of the gripper are obtained through position feedback. An algorithm is developed to calculate the deformation of the beam with the use of touching force and torque. With the help of the beam deformation, the touching point on the surface can be calculated by the proposed algorithm. With the touching points, the object's surface can be modelled by the modelling algorithm, which consists of planar and curvilinear surface modelling methods. Finally, for the planar surface, the surface boundary which is in the robot workspace is defined by a boundary defining algorithm. Then the force/position trajectories for the required tasks on the object are generated and modified with the path planning program.

In most of the existing work on the recognition of an object with a robot, the research usually focuses on the measuring method. But to enable the robot to fulfill the specific task on the surface, such as cleaning the boat body, drawing curves or polishing etc., it is essential and practical to develop the programming interface so that the operator can generate and modify the task trajectory, then simulate it on the computer to verify the task feasibility. In this thesis, such a graphics user interface

is developed.

For the tasks which involve extensive contact with the environment, it is vitally important to control the forces of interaction between the manipulator and the environment. A force/position control algorithm, therefore, is desired. The force/position control algorithm used in this thesis is the impedance control scheme which was proposed by Dr.Yury Stepanenko and Dr. Glen Field in [1]. An algorithm for the reference trajectory generation is presented to let the robot fulfill the specific task on the surface with the impedance controller.

1.3 The thesis outline

The thesis consists of seven chapters. A novel algorithm for measuring an arbitrary oriented object surface by a robot is presented in chapter 2. It is based on tactile sensing and force control. An elastic stick held by the end-effector of the robot is used to touch the surface of the object in order to gather the surface data. The deformation of the elastic beam is calculated by an algorithm based on the use of touching forces and torques. Combining with the deformation with the feed back based knowledge of the gripper pose allows the orientation or the equation of the object surface to be found.

The third chapter focuses on the discussion of how to define the path to let the robot detect an oriented object which has a planar surface, how to measure the boundary of the object surface in the robot workspace, how to graphically generate planar curves, such as circle, oval, rectangle, cubic curves and words, and map them into the robot joint space so that the robot can draw them on the object surface.

The Beta2-Spline method is employed to model the curvilinear surface in chapter 4. A simple method is given to let the robot gather the points on the surface for the

Beta2-Spline method. Because the curvilinear surface is approximated by the numerical method, it is hard to get the analytical equation of the surface curve. Chapter 4 gives the algorithm to generate the curves on the curvilinear surface according to the numerical points in order to process the surface along the curves.

In chapter 5, an algorithm is presented to generate the force/position reference trajectory for using the impedance control in the robot tracking task on the object surface. The impedance control scheme [1] requires neither a dynamic model or torque controlled actuators. This special advantage allows implementation of the method on existing industrial robot designs.

The sixth chapter describes the apparatuses which are used in the experiments for the presenting of algorithms. A novel software package with graphics animation is specifically developed for the implementation of the above algorithms on the REIS industry robot. The package is developed with SunPhigs in C. We call this package as "Graphics Animation & Path Creation Interface (**GAPCI**)".

Chapter 7 concludes the thesis with a short summary and recommendations for future work.

1.4 Thesis contribution

The contribution of the thesis can be summarized as follows:

(1). A new algorithm for measuring and modelling a three dimensional arbitrary oriented surface by the robot is presented.

(2). A novel graphics software package has been developed and implemented for industrial robots. The package provides the automatic performance of the following operations for the robots:

- automatic search, measurement and identification of objects in the working area.

- displaying the object surface, visually generating and modifying (rotating, shifting, shrinking and enlarging) the curves drawn by the robot on the surface.

- automatic programming of the robot actuators' motion, which provides a desired end-effector path along the object surface.

- providing an animation of the motion of industrial robots using three dimensional computer graphics. This package can interpret the user's commands and display various motions of the robot.

- generating the nominal control input for an controller, which provides the desired force control along the processing trajectory. In particular, this software allows the automation of such important industrial operations as processing (grinding, polishing, cleaning, etc.) of curvilinear surfaces with required and controlled force interaction between the processing tool and the surface.

Chapter 2

Measurement of a surface by sensing the contact force

2.1 Introduction

In this chapter, an algorithm is presented to detect the points on an arbitrarily three dimensional object surface. To reduce the touching force and avoid damaging the object, a soft elastic beam is held by a robot end-effector to touch the object surface. The robot will stop moving when the magnitude of the touching force exceeds a specified value. The applied forces and torques are sensed by a force sensor mounted between the robot's gripper and it's mounting flange. The touching forces and torques are used to calculate the deformation of the beam. Thus, the touching point can be determined by using the deformation. To ensure the beam deformation occurs in a plane, called the *deformation plane*, the beam should be a cylinder.

2.2 Study and solution of the two dimension problem

Let us consider the deformation of the elastic beam in the coordinate plane, illustrated in Figure 2.1. The touching force $F(f_z, f_x)$ and torque m_y are read by the force sensor fixed that the gripper is mounted upon.

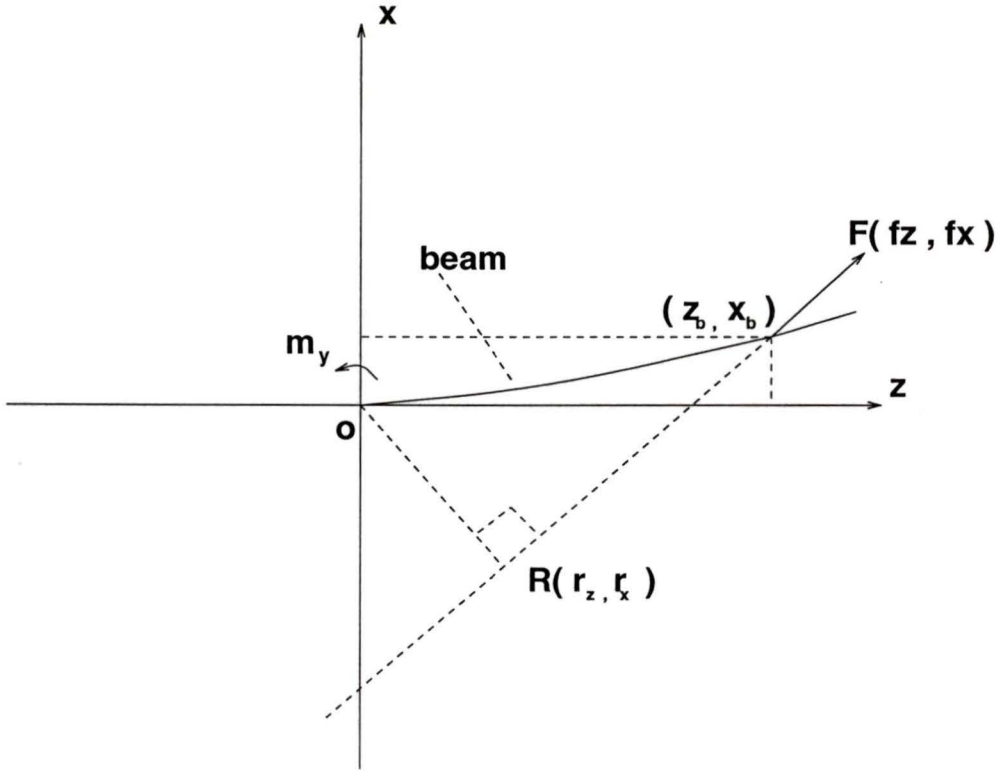

Figure 2.1: 2-D problem

If $R(r_z, r_x)$ is the vector of the moment arm, the following relationship is true:

$$m_y = r_z f_x - r_x f_z \tag{2.1}$$

$$R \cdot F = r_z f_z + r_x f_x = 0 \tag{2.2}$$

On the other hand, if we take $F(f_z, f_x)$ as the direction numbers, we can get the line equation which passes through **touching point** (z_b, x_b) and $R(r_z, r_x)$. We also can get the deformation curve of the elastic beam according to the Euler beam equation:

$$\frac{z - r_z}{f_z} = \frac{x - r_x}{f_x} \tag{2.3}$$

$$x = \frac{f_x}{6EI}(3z_b - z)z^2 \tag{2.4}$$

If we force $z = z_b$ and $x = x_b$, the solutions of the 2-D problem can be derived from (2.1), (2.2), (2.3) and (2.4):

$$x_b = \frac{f_x z_b^3}{3EI} \tag{2.5}$$

$$\begin{aligned} z_b = &\sqrt[3]{-\frac{3EIm_y}{2f_x f_z} + \sqrt{(\frac{3EIm_y}{2f_x f_z})^2 - (\frac{EI}{f_z})^3}} \\ &+\sqrt[3]{-\frac{3EIm_y}{2f_x f_z} - \sqrt{(\frac{3EIm_y}{2f_x f_z})^2 - (\frac{EI}{f_z})^3}} \end{aligned} \tag{2.6}$$

Here, z_b and x_b are relative to the reference coordinate frame which is fixed on the gripper. They can be transformed into physical coordinate frame $o - x_0 y_0 z_0$ with the forward kinematics.

2.3 Study and solution of the three dimension problem

Now, let us consider the three dimension problem which is illustrated in Figure 2.2. We take the actual coordinate frames used on our robot, i.e., six joints. The frame on the gripper is the 6th coordinate frame $o_6 - x_6y_6z_6$. Thus, the force F, torque M and torque arm R become $F(f_{x_6}, f_{y_6}, f_{z_6})$, $M(m_{x_6}, m_{y_6}, m_{z_6})$ and $R(r_{x_6}, r_{y_6}, r_{z_6})$. We have the relation similar to the 2-D situation:

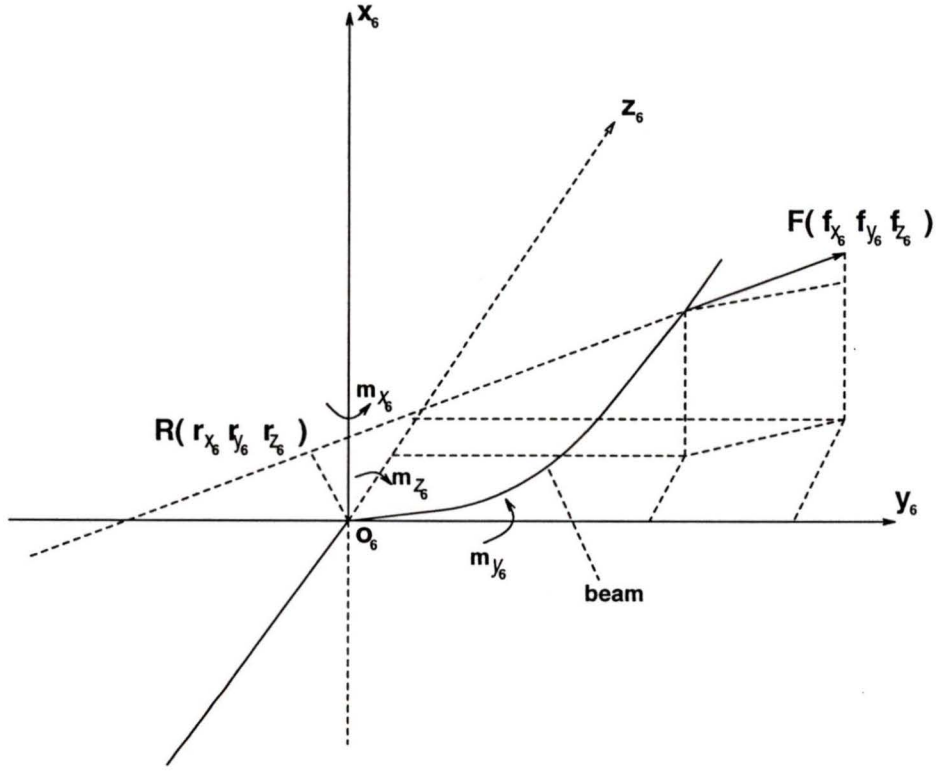

Figure 2.2: 3-D problem

$$m(m_{x_6}, m_{y_6}, m_{z_6}) = R \times F \tag{2.7}$$

$$\begin{cases} m_{x_6} &= r_{y_6} f_{z_6} - r_{z_6} f_{y_6} \\ m_{y_6} &= r_{z_6} f_{x_6} - r_{x_6} f_{z_6} \\ m_{z_6} &= r_{x_6} f_{y_6} - r_{y_6} f_{x_6} \end{cases} \tag{2.8}$$

$$R \cdot F = r_{x_6} f_{x_6} + r_{y_6} f_{y_6} + r_{z_6} f_{z_6} = 0 \tag{2.9}$$

In the light of (2.8) and (2.9), $R(r_{x_6}, r_{y_6}, r_{z_6})$ can be solved as:

$$\begin{aligned} r_{x_6} &= \frac{m_{z_6} f_{y_6} - m_{y_6} f_{z_6}}{{f_{x_6}}^2 + {f_{y_6}}^2 + {f_{z_6}}^2} \\ r_{y_6} &= \frac{m_{x_6} f_{z_6} - m_{z_6} f_{x_6}}{{f_{x_6}}^2 + {f_{y_6}}^2 + {f_{z_6}}^2} \\ r_{z_6} &= \frac{m_{y_6} f_{x_6} - m_{x_6} f_{y_6}}{{f_{x_6}}^2 + {f_{y_6}}^2 + {f_{z_6}}^2} \end{aligned} \tag{2.10}$$

Here, we require the elastic beam to be a cylinder. In the light of this condition and the results of the previous paragraph, it is not difficult to get the solutions of the three dimension problem.

Since the elastic beam is a cylinder, its deformation constantly happens in a plane (according to material mechanics), and this plane is defined by the beam and force $F(f_{x_6}, f_{y_6}, f_{z_6})$, namely, the **deformation plane**. If we can get a new frame which has two axes in this deformation plane, we can use the results (2.5) and (2.6). First, let us turn the frame $o_6 - x_6 y_6 z_6$ around z_6-axis with the angle θ as to form a new coordinate frame $o'_6 - x'_6 y'_6 z'_6$ which is illustrated in Figure 2.3 and whose x'_6-axis in the **deformation plane**.

In fact, the *deformation plane* can also be defined by the force $F(f_{x_6}, f_{y_6}, f_{z_6})$ and the torque vector $R(r_{x_6}, r_{y_6}, r_{z_6})$ uniquely. So, if we let $i = (cos\theta, -sin\theta, 0)$, we know F, R and i are in the same plane (deformation plane). That is :

$$\begin{vmatrix} f_{x_6} & f_{y_6} & f_{z_6} \\ cos\theta & -sin\theta & 0 \\ r_{x_6} & r_{y_6} & r_{z_6} \end{vmatrix} = 0 \tag{2.11}$$

So, we have the results:

$$f_{z_6} r_{y_6} cos\theta - f_{x_6} r_{z_6} sin\theta + f_{z_6} r_{x_6} sin\theta - f_{y_6} r_{z_6} cos\theta = 0 \tag{2.12}$$

Substituting m_{x_6} and m_{y_6} from equation (2.8) yields

$$tan\theta = \frac{m_{x_6}}{m_{y_6}} \tag{2.13}$$

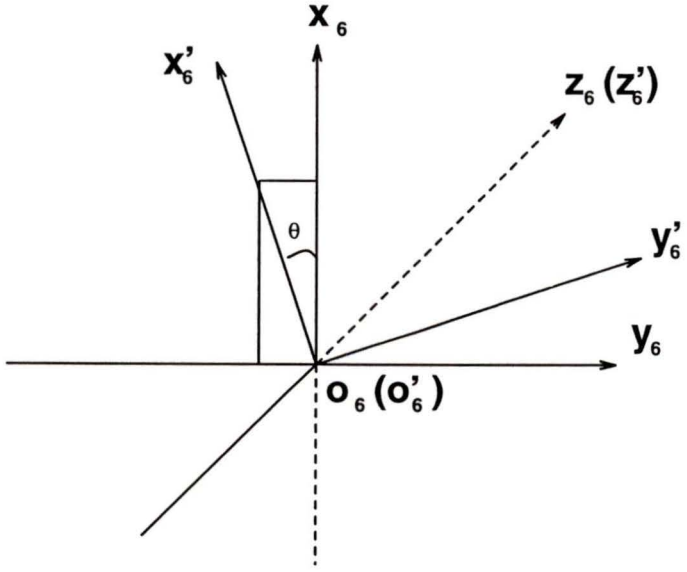

Figure 2.3: First Transformation

The orientation transformation [From $o_6 - x_6y_6z_6$ to $\acute{o}_6 - \acute{x}_6\acute{y}_6\acute{z}_6$] is :

$$T_{\acute{6}}^{6} = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.14}$$

Then, we can get the transformations :

$$\begin{pmatrix} f_{\acute{x}_6} \\ f_{\acute{y}_6} \\ f_{\acute{z}_6} \end{pmatrix} = T_{\acute{6}}^{6} \cdot \begin{pmatrix} f_{x_6} \\ f_{y_6} \\ f_{z_6} \end{pmatrix} = \begin{pmatrix} f_{x_6}cos\theta - f_{y_6}sin\theta \\ f_{x_6}sin\theta + f_{y_6}cos\theta \\ f_{z_6} \end{pmatrix} \tag{2.15}$$

$$\begin{pmatrix} r_{\acute{x}_6} \\ r_{\acute{y}_6} \\ r_{\acute{z}_6} \end{pmatrix} = T_{\acute{6}}^{6} \cdot \begin{pmatrix} r_{x_6} \\ r_{y_6} \\ r_{z_6} \end{pmatrix} = \begin{pmatrix} r_{x_6}cos\theta - r_{y_6}sin\theta \\ r_{x_6}sin\theta + r_{y_6}cos\theta \\ r_{z_6} \end{pmatrix} \tag{2.16}$$

Second, we turn the coordinate frame $\acute{o}_6 - \acute{x}_6\acute{y}_6\acute{z}_6$ around $\acute{x}_6 - axis$ with the angle β to form another coordinate frame $\ddot{o}_6 - \ddot{x}_6\ddot{y}_6\ddot{z}_6$, which has two axes in the **deformation plane**, illustrated in Figure 2.4.

The orientation coordinate transformation [From $\acute{o}_6 - \acute{x}_6\acute{y}_6\acute{z}_6$ to $\ddot{o}_6 - \ddot{x}_6\ddot{y}_6\ddot{z}_6$] is :

$$T_{\ddot{\acute{6}}}^{\acute{6}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\beta & -sin\beta \\ 0 & sin\beta & cos\beta \end{pmatrix} \tag{2.17}$$

Then,

$$\begin{pmatrix} f_{\ddot{x}_6} \\ f_{\ddot{y}_6} \\ f_{\ddot{z}_6} \end{pmatrix} = T_{\ddot{\acute{6}}}^{\acute{6}} \cdot \begin{pmatrix} f_{\acute{x}_6} \\ f_{\acute{y}_6} \\ f_{\acute{z}_6} \end{pmatrix} = \begin{pmatrix} f_{\acute{x}_6} \\ f_{\acute{y}_6}cos\beta - f_{\acute{z}_6}sin\beta \\ f_{\acute{y}_6}sin\beta + f_{\acute{z}_6}cos\beta \end{pmatrix} \tag{2.18}$$

$$\begin{pmatrix} r_{\ddot{x}_6} \\ r_{\ddot{y}_6} \\ r_{\ddot{z}_6} \end{pmatrix} = T_{\ddot{6}}^{\acute{6}} \cdot \begin{pmatrix} r_{\acute{x}_6} \\ r_{\acute{y}_6} \\ r_{\acute{z}_6} \end{pmatrix} = \begin{pmatrix} r_{\acute{x}_6} \\ r_{\acute{y}_6} cos\beta - r_{\acute{z}_6} sin\beta \\ r_{\acute{y}_6} sin\beta + r_{\acute{z}_6} cos\beta \end{pmatrix} \tag{2.19}$$

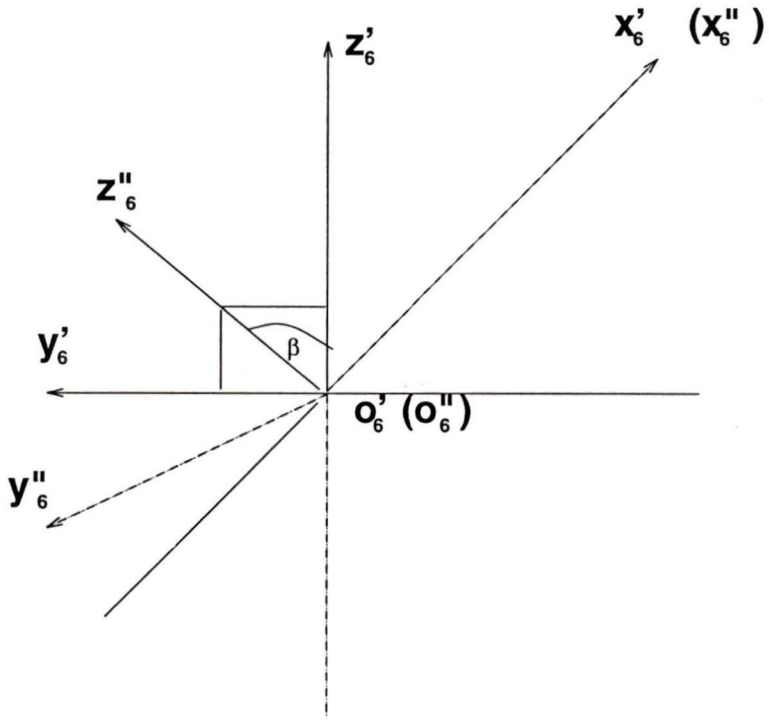

Figure 2.4: Second Transformation

The coordinate frame we are looking for must have:

$$f_{\ddot{y}_6} = 0 \tag{2.20}$$

$$r_{\ddot{y}_6} = 0 \tag{2.21}$$

From (2.18) and (2.20), we know β should be :

$$tan\beta = \frac{f_{\acute{y}_6}}{f_{\acute{z}_6}} \tag{2.22}$$

According to (2.10), (2.12), (2.15) and (2.16), we find that equation (2.21) is satisfied automatically and we have the result :

$$\begin{aligned} \ddot{R} \cdot \ddot{F} &= r_{\ddot{x}_6} f_{\ddot{x}_6} + r_{\ddot{z}_6} f_{\ddot{z}_6} \\ &= r_{\acute{x}_6} f_{\acute{x}_6} + r_{\acute{y}_6} f_{\acute{y}_6} + r_{\acute{z}_6} f_{\acute{z}_6} \\ &= r_{x_6} f_{x_6} + r_{y_6} f_{y_6} + r_{z_6} f_{z_6} \\ &= 0 \end{aligned} \tag{2.23}$$

Finally, we get what we want — frame $\ddot{o}_6 - \ddot{x}_6\ddot{y}_6\ddot{z}_6$. In this frame, we can use the results (2.5) and (2.6).

The orientation transformation [from $o_6 - x_6y_6z_6$ to $\ddot{o}_6 - \ddot{x}_6\ddot{y}_6\ddot{z}_6$] is :

$$\begin{aligned} T_{\ddot{6}}^{6} &= T_{\ddot{6}}^{\acute{6}} \cdot T_{\acute{6}}^{6} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\beta & -sin\beta \\ 0 & sin\beta & cos\beta \end{pmatrix} \cdot \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} cos\theta & -sin\theta & 0 \\ cos\beta sin\theta & cos\beta cos\theta & -sin\beta \\ sin\beta sin\theta & sin\beta cos\theta & cos\beta \end{pmatrix} \end{aligned} \tag{2.24}$$

On the other hand:

$$\begin{aligned} T_{6}^{\ddot{6}} &= T_{\acute{6}}^{6} \cdot T_{\ddot{6}}^{\acute{6}} \\ &= \begin{pmatrix} cos\theta & sin\theta cos\beta & sin\theta sin\beta \\ -sin\theta & cos\theta cos\beta & cos\theta sin\beta \\ 0 & -sin\theta & cos\theta \end{pmatrix} \end{aligned} \tag{2.25}$$

Now, we have the transformation formulas :

$$\begin{pmatrix} f_{\ddot{x}_6} \\ f_{\ddot{y}_6} \\ f_{\ddot{z}_6} \end{pmatrix} = T_{\ddot{6}}^{6} \cdot \begin{pmatrix} f_{x_6} \\ f_{y_6} \\ f_{z_6} \end{pmatrix}$$

$$= \begin{pmatrix} cos\theta & -sin\theta & 0 \\ cos\beta sin\theta & cos\beta cos\theta & -sin\beta \\ sin\beta sin\theta & sin\beta cos\theta & cos\beta \end{pmatrix} \cdot \begin{pmatrix} f_{x_6} \\ f_{y_6} \\ f_{z_6} \end{pmatrix} \tag{2.26}$$

$$\begin{pmatrix} r_{\ddot{x}_6} \\ r_{\ddot{y}_6} \\ r_{\ddot{z}_6} \end{pmatrix} = T_{\ddot{6}}^{6} \cdot \begin{pmatrix} r_{x_6} \\ r_{y_6} \\ r_{z_6} \end{pmatrix}$$

$$= \begin{pmatrix} cos\theta & -sin\theta & 0 \\ cos\beta sin\theta & cos\beta cos\theta & -sin\beta \\ sin\beta sin\theta & sin\beta cos\theta & cos\beta \end{pmatrix} \cdot \begin{pmatrix} r_{x_6} \\ r_{y_6} \\ r_{z_6} \end{pmatrix} \tag{2.27}$$

In the light of (2.20) and (2.21), we can get torque in the frame $\ddot{o}_6 - \ddot{x}_6\ddot{y}_6\ddot{z}_6$ directly,

$$\begin{pmatrix} m_{\ddot{x}_6} \\ m_{\ddot{y}_6} \\ m_{\ddot{z}_6} \end{pmatrix} = \begin{pmatrix} 0 \\ r_{\ddot{z}_6} f_{\ddot{x}_6} - r_{\ddot{x}_6} f_{\ddot{z}_6} \\ 0 \end{pmatrix} \tag{2.28}$$

Here,

$$tan\beta = \frac{f_{\acute{y}_6}}{f_{\acute{z}_6}}$$

$$tan\theta = \frac{m_{x_6}}{m_{y_6}}$$

$$
\begin{aligned}
cos\theta &= \frac{m_{y_6}}{\sqrt{(m_{y_6})^2 + (m_{x_6})^2}} \\
sin\theta &= \frac{m_{x_6}}{\sqrt{(m_{y_6})^2 + (m_{x_6})^2}}
\end{aligned}
$$

$$
\begin{aligned}
cos\beta &= \frac{f_{z_6}\sqrt{(m_{y_6})^2 + (m_{x_6})^2}}{\sqrt{((m_{y_6})^2 + (m_{x_6})^2)(f_{z_6})^2 + (f_{x_6}m_{x_6} + f_{y_6}m_{y_6})^2}} \\
sin\beta &= \frac{f_{x_6}m_{x_6} + f_{y_6}m_{y_6}}{\sqrt{((m_{y_6})^2 + (m_{x_6})^2)(f_{z_6})^2 + (f_{x_6}m_{x_6} + f_{y_6}m_{y_6})^2}}
\end{aligned}
$$

2.4 Compensation for the gravity and noise effect

The robot end-effector is connected to the robot arm through the force sensor and the gravity of the end-effector always exerts on the force sensor. The hardware noise also effects the force sensor. Therefore, we have to deduct the force of the gravity and the noise from the force sensor values while the robot is touching or searching an object.

Because the relative mass center location and the weight of the end-effector are constant, for any end-effector orientation, we can calculate the components of the weight and the weight torque in the force sensor coordinates frame if we know the sensor homogeneous transformation matrix related to the robot base frame. On REIS V15 industry robot, the homogeneous transformation matrix of the force sensor coordinate frame is the same as the end-effector's, thus, the transformation relationship is:

$$F_w^e = [R\_0e]^{-1} \cdot F_w^b$$

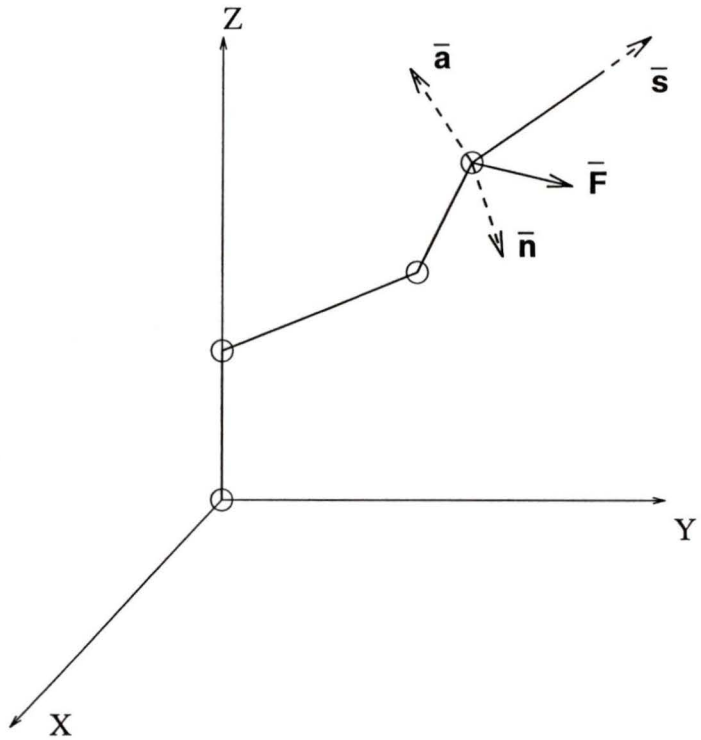

Figure 2.5: The end-effector orientation with respect to the base frame

Here,

$$[R\_0e] = (\vec{n}, \vec{s}, \vec{a}) = \begin{pmatrix} n_x & s_x & a_x \\ n_x & s_x & a_x \\ n_x & s_x & a_x \end{pmatrix}$$ The orientation matrix of the end-effector

F_w^e The gravity vector of the robot end-effector relative to the end-effector coordinate frame

F_w^b The gravity vector of the robot end-effector relative to the robot base coordinate frame

The torque of the gravity is :

$$M_w^e = R_w^e \cdot F_w^e \tag{2.29}$$

The gravity F_w^b and torque arm $R_w^e = (0, 0, r)$ can be measured in advance and the force F_w^e is:

$$F_w^e = (-n_z mg, -s_z mg, -a_z mg) \tag{2.30}$$

because $F_w^b = (0, 0, -mg)$.

To deduct the noise effect, we get the sum of fifty samples at each time interval and divide by fifty to get final gross touching force F_{ft}. Finally, the real touching force and torque are:

$$F^e = F_{ft} - F_w^e$$
$$M^e = M_{ft} - M_w^e$$

2.5 Summary

An algorithm is developed to calculate the contact point between the elastic beam and the object surface based on the use of touching forces and torques. The orientation or the equation of the object surface can be found by using this algorithm.

Chapter 3

Working on an arbitrarily oriented planar surface

3.1 Introduction

With the algorithm introduced in chapter 2, the touching points on the object surface can be calculated. However, as the robot has no knowledge of the object position, a searching method should be presented. In this chapter, we will discuss how to let the robot detect an arbitrarily oriented object which has a planar surface, how to measure the boundary of the surface in the robot workspace, how to graphically generate planar curves—such as circles, ovals, rectangles, cubic curves and words on the surface—then map them into the robot joint space.

3.2 Searching and modelling the planar surface

To let the robot get the knowledge of the object—ascertaining such things as position, orientation and boundary inside the robot working space—searching and measuring methods should be developed. Here, two searching methods are presented. The robot will stop searching if any of its joints exceed their displacement limits, or the touching force sensed by the force sensor is over a specified value, which means the robot has found the object.

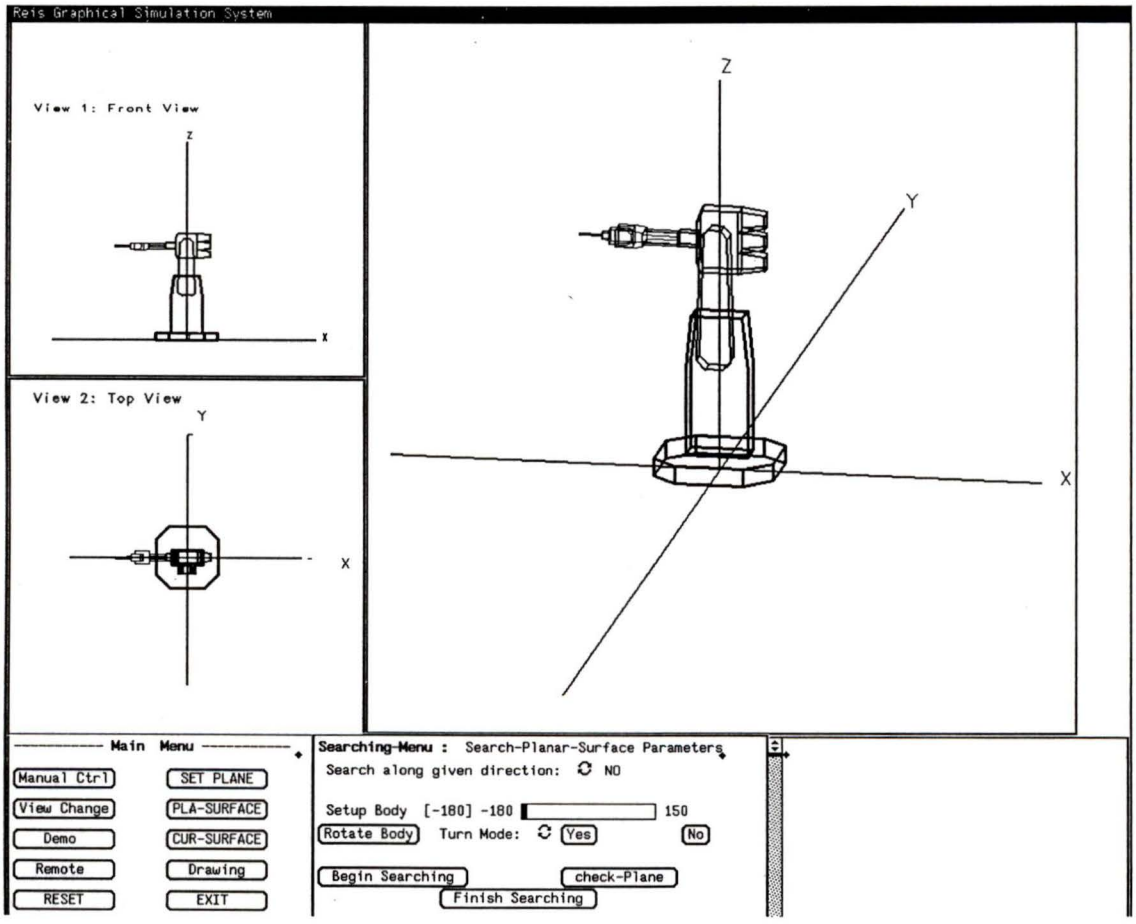

Figure 3.1: Searching an object control panel

The searching parameters in both methods will be set up by **GAPCI** shown as Figure 3.1.

3.2.1 Search along given direction

In this searching method, the tip of the end-effector will move along a specific direction (S=**X**,**Y**,**Z**) toward the object surface. The moving interval is determined by the following linear equation with the starting point(x_0, y_0, z_0):

$$\frac{x - x_0}{\mathbf{X}} = \frac{y - y_0}{\mathbf{Y}} = \frac{z - z_0}{\mathbf{Z}} = t$$

The end-effector orientation needs to be controlled so that the robot will touch the object smoothly. It can be done if the gripper on the end-effector is parallel to the ground and inside the plane composed of the robot arm2 and arm3, as shown in Figure 3.3.

To determine the searching direction and starting point, we can adjust the robot joint angles by **GAPCI** so as to let the end-effector point to the object shown in Figure 3.2. The tip position and orientation matrix will be known according to the robot forward kinematics. R^e is with respect to the robot base frame. The searching direction is the third column of R^e:

$$S = [R^e]_3$$

and the position of the end-effector tip is taken as the starting point (x_0, y_0, z_0).

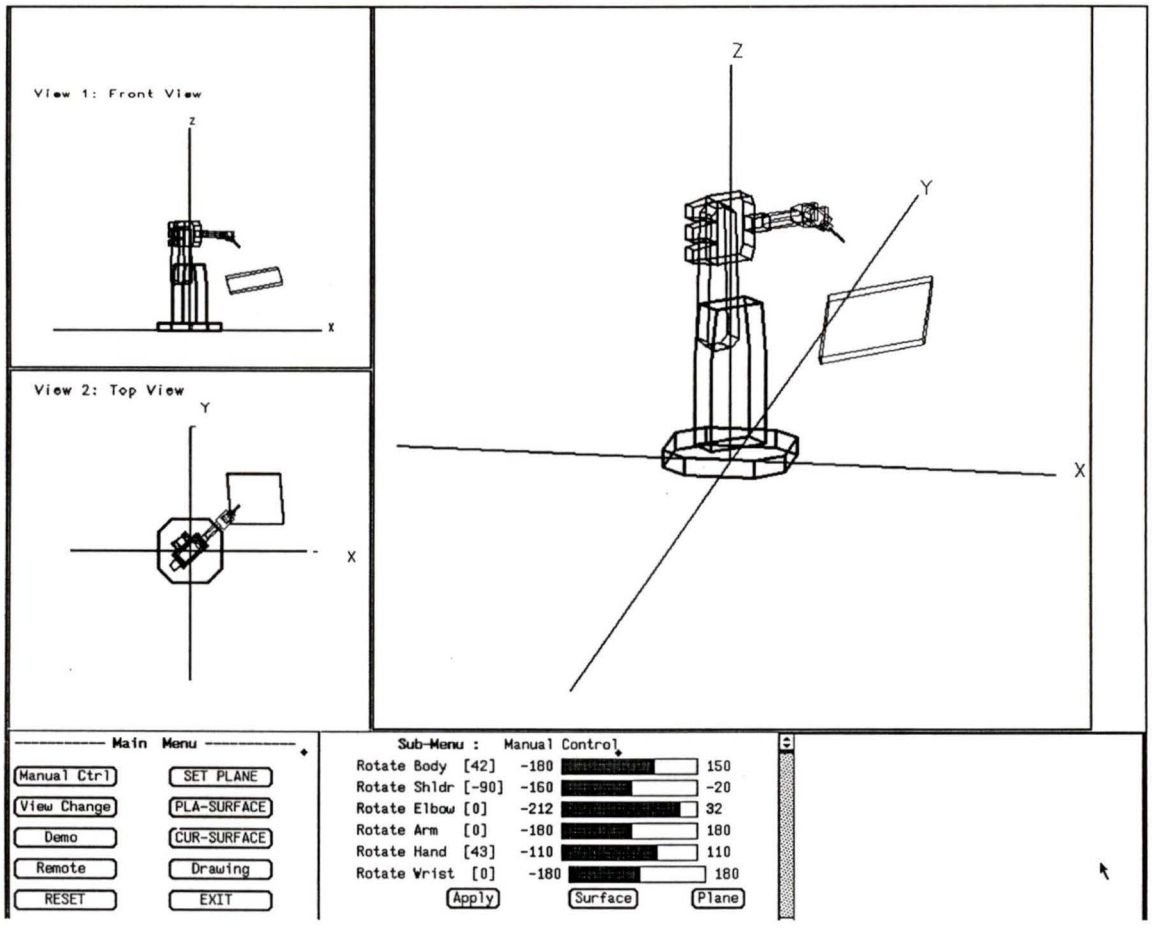

Figure 3.2: Joint adjusting panel

To get the end-effector orientation matrix, searching direction and starting point associated with the above condition, we need to operate the robot as follows:

(1). Set the robot to its home position. The initial orientation of the end-effector is :

$$R_0^e = \begin{vmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{vmatrix}$$

(2). Adjust θ_1 and θ_5 until the end-effector points to the object. So the current R^e is(as required):

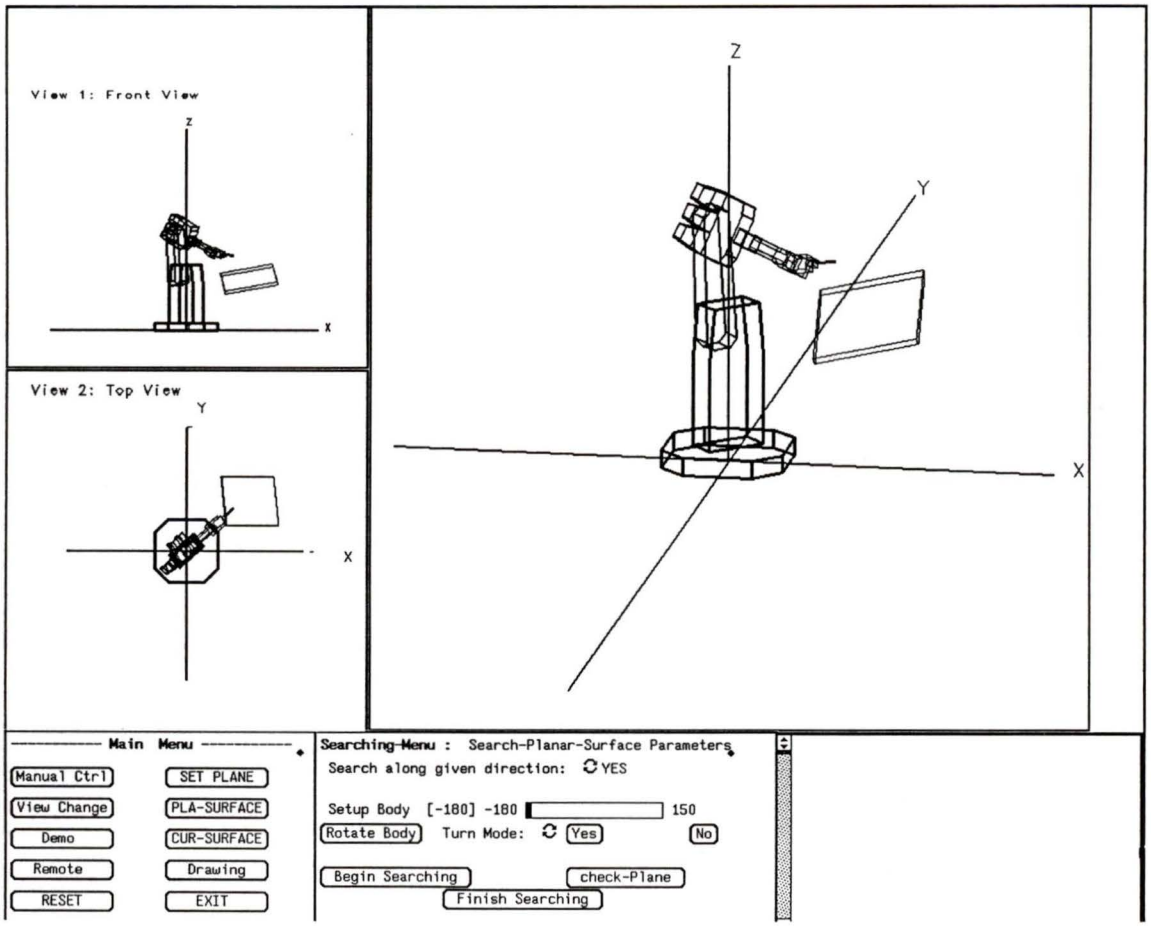

Figure 3.3: Searching the object with specific orientation

$$R^e = Rot(z,\theta_1)\cdot R_0^e \cdot R(y,-\theta_5) = \begin{vmatrix} cos\theta_5.sin\theta_1 & -cos\theta_5.sin\theta_1 & -sin\theta_5 \\ sin\theta_1 & cos\theta_1 & 0 \\ sin\theta_5.cos\theta_2 & -sin\theta_5.sin\theta_1 & cos\theta_5 \end{vmatrix}$$

And the searching direction is:

$$\mathbf{S} = (\mathbf{X},\mathbf{Y},\mathbf{Z}) = [\mathbf{R^e}]_\mathbf{3}$$

(3). The starting point(x_0, y_0, z_0) is the current position of the end-effector tip which is determined by the forward kinematics.

(4). The orientation matrix for the end-effector now is:

$$R^e_{search} = Rot(z, \theta_1) \cdot R^e_0 = \begin{vmatrix} 0 & -sin\theta_1 & -cos\theta_1 \\ 0 & cos\theta_1 & -sin\theta_1 \\ 1 & 0 & 0 \end{vmatrix}$$

The end-effector will keep this orientation while the robot is searching the object.

Using the main menu of the **GAPCI**, the user can adjust the six joints of the robot as shown in Figure 3.3. And the user can select the searching method by clicking on the circle "Searching along given direction" and start the searching procedure by pressing the button "Begin Searching" shown in Figure 3.1. The robot's inverse kinematics is involved in this searching method; therefore, the robot cannot move far along specific searching directions because of the joints' limit.

3.2.2 Search in the specific plane

In this searching method, the joint angles are directly assigned instead of using the inverse of the Jacobian matrix. Therefore, it can search extensively in the robot workspace.

The idea is that the whole working space is divided into several "searching planes" which are vertical to the ground and through the Z axis of the base frame. Then the robot starts searching for the object in each "searching plane". To define the searching plane, the user only needs to turn the robot joint angle θ_1 from its "home" position. This can be done with **GAPCI** by changing the slide value on the property "Setup body" shown in Figure 3.4. In the searching plane, joint θ_2, θ_3 and θ_5 are updating with a particular order so that the robot can search the farthest area inside the

searching plane. The searching procedure is composed of two steps, called searching cycle, shown in Figure 3.5.

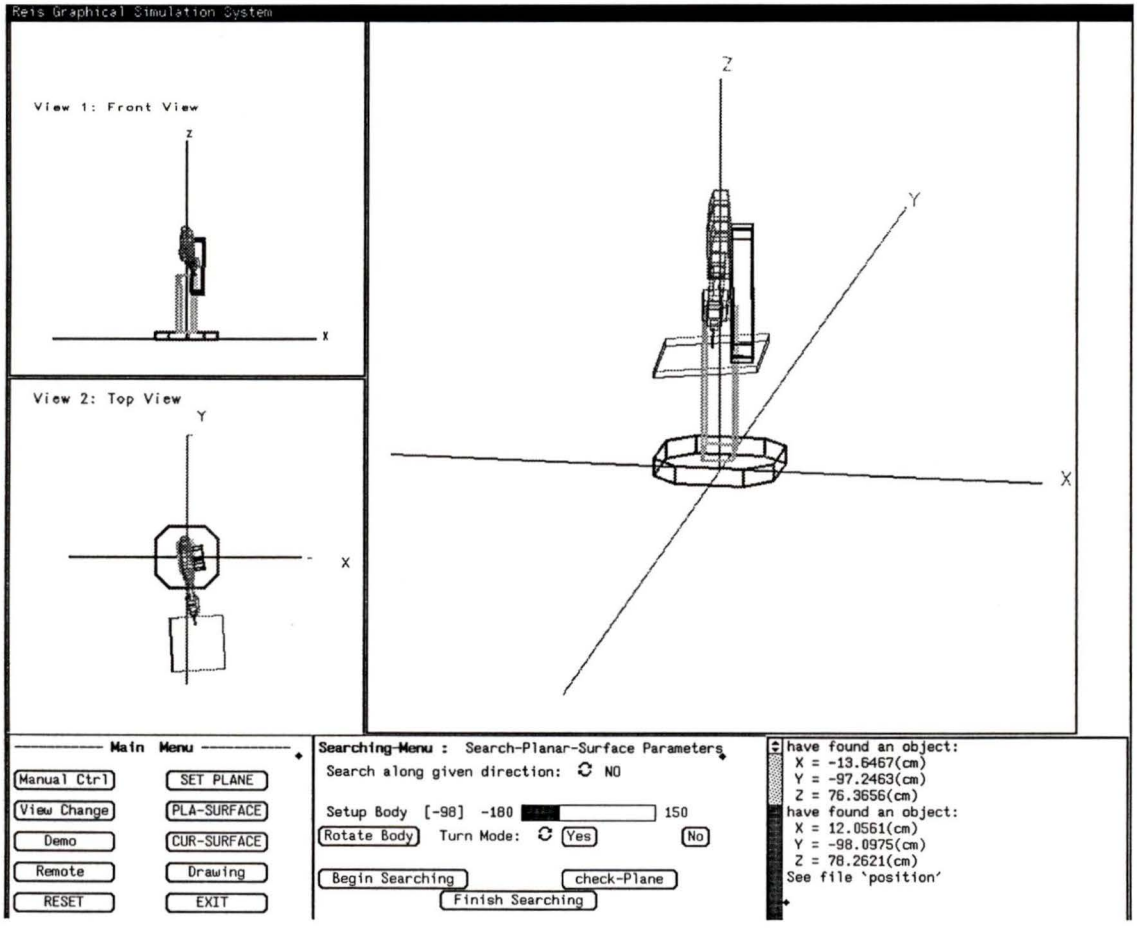

Figure 3.4: Panel for searching in a specific plane

In the first step, the robot starts searching at position A_1 and only joint θ_3 varies with the constant delta angle until the tip of the end-effector touches an object or the tip gets to position B_1(due to limitation of θ_3). If no object was touched at position B_1, the robot will change joint θ_5 with the constant delta angle until an object is contacted or the tip of the arm gets to position C_1(the limitation of θ_5). The robot

will go to the second step if no object is found.

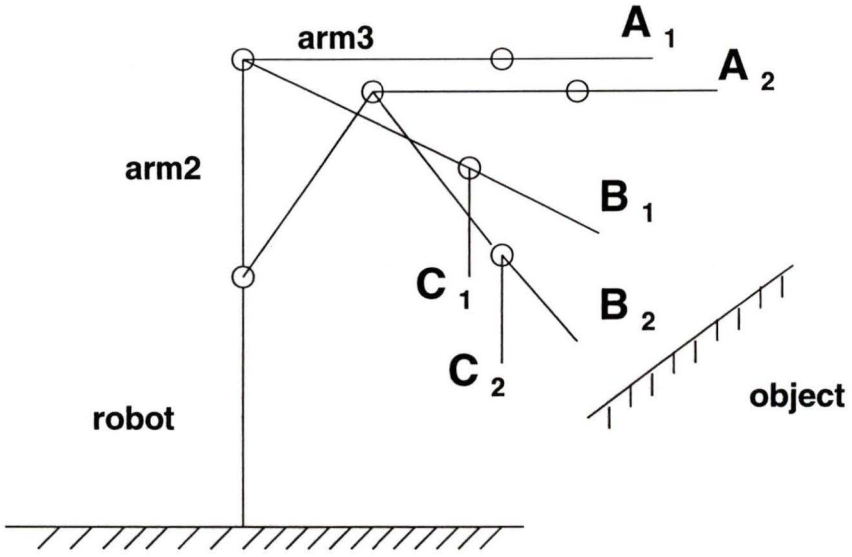

Figure 3.5: The robot searching cycle

In the second step, joint θ_2 changes with the positive delta angle, and joint θ_5 varies by adding the negative constant delta angle until the tip gets to position B_2 or touches an object. If no object is found and no joint exceeds its limit, the robot will gradually change joint θ_3 with the negative constant delta angle until the beam gets to position A_2 or touches an object. If no object is found in this search cycle, the robot will change joint θ_2 with the positive delta angle, and switch to the next searching cycle until it finds an object or stops searching when joint θ_5 gets to its limit value. If there is no object in this searching plane, the robot will go to the next searching plane where it continues the searching procedure until it finds an object or it finishes searching the whole workspace.

The user can order the robot to go to the next searching plane counterclockwise by pressing "NO" in the option "Turn Mode". Otherwise, the robot will go along clockwise. The interval angle between the two searching planes can be adjusted.

3.2.3 Modelling the planar surface

It is known that the equation of a planar surface is determined by three surface points which are not on one line. A method is presented here to let the robot get the other two points after it finds the first point on the surface.

(1). Lifting the robot end-effector up by altering joint θ_5;

(2). Then moving the end-effector forward (backward) to the surface with different value of the joint θ_2;

(3). Pointing the end-effector, which holds the elastic beam, to the new point on the surface and searching down for the second point;

(4). After the second point is found, the end-effector will be lifted up again and the robot will turn its body along counterclockwise or clockwise(based on the option for "Turn Mode").

(5). Then the robot will move the elastic beam down and forward to the surface for touching the third point. After the robot finishes gathering the three points, a graphical rectangle picture will be shown up to simulate the position and orientation of the object surface by **GAPCI**, as shown in Figure 3.4.

To transform the planar curve trajectory into the robot joint space, a transformation matrix between the planar surface and the robot base frame should be set up. The following algorithm is used to build up the matrix:

Assuming there are three points(A,B,C) that are detected on the surface.

(1). The first touching point A, as shown in Figure 3.6, is used as the origin $\mathbf{O_p}$ for the planar surface coordinate frame.

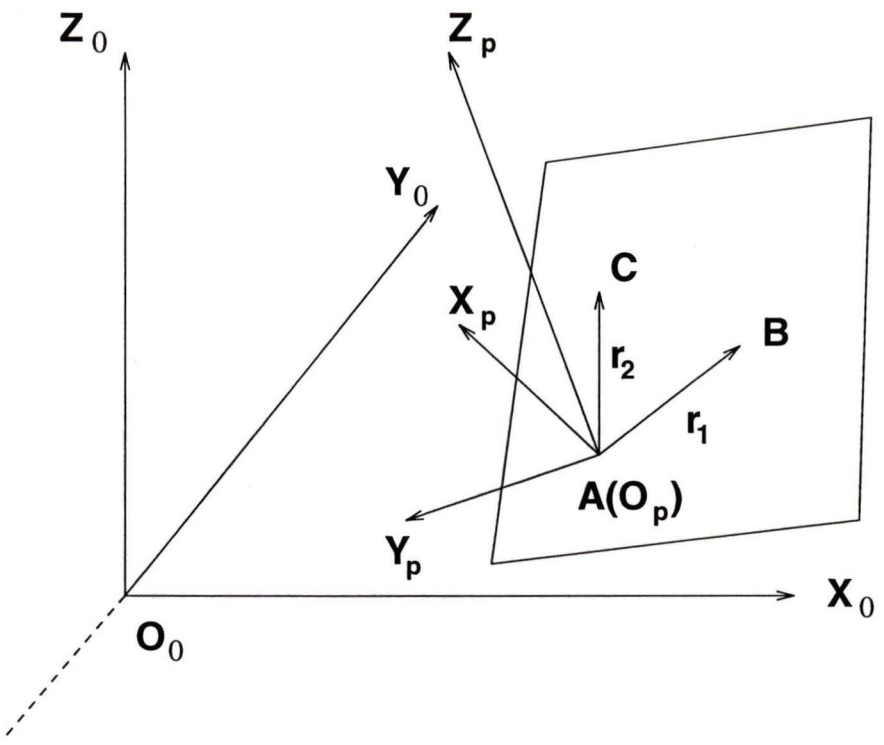

Figure 3.6: The object surface coordinate frame

(2). Points B and C are connected to point A separately generating two vectors: $\vec{r_1}, \vec{r_2}$.

(3). Getting the planar surface normal vector $\vec{\mathbf{n_p}}$ by:

$$\vec{\mathbf{n_p}} = \vec{\mathbf{r_1}} \times \vec{\mathbf{r_2}} \quad (if \quad n_z^p > 0)$$
$$\vec{\mathbf{n_p}} = \vec{\mathbf{r_2}} \times \vec{\mathbf{r_1}} \quad (if \quad n_z^p < 0)$$

Finally, the normalized vector $\vec{\mathbf{n_p}}$ is:

$$n_x^p = \frac{n_x^p}{\sqrt{(n_x^p)^2 + (n_y^p)^2 + (n_z^p)^2}}$$
$$n_y^p = \frac{n_y^p}{\sqrt{(n_x^p)^2 + (n_y^p)^2 + (n_z^p)^2}}$$
$$n_z^p = \frac{n_z^p}{\sqrt{(n_x^p)^2 + (n_y^p)^2 + (n_z^p)^2}}$$

(4). Taking the normal vector as $\mathbf{Z_p - axis}$ for the planar surface coordinate frame:

$$\vec{\mathbf{n_p}} = (n_x^p, n_y^p, n_z^p)$$

(5). If $\vec{\mathbf{n_p}}$ is not parallel with $\mathbf{Z_0 - axis}$, we define $\mathbf{X_p - axis}$ as :

$$\vec{\mathbf{a_p}} = \vec{\mathbf{n_p}} \times \mathbf{Z_0}$$

Here, $\mathbf{Z_0 - axis}$ is for the robot baseframe.

If $\vec{\mathbf{n_p}}$ is parallel with $\mathbf{Z_0 - axis}$, we define $\mathbf{X_P - axis}$ as:

$$\vec{\mathbf{a_p}} = (1, 0, 0)$$

Actually, in the practical situation it is hard to tell if $\vec{\mathbf{n_p}}$ is parallel with $\mathbf{Z_0 - axis}$ or not by numeric value. Therefore,

if $n_z^p > 0$, and $n_z^p < cos(2^0)$

or $n_z^p < 0$, and $n_z^p > -cos(2^0)$

We take $\vec{\mathbf{n_p}}$ as parallel with $\mathbf{Z_0 - axis}$.

(6). Finally, we define $\mathbf{Y_p - axis}$ for the plane as :

$$\vec{\mathbf{s_p}} = \vec{\mathbf{n_p}} \times \vec{\mathbf{a_p}}$$

(7). Now, we can get the *homogeneous transformation matrix* from the planar surface frame to the robot baseframe:

$$\mathbf{T_0^p} = \begin{pmatrix} \mathbf{R_0^p} & \mathbf{O_p} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} = \begin{pmatrix} \vec{\mathbf{a_p}} & \vec{\mathbf{s_p}} & \vec{\mathbf{n_p}} & \mathbf{O_p} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Where $\mathbf{O_p}$ is the coordinates of $\mathbf{A}$ vector locating the origin of the $\mathbf{P}$ frame with respect to the $\mathbf{O}$ frame.

3.3 Determination of the boundary of the planar surface

Due to potential singularities of the inverse of Jacobian and joint motion limitation, it is essential to know the surface boundary in the robot working space. With the knowledge of the boundary, we can create the curves inside the surface and calculate their trajectories. An algorithm is derived to manipulate the robot into detecting the boundary of the object surface.

3.3.1 Searching along an intersection line

To use the smallest number of joint angles as possible for determining the surface boundary, we still divide the robot work space into several *searching places* as mentioned before. With this method, the user can let the robot extensively touch the surface and only needs to calculate three joint values for each touching step. Each *searching place* intersects the object surface with a linear line l_1 and a coordinate frame $O - X_sY_sZ_s$ on the planar surface can be defined as shown in Figure 3.7. The robot will touch the object along these intersection lines until it finishes measuring the work space. Finally, the boundary is defined by the touching points.

The object surface transformation matrix to the baseframe is:

$$\mathbf{T_0^p} = \begin{pmatrix} \mathbf{R_0^p} & \mathbf{O_p} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} = \begin{pmatrix} \vec{\mathbf{a_p}} & \vec{\mathbf{s_p}} & \vec{\mathbf{n_p}} & \mathbf{O_p} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

So the equation for the planar surface in the baseframe is:

$$\vec{\mathbf{n_p}} \cdot \vec{\mathbf{r}} = \vec{\mathbf{n_p}} \cdot \mathbf{O_p} \tag{3.1}$$

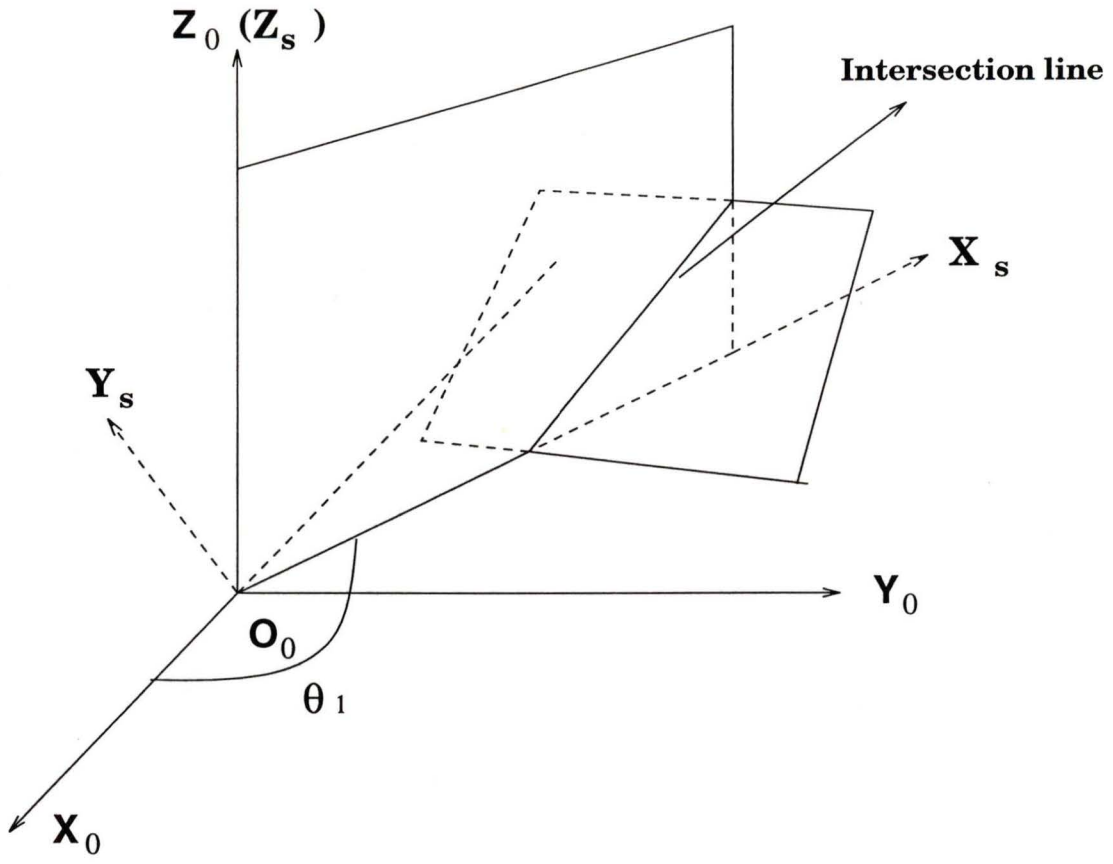

Figure 3.7: Intersection line and searching plane frame

On the other hand, the transformation matrix from the searching plane frame to the robot baseframe is:

$$\mathbf{T}_{\mathbf{0}}^{\mathbf{s}} = \begin{pmatrix} \mathbf{R}_{\mathbf{0}}^{\mathbf{s}} & \mathbf{O_s} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} = \begin{pmatrix} cos\theta_1 & -sin\theta_1 & 0 & 0 \\ sin\theta_1 & cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

Thus, the equation for the *searching plane* in the baseframe is:

$$\vec{\mathbf{s_s}} \cdot \vec{\mathbf{r}} = \vec{\mathbf{s_s}} \cdot \mathbf{O_s} = 0 \tag{3.3}$$

Now, the intersection line l_1 is derived from (3.1), and (3.3):

$$\begin{cases} x \cdot n_x^p + y \cdot n_y^p + z \cdot n_z^p = \vec{\mathbf{n_p}} \cdot \mathbf{O_p} \\ -x \cdot sin\theta_1 + y \cdot cos\theta_1 = 0 \end{cases} \tag{3.4}$$

Here, we assume $n_z^p \neq 0$ is always correct.

3.3.2 The equation of the intersection line

If we force all robot arm segments into the *searching plane*, it is always true that joints angles 4 and 6 are equal to zero. Thus, only joint 2, 3 and 5 need be updated. Under this condition, all robot arm segments and the intersection line are in the same plane—the *searching plane*. To simplify the calculation procedure, we can design the searching point and relative robot joint with respect to the *searching plane* instead of the robot baseframe.

The point transformation from the searching plane frame to the baseframe is:

$$\begin{aligned} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} &= T_0^s \cdot \begin{pmatrix} x_s \\ y_s \\ z_s \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} x_s \cdot cos\theta_1 - y_s \cdot sin\theta_1 \\ x_s \cdot sin\theta_1 + y_s \cdot cos\theta_1 \\ z_s \\ 1 \end{pmatrix} \end{aligned} \tag{3.5}$$

Substituting (3.5) into (3.4), the intersection line equation in the *searching plane* coordinate frame is :

$$\begin{cases} y_s & = & 0 \\ x_s(n_x^p cos\theta_1 + n_y^p sin\theta_1) + z_s n_z^p & = & \vec{\mathbf{n_p}} \cdot \mathbf{O_p} \end{cases} \tag{3.6}$$

3.3.3 The configuration of the robot joint

After knowing the line equation in the *searching plane*, we can design the robot joint angle for each searching step along the line, as shown in Figure 3.8. Here, only joints 2, 3 and 5 need to be calculated while joint 1 keeps constant value and joint $\theta_4 = \theta_5 = 0$. In each searching step, we force the elastic beam to be vertical to the intersection line so that the following result is true:

$$\alpha_1 + \alpha_2 = \beta + \alpha_3 \tag{3.7}$$

Considering the robot joint sign, we have the results:

$$\begin{cases} \alpha_1 & = & -\theta_2 \\ \alpha_2 & = & -\theta_3 \\ \alpha_3 & = & \theta_5 \end{cases} \tag{3.8}$$

From (3.6), β can be calculated:

$$\beta = Atan2[\frac{-(n_x^p \cdot cos\theta_1 + n_y^p \cdot sin\theta_1)}{n_z^p}] \tag{3.9}$$

The condition for β is: $0 \leq \beta < \frac{\pi}{2}$. This means the surface should face to the robot with the orientation shown as in Figure 3.8.

Based upon the above condition and results, we can let the tip of the beam satisfy the following equation when the beam is perpendicular to the intersection line:

$$\begin{cases} X_A & = & l_1 \cdot cos\alpha_1 + l_2 \cdot sin(\alpha_1 + \alpha_2) + l_3 \cdot sin\beta \\ Z_A & = & l_1 \cdot sin\alpha_1 - l_2 \cdot cos(\alpha_1 + \alpha_2) - l_3 \cdot cos\beta \end{cases} \tag{3.10}$$

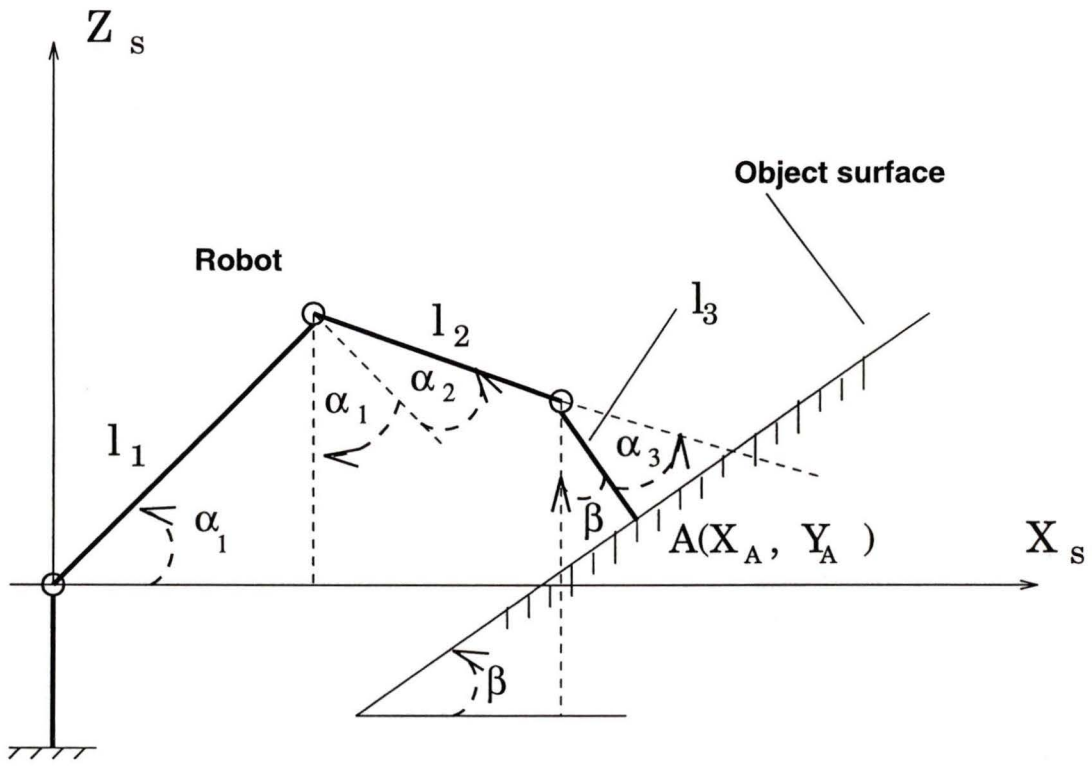

Figure 3.8: The robot configuration for each searching step

l_1: the length of the robot arm segment 2

l_2: the length of the robot arm segment 3

l_3: the length of the gripper and the tool

Because X_A and Z_A also satisfy the equation (3.6), the relationship between α_1 and α_2 can be obtained by substituting the X_A and Z_A into (3.6):

$$P \cdot sin(\alpha_1 + \alpha_2) - Q \cdot cos(\alpha_1 + \alpha_2) \;=\; A_1 - A_2 \tag{3.11}$$

Here,

$$\begin{aligned} A_1 &= \vec{\mathbf{n_p}} \cdot \mathbf{O_p} - n_z^p \cdot (l_1 \cdot sin\alpha_1 - l_3 \cdot cos\beta) \\ A_2 &= (l_1 \cdot cos\alpha_1 + l_3 \cdot sin\beta)(n_x^p \cdot cos\theta_1 + n_y^p \cdot sin\theta_1) \\ P &= l_2 \cdot (n_x^p \cdot cos\theta_1 + n_y^P \cdot sin\theta_1) \\ Q &= n_z^p \cdot l_2 \end{aligned}$$

Considering the robot joint limit, the following result is derived from (3.11):

$$\alpha_1 + \alpha_2 = Atan2[\frac{P}{-Q}] + Atan2[\frac{\sqrt{Q^2 + P^2 - (A_1 - A_2)^2}}{A_1 - A_2}] \qquad (3.12)$$

NOTE: The value of α_1 should be defined in advance when (3.11) is used.

3.3.4 Implementation of the determining procedure

Before the robot starts determining the boundary, the initial position for the robot should be defined. Otherwise, the robot joint angles cannot be calculated only by (3.8), (3.9) and (3.12). In the present method, the robot's initial position is set up with the following joint values:

$$\begin{aligned} \theta_1 &= \theta_1^* \\ \theta_2 &= -28^0 \\ \theta_3 &= -28^0 \\ \theta_4 &= 0 \\ \theta_5 &= 0 \\ \theta_6 &= 0 \end{aligned}$$

θ_1^* is the value for joint 1 at which the robot starts searching the object.

The initial position is shown in Figure 3.9.

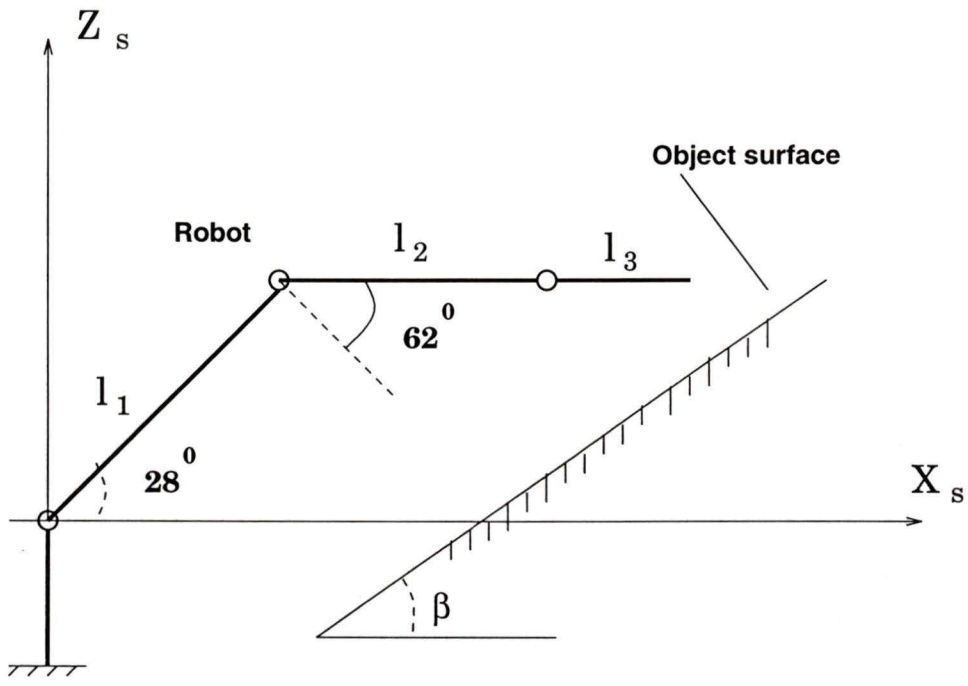

Figure 3.9: The robot initial position for determining the boundary

An integer variable N_p with zero as its initial value is applied to express how many searching planes the robot has been working in. Another integer variable R_b is used to monitor the robot body rotation direction with the regulation:

$R_b = 0$: The robot turns its body counterclockwise

$R_b = 1$: The robot turns its body clockwise

The initial value of R_b is zero.

The whole procedure for determining the boundary is divided into the following steps:

(1). The formulas (3.9) and (3.12) are used to calculate the joint angle θ_3^* and θ_5^* under conditions (3.8) and $\theta_1 = \theta_1^*$. The first desired position for the robot, where

the beam is vertical to the intersection line and the tip of the beam is on the line, is:

$$\begin{aligned} \theta_1 &= \theta_1^* \\ \theta_2 &= -28^0 \\ \theta_3 &= \theta_3^* \\ \theta_4 &= 0 \\ \theta_5 &= \theta_5^* \\ \theta_6 &= 0 \end{aligned} \tag{3.13}$$

(2). Prior to the desired position (3.13), the robot is put at the preliminary position with the joint values:

$$\begin{aligned} \theta_1 &= \theta_1^* \\ \theta_2 &= -28^0 \\ \theta_3 &= \theta_3^* \\ \theta_4 &= 0 \\ \theta_5 &= \theta_5^* - \Delta\theta_5 \\ \theta_6 &= 0 \end{aligned}$$

$\Delta\theta_5$ is a positive value defined by the operator so that the beam is over the intersection line, as shown in Figure 3.10.

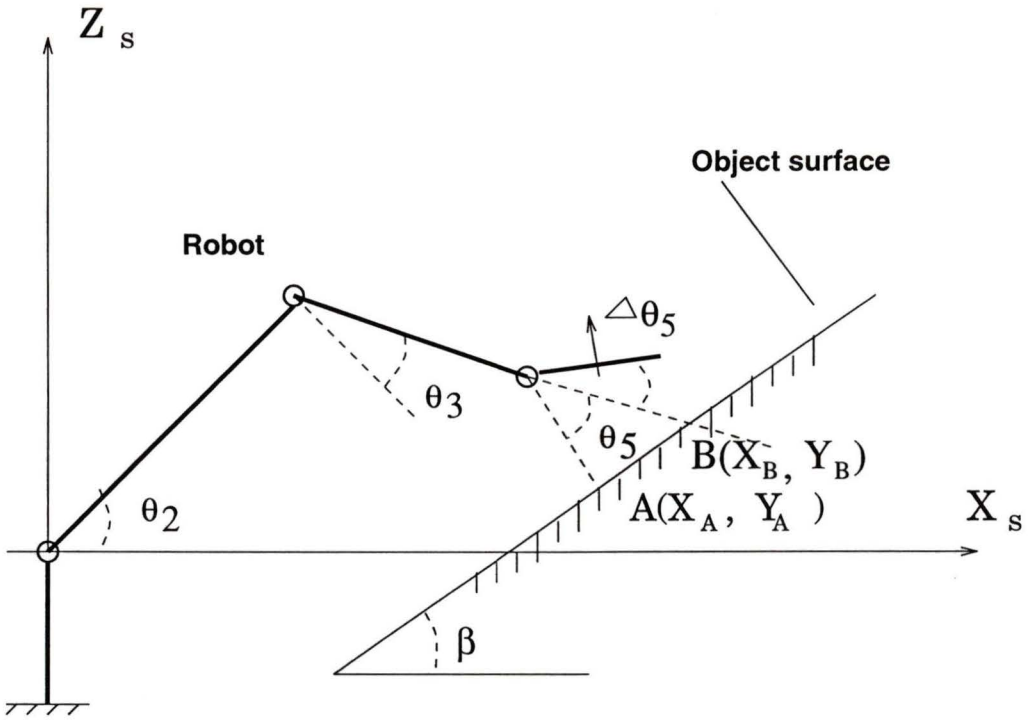

Figure 3.10: The robot preliminary position at each searching step

(3). Joint 5 is changed by adding a positive value so as to let the beam touch in front of the object surface. If the beam cannot contact the object between point A and point B in Figure 3.10, go to step (4), otherwise, go to step (5).

(4). A particular direction vector, which points to the robot along the intersection line, is designed to obtain the next position for the beam tip. The length of the vector depends on the operator. For instance, in the present searching algorithm, the length is defined as 2 cm. If the previous intersection point, the next point and the length of the direction vector are defined as (X_p, Z_p), (X_n, Z_n) and L separately, the following results are correct in the searching plane frame, as shown in Figure 3.11:

$$\begin{aligned} X_n &= X_p - L \cdot cos\beta \\ Z_n &= Z_p - L \cdot sin\beta \end{aligned} \tag{3.14}$$

Based upon the conditions of $\theta_4 = \theta_6 = 0$ and the beam should be vertical to the intersection line, the orientation of the end-effector with respect to the searching plane frame is:

$$R_s^e = \begin{pmatrix} cos\beta & 0 & sin\beta \\ 0 & -1 & 0 \\ sin\beta & 0 & -cos\beta \end{pmatrix} \tag{3.15}$$

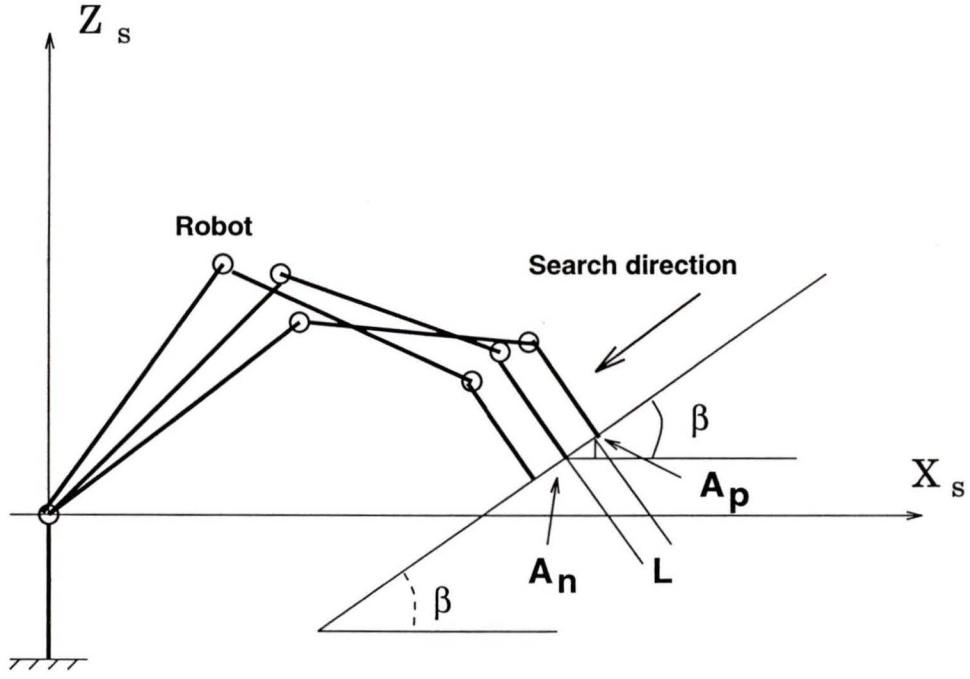

Figure 3.11: Detecting the boundary along the intersection line

With the result (3.2), both (3.14) and (3.15) can be transformed from the searching plane frame to the robot baseframe by:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = T_0^s \cdot \begin{pmatrix} X_n \\ 0 \\ Z_n \\ 1 \end{pmatrix} \tag{3.16}$$

$$\mathbf{R}_0^e = \mathbf{R}_0^s \cdot \mathbf{R}_s^e \tag{3.17}$$

Using the results (3.18) and (3.17), the new desired position as mentioned in (3.13) can be calculated with the robot's inverse kinematics.

Repeat step (2) to (4), if the robot can contact the object, go to step (5); otherwise, go to step (7) before the robot joint θ_2,θ_3 and θ_5 exceed their angle limit.

(5). Let $N_p = N_p + 1$. In case the robot has touched the object, the touching position is recorded in the data file as N_p as its index number then go to step (6). Only if $N_p = 1$, the current joint angles will be stored as *Medium Joint Position*. If the robot can not touch the object along this intersection line, go to step (7).

(6). Lift the beam up by adding a specific negative value to joint 5. If $R_b = 0$, the robot body is rotated along counterclockwise with a specific constant value. Otherwise, the robot body is rotated along clockwise.

If $N_p = 1$, keep the robot's current joint angle as "preliminary position", then go to step (3).

If $N_p > 1$, we need to define joint 2 so as to find the robot's next "preliminary position". Comparing the length to the origin of the XY coordinate plane for the two points B_{N_p} and B_{N_p+1}, as shown in Figure 3.12, if $R_{N_p} > R_{Np+1}$, for getting point B_{N_p+2}, joint 2 is given a positive incremental value. Otherwise, joint 2 is given negtive incremental value, the final θ_2 is named as θ_2^*. Then go to step (1), but $\theta_2 = \theta_2^*$ instead of -28 degrees.

(7). If the robot joints cannot satisfy condition (3.11), or any joint angle exceeds its limit, or the robot joints already have got to the specific limit values defined by the operator, it is assumed that no part of the object surface is along this intersection line. In the present searching method, a specific limit value is $-70^0 < \theta_2 \leq -28^0$.

When the robot cannot touch the object along the intersection line, the current joint angles are stored as terminal position Θ.

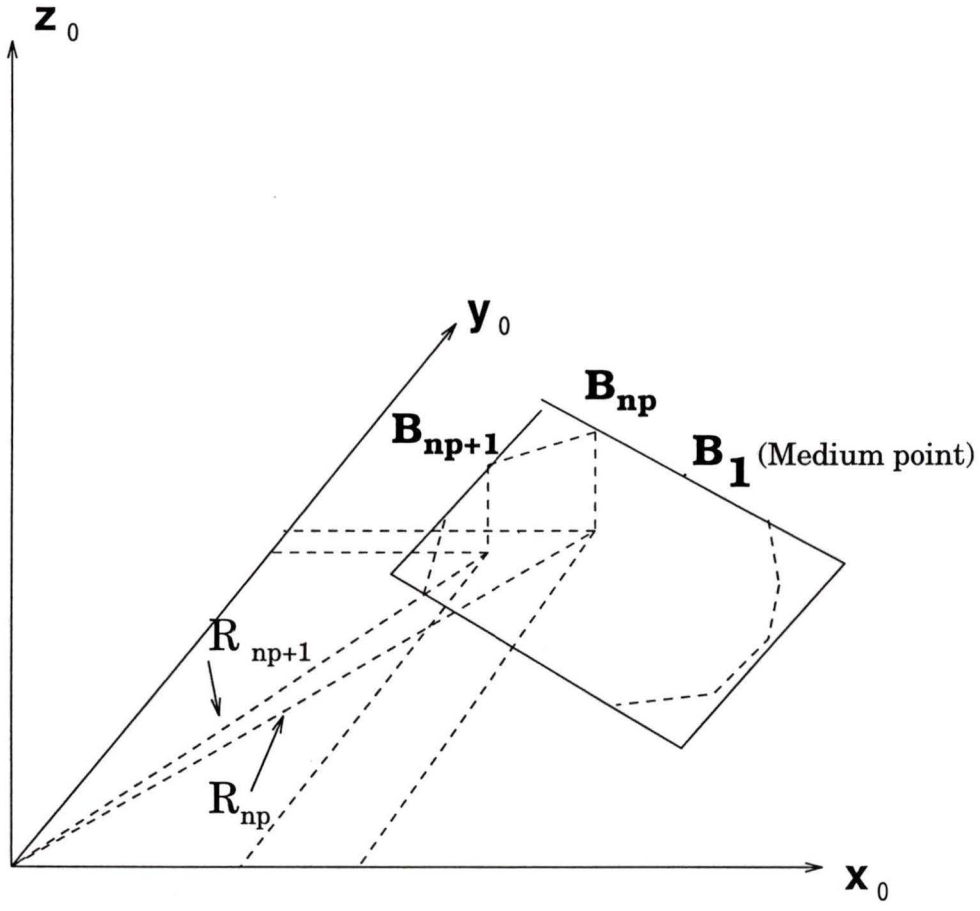

Figure 3.12: Measuring the boundary of the object surface

If $R_b = 0$, the robot will turn its body (by changing joint 1) clockwise to the object until it touches the object. This touching point is recorded in the data file and an integer variable N_1 is used by $N_1 = N_p$. N_1 expresses how many boundary points there are from the *Medium Joint Position* to the final point in the counterclockwise direction. Go to step (8).

If $R_b = 1$, the robot will turn its body (by changing joint 1) counterclockwise to the object until it touches the object. This touching point is recorded in the data file

and an integer variable N_2 is used by $N_2 = N_p$. N_2 expresses the number of boundary points from the *Medium Joint Position* to the final point. Go to step (8).

(8). The robot is put back to position Θ and joint 5 is forced back to zero. Consequently, the robot goes to the *Medium Joint Position* with $\theta_5 = 0$. If $R_b = 1$, the robot finishes determining the boundary and goes to step (9). Otherwise, $R_b = 1$ and $N_p = 1$, the robot goes to step (6).

(9). The boundary points in the data file are re-arranged and put into a one dimensional array according to the index number N_1 and N_2. All of the points are calculated with respect to the robot baseframe. The first point is at the position whose index is N_1, the last point is the position whose index is N_2. The array will be used to display the boundary on the screen. The whole procedure of determining the boundary is shown in Figure 3.12.

(10). By selecting the button "Define Bound" in the property "Check Plane" in **GAPCI**, the robot will measure the object surface boundary from step (1) to step (9). Figure 3.13 shows a table boundary measured by the Reis robot at the Robotic Adaptive Telesystem Laboratory, University of Victoria.

3.3.5 Display of the surface boundary on the screen

Besides displaying the boundary in the three projection coordinate frames shown in Figure 3.13, **GAPCI** also shows the boundary in two dimensions as seen in Figure 3.14 on the right lower canvas so that the operator can easily generate or modify the planar curves inside the boundary. To display the boundary on the screen as shown in Figure 3.13, the boundary points should be transformed from the baseframe into the screen coordinate frame with a shrinking matrix.

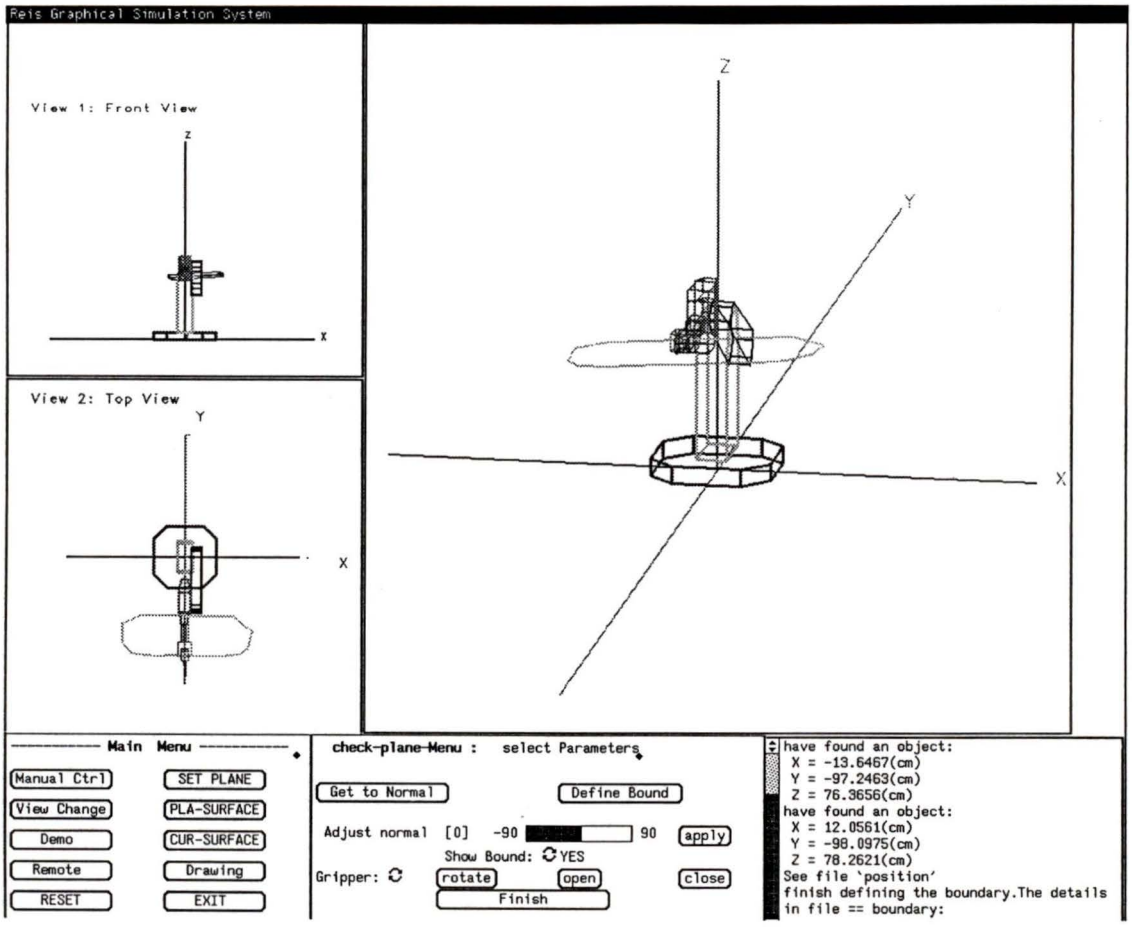

Figure 3.13: Display the boundary of the object surface

To display the boundary on the lower canvas shown in Figure 3.14, two coordinate transformation steps are used:

(1). Transforming the boundary points from the baseframe into the surface

coordinate frame by:

$$\begin{pmatrix} x_s \\ y_s \\ z_s \\ 1 \end{pmatrix} = [T_0^s]^{-1} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.18}$$

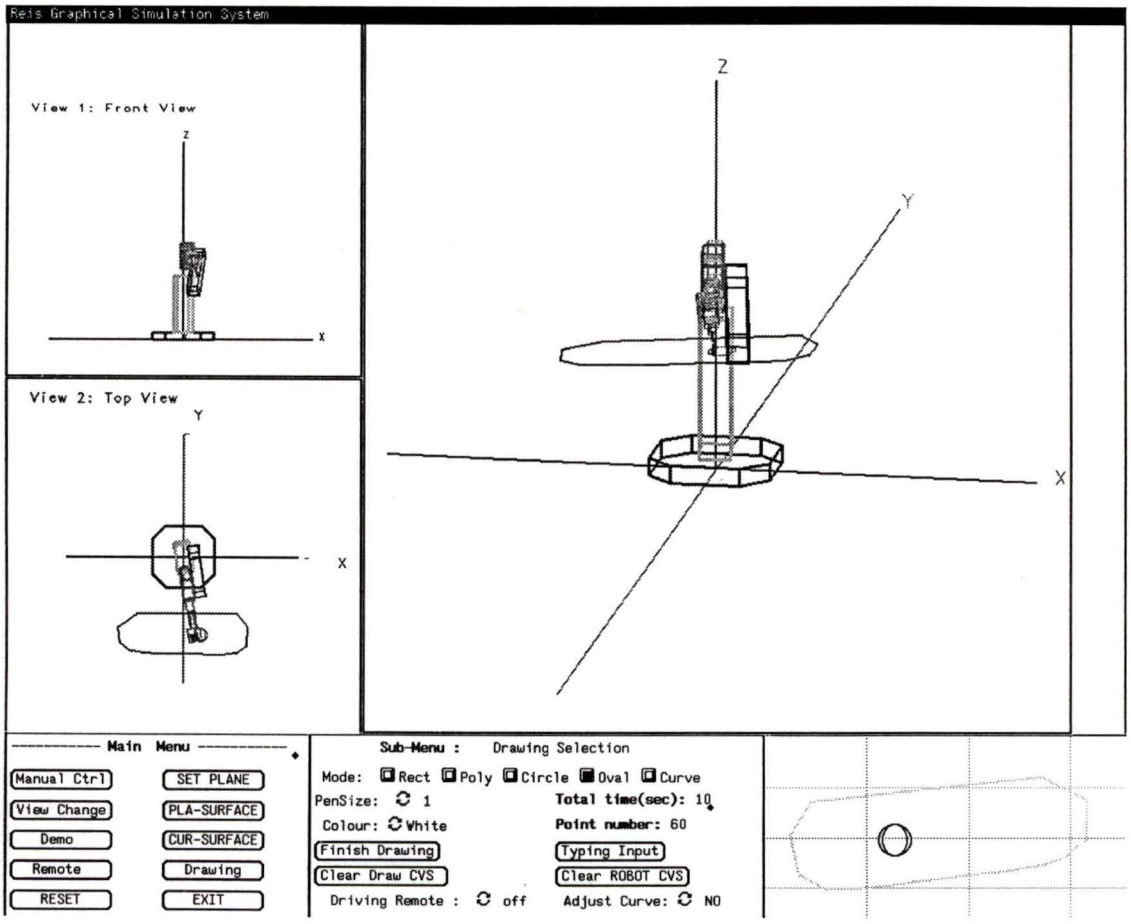

Figure 3.14: Generating curves inside the boundary

(2). Translating the boundary points from surface frame to the lower canvas frame. The lower canvas size and coordinate frame is shown in Figure 3.15.

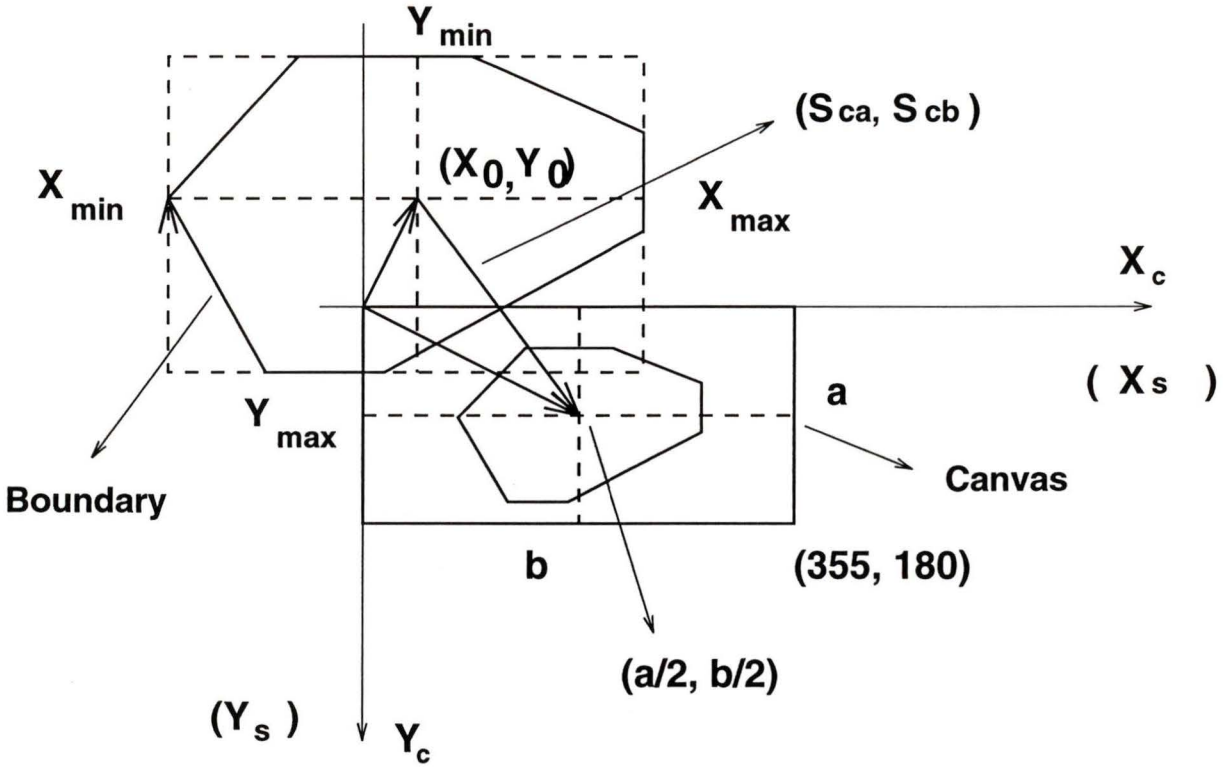

Figure 3.15: Expressing boundary in canvas frame

To show the boundary on this canvas and graphically generate curves on it, we should calculate the transformation matrix between the canvas coordinate $O-X_cY_cZ_c$ and the surface coordinate $O-X_sY_sZ_s$. The matrix only transforms the X, Y coordinate.

(a). If the X_s, Y_s should be expressed in the same unit as the X_c, Y_c, then the transformation matrix is:

$$T_1 = \begin{pmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The boundary can be expressed in the canvas frame by modifying the points with T_1, as shown in Figure 3.15.

(b). Find the X_{min}, X_{max} and Y_{min}, Y_{max} on the boundary and define the proportion constant between the canvas and the box frame which covers the boundary, as shown in Figure 3.15.

We let $a = 180$, $b = 355$ and select the two proportion values P_1, P_2 to satisfy the following results:

$$\begin{cases} P_1 \cdot |X_{min} - X_{max}| &= 3a/4 \\ P_2 \cdot |Y_{min} - Y_{max}| &= 3b/4 \end{cases}$$

That is:

$$\begin{cases} P_1 &= 3a/(4|X_{min} - X_{max}|) \\ P_2 &= 3b/(4|Y_{min} - Y_{max}|) \end{cases}$$

The proportion matrix is:

$$T_2 = \begin{pmatrix} P_1 & 0 & 0 \\ 0 & P_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(c). The center coordinate of the box frame which covers the boundary is:

$$\begin{cases} X_0 &= (X_{min} - X_{max})/2 \\ Y_0 &= (Y_{min} - Y_{max})/2 \end{cases}$$

To shift the boundary inside the lower canvas, the shifting components are:

$$\begin{cases} S_{ca} &= \frac{a}{2} - X_0 \\ S_{cb} &= \frac{b}{2} - Y_0 \end{cases}$$

The shifting matrix is:

$$T_3 = \begin{pmatrix} 1 & 0 & S_{ca} \\ 0 & 1 & S_{cb} \\ 0 & 0 & 1 \end{pmatrix}$$

(d). The final transformation matrix which projects the boundary inside the canvas is:

$$T_c^s = T_3 \cdot T_2 \cdot T_1 \tag{3.19}$$

The boundary is projected inside the lower canvas by means of modifying the points by $\mathbf{T_c^s}$, as shown in Figure 3.14.

3.4 Planar curves generation with screen graphical input

A graphical input property is provided in **GAPCI** so that the user can generate the curves such as circle, oval, rectangle, cubic curve and words inside the boundary, as shown in Figure 3.16

GAPCI provides the operator with two ways to generate the curves inside the boundary:

(1). Typing the geometric parameters for the curves by selecting the property "Typing Input" in Figure 3.16. If the drawing "Mode" is one out of "Rect", "Circle" and "Oval", the "Type-Input-Menu2" will show up so that the operator can input the desired parameters, as shown in Figure 3.17.

If the drawing "Mode" is one out of "Poly" and "Curve", the "Type-Input-Menu1" will be displayed in Figure 3.18.

In this drawing method, all of the curves are generated in the surface coordinate frame, then projected inside the canvas frame by the matrix $\mathbf{T_c^s}$. Because the origin of the surface coordinate frame is one out of three touching points, the origin for the

curves is always inside the boundary. That means the curves always can be generated inside the boundary by modifying the parameters if they are not proper.

(2). The operator can also use the mouse to assign geometric points for the curves. In this drawing method, the points on the curve need to be translated into the surface frame with $[\mathbf{T}_{\mathbf{c}}^{\mathbf{s}}]^{-1}$ since it was generated in the canvas coordinate frame.

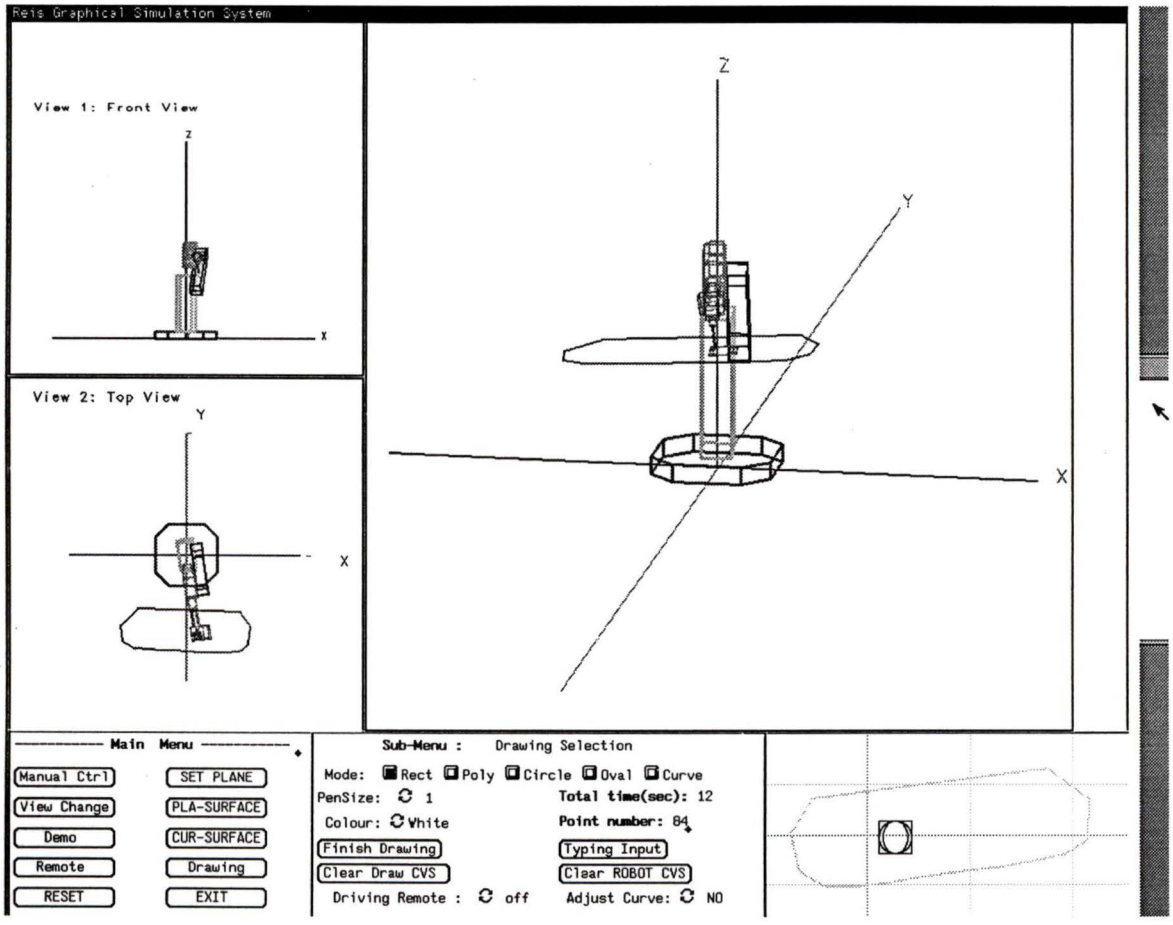

Figure 3.16: Curve generation with graphical input

3.5 Graphically processing the planar curves

After getting the geometric parameters for the curves, **GAPCI** gives some properties so as to help the operator check that the joint limits are satisfied and modify the curves inside the boundary.

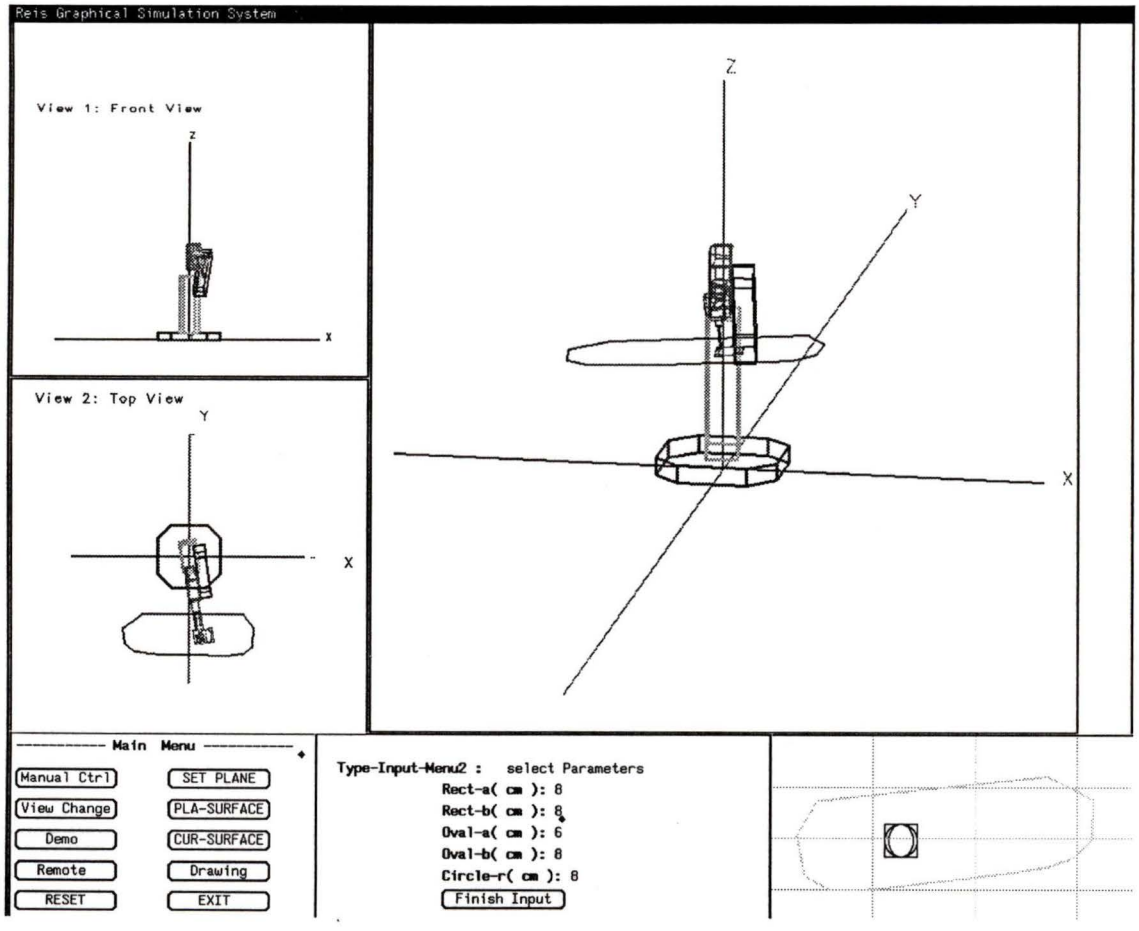

Figure 3.17: Input circle, rectangle and oval parameters

3.5.1 Animation of the drawing task

Because the robot end-effector is forced to be vertical to the object surface when the robot is drawing curves on the surface, some curves in a special positions will cause a robot joint to go over its limit. If the curves data are transported to the robot without checking the geometric limit, the robot may stop drawing or damage the surface. **GAPCI** gives the operator an animation property so that the operator can animate the drawing task with the graphical robot.

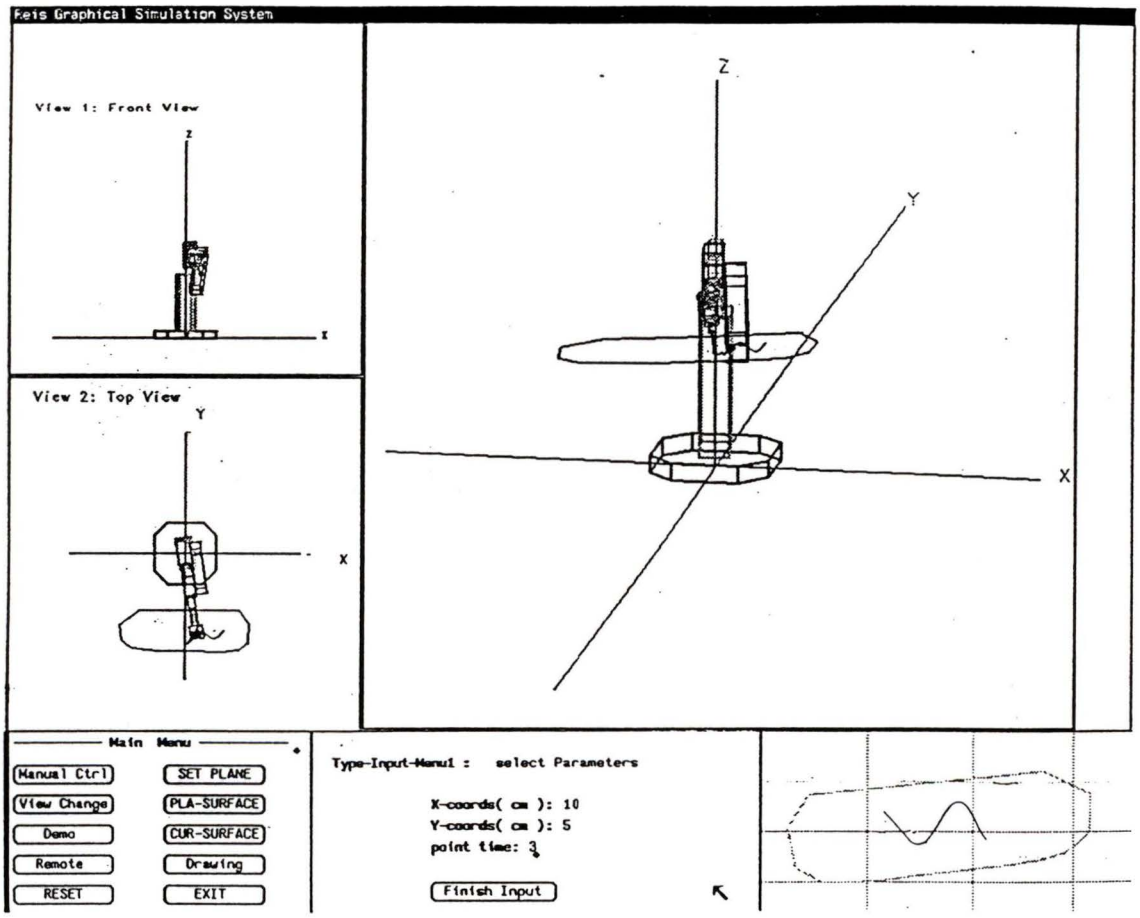

Figure 3.18: Input polyline, cubic curve parameters

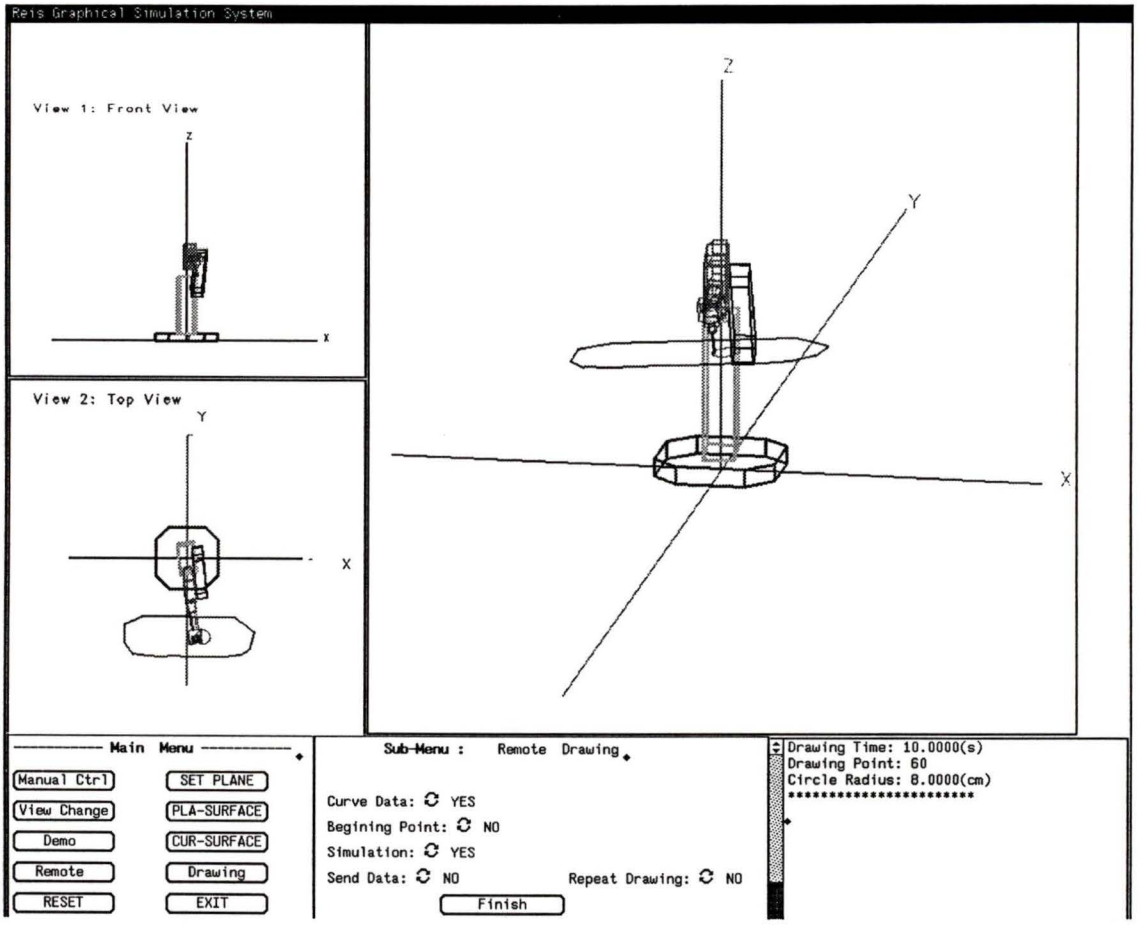

Figure 3.19: Simulation of the robot's drawing task

If the operator chooses property "Driving Remote: On" in panel "Drawing Selection" as shown in Figure 3.16, a panel called "Remote Drawing" will show up which is seen in Figure 3.19 so that the operator can chose "Simulation" to make the graphical robot draw the curves on the graphical surface. If the graphical robot can finish drawing the curves, the real robot can also draw the curves on the physical surface. Then the operator needs to chose "Beginning Point:Yes" to let the robot end-effector be vertical and go forward to the surface. When the tip of the pen held by the end-effector hits the surface, the robot will stop moving and start drawing

curves on the surface if the button "Send Data:Yes" is selected.

If the point is beyond the joint limit, the graphical robot will stop drawing the curve and then a message shown on the right lower window to remind the user what is wrong with the drawing procedure. In this situation, the operator can select "Adjust Curve:Yes" to modify the curves.

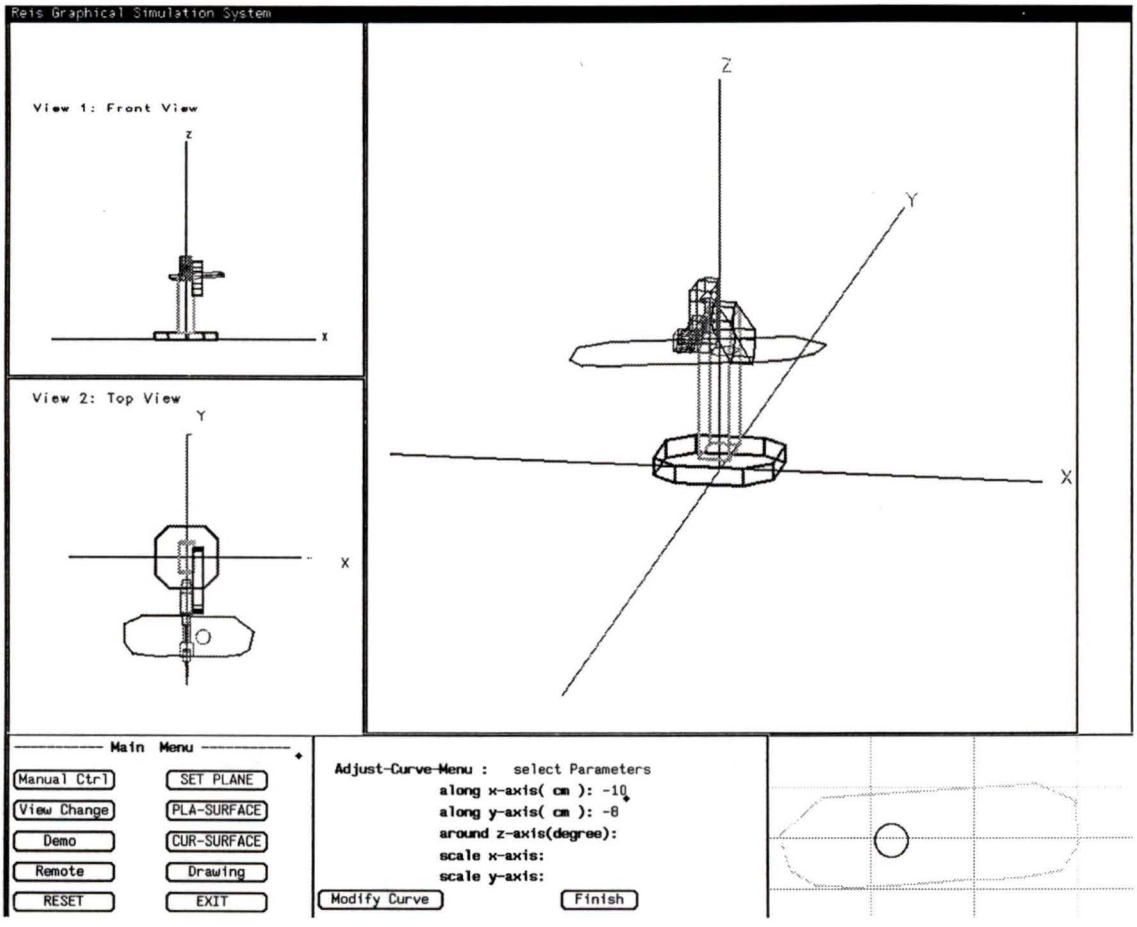

Figure 3.20: Modifying curves inside the boundary

3.5.2 Shifting, rotating and shrinking the planar curves

Once the property "Adjust Curve:Yes" is activated in Figure 3.16, the panel "Adjust-Curve-Menu" will be displayed as shown in Figure 3.20. With the help of this panel, the operator can shift, rotate and shrink or enlarge the planar curves inside the boundary. The satisfactory curves data will be sent to the path planning program for getting the joint space trajectory of the robot.

3.6 Summary

The algorithms are developed in this chapter to:

(1). define the path to let the robot detect an oriented object which has a planar surface

(2). measure the boundary of the object surface in the robot workspace

(3). graphically generate planar curves, such as circle, oval, rectangle, cubic curves and words

(4). convert task into the robot joint space so that the robot can draw them on the object surface.

Chapter 4

Measuring and modelling of a curvilinear surface

4.1 Introduction

To let the robot process a surface such as in boat body cleaning, polishing, grinding or drawing curves on the surface, etc., we need know the surface equation. A numerical method will be used to approximate the curvilinear surfaces with the reference points. Since the robot is used to collect the reference points on the surface, it is essential to use a simple and practical numerical method. Two conditions should be considered for selecting the approximation method. One is that the method for getting the reference points should be simple. The other condition is to allow the operator to easily generate the curves on the curvilinear surface according to the numerical points.

Based on the above conditions, the Beta2-Spline interpolation method is selected to model the curvilinear surface. The specific measuring method is designed for the robot to get the surface points for the interpolation method. A numerical algorithm

is presented to generate the curve on the curvilinear surface.

4.2 Beta2-Spline interpolation method

The Beta2-spline curve or surface is a special case of the Beta-spline curve and space. Its first derivative and curvature are continuous at the *joints* curves or *borders* for surfaces.

For a cubic Beta2-spline curve, the *ith* segment of it takes the form

$$\mathbf{c_i}(\beta_2, \mu) = \sum_{r=0}^{3} \mathbf{v_{i+r}} b_r(\beta_2, \mu) \quad (0 \leq \mu \leq 1) \tag{4.1}$$

The basis functions $b_r(\beta_2, \mu)$ are:

$$\begin{aligned}
b_0(\beta_2, \mu) &= 2\gamma(1-\mu)^3 \\
b_1(\beta_2, \mu) &= \gamma(\beta_2 + 8 + \mu^2(-3(\beta_2+4) + 2\mu(\beta_2+3))) \\
b_2(\beta_2, \mu) &= \gamma(2 + \mu(6 + \mu(3(\beta_2+2) - 2\mu(\beta_2+3)))) \\
b_3(\beta_2, \mu) &= 2\gamma\mu^3
\end{aligned}$$

where,

$$\gamma = \frac{1}{\beta_2 + 12} \tag{4.2}$$

For a cubic Beta2-spline surface, the *ith* segment of it takes the form

$$\mathbf{c_{i,j}}(\beta_2, \mu, \omega) = \sum_{r=0}^{3} \sum_{g=0}^{3} \mathbf{v_{i+r,j+g}} b_g(\beta_2, \mu) b_r(\beta_2, \omega) \quad (0 \leq \mu \leq 1, 0 \leq \omega \leq 1) \tag{4.3}$$

The parameter β_2 in the Beta2-spline curve and surface represents *tension*. Increasing the *tension* makes the curve or surface more polygonal. In the extreme case,

when β_2 approaches infinity, the Beta2-spline curve will coincide with its control polygon and the Beta2-spline surface will coincide with its control mesh. When β_2 is equal to zero, a Beta2-spline function will become a Beta-spline function. These features of the Beta2-spline function make it an attractive tool of curve and surface modelling.

4.3 Beta2-Spline interpolation problem

4.3.1 Beta2-Spline curve interpolation problem

Given n points $\mathbf{p_i}$ $(i = 1, 2, ..., n)$, find a piecewise cubic Beta2-spline curve $\mathbf{c_l}(\beta_2, \mu)$ $(l = 1, 2, ..., n-1)$ which passes through points $\mathbf{p_i}$.

This problem can be solved by finding control vertices $\mathbf{v_j}$ $(j = 0, 1, 2, ..., n+1)$ of the curve $\mathbf{c}$ and letting its joint points $\mathbf{r_i}$ coincide with $\mathbf{p_i}$. That is

$$\mathbf{r_i} = \mathbf{c_i}(\mu = 0) = \mathbf{p_i} \quad (i = 1, 2, ..., n-1)$$

and,

$$\mathbf{r_n} = \mathbf{c_{n-1}}(\mu = 1) = \mathbf{p_n} \quad (i = 1, 2, ..., n-1)$$

These relations are illustrated in Figure 4.1.

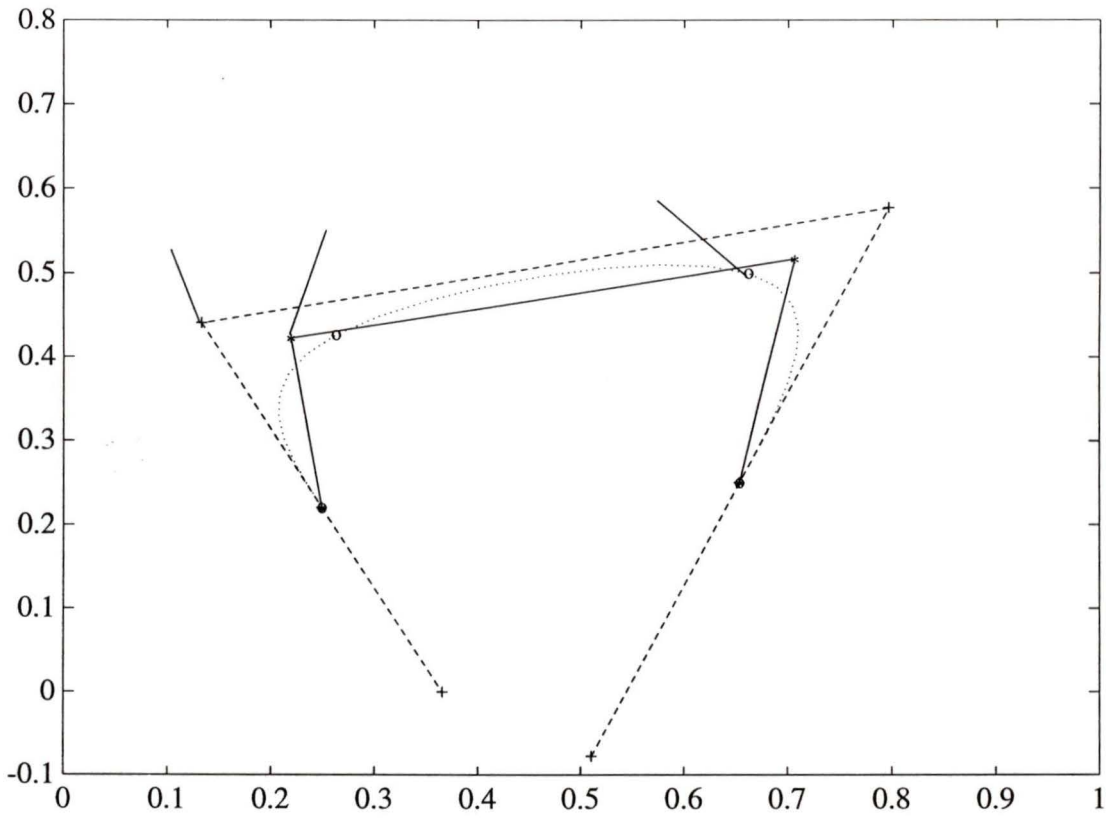

Figure 4.1: Beta2spline curve and its control joint points

4.3.2 Beta2-Spline surface interpolation problem

Given $n \times m$ points $\mathbf{p_{i,j}}$ (i=1,2,...,n, j=1,2,...,m), find a piecewise cubic Beta2-spline surface $\mathbf{s_{k,l}}(\beta_2, \mu, \omega)$ (k=1,2,...,n-1, l=1,2,...,m-1) which passes through points $\mathbf{p_{i,j}}$.

This problem can be solved by finding control mesh $\mathbf{v_{i,j}}$ ($i = 0, 1, 2, ..., n+1, j = 0, 1, 2, ..., m+1$) of the surface $\mathbf{s}$, and letting its border points $\mathbf{r_{i,j}}$ coincide with $\mathbf{p_{i,j}}$.

That is

$$\mathbf{r_{i,j}} = \mathbf{s_{i,j}}(\mu = 0, \omega = 0) = \mathbf{p_{i,j}} \quad (i = 1, 2, ..., n-1, j = 1, 2, ..., m-1)$$

and,

$$\begin{aligned}
\mathbf{r_{1,m}} &= \mathbf{s_{1,m-1}}(\mu = 0, \omega = 1) = \mathbf{p_{1,m}} \\
\mathbf{r_{n,1}} &= \mathbf{s_{n-1,1}}(\mu = 1, \omega = 0) = \mathbf{p_{n,1}} \\
\mathbf{r_{n,m}} &= \mathbf{s_{n-1,m-1}}(\mu = 1, \omega = 1) = \mathbf{p_{n,m}} \\
\mathbf{r_{n,j}} &= \mathbf{s_{n-1,j}}(\mu = 1, \omega = 0) = \mathbf{p_{n,j}} \\
\mathbf{r_{i,m}} &= \mathbf{s_{i,m}}(\mu = 0, \omega = 1) = \mathbf{p_{i,m}}
\end{aligned}$$

These relations are illustrated in Figure 4.2.

4.4 Interpolation error definition

4.4.1 The Beta2-spline curve

For the cubic Beta2-spline curve with n+2 control points $\mathbf{v_i}$ $(i = 0, 1, ..., n+1)$, the interpolation error of *kth* iteration $\mathbf{e^k}$ can be calculated parametrically as the difference between the original data vertices $\mathbf{p_i}$ $(i = 1, 2, ..., n)$ and the corresponding joint points $\mathbf{r_i^k}$ $(i = 1, 2, ..., n)$ of parametric Beta-Spline approximation curve of *kth* iteration:

$$\mathbf{e_i^k} = \mathbf{r_i^k} - \mathbf{p_i} \quad (1 \leq i \leq n) \tag{4.4}$$

According to the local support property of Beta2-spline,

$$\mathbf{r_i^k} = B_1 \mathbf{v_{i-1}^k} + B_2 \mathbf{v_i^k} + B_3 \mathbf{v_{i+1}^k} \quad (0 < i < n) \tag{4.5}$$

where,

$$B_1 = 2\gamma$$

$$B_2 = \gamma(\beta_2 + 8) = 1 - 4\gamma$$
$$B_3 = 2\gamma$$

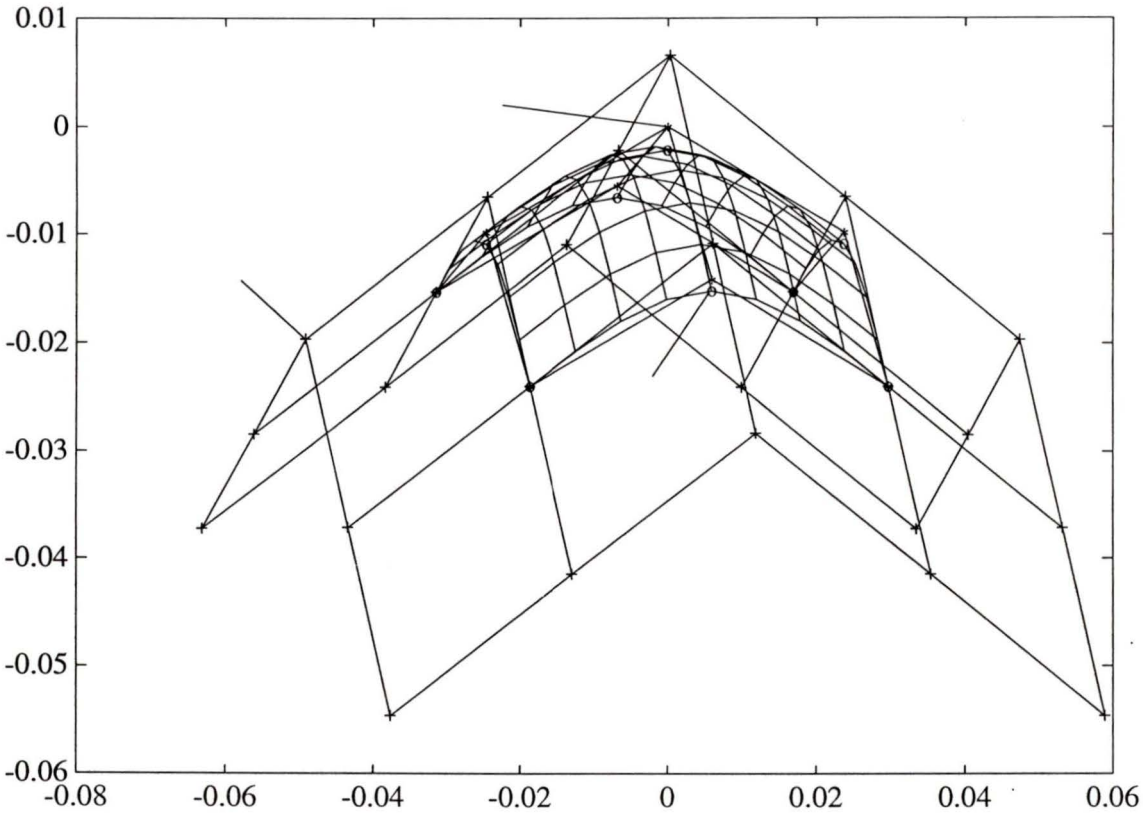

Figure 4.2: Beta2spline surface and its control joint points

Substituting B_1, B_2 and B_3 into equation (4.5):

$$\mathbf{r_i^k} = 2\gamma\mathbf{v_{i-1}^k} + (1 - 4\gamma)\mathbf{v_i^k} + 2\gamma\mathbf{v_{i+1}^k} \quad (1 < i < n)$$

At the end points, for non-periodical curve,

$$\begin{aligned} \mathbf{r_1^k} &= \mathbf{p_1} \\ \mathbf{r_n^k} &= \mathbf{p_n} \end{aligned} \tag{4.6}$$

4.4.2 The Beta2-spline surface

For Beta2-spline surface with $n \times m$ control points, the interpolation error of *kth* iteration $\mathbf{e^k}$ is also defined parametrically as the difference between the original data vertices $\mathbf{p_{i,j}}$, and the current parametric Beta2-spline approximation surface border points $\mathbf{r^k_{i,j}}$ of *kth* iteration:

$$\mathbf{e^k_{i,j}} = \mathbf{r^k_{i,j}} - \mathbf{p_{i,j}} \quad (1 \leq i \leq n, 1 \leq j \leq m) \tag{4.7}$$

For non-periodical surface, points $\mathbf{r^k_{i,j}}$ on the surface boundary are given by:

$$\begin{aligned} \mathbf{r_{1,j}} &= \mathbf{p_{1,j}} \quad 0 \leq j \leq m \\ \mathbf{r_{n,j}} &= \mathbf{p_{n,j}} \quad 0 \leq j \leq m \\ \mathbf{r_{i,m}} &= \mathbf{p_{i,m}} \quad 0 \leq i \leq n \\ \mathbf{r_{i,1}} &= \mathbf{p_{i,1}} \quad 0 \leq i \leq n \end{aligned} \tag{4.8}$$

Due to the local support property of $Beta2-spline$ surfaces, the other joint points of $\mathbf{r_{i,j}}$ on the current surface are:

$$\mathbf{r^k_{i,j}} = B_{i-1}\mathbf{w^k_{i-1,j}} + B_i\mathbf{w^k_{i,j}} + B_{i+1}\mathbf{w^k_{i+1,j}} \tag{4.9}$$

or,

$$\mathbf{r^k_{i,j}} = 2\gamma\mathbf{w^k_{i-1,j}} + (1-4\gamma)\mathbf{w^k_{i,j}}2\gamma\mathbf{w^k_{i+1,j}} \tag{4.10}$$

Where,

$$\begin{aligned} \mathbf{w^k_{i-1,j}} &= 2\gamma\mathbf{v^k_{i-1,j-1}} + (1-4\gamma)\mathbf{v^k_{i-1,j}} + 2\gamma\mathbf{v^k_{i-1,j+1}} \\ \mathbf{w^k_{i,j}} &= 2\gamma\mathbf{v^k_{i,j-1}} + (1-4\gamma)\mathbf{v^k_{i,j}} + 2\gamma\mathbf{v^k_{i,j+1}} \\ \mathbf{w^k_{i+1,j}} &= 2\gamma\mathbf{v^k_{i+1,j-1}} + (1-4\gamma)\mathbf{v^k_{i+1,j}} + 2\gamma\mathbf{v^k_{i+1,j+1}} \end{aligned}$$

Substituting $\mathbf{w^k_{i-1,j}}$, $\mathbf{w^k_{i,j}}$ and $\mathbf{w^k_{i+1,j}}$ into equation (4.10):

$$\begin{aligned} \mathbf{r^k_{i,j}} = {} & (1-4\gamma)^2\mathbf{v^k_{i,j}} + 4\gamma^2(\mathbf{v^k_{i-1,j-1}}\mathbf{v^k_{i-1,j+1}} + \mathbf{v^k_{i+1,j-1}} + \mathbf{v^k_{i+1,j+1}}) + \\ & 2\gamma(1-4\gamma)(\mathbf{v^k_{i-1,j}} + \mathbf{v^k_{i,j-1}} + \mathbf{v^k_{i,j+1}} + \mathbf{v^k_{i+1,j}}) \qquad (1 < i < n, 1 < j < m) \end{aligned}$$

4.5 Error-adjusting iterative algorithm

To calculate the control point $\mathbf{v_i}$ for a Beta2-spline curve or $\mathbf{v_{i,j}}$ for a Beta2-spline surface, an error-adjusting iterative algorithm is presented below. The calculation for the interpolation error is based on (4.4) for curve or (4.7) for surface until it gets into the defined range. Then the control points are also obtained.

(1). Input original data vertices $\mathbf{p}$ and tolerance ϵ.

(2). Store $\mathbf{p}$ in $\mathbf{v^0}$, the initial control polygon or control mesh. Let $k = 0$, and start iteration.

(3). Calculate the current curve joint points or border points $\mathbf{r^k}$ corresponding to $\mathbf{v^k}$.

(4). Compute the error between the original data vertices $\mathbf{p}$ and current $\mathbf{r^k}$.

$$\mathbf{e^k} = \mathbf{r^k} - \mathbf{p}$$

(5). Check if

$$||\mathbf{e^k}|| = max(||\mathbf{e_1^k}||, ||\mathbf{e_2^k}||, ..., ||\mathbf{e_n^k}||) < \epsilon$$

If it is true go to step (6); otherwise, set $\mathbf{v^{k+1}} = \mathbf{v^k} - \mathbf{e^k}$, go to step (3) for the next iteration.

(6). For Beta2-spline curve, in order to ensure end point condition equation (4.6), $\mathbf{v_0}$ and $\mathbf{v_{n+1}}$ are determined by $\mathbf{v_0} = \mathbf{2v_1} - \mathbf{v_2}, \mathbf{v_{n+1}} = \mathbf{2v_n} - \mathbf{v_{n-1}}$.

For Beta2-spline curve, in order to ensure boundary condition equation (4.8), end points of the control mesh can be arranged as:

$$\mathbf{v} = \begin{pmatrix} \mathbf{v_{0,0}} & \mathbf{v_{0,j}} & \mathbf{v_{0,m+1}} \\ \mathbf{v_{i,0}} & \acute{\mathbf{v}} & \mathbf{v_{i,m+1}} \\ \mathbf{v_{n+1,0}} & \mathbf{v_{n+1,j}} & \mathbf{v_{n+1,m+1}} \end{pmatrix}$$

Where,

$$\acute{\mathbf{v}} = \begin{pmatrix} \mathbf{p_{1,1}} & \cdots & \mathbf{p_{1,m}} \\ & \mathbf{v_{2,2}} \quad \cdots \quad \mathbf{v_{2,m-1}} & \\ \vdots & \vdots \qquad\qquad \vdots & \vdots \\ & \mathbf{v_{n-1,2}} \quad \cdots \quad \mathbf{v_{n-1,m-1}} & \\ \mathbf{p_{n,1}} & \cdots & \mathbf{p_{n,m}} \end{pmatrix}$$

$$\mathbf{v_{i,0}} = 2 \begin{pmatrix} \mathbf{p_{1,1}} \\ \mathbf{p_{2,1}} \\ \vdots \\ \mathbf{p_{n-1,1}} \\ \mathbf{p_{n,1}} \end{pmatrix} - \begin{pmatrix} \mathbf{p_{1,2}} \\ \mathbf{v_{2,2}} \\ \vdots \\ \mathbf{v_{n-1,2}} \\ \mathbf{p_{n,2}} \end{pmatrix}$$

$$\mathbf{v_{i,m+1}} = 2 \begin{pmatrix} \mathbf{p_{1,m}} \\ \mathbf{v_{2,m}} \\ \vdots \\ \mathbf{v_{n-1,m}} \\ \mathbf{p_{n,m}} \end{pmatrix} - \begin{pmatrix} \mathbf{p_{1,m-1}} \\ \mathbf{p_{2,m-1}} \\ \vdots \\ \mathbf{p_{n-1,m-1}} \\ \mathbf{p_{n,m-1}} \end{pmatrix}$$

$$\mathbf{v_{0,j}} = \left(2\mathbf{p_{1,1}} - \mathbf{p_{2,1}},\ 2\mathbf{p_{1,2}} - \mathbf{v_{2,2}},\ \cdots,\ 2\mathbf{p_{1,m-1}} - \mathbf{v_{2,m-1}},\ 2\mathbf{p_{1,m}} - \mathbf{p_{2,m}} \right)$$

$$\mathbf{v_{n+1,j}} = \left(2\mathbf{p_{n,1}} - \mathbf{p_{n-1,1}}, 2\mathbf{p_{n,2}} - \mathbf{v_{n-1,2}}, \cdots, 2\mathbf{p_{n,m-1}} - \mathbf{v_{n-1,m-1}}, 2\mathbf{p_{n,m}} - \mathbf{p_{n-1,m}} \right)$$

$$\mathbf{v_{0,0}} = \left(2\mathbf{v_{0,1}} - \mathbf{v_{0,2}} \right)$$

$$\mathbf{v_{0,m+1}} = \left(2\mathbf{v_{0,m}} - \mathbf{v_{0,m-1}} \right)$$

$$\mathbf{v_{n+1,0}} = \left(2\mathbf{v_{n,1}} - \mathbf{v_{n-1,2}} \right)$$

$$\mathbf{v_{n+1,m+1}} = \left(2\mathbf{v_{n+1,m}} - \mathbf{v_{n+1,m-1}} \right)$$

(7). Generate the Beta2-spline cubic curve 4.1 or surface 4.3 based on the obtained control polygon or control mesh.

4.6 Algorithm of measuring the curvilinear surface

The only condition for using the Beta2-Spline interpolation method is that the original surface points should be given in m columns or n rows and the number of points should be the same in each column or row. Therefore, the searching method has been designed to let the robot touch the surface with the same step along each column (or row) so as to get the same numbers of points.

4.6.1 The orientation of the robot end-effector

To measure the curvilinear surface, the elastic beam held by the robot gripper will touch the surface along defined paths. But as this defined path is usually designed in the Cartesian coordinate, it should be transformed into the robot joint space with the robot inverse kinematics. By using the inverse kinematics, the orientation of the robot end-effector should be given for each measuring point. Let us assume that the

object is put in the area formed by the third or fourth phase of the robot base frame, which is illustrated in Figure 4.3.

The orientation of the end-effector must be:

$$\mathbf{R}_{\mathbf{0}}^{\mathbf{e}} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix} \tag{4.11}$$

It means the elastic beam is always vertical to the ground while the robot is measuring the object surface.

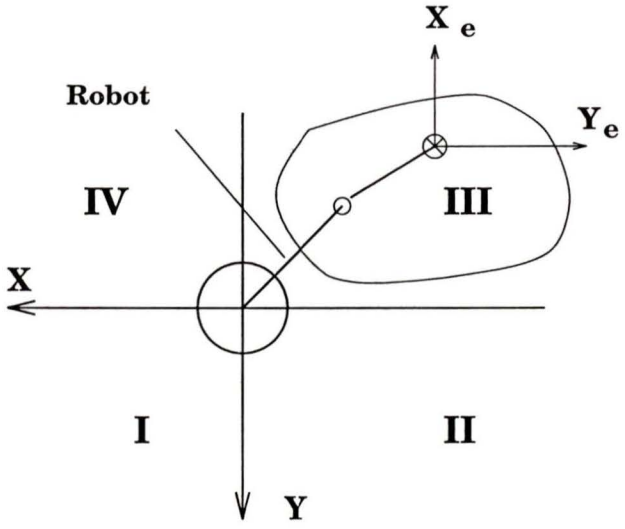

Figure 4.3: End-effector orientation for measuring the surface

4.6.2 Partitioning of the curvilinear surface

For the convex surface illustrated in Figure 4.4, the surface is partitioned into two parts. One is the *back area* which is behind the robot. The other one is the *front area* which is in front of the robot. First, the robot will measure the *back area*, then work on the *front area*.

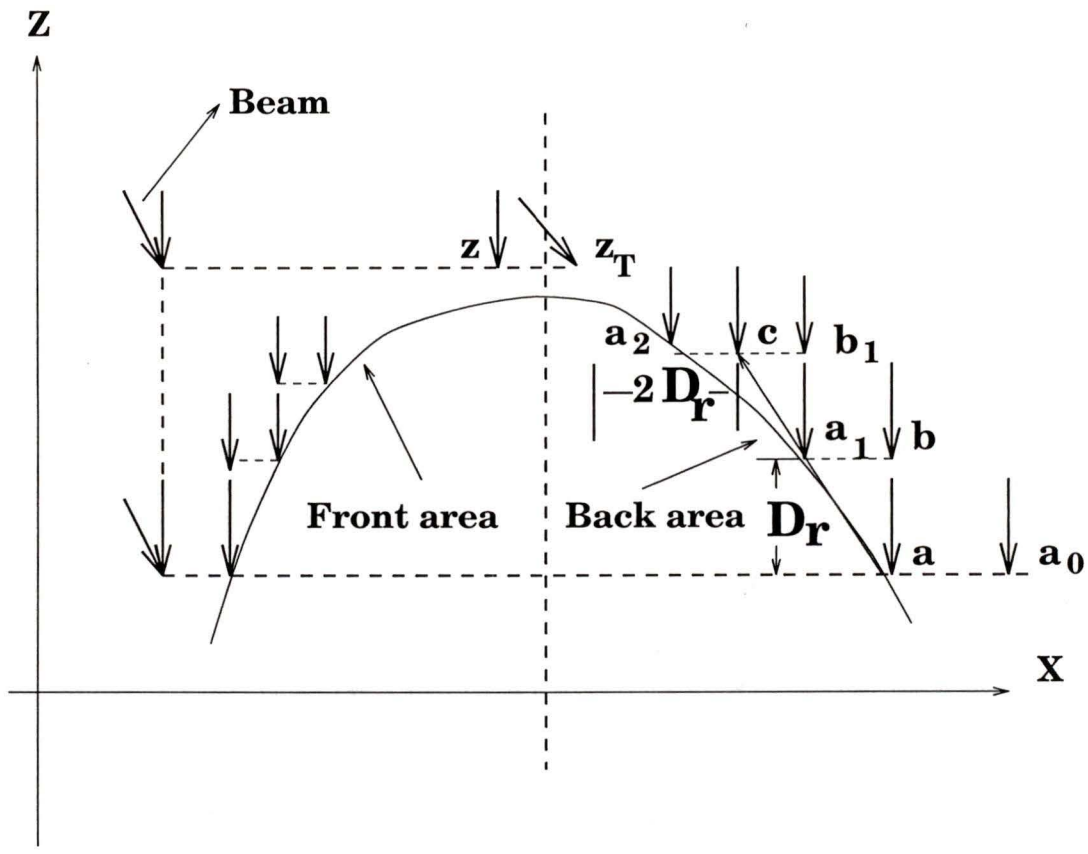

Figure 4.4: Measuring procedure for the convex surface

4.6.3 Set up of parameters for the measurement

To let the robot measure the whole surface column by column, an angle α is used to define the measuring direction, which is shown in Figure 4.5. The interval length between two columns and two rows is defined as D_c and D_r. M stands for the number of columns and N stands for the number of rows. M_c and N_r are used to record the current numbers of columns and rows. To indicate the measuring partition, the integer N_p is to be selected.

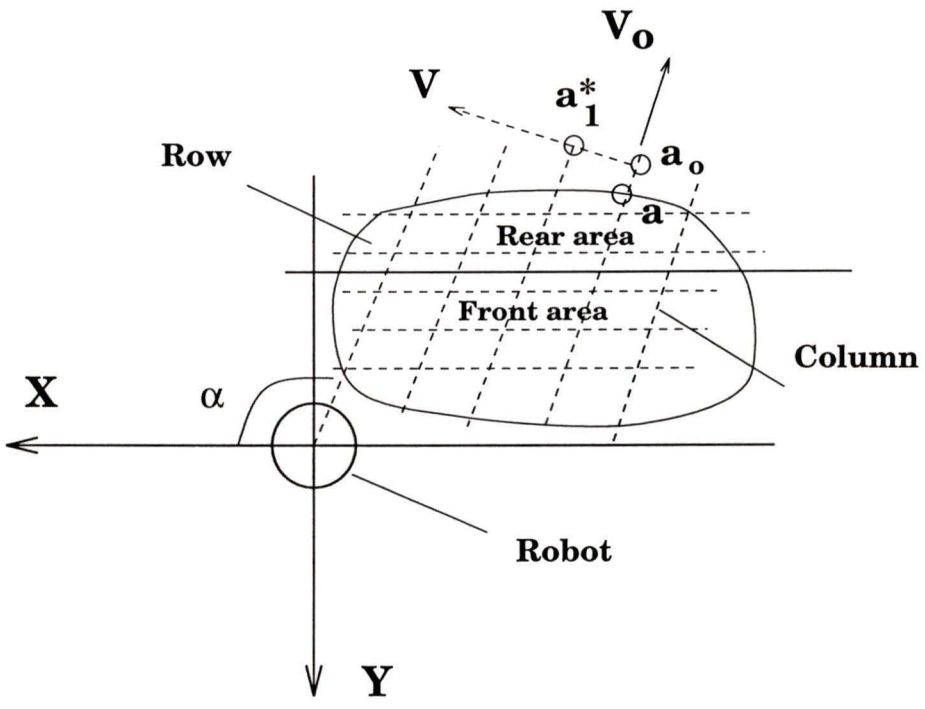

Figure 4.5: Parameter definition for measuring

4.6.4 Measuring the convex surface

1. Put the beam tip on the beginning point a_0 and let $N_p = 1, M_c = 1, N_r = 0$. Measure the back area on the surface first;

2. Use orientation (4.11) as the end-effector's current orientation, and calculate the joint angle by calling the inverse kinematics subroutine with the beam vertical to the ground;

3. Calculate the next beam tip (x, y) coordinate with the measuring direction vector $\mathbf{V} = (cos\alpha, sin\alpha)$ $(-180 < \alpha < 0)$ as illustrated in Figure 4.5 and the specific moving length. Only (x, y) coordinates vary in each measuring interval.

The measuring path vector is:

$$\mathbf{r} = \mathbf{r_0} + t \cdot \mathbf{V}$$

If $t < 0$, the beam goes forward to the robot and is measuring the back area. If $t > 0$, the beam goes away from the robot and is measuring the front area.

4. Calculate the joint angles by inverse kinematics, move the robot to the next point. If the beam can touch the surface, go to 5, otherwise, go to 3.

5. Write down the touching point as a in the data file with the index number $N_r = N_r + 1$. If $M_c = M$, mark the a as a_{end} which will be used as a reference point in the measuring procedure for the front area. Lift the beam along **Z**-axis with the length D_r to the position b, which is illustrated in Figure 4.4. Go to step 6.

6. If $N_r = 1$, move the beam forward to the surface as step 3 until the beam touches the surface. Then go to 5. If $N_r > 1$, calculate the pre-beginning point c by $\mathbf{c} = \mathbf{b} - \mathbf{a}$. Move the beam from b to c directly. Go to step 7.

7. Move the beam forward to the surface from point c. If the beam can touch the surface within the length $2D_r$ which is illustrated in Figure 4.4, go to 5; Otherwise, go to 8.

8. If $N_p = 1$, it is assumed that the beam is over the top of the surface. The position is named z as illustrated in Figure 4.4. Let the beam go back from z with the length D_r to the point z_T and rotate the beam around the Y_e axis with a specific angle β. The rotation can be done by changing the orientation

for the end-effector as:

$$R_0^e = RR_0^e \begin{pmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{pmatrix} \tag{4.12}$$

Go to 9 if $N_p = 0$, else go to 10.

9. Rotate the beam from point z_T onto the surface by changing robot joint 5 until the beam touches the surface again. Write the touching point into the data file and put the beam tip back to position z_T. Go to 10.

10. If $M_c \neq M$, the beam is put back to the point $a^* = (X^*, Y^*)$ which is the distance of D_r to point a and move to the *next column* beginning position $a_1^* = (X_1^*, Y_1^*)$ along the column direction with length D_c, which is illustrated in Figure 4.5.

$$\begin{aligned} (X_1^*, Y_1^*) &= (X^*, Y^*) + D_c V \\ V &= (-sin\alpha, cos\alpha) \end{aligned}$$

Let $M_c = M_c + 1$ and repeat step 2 to 10. If $M_c = M$, and $N_p = 1$, let $N_p = 0, M_c = 1$ and go to 11 for measuring the front area. Otherwise, the whole procedure is completed.

11. Let the beam be vertical to the ground at point z_T by giving the new value to the end-effector as (4.11).

12. Force the beam to go forward to the robot at position z_{fb} from which the distance is $L = (N - N_{fp})D_r$ to the point z_T. Let $N_{fp} = 1$.

13. Move the beam forward to the ground along the Z-axis with the current end-effector orientation until the beam touches the surface or the beam tip gets to point a_{end}. Go to step 2.

4.6.5 Arrangement of the touching points

After finishing the whole measuring procedure, all of the touching points are rearranged along the columns and rows and the lowest number of points on the column is selected as the standard row amount. The physical curvilinear surface is approximated by the Beta2-spline interpolation method with the rearranged points and specific β, tolerance ϵ and the interpolation number's μ and ω.

4.6.6 Implementation of the measuring procedure

The above-mentioned algorithm can be interactively performed on the Reis robot with the **GAPCI**

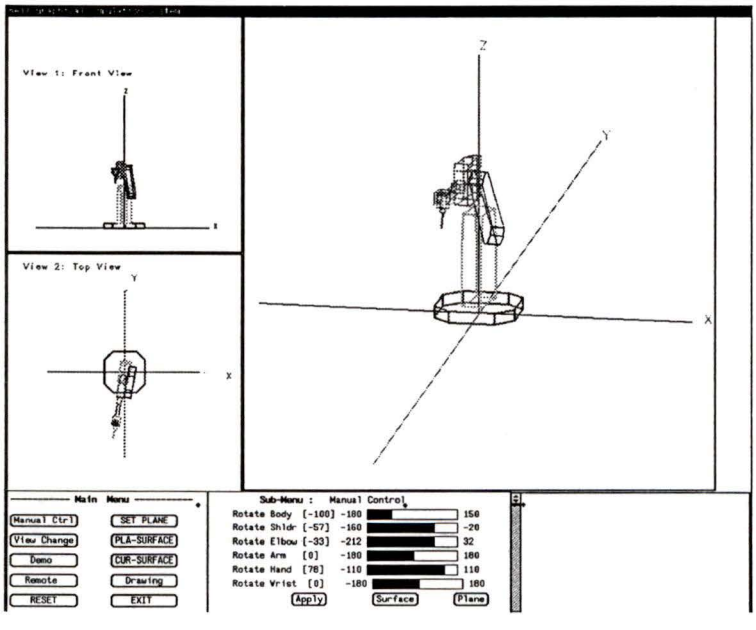

Figure 4.6: Setting up initial position for the measurement

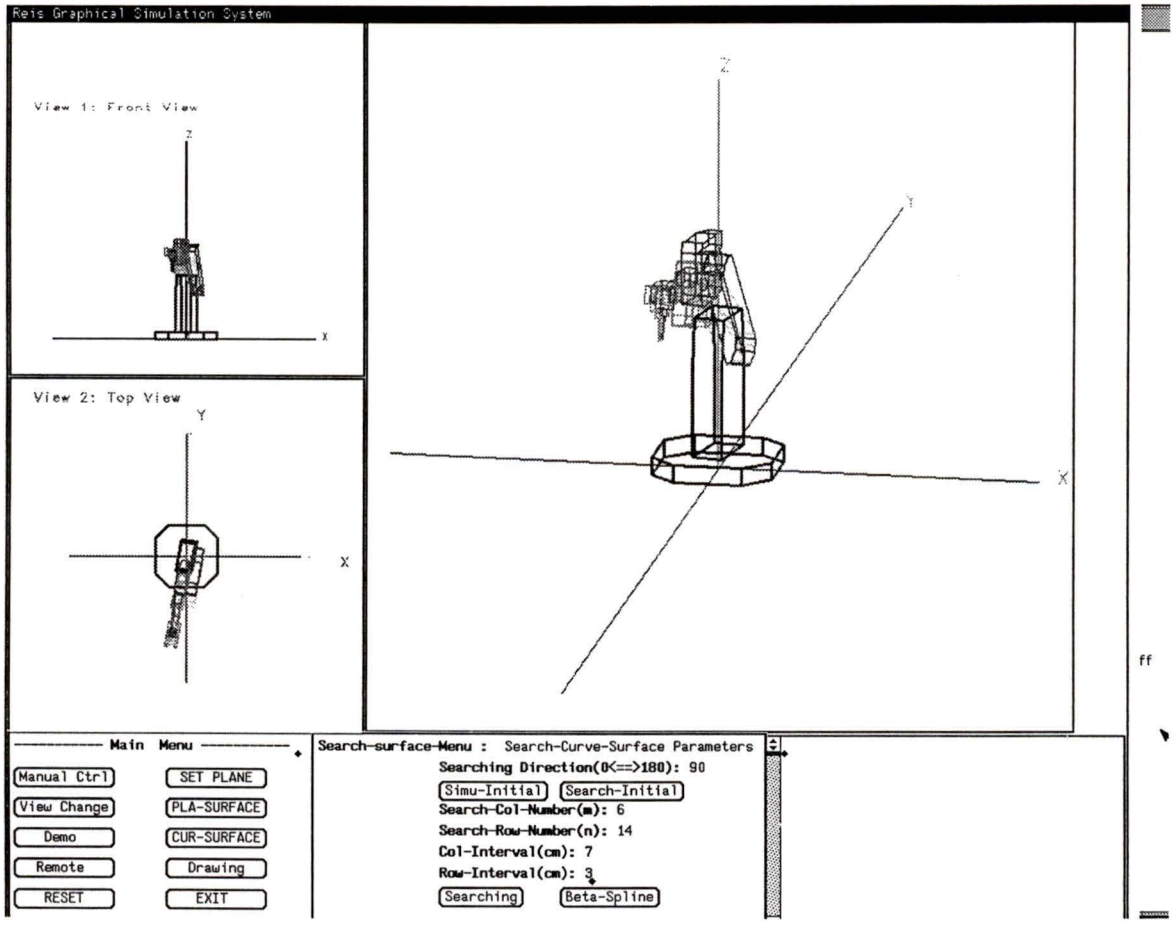

Figure 4.7: Setting up measuring parameters

1. Put the beam tip at the desired point on the "Manual Control" panel, illustrated in Figure 4.6;

2. select the parameters for the measuring procedure on the "Search-Curve-Menu" panel, illustrated in Figure 4.7;

3. select the parameters for using the Beta2-spline interpolation method on the "Surface -Generate-Menu" panel, as illustrated in Figure 4.8;

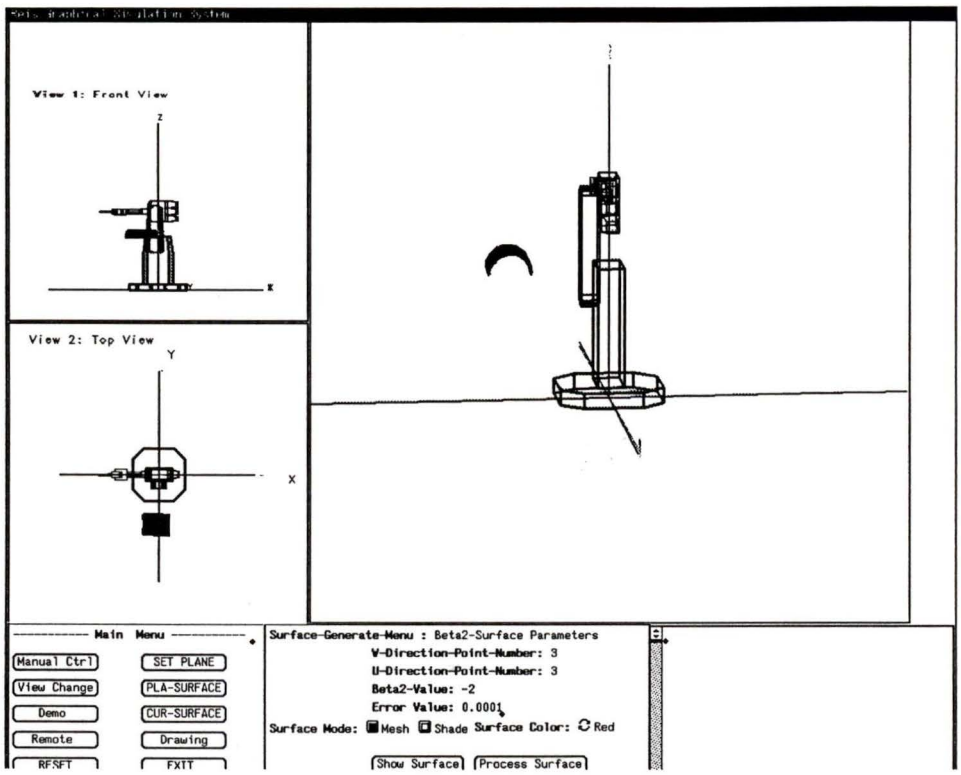

Figure 4.8: Inputting the parameters for Beta2-spline

By changing the parameters on the "Surface-Generate-Menu" panel for the Beta2-spline, a more precise surface may be obtained, which is illustrated in Figure 4.9;

4.7 Creating the curves on the surface with the discrete set of data

When we let the robot process the curvilinear surface such as when drawing a curve on it, the points on the surface curve should be known either by the

analytic curve equation or a numerical method.

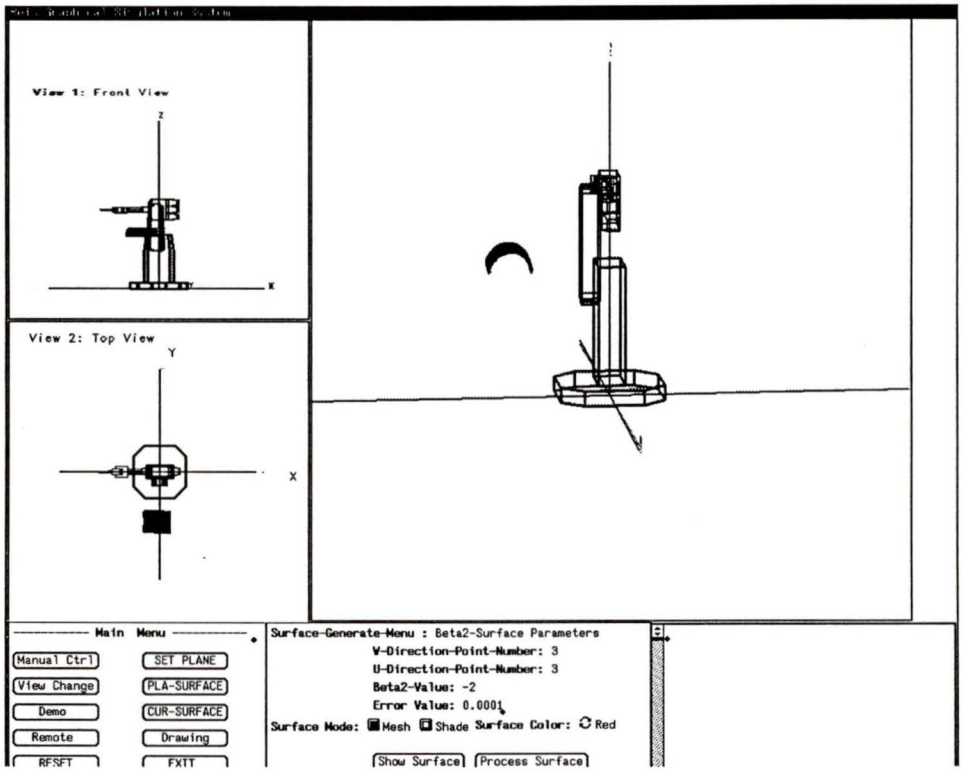

Figure 4.9: Interpolating more points on the surface

The surface curve in the present algorithm will be generated by the intersection of the surface and the plane which is perpendicular to the ground. It is hard to get an analytic expression of the intersection curve because the surface is approximated by means of a numerical method. The only way to get the point on the curve is to develop a numerical method. The idea is to define a plane which is through the z axis and vertical to the ground. The intersection of the surface and the plane is a curve on the surface. Our numerical algorithm can be explained with Figure 4.10.

1. The plane equation will be:

$$Ax + By = D \tag{4.13}$$

It is also the line equation of the projection of the plane in the XYcoordinate plane.

2. On each row(column), find the point from which the distance to the plane is the shortest between the other points on the row(column). Mark the point with the row(column) index number.

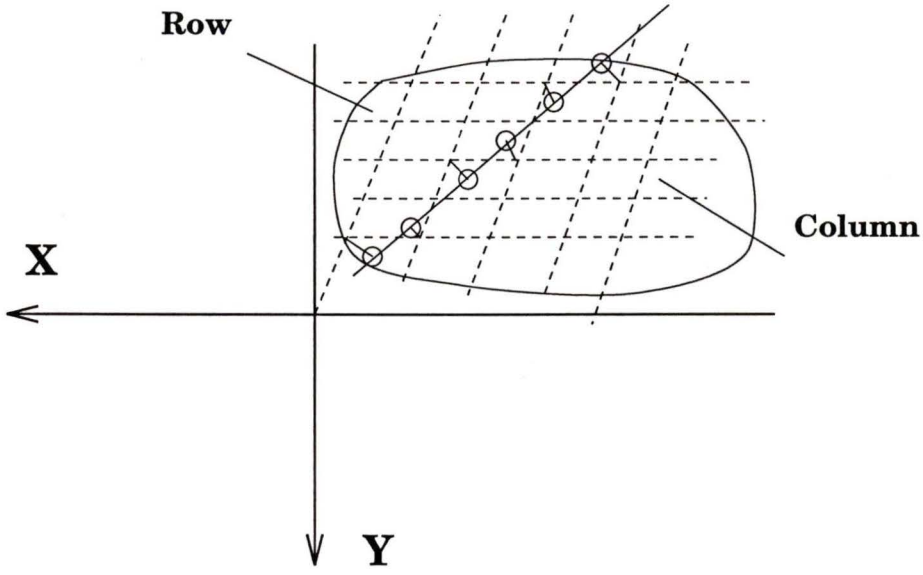

Figure 4.10: Calculating points for the intersection curve

3. If the points are assessed along the row, the desired point on the surface will be $S_{i,j}$ (i is the row index, j is the column index). Check if $S_{i,j}$ is on the plane by calculating the distance P_d:

$$P_d = \frac{|Ax_s + By_s - D|}{\sqrt{A^2 + B^2}} \tag{4.14}$$

If $P_d = 0$ or $P_d < 0.005$ (m), it is assumed that $S_{i,j}$ is on the plane, the point is recorded as $P_i = S_{i,j}$, then the next row is checked. Otherwise, find another surface point which is on the other side of the plane by calculating the value

$$f(x_s, y_s) = Ax_s + By_s - D$$

with the three points $S_{i,j-1}, S_{i,j}$ and $S_{i,j+1}$. Assuming $f(S_{i,j}) \cdot f(S_{i,j+1}) < 0$, the two points are on the different side of the plane.

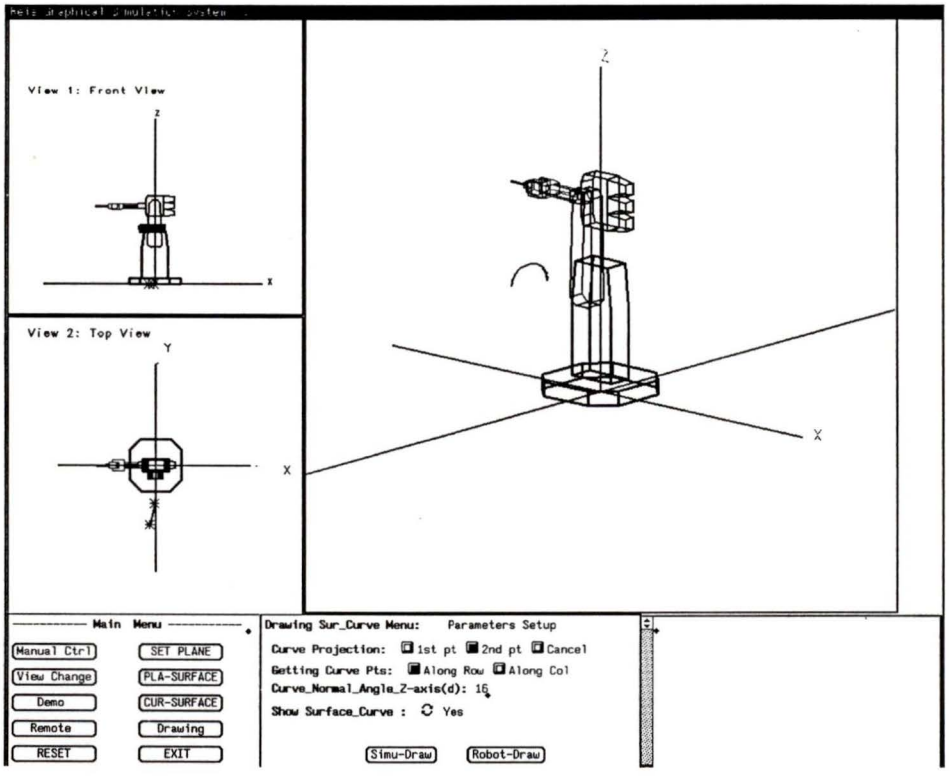

Figure 4.11: Creating the intersection curve on the surface

4. Connecting the two points $S_{i,j}, S_{i,j+1}$ produces line L_r:

$$\begin{cases} x &= S_x^{i+1} + t(S_x^i - S_x^{i+1}) \\ y &= S_y^{i+1} + t(S_y^i - S_y^{i+1}) \\ z &= S_z^{i+1} + t(S_z^i - S_z^{i+1}) \end{cases} \tag{4.15}$$

The intersection point between the plane and the line L_r is obtained by means of the calculation:

$$t = \frac{D - AS_x^{i+1} - BS_y^{i+1}}{A(S_x i - S_x^{i+1}) + B(S_y i - S_y^{i+1})}$$

Substituting t into (4.15), the intersection point P_i can be found.

5. Repeat step 2 to step 4 until all rows are completed. A set of points P_i (i=1,2,...,N) on the intersection curve will be obtained.

With the **GAPCI**, the operator can generate the plane and select the path of the points along the row or column on the screen via the panel "Drawing Sur_Curve Menu" illustrated in Figure 4.11.

4.8 Summary

The Beta2-Spline method is employed to model the curvilinear surface. A simple method is given to let the robot gather the points on the surface for the Beta2-Spline method. A algorithm is developed to generate the curves on the curvilinear surface according to the numerical points in order to process the surface along the curves.

Chapter 5

Processing the surface with force/position control

5.1 Introduction

Position control strategies are adequate for tasks such as transfer of materials, spot welding, spray painting and seam welding where the manipulator is not interacting significantly with objects in the environment. In these cases the interaction forces are negligible. However, tasks such as assembly, grinding, polishing and drawing etc., which involve extensive contact with the environment, are better handled by controlling the forces of interaction between the manipulator and the environment directly. For example, consider a manipulator cleaning a boat body with a mop, or writing with a felt tip marker. If the stiffness of the end-effector, tool, or environment is high, it becomes increasingly difficult to perform operations in which the manipulator contacts a surface with a pure position control scheme. For a rigid structure such as a robot, a slight position error could lead to extremely large forces of interaction with

disastrous consequences. The above applications are typical in that they involve both force control and trajectory control. It is clear that we should modify the position commands based on externally sensed force information. Therefore a force feedback control algorithm should accept force and motion commands, measure forces and positions, and produce motion commands for the manipulator.

Our objective is to let the Reis industrial robot track along an arbitrarily orientated planar or a curvilinear surface while maintaining a range of admissible contact force. For example, the robot draws curves on the planar surface with a point pen held by the gripper. Due to the uncertainties from the environment and the robot, we have to control both positions and contact force so as to avoid damaging the pen and the surface of the object. A force/position control algorithm, therefore, is desired. The two most popular proposed approaches are hybrid position/force control [19] [20] [21] and impedance control [1], [22] [23] [24] [25]. But most force control schemes require a dynamic model of the robot and the torque controlled actuators. This is not true in practice for two reasons. First, there are always uncertainties in modelling due to limitation in measurement techniques. Second, it is not correct to assume that any dynamic model is complete. Certain aspects of system behavior are not known and hence are not modeled. Due mainly to the above limitation, there had been few reports of practical implementation of impedance control on existing industrial manipulators until a new impedance control scheme was proposed by Dr.Yury Stepanenko and Dr. Glen Field in [1]. The paper [1] describes the implementation of impedance control using a model reference approach which requires neither a dynamic model nor torque controlled actuators. This special advantage facilitates the implementation of the impedance control scheme on existing industrial robot designs.

In this chapter, an algorithm is presented to generate the force/position reference trajectory for using the impedance control [1] in the robot tracking task on the object

surface.

5.2 Hybrid force/position control

Paul and Shimano [17] first investigated the problem of force/position control. Mason [18] presented the concepts of constraint frame which put the constraints into two categories: natural constraint and artificial constraints. Based on the concepts he presented a general control theory for compliant motion control. Based upon Mason's work, Raibert and Craig [19] presented a hybrid force/position control method which can design the control laws such as PID for position and force loops separately.

In the hybrid force/position control scheme, the task space is split into two subspaces: positions are manipulated and controlled along those directions in which it is impossible to apply an arbitrary force; forces are activated and controlled along the directions in which arbitrary motion is not possible. For hybrid control a **selection matrix**, S, is used to determine which DOF is to be position controlled, which is to be force controlled. The matrix S is diagonal with 1's on the diagonal entries corresponding to degrees of freedom in the compliance frame that are to be position controlled. Note then that the zeros on the diagonal entries in S correspond to the degrees of freedom that are to be force controlled. In this way each degree-of-freedom is uniquely specified as being either position controlled or force controlled. Though the hybrid force/position control scheme allows specification of both position and force reference signals, it causes some of the degree of freedom to be lost when it is under force control.

5.3 Impedance force/position control

Impedance control is a general approach to control in which the robot is under position control, and external force on the end-effector is related to the position and velocity of the robot through an impedance function. The advantage of this system is that it explicitly accounts for the dynamic interaction between the manipulator end-effector and the environment. Impedance control attempts to force the end-effector to behave as a multidimensional linear spring-mass-damper system whose parameters (inertia-damping-stiffness) can be specified arbitrarily. The goal of impedance control is to regulate the force of interaction which may vary due to uncertainties in the location of the point of contact and stiffness characteristics of both the robot and the environment. The impedance control scheme is not to follow a commanded force trajectory and any distinction between the force controlled subspace and position controlled subspace is ignored. In an impedance control system, the end-effector force can only be controlled indirectly, and a reference force cannot generally be specified.

5.3.1 The manipulator model in task space

Since a manipulator task specification, such as drawing a curve on an object surface, or cleaning a boat body, is typically given relative to the end-effector, it is natural to attempt to derive the control algorithm directly in the task space, rather than joint space or actuator space. This is even more important in force control than it is in position control due to the nature of the interaction of the manipulator with the environment.

The dynamics of an n d.o.f manipulator in joint space can be described by:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) + f(\dot{q}) = \tau \tag{5.1}$$

where τ is the $n \times 1$ vector of the joint torque supplied by the actuators; q is the joint displacement vector. H(q) is the $n \times n$ symmetric, positive definite mass matrix; $C(q, \dot{q})$, g(q) and $f(\dot{q})$ represent torques due to centrifugal, gravity, and friction forces, respectively.

If the manipulator is in contact with the environment then the dynamics equation (5.1) must be modified to include the actuator reaction torque $J^T F$ corresponding to the end-effector force F_e. Thus the dynamics equation of the manipulator becomes:

$$H(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + f(\dot{q}) = \tau - J^T F_e \tag{5.2}$$

where J is the $n \times 6$ configuration dependent Jacobian which relates the joint velocity to the linear and angular velocities of the end-effector; and F_e presents the force exerted by the end-effector on the environment and measured by a wrist force sensor. Now, we can rewrite these equations directly in terms of the end-effector coordinates as follows. Assuming $n = 6$, the relationship between the vector of end-effector coordinates X and the vector of joint coordinates **q** is given as:

$$X = L(q) \tag{5.3}$$

where the function L denotes the forward kinematic equations; X is a six-dimensional vector indicating the position and orientation of the manipulator end-effector. Taking the first and second derivatives of (5.3) yields:

$$\dot{X} = J(q)\dot{q} \tag{5.4}$$

$$\ddot{X} = J(q)\ddot{q} + \dot{J}\dot{q} \tag{5.5}$$

Substituting into (5.2), the robot dynamics equation in task space is given by [20]:

$$H_x(x)\ddot{x} + C_x(x,\dot{x})\dot{x} + g_x(x) + f_x(\dot{x}) = F - F_e \tag{5.6}$$

where,

$$\begin{aligned} F &= J^{-T}\tau \\ H_x &= J^{-T}HJ^{-1} \\ C_x(x,\dot{x}) &= J^{-T}(C - HJ^{-1}\dot{J})J^{-1} \\ g_x &= J^{-T}g \\ f_x(\dot{x}) &= J^{-T}f \end{aligned}$$

5.3.2 Conventional impedance control

Impedance control does not attempt to track motion or force trajectories but rather attempts to regulate the relationship between the velocity and force, **mechanical impedance**. It is solely a position control scheme, with adjustments made to react to contact forces. Positions are commanded, and impedances are adjusted to obtain the proper force response. Using the common mechanical/electrical analog that equates force with voltage and velocity with current the ratio of force to velocity(torque to angular velocity) is refer-ed to as the mechanical impedance of the system. In the frequency domain this is represented by:

$$\frac{F_e(s)}{V(s)} = Z(s) \tag{5.7}$$

Where $F_e(s)$, V(s), and Z(s) are, respectively, the force, velocity and impedance. In terms of the position X(s) we may write

$$\frac{F_e(s)}{X(s)} = sZ(s) \tag{5.8}$$

In the linear case a desired impedance might be specified as

$$sZ(s) = -(Ms^2 + Bs + K) \tag{5.9}$$

The transfer function relation (5.8) then corresponds to the differential equation

$$M\ddot{x} + B\dot{x} + Kx = -F_e \tag{5.10}$$

Equation (5.10) then represents the desired response of the manipulator. The constants M, B, and K represent desired inertia, damping, and stiffness, respectively, and it is the task of the impedance control system to produce the actual response of the system represented by (5.10).

In the impedance control, the desired dynamics at end-effector are usually specified by the second order dynamics:

$$M(\ddot{x} - \ddot{x^d}) + B(\dot{x} - \dot{x^d}) + K(x - x^d) = -F_e \tag{5.11}$$

where x^d is the virtual trajectory which often coincides with the desired trajectory when no contact occurs. Given the manipulator dynamics (5.6) the appropriate control to achieve the target dynamics (5.11) is given by

$$\begin{aligned} \tau &= F_e + C_x\dot{x} + g_x + f_x + H_x\{\ddot{x^d} - M^{-1}[B(\dot{x} - \dot{x^d}) \\ &\quad + K(x - x^d) + F_e]\} \end{aligned} \tag{5.12}$$

The conventional impedance control requires the dynamics model of the manipulator and torque controlled actuators, which are usually difficult to know in practice. Though some methods have been proposed to estimate the dynamic parameters of

the manipulator [24] [25], there are few reports of the practical implementation of impedance control on existing industrial manipulators. Fortunately, this problem has been solved in [1].

5.3.3 Model reference impedance control

Impedance control will be implemented using the model reference impedance control approach presented by Dr. Yury Stepanenko and Dr. Glen Field. The forces and torques measured by a wrist mounted force sensor are input to a six degree of freedom spring-mass-damper model. The output of the model represents how the path of the end effector is to be modified in the Cartesian end effector frame.

5.4 Force/Position tracking with impedance control

No matter what is used, the method described in 5.3.2 or the method of 5.3.3, usually we cannot track both a specific motion and reference forces. When the robot contacts the environment, the force $F_e \neq 0$ in the equation 5.11, thus the error $e = x - x^d \longrightarrow 0$ at the steady state can never occur. This means we can not follow the desired motion x^d. On the other hand, in the impedance control law (5.12), we do not consider any reference force F_r. If we intend to track both reference position and contact forces, we have to develop an algorithm to modify the desired trajectory for the impedance controller. Here, the concept of hybrid control is applied; that means we will split the task space into a force control subspace and a position control subspace. In some applications, it is easy and necessary to split the task space. As for our drawing task, we are only concerned with the x,y coordinates and

the normal contact force with respect to the object's surface frame. Thus the force control subspace can be defined along the surface normal direction.

5.4.1 Reference force tracking

To let the robot track reference forces while maintaining a desired impedance relationship between position error and external force, we present an algorithm to modify the desired trajectory for the impedance controller.

Let $\mathbf{p^r}$ be a given reference force input, and $\mathbf{x^r}$ be the referenced position trajectory. Suppose that $\mathbf{x^d}$ is calculated by

$$\mathbf{x^d} = \mathbf{x^r} + \mathbf{\Delta x^r} \tag{5.13}$$

where the purpose of using $\mathbf{\Delta x^r}$ is to modify $\mathbf{x^r}$ in such a way that the resulting $\mathbf{x^d}$ will drive the robot to produce a contact force $\mathbf{p} = \mathbf{p^r}$ in a steady state. $\mathbf{\Delta x^r}$ must be created on-line to enable $\mathbf{p}$ to converge to $\mathbf{p^r}$.

Now, we define the force error σ as $\sigma = \mathbf{p^r} - \mathbf{p}$, then the quadratic performance function ξ as $\xi = \frac{1}{2}\sigma^T\sigma$.

When a spring is used to connect a ball-pen to the robot gripper, we could assume that the environment we meet is a pure stiffness, that is

$$\mathbf{p} = \mathbf{K^e}(\mathbf{x} - \mathbf{x^e}) \tag{5.14}$$

where $\mathbf{K^e}$ is a diagonal stiffness matrix with non-negative constant terms, and $\mathbf{x^e}$ is the position of the ball-pen. It is obvious that the force will be zero if the end effector is not in contact with the environment.

Force tracking can be realized by minimizing σ w.r.t $\mathbf{\Delta x^r}$. It is shown in [21] that the force $\mathbf{p}$ in steady state can be written as

$$\mathbf{p} = \mathbf{K^{eq}}(\mathbf{x^d} - \mathbf{x^e}) \tag{5.15}$$

where $\mathbf{K^{eq}} \equiv \mathbf{K}(\mathbf{K} + \mathbf{K^e})^{-1}\mathbf{K^e}$. If we select proper parameters based on the methods in [1], the reference model of the mass-spring-damper for the robot will be in a steady state soon enough. Then we can use the steady state $\mathbf{p}$ (5.11) in the performance function ξ. Thus if we obtain ξ as a function of $\mathbf{\Delta x^r}$, we get

$$\begin{aligned} \xi(\mathbf{\Delta x^r}) &= \frac{1}{2}(\mathbf{p^r} - \mathbf{K^{eq}}(\mathbf{x^r} + \mathbf{\Delta x^r} - \mathbf{x^e}))^{\mathbf{T}} \\ &\quad (\mathbf{p^r} - \mathbf{K^{eq}}(\mathbf{x^r} + \mathbf{\Delta x^r} - \mathbf{x^e})) \end{aligned} \tag{5.16}$$

$\mathbf{F^e}$ will track $\mathbf{p^r}$ if $\mathbf{\Delta x^r}$ is appropriately selected to minimize $\xi(\mathbf{\Delta x^r})$ along the trajectory $\mathbf{x^r}$. Since $\xi(\mathbf{\Delta x^r})$ is quadratic, we can use a simple steepest descent algorithm to adjust $\Delta\dot{\mathbf{x}}^{\mathbf{r}}$, i.e. $\Delta\mathbf{x^r} = -\beta\eta$, where β is the step size and η can easily be shown to be

$$\eta_i = \frac{\delta\xi}{\delta\Delta x_i^r} = -k_i^{eq}\sigma_i \tag{5.17}$$

Finally, we can write

$$\Delta\dot{\mathbf{x}}^{\mathbf{r}} = \Gamma\sigma = \Gamma(\mathbf{p^r} - \mathbf{p}) \tag{5.18}$$

where Γ is an $n \times n$ constant gain matrix whose values must be non-negative. A zero value for the i_{th} diagonal element of Γ turns trajectory modification off in the i_{th} coordinate axis, so that $x_i^d = x_i^r$.

Upon the above conditions the steepest descent algorithm 5.18 will continuously force the performance function ξ toward zero if no zero elements are on the diagonal of Γ; consequently, the force error σ will approach zero in directions with a non-zero γ_i, i.e. $p_i \to p_i^r$. γ_i can be obtained by a one dimension search method.

5.4.2 Reference position tracking

When the contact force $\mathbf{F^e} \neq \mathbf{0}$, the position error $e_i = x_i - x_i^d = \frac{f_i}{K_i} \neq 0$ in the steady state due to equation (5.11). Therefore the impedance control does not track the desired trajectory. To track the desired motion, we give a simple method to modify the original reference trajectory in the position subspace. According to the equation (5.13) and (5.11), in the steady state, we have the following result

$$\begin{aligned} x_i - x_i^d &= x_i - (x_i^r + \Delta x_i^r) \\ &= \frac{f_i^e}{K_i} \end{aligned} \tag{5.19}$$

To let $x_i - x_i^r = 0$, we should select $\Delta x_i^r = -\frac{f_i^e}{K_i}$. The contact force f_i^e is measured by force sensor of the robot.

5.5 Path planning for reference trajectory

To let the robot track the desired curve on the planar or curvilinear surface with the impedance control, the tracking time and the velocity of the end-effector should be considered a desired reference trajectory. Here, we use *Linear Segments with Parabolic Blends*—**LSPB** and *Polynomial Function* to generate the smooth curve trajectory on the planar or curvilinear surface. The path planning method is shown as Figure 5.1

and Figure 5.2. In our approach, all of the trajectories are generated with respect to the Cartesian coordinate frame.

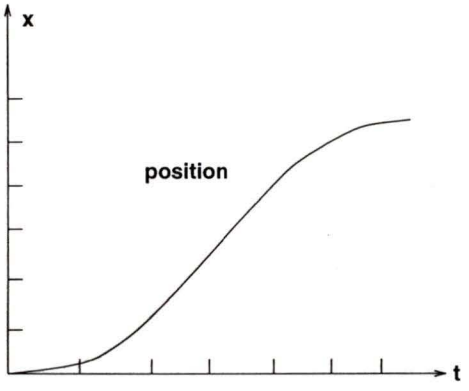

Figure 5.1: LSPB Trajectory

5.5.1 The transformation of the planar curves

For the planar surface, the trajectory of the curve is created in the surface coordinate frame first, then it will be transferred into the baseframe by the homogeneous matrix of the planar surface. Since the homogeneous matrix is known and the end-effector of the robot is supposed to be vertical to the surface as the robot tracks the curve on the surface, the orientation of the end-effector can be obtained by the planar surface orientation. The points on the trajectory, therefore, can be transferred into the robot joint space by the inverse kinematics.

We already know the homogeneous transformation matrix which is from the planar surface to the baseframe in the paragraph 3.2.3:

$$\mathbf{T}_{\mathbf{0}}^{\mathbf{p}} = \begin{pmatrix} \mathbf{R}_{\mathbf{0}}^{\mathbf{p}} & \mathbf{O_p} \\ \mathbf{0} & \mathbf{1} \end{pmatrix}$$

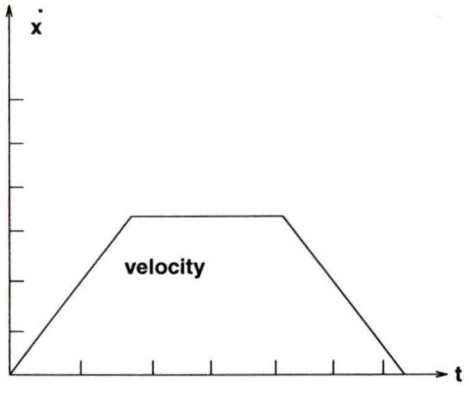

Figure 5.2: Velocity Profile for LSPB Trajectory

Since $\mathbf{T_0^p}$ is constant matrix, the translation results are:

$$\mathbf{r} = \mathbf{R_0^p} \cdot \mathbf{r_p} + \mathbf{O_p} \tag{5.20}$$

$$\dot{\mathbf{r}} = \mathbf{R_0^p} \cdot \dot{\mathbf{r_p}} \tag{5.21}$$

$\mathbf{r_p}$ is the vector for the point in the planar surface coordinate frame.

$\mathbf{r}$ is the vector for the point in the base coordinate frame.

$\dot{\mathbf{r}}$ and $\dot{\mathbf{r_p}}$ are the velocity vectors.

5.5.2 Trajectory generation of circle and oval

For a circle or an oval which is on the planar surface, the parameter equations can be expressed in the following format:

$$\begin{cases} x_p = R \cdot cos\theta \\ y_p = R \cdot sin\theta \\ z_p = 0 \end{cases} \tag{5.22}$$

R is the radius of the circle.

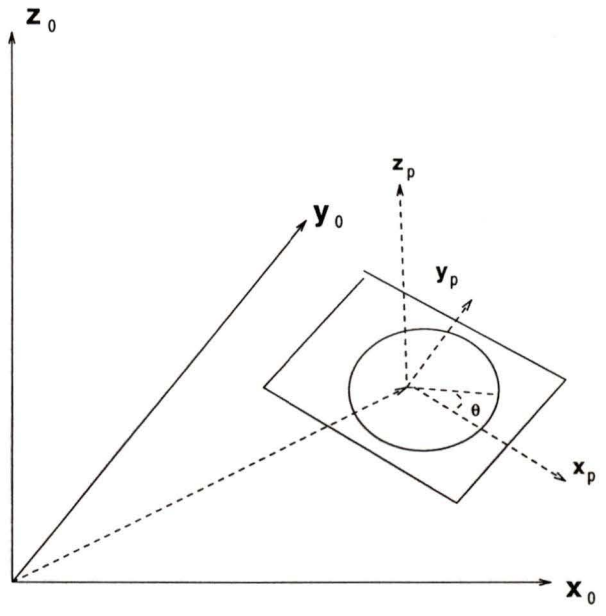

Figure 5.3: Circle Interpolation

$$\begin{cases} x_p &= a \cdot cos\theta \\ y_p &= b \cdot sin\theta \\ z_p &= 0 \end{cases} \tag{5.23}$$

a is the long and b the short axis for the oval.

The interpolations for the circle and oval are shown in Figure 5.3 and 5.4

The reference trajectory for the circle and oval can be obtained by **LSPB**. Let us suppose the robot finishes drawing the circle or oval in T seconds and the time T is divided into three parts:

- In the first interval(t=T/3), $\theta = \pi/2$
- In the second interval(t=2T/3), $\theta = 3\pi/2$

- In the third interval(t=T), $\theta = 2\pi$

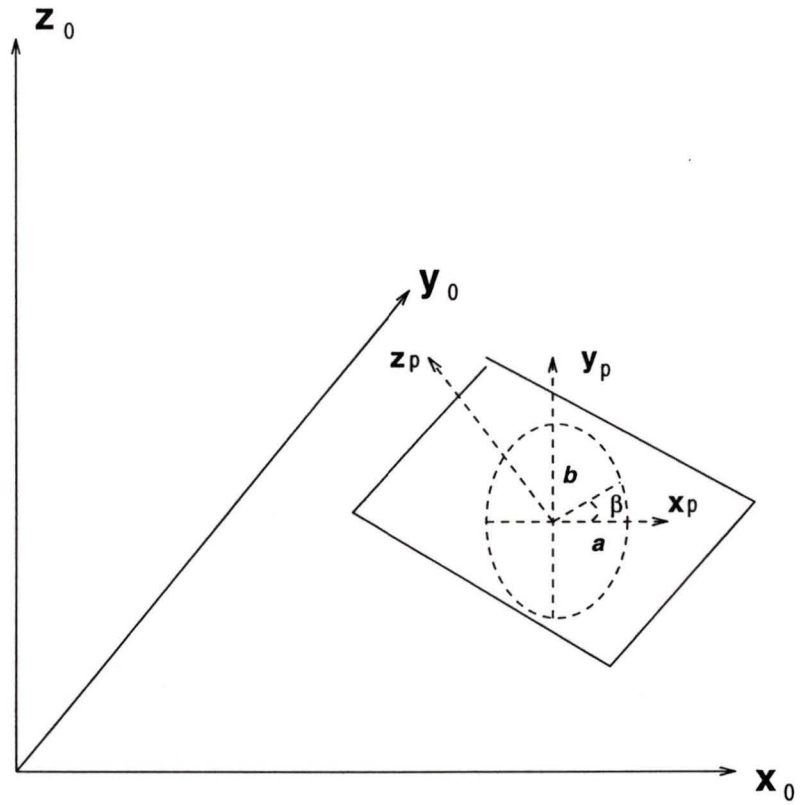

Figure 5.4: Oval Interpolation

Based on **LSPB**, the angular velocity for the θ can be expressed as:

$$\begin{cases} \dot{\theta} = At & (0 \le t \le T/3) \\ \dot{\theta} = B & (T/3 \le t \le 2T/3) \\ \dot{\theta} = -At + C & (2T/3 \le t \le T) \end{cases} \tag{5.24}$$

With above conditions, θ can be expressed in time t:

$$\begin{cases} \theta = 4.5\pi T^{-2}t^2 & (0 \le t \le T/3) \\ \theta = 3\pi T^{-1}t - 0.5\pi & (T/3 \le t \le 2T/3) \\ \theta = -4.5\pi T^{-2}t^2 + 9\pi T^{-1}t - 2.5\pi & (2T/3 \le t \le T) \end{cases} \tag{5.25}$$

5.5.3 Trajectory generation of rectangle and polyline

For rectangle or polyline on the planar surface, we can express the y coordinate with x, and express the x coordinate with time t by **LSPB** shown in Figure 5.5 and Fig 5.6, or express x with y and y with time t. For example, to express the side L_1 in Fig 5.5, we have the results:

$$\begin{cases} y = a \\ -b \leq x \leq b \\ t = T \end{cases}$$

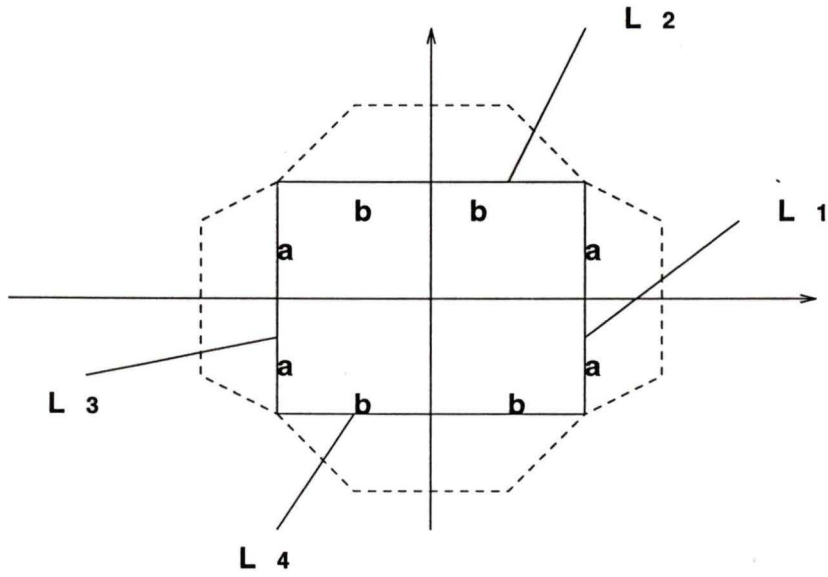

Figure 5.5: LSPB Velocity Profile for Rectangle

and the conditions

$$\begin{cases} x = b & t = 0 \\ x = 0.5b & t = T/3 \\ x = -0.5b & t = 2T/3 \\ x = -b & t = T \end{cases}$$

Therefore, the x coordinate can be expressed in t by **LSPB**:

$$\begin{cases} x = -4.5bT^{-2}t^2 + b & 0 \le t \le T/3 \\ x = -3bT^{-1}t + 1.5b & T/3 \le t \le 2T/3 \\ x = 4.5bT^{-2}t^2 - 9bT^{-1}t + 3.5b & 2T/3 \le t \le T \end{cases}$$

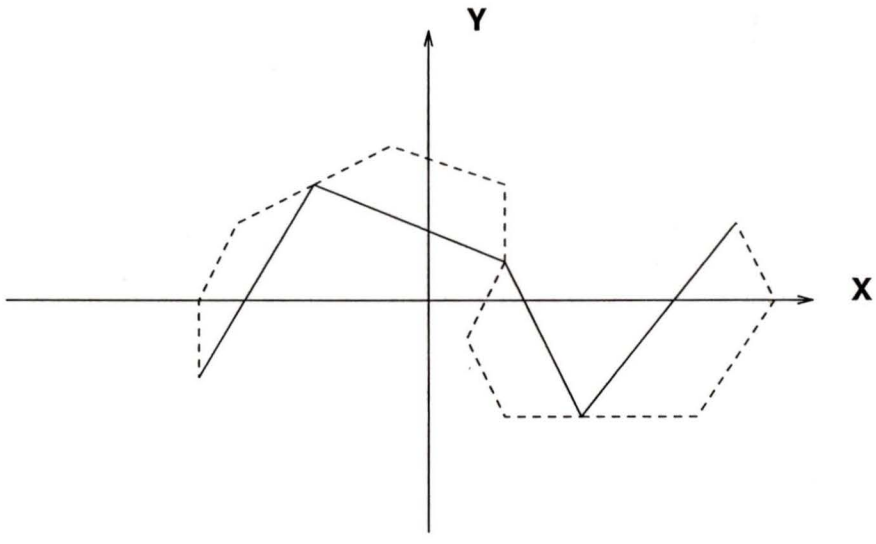

Figure 5.6: LSPB Velocity Profile for Polyline

5.5.4 Trajectory generation of a cubic curve

To generate a smooth curve on the planar surface upon the defined geometric points and other conditions such as drawing time, velocity, etc., we use the *Cubic Polynomial Function.* First, we express coordinate y in x :

$$\mathbf{y = b_0 + b_1 \cdot (x - x_0) + b_2 \cdot (x - x_0)^2 + b_3 \cdot (x - x_0)^3}$$

Here, we define the following conditions for getting b_0, b_1, b_2, b_3:

$$\begin{aligned} y_x(x_0) &= 0 \\ y_x(x_f) &= 0 \\ y(x_0) &= Y_1 \\ y(x_f) &= Y_2 \end{aligned}$$

Then we still use cubic polynomial function to express x coordinate in time t.

$$\mathbf{x = a_0 + a_1 \cdot (t - t_0) + a_2 \cdot (t - t_0)^2 + a_3 \cdot (t - t_0)^3}$$

Here, we define the following conditions for getting a_0, a_1, a_2, a_3:

$$\begin{aligned} x_t(t_0) &= 0 \\ x_t(t_f) &= 0 \\ x(t_0) &= X_1 \\ x(t_f) &= X_2 \end{aligned}$$

where, the coefficients for the above cubic polynomial functions are:

$$\begin{aligned} a_0 &= X_1 \\ a_1 &= 0 \\ a_2 &= 3(X_2 - X_1)/(t_f - t_0)^2 \\ a_3 &= 2(X_1 - X_2)/(t_f - t_0)^3 \end{aligned}$$

$$\begin{aligned} b_0 &= Y_1 \\ b_1 &= 0 \\ b_2 &= 3(Y_2 - Y_1)/(x_f - x_0)^2 \\ b_3 &= 2(Y_1 - Y_2)/(x_f - x_0)^3 \end{aligned}$$

In our Graphics Animation & Path Creation Interface(**GAPCI**), we select two ways to input the data for generating the smooth cubic curves shown in Figure 3.17 and 3.18:

(1). For geometric condition, we assume at the controlling point, the end-effector's velocity to be equal to zero:

$$(X_1, Y_1, t_1), (X_2, Y_2, t_2), ..., (X_n, Y_n, t_n)$$

(2). Position and average time T, we still assume at the controlling point, the end-effector's velocity to be equal to zero:

$$(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)$$

$$\delta t = T/(n-1) = t_i$$

5.5.5 The trajectory interpolation of the curvilinear curves

In the section 4.7, we already derived the algorithm to get the points for forming the curve on the curvilinear surface. These points are in the baseframe. We use *Cubic polynomial Function* to connect these points and get the trajectory for the surface curves as we did in the last section.

5.6 Summary

An algorithm is presented to generate the force/position reference trajectory for using the impedance control in the robot tracking task on the object surface.

Chapter 6

Experimental apparatus

6.1 Graphics animation & path creation interface

The Graphics Animation & Path Creation Interface is developed with SunPhigs in the Xwindows environment running on UNIX, shown as Figure 6.1.

SunPhigs is a system which combines modelling with two-dimensional and three-dimensional computer graphics. It gives application programmers the ability to construct models which have a complex logical structure, and then to create pictures of models with which users can interact.

SunPhigs has the properties for real-time manipulation of pictures and off-line storage of pictures and models. Besides, it is implemented as a function package which can be called by conventional programming languages such as C and Fortran. These advantages allow us to build a graphics user interface and connect it to the existing C code programs.

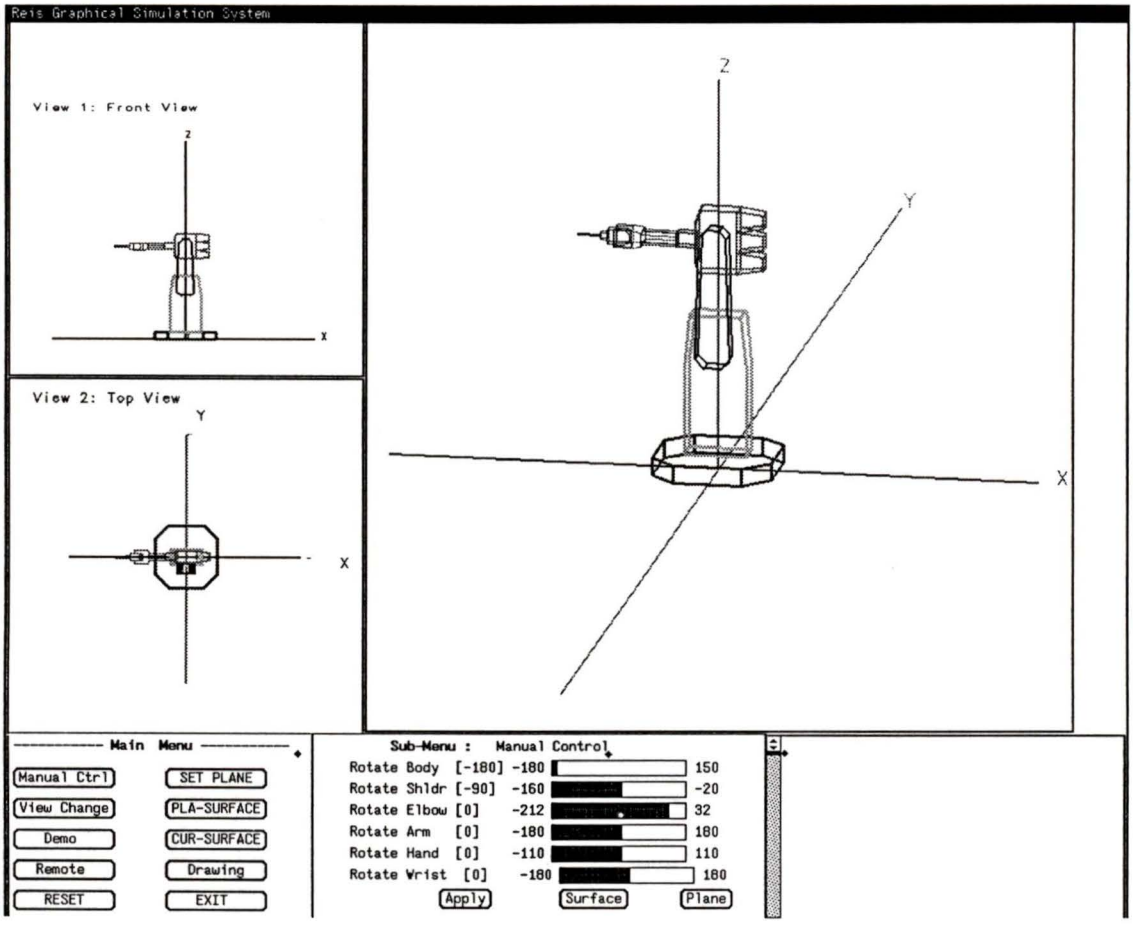

Figure 6.1: Graphics Animation & Path Creation Interface

The development of **GAPCI** was motivated by the desire to provide the operator with a simple graphical means of specifying the robot trajectory and with an interactive way of controlling the robot from a remote location. The **GAPCI** consists of the following properties:

- Animating the motion of industrial robots using three dimensional computer graphics. Interpreting the user's commands and displaying various motions of the robot, which are shown in the previous chapters. Changing the graphics

view as shown in Figure 6.2.

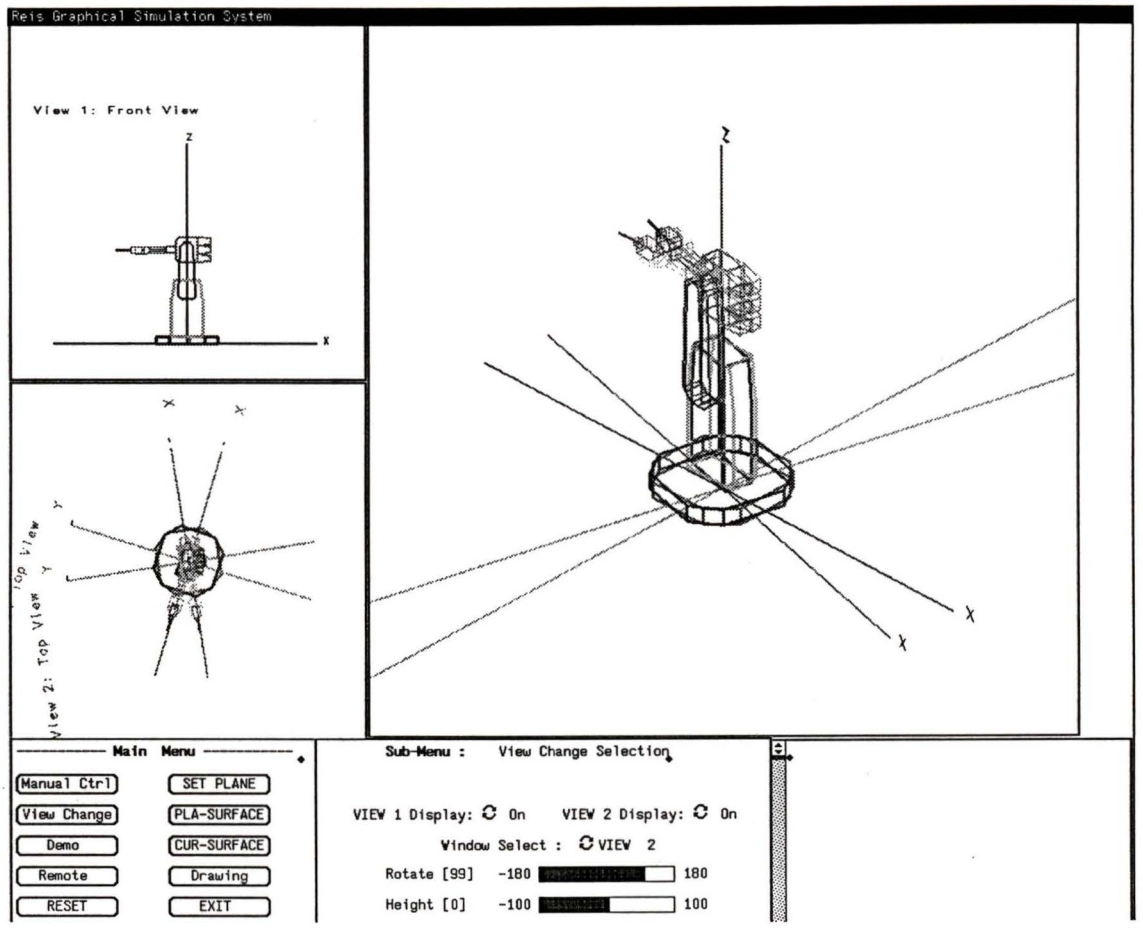

Figure 6.2: Changing view of the robot

- Automatically searching, measuring and identifying objects in the robot working area.
- Displaying the object surface, graphically generating and modifying(rotating, shifting,shrinking and enlarging) the curves drawn by the robot on the surface.
- Exchanging the data from both sides of the robot and the **GAPCI** by the

computer network communication on the Ethernet. This property is optional as shown in Figure 6.3.

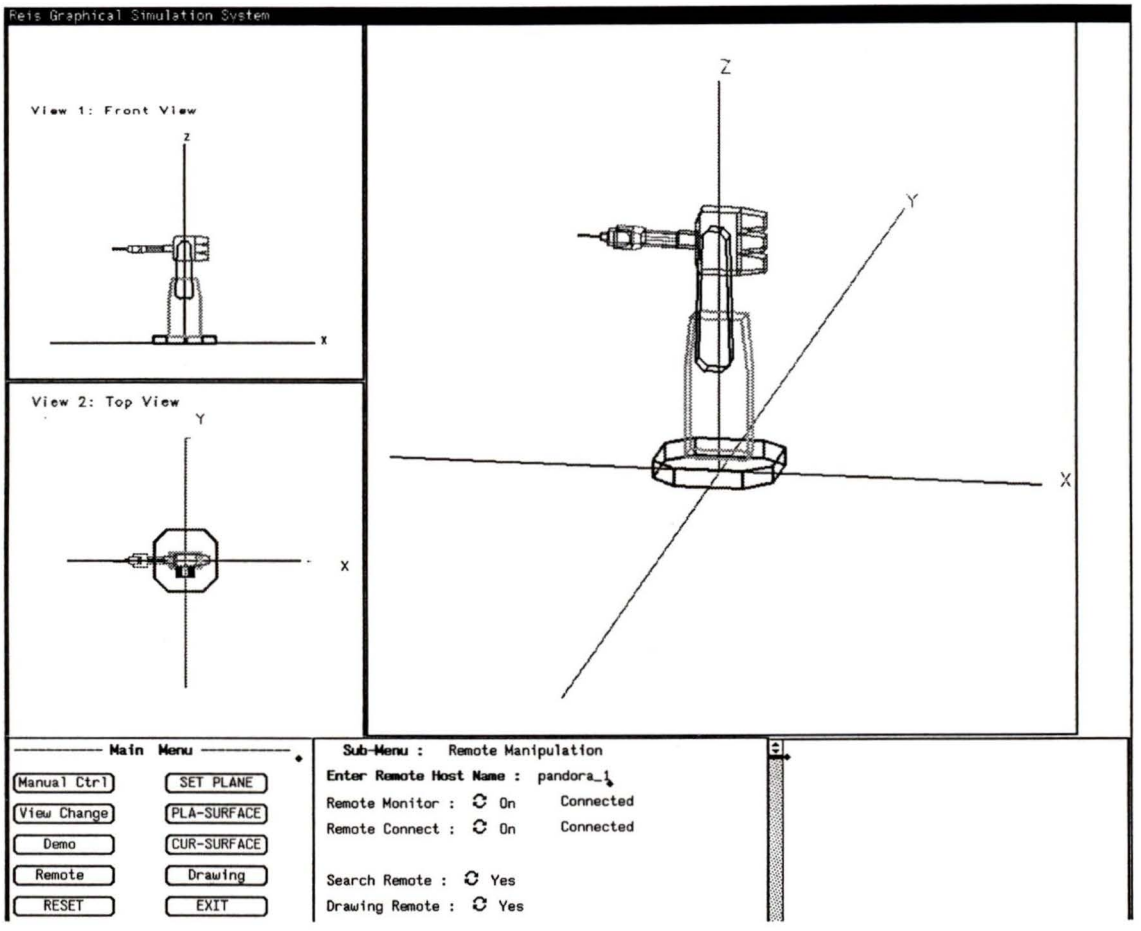

Figure 6.3: Connecting GAPCI to the robot

- Automatically programming the robot actuators' motion, which provides a desired end-effector path along the object surface.
- Generating the nominal control input for the robotic impedance controller, which provides the desired force control along the processing trajectory. In

particular, this software allows the automation of such important industrial operations as processing (grinding, polishing, cleaning, etc.) of curvilinear surfaces with required and controlled force interaction between the processing tool and the surface.

6.2 The Robot graphics animation & teleoperation system

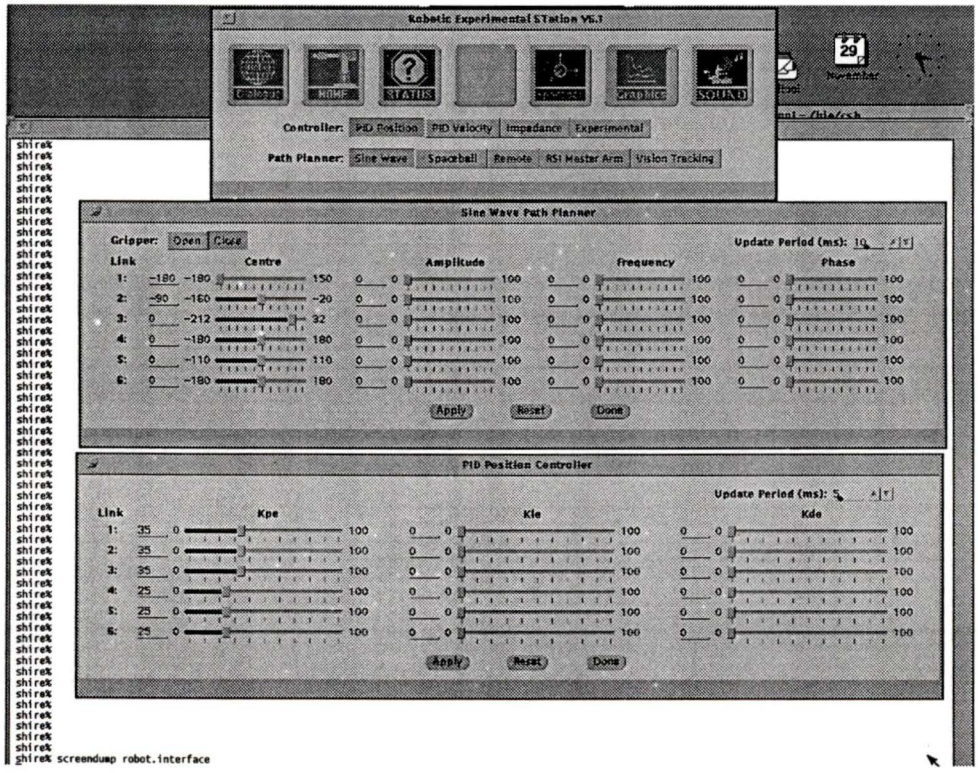

Figure 6.4: The robot computer control interface

The Graphics Animation & Path Creation Interface, a host workstation, the Telerobotic Experimental STation(TEST) as shown in Figure 6.5 consist of the Robot

Graphical Animation and Teleoperation System (RGATS) as shown in Figure 6.4, which is an architecture featuring the 3-D graphical display of a robot on a host workstation and teleoperation over a network between the host computer and the computer controlling the robot.

The Telerobotic Experimental STation(TEST) was developed in the Adaptive Robotic System Laboratory(ARTLAB) at the University of Victoria. TEST's hardware is composed of a Reis V15 industrial manipulator controlled by a multiprocessor VME bus computer running the VxWorks multiprocessor, multitasking operating system. The control computer is attached to the campus Ethernet, thereby enabling user interaction via a Sun workstation running X-window. See Figure 6.5.

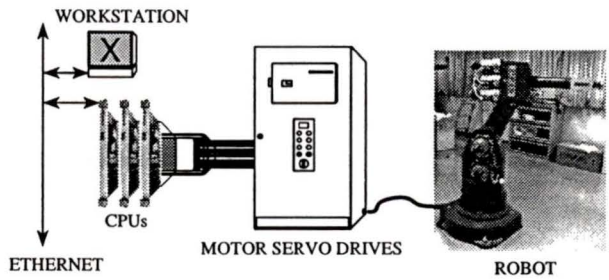

Figure 6.5: TEST Hardware

6.3 The robot detector

The steel elastic stick held by the end-effector of the robot to touch the object surface for gathering the surface data as shown in the Figure 6.6.

$$\begin{aligned} Elasticity(E) &= 30 \times 10^6(psi) = 206.8(GPa) \\ Length(L) &= 200(mm) \end{aligned}$$

$$
\begin{aligned}
Diameter &= 3.175(mm) \\
I &= \frac{\pi . D^4}{64} \\
\Delta y &= \frac{FL^3}{3EI}
\end{aligned}
$$

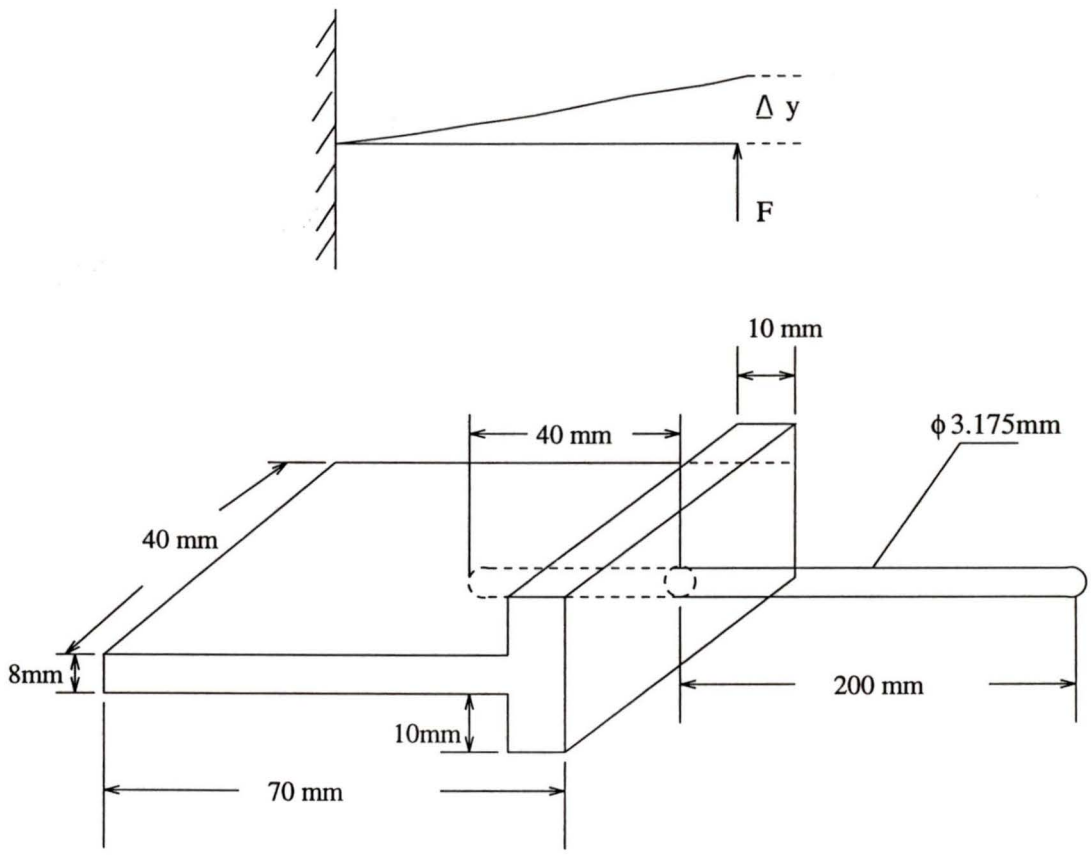

Figure 6.6: The robot detector

6.4 The Reis V15 industry robot

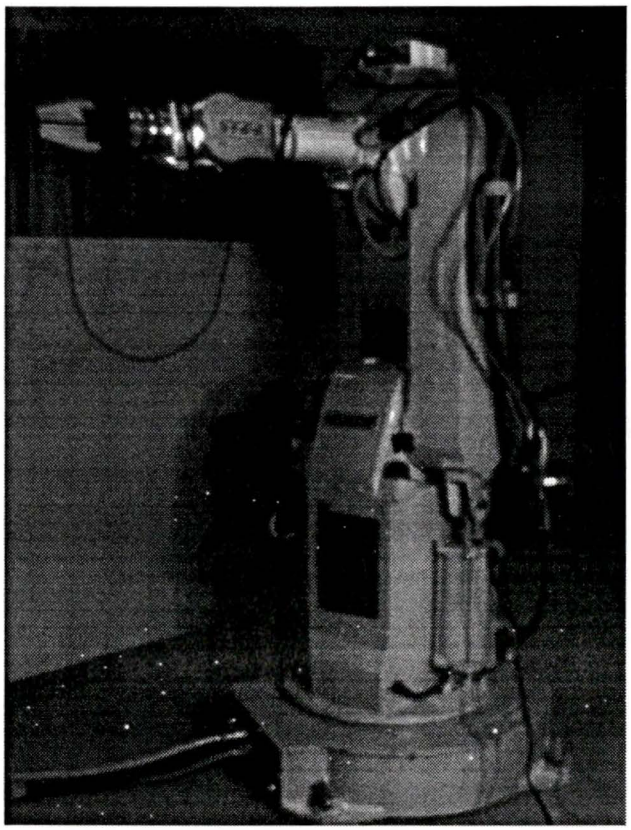

Figure 6.7: Reis V15 industry robot

The Reis V15 industry robot is used in our experiments. The robot is a general purpose six degree-of-freedom arm with the common 6R configuration; that is to say there are six revolute joints with the last three joints forming an Euler wrist. Each joint of the robot is driven by a DC motor through a harmonic drive with a 100:1 reduction. Each joint is also provided with a tachogenerator for velocity feedback, an optical position encoder for position feedback, and an optical homing indicator which is used to move the arm to a reference position on start up. A six axis force sensor is fixed on the end-effector of the robot for measuring touching force on the robot

mounting flange.

6.5 The physical objects and experiments

A table and a half cylinder were put in the Reis robot workspace in order to test the proposed algorithms and **GAPCI**. The experiment results are shown in the Figure 3.13 and Figure 4.8 separately. The errors for the position are less than 1.5 cm. In the experiment, a circle, oval, rectangle and cubic curve were created by the **GAPCI** and drawn by the Reis robot as shown in Figure 3.16 and Figure 3.18; a curve was generated on the cylinder surface by the **GAPCI** and tracked by the robot shown in Figure 4.11.

Chapter 7

Conclusion and Suggestions for Future Work

7.1 Thesis Summary

In this thesis, we have developed and implemented some simple and efficient algorithms for measuring, modelling and processing an arbitrary oriented object by the robots. A simple calculation algorithm of the deformation of the elastic stick is presented.

A computer graphics software package has been developed and implemented for industrial robots. The package provides the automatic performance of the following operations for the robots:

- automatic search, measurement and identification of objects in the working area.
- Displaying the object surface, visually generating and modifying(rotating, shifting,shrinking and enlarging) the curves drawn by the robot on the surface.

- automatic programming of the robot actuators motion, which provides a desired end-effector path along the object surface.
- providing an animation of the motion of industrial robots using three dimensional computer graphics. This package can interpret the user's commands and display various motions of the robot.
- generating the nominal control input for robotic impedance controller, which provides the desired force control along the processing trajectory. In particular, this software allows the automation of such important industrial operations as processing (grinding, polishing, cleaning, etc.) of curvilinear surfaces with required and controlled force interaction between the processing tool and the surface.

7.2 Recommendation for Future Works

In this thesis, the acceleration of the end-effector of the robot has not been considered, so that the measuring speed of the end-effector could not be fast. Future work could increase the measuring speed by deducing the effect of acceleration of the end-effector to the force sensor.

The measuring method for the curvilinear surface is effective on the convex surface. It would be desirable to develop a measuring method which can be used for any kind of surface in the future work.

Bibliography

[1] G. Field and Y. Stepanenko, "Robot Impedance Control Without a Dynamic Model: A Model Reference Approach", IASTED Conference on Control and Robotics, August, 1992, pp.110-113

[2] Merlet, J-P., "C-Surface Applied to the Design of an Hybrid Force-Position Robot Controller," Proc. 1987 IEEE Int. Conf. on Robotics and Automatic Control Conf., 1976, pp. 694-699

[3] Blauer, M., and Belanger, P. R. "State and Parameter Estimation for Robotic Manipulators Using Force Measurements," IEEE Trans. on Automatic Control, vol. AC-32, no.12,December 1987.

[4] Ish-Shalom, J., "The CS Language Concept: A New Approach to Robot Motion Design," The International Journal of Robotics Research, Vol. 4, No. 1, Spring 1985.

[5] Kelly A. Korzeniowski and Willian A. Wolovich, "3-D Surface Tracking and Recognition Using a Dual-Drive Hybrid Controller", IASTED Conference on Control and Robotics, August, 1992, pp.125-128.

[6] O. Khatib, "A Unified Approach for Motion and Force Control of Robotic Manipulators: The Operational Space Formulation," IEEE Journal of Robotics and

Automation, Vol. 3, no. 1, 1987, pp.43-53.

[7] P.K.Allen and K.S.Robert, "Haptic Object Recongnition Using a Multi-Fingered Dextrous Hand", IEEE Conference on Robotics and Automation, vol.1, May 1989, pp.342-347.

[8] P.J. Bell, "Generalizing the Hough Transform to Detect Arbitrary Shapes", Pattern Recognition, vol.13, 1981, pp.111-122

[9] R.C.Bolles and R.A.Cain, "Recognizing and Locating Partially Visible Objects: the Local-Feature-Focus Method," Int. J. Robotics Res., vol. 1, Fall 1982, pp.57-82

[10] W. A. Perkins, "A Model-Based Vision System for Industrial Parts," IEEE Trans. Computers, vol. C-27, Feb. 1978, pp.126-143,

[11] J. L. Turney, T. N. Mudge, and R. A. Volz, "Recognizing Partially Occluded Parts," IEEE Trans. Patt. Anal. Mach. Intell., vol. PAMI-7, July 1985, pp.410-421

[12] H. H. Baker and T. O. Bindford, "Depth From Edge and Intensity Based Stero," in Proc. 7th Joint conf. Art. Int. IJCAI, vol. II, Aug.24-28,1981, pp.631-636.

[13] R. A. Jarvis, "Range Finding Techniques for Computer Vision," IEEE Trans. Pattern Anal. Machines Intell., vol. PAMI-5, Mar. 1983, pp. 122-140,

[14] D. Poussart, "3D Shape Acquisition and Processing for Prosthesis Design," Tutorial on Medical Image, Graphics Interface, Montreal, Canada, May 1985.

[15] D. Nitzan, A. E. Brain, and R. O. Duda, "The Measurement and Use of Registered Reflectance and Range Data in Scene Analysis," Proc. IEEE, vol. 65, Feb. 1977, pp.206-220

[16] D. H. Ballard and C. M. Brown, Computer Vision. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[17] R.P.C. Paul and B. Shimano,"Compliance and Control," in Proc. Joint Automatic Control Conf.,San Francissco, 1976, pp.694-699

[18] M.T. Mason, "Compliance and Force Control for Computer Controlled Manipulator", IEEE Trans. on System, Man, and Cybernetics. SMC-11, (6) 1981, pp.418-432

[19] M.H.Raibert and J.J. Craig, "Hybrid Position/Force Control of Manipulator," ASME J.of Dynamic System, Measurement and Control, Vol. 102, 1981,pp 126-133.

[20] Craig, J. " Introduction to Robotics Mechanics and Control" , Addsion-Wesley Publishing Company, 1986.

[21] Spong M., and Vidysagar, M. "Robot Dynamics and Control", J.Wiley & Son., 1989.

[22] N.H. Hogan, " Impedance Control: An approach to Manipulaton: Part I - Theory, PART II - Implementation, Part III -application" ASMEJ. of Dynamic, Systems, Measurement, and Control, Vol. 107, 1985, pp.1-24.

[23] Lasky, T.A., and Hsia, T.C. "On Force-Tracking Impedance Control of Robot Manipulators", Procedings of the IEEE Conference on Robotics and Automation, Sacramento California, April 1991, pp.274-280.

[24] Liu, G.J., and Goldenberg, A.A. "Robust Hybrid Impedance Control of Robot Manipulators", Procedings of the IEEE Conference on Robotics and Automation, Sacramento California, April 1991, pp.287-292.

[25] Lu, W.-S., and Meng, Q.-H. "Impedance Control with Adaptation for Robotic Manipulations", IEEE Transactions on Robotics and Automation, Vol.7, No.3, June 1991.

[26] Witney, D.E. "Force Feedback Control of Manipulator Fine Motions", ASME Journal of Dynamic Systems, Measurement and Control, June 1977, Vol.102, pp.91-97

[27] Ogata, Katsuhiko, "Discrete-Time Control Systems", Prentice-Hall, Inc; 1987.

[28] Payandeh, S., and Goldenberg, A.A. "A Robust Force Controller: Theory and Experiments", Procedings of the IEEE Conference on Robotics and Automation, Sacramento California, April 1991,pp.42-47.

[29] W. Richard Stevens, " UNIX Network Programming ", Prentice Hall Inc., 1990.

[30] T.L.J. Howard and W.T.Hewitt, "A Practical Introduction to Phigs and Phigs Plus", Addison-Wesley Publishers Ltd., 1991.

VITA

Surname: Tao

Given Names: Wei

Place of Birth: Guangzhou, China

Date of Birth: October 28,1959

Educational Institutions Attended:

University of Victoria	1992 to 1994
Jilin Institute of Technology	1979 to 1983

Degrees Awarded:

B.A., Jilin Institute of Technology, China, 1983

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis:

Robotic Measurement and Processing of Arbitrary Oriented Surfaces

Author: [redacted]

Wei Tao

December 16, 1996

Date