An Approach to Generate Geometric Models from Multiple Range Images

by

Helai Yao
M.S., Beijing Institute of Technology, 1986
B.A., Beijing Polytechnic University, 1982

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the
Department of Mechanical Engineering

We accept this thesis as conforming to the required standard

Dr. Z. Dong, Co-Supervisor (Dept. of Mechanical Engineering)

Dr. R. P. Podhorodeski, Co-Supervisor (Dept. of Mechanical Engineering)

Dr. G. F. McLean, Departmental Member (Dept. of Mechanical Engineering)

Dr. G. W. Vickers, Departmental Member (Dept. of Mechanical Engineering)

Dr. K. F. Li, Outside Member (Dept. of Electrical and Computer Engineering)

Dr. A. K. C. Wong, External Examiner (Dept. of Systems Design Engineering,

University of Victoria

Supervisors: Dr. R. Podhorodeski and Dr. Z. Dong

Abstract

The research described in this dissertation focuses on the development of a new approach for the generation of geometric models from multiple-view range image data.

Through intensive comparison and evaluation of different representations, the cross-section contour based representation is concluded to be ideal for modeling with range image data. The representation is shown to be at an intermediate level – compatible with both the low-level of range image data and with the need to provide relatively high-level geometric and topological information in models.

A new concept of generating partial models within device frames, frames associated with the working principle and geometry of a range sensor, is introduced. The range data are well distributed in the device frame. This good data distribution facilitates computations relevant to rendering the cross-sections required by the representation and relevent to identifying occlusions present in the image. Methodology for merging the partial models with a current global model is developed to allow the incorporation of redundancy between the partial model and the current global model and to allow growth of the global model. A simulation of the ERIM imaging-radar based range sensor, a prototype triangulation-based range sensor developed for this research and a commercial HYMARC range sensing system are used for approach verification. The device frames associated with the sensors are derived, and used to test the modeling approaches and the developed system.

The presented research: demonstrates the suitability of the cross-section based representation for range-image based modeling systems; introduces a new concept and associated methods for generating cross-section contour models in range sensor device frames to take advantage of well distributed data; develops a series of algorithms

for partial modeling in the device frame and for global model integration; and demonstrates the feasibility of the developed new approaches for applications by testing the system for multiple sensor types.

Examiners:

Dr. Z. Dong, Co-Supervisor (Dept. of Mechanical Engineering)

Dr. R. P. Podhorodeski, Co-Supervisor (Dept. of Mechanical Engineering)

Dr. G. F. McLean, Departmental Member (Dept. of Mechanical Engineering)

Dr. G. W. Vickers, Departmental Member (Dept. of Mechanical Engineering)

Dr. K. F. Li, Outside Member (Dept. of Electrical and Computer Engineering)

Dr. A. K. C. Wong, External Examiner (Dept. of Systems Design Engineering,

Table of Contents

List of Tables

List of Figures

Acknowledgements

I must thank my co-supervisors: Professor R. Podhorodeski and Professor Z. Dong for their guidance during this research program and their comments on the work.

I wish to express my gratitude to the CAD/CAM, and the image processing and computer vision research groups, especially to Professor G. W. Vickers and Professor G. F. McLean, for the use of their equipment.

I also wish to acknowledge the financial support of the Natural Sciences and Engineering Council of Canada (NSERC), the Institute for Robotic and Intelligent Systems (IRIS) and Precarn Associates Inc.

Finally, I would like to thank my parents, wife and children for their encouragement.

Chapter 1

Introduction

1.1 Overview

A description of the geometric characteristics of an object or working environment is required for intelligent robotic systems, manufacturing and various other application fields. Advanced robotic and reverse engineering systems further require the geometric modeling to be conducted in conjunction with the machine vision system to support object recognition, intelligent decision making, rapid replication and accurate inspection.

Geometric modeling refers to a collection of methods used to define the shape and other geometric characteristics of an object [43]. Its goal is to generate models in a computer system for the structures and objects in the real world. Such a model is actually described in a representation including the geometric and topological information of the structures and objects.

Traditionally, a geometric modeling approach is based on an assumption that the relevant geometric characteristics of an object to be modeled and/or the

relevant modeling results are well known by the user. However, the traditional geometric modeling techniques are challenged by a new demand of perceiving and modeling existing objects and/or working environments. This demand comes from the fact that the geometries of objects to be modeled may be partially or totally unknown. A positive solution for satisfying the demand is to combine geometric modeling techniques with 3D active sensing techniques. The research associated with this dissertation has developed and implemented a new approach that is able to generate multiple-view 3D image data based surface models for describing geometric characteristics of existing objects with complicated curved surfaces.

This introductory chapter: provides a brief review of geometric modeling with traditional modeling approaches, and describes the research motivation driven by the need for more flexible modeling technologies (Section 1.2); discusses the general problems that must be solved by a 3D vision data based geometric modeling approach (Section 1.3); reviews previous research on geometric representations used by both traditional geometric modeling methods and computer vision systems, on range sensors and their device geometries, and on multiple-view fusion technologies (Section 1.4); and finally states the research objective (Section 1.5). In addition, Section 1.6 gives an outline of the remaining contents of the dissertation.

1.2 Traditional Geometric Modeling Approachs and Challenging Desires

1.2.1 Information Embedded in a Geometric Model

A geometric model can describe objects with their shape, dimension, area, volume, position, orientation, and topological relationships. In general, various geometric

models can be classified into three major categories: wireframe models; surface models; and solid models.

Different types of geometric models have different capabilities of representing geometric information. A geometric model with a simple form is usually at low level in terms of the geometric information it can supply, while a sophisticated geometric representation at the high level can provide much more detail.

A wireframe model has the simplest model form, and is composed of object edges that are surface intersections and/or surface outlines. Wireframe models may include straight lines, analytical curves and free parametric curves. Object descriptions provided by a wireframe model are primarily limited to the edge related features. To overcome the ambiguity introduced by such a model, human intelligence is often needed.

In contrast, a surface model is able to intrinsically provide more detailed characteristics data such as surface shape, curvature and surface points. A surface model can be built using either regular analytic surfaces such as a plane, a cylinder or a sphere, or sculptured surfaces defined by spline functions or surface meshes. Both wireframe and surface models supply few of the topological relationships among their component elements: points, boundaries and surfaces.

A solid model, as a more sophisticated geometric model, provides a complete description of the objects allowing detailed geometric and topological information. The dimensions, areas, volumes, surface shape descriptions, surface neighborhood relationships, and relative object positions and orientations can be obtained from the database of a solid model. The interior and exterior spaces of an object can also be detected.

1.2.2 Traditional Approach to Geometric Model Generation

Construction of a geometric model requires initial data input. Sophisticated modeling approaches such as solid modeling require more detailed information about the object to be modeled. This initial data includes both geometry and topology information of the object.

Traditionally, geometric models for various objects are created using a CAD/CAM system. The modeling approach has four primary components: 1) symbol structures that represent solid objects; 2) processes that use such representations for answering geometric questions about the objects; 3) input facilities for creating and editing object representations and for evoking processes; and 4) output facilities and representations of results [53].

The creation of a geometric model is usually done interactively by users of the geometric modeling system through graphics input and output devices such as a mouse, a 2D digitizer, a trackball, a joystick, a lightpen, a keyboard, a graphics terminal or a plotter. Since the users of the graphics systems well understand the geometry and topology of objects that they want to model, they can effectively use the input devices to provide the object's geometry and topology to the geometric modeling system [20, 43, 76].

This modeling procedure can be illustrated using the flange example in Figure 1.1. When a solid model for the flange is built, the shape, dimensions and location of various surfaces representing the central hole, keyway, pilots, chamfers and so on, the surface adjacency relationships indicating object topology, and the surface orientation identifying the interior and exterior of the object are specified by the user.

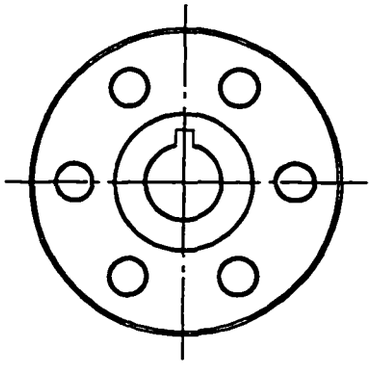


Figure 1.1: Modeling with known object geometry and topology

1.2.3 The Need for More Flexible Modeling Approaches

Although the traditional geometric modeling approaches have been used broadly in design and manufacturing, they cannot satisfy some key requirements of geometric model generation imposed by intelligent robotic applications, advanced manufacturing, and many other applications [12, 22, 45]. These requirements include: 1) modeling objects with known geometries, but in unstructured environments; and 2) modeling existing objects whose geometries are not known [72].

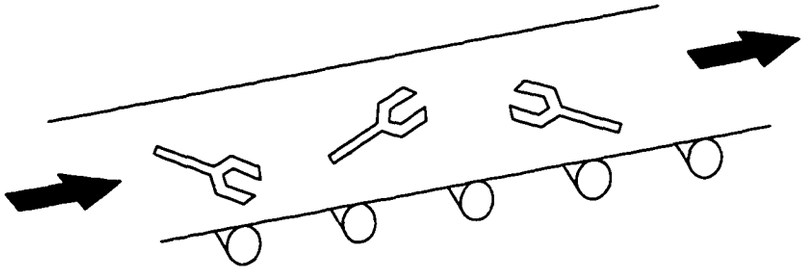


Figure 1.2: Objects with known geometry in a unstructured environment

Figure 1.2 shows an example for case one. A cast part has been designed,

and a geometric model for the ideal geometry of the part has been created. In manufacturing, the shape and dimensions of produced parts need to be measured automatically and to be compared with designed dimensions given in the ideal model. However, no direct match between the real cast part and its ideal model exists due to the different orientation and/or scale between them. The creation of a geometric model based upon the real geometry of the produced part is needed for inspecting the dimensions of the manufactured part. Without human interaction the traditional modeling approaches are not able to construct this type of real geometric model. Other similar situations may include a robot picking up parts from a bin and automatic vehicles moving in unstructured surroundings.

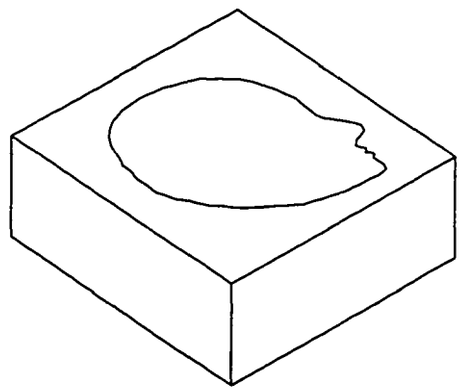


Figure 1.3: An object with unknown geometry

An example of the second case is illustrated in Figure 1.3 where an injection mould of a face sculpture is desired. Similar situations can be found with worn part surfaces or with deformation of formed sheet metal parts. Since the shape is irregular and the exact geometry and topology of the object are unknown, the initial object description cannot be explicitly input into a CAD/CAM system with traditional approaches. Without a geometric representation, the tool path for machining and CNC codes cannot be generated. A solution to solve this problem is to model the object based

on object surface measurements. Although coordinate measuring machines (CMM) are used for measuring object profiles in industry, applications of CMM are limited by its mechanical contact working principle and the degrees of freedom of the machine and its touch probe. Application of active 3D vision sensors can eliminate these drawbacks due to their non-contact measuring principle.

An active vision sensor emits an energy beam, a laser beam in most cases, onto the surface of objects. Coordinate data (referred to as range data) of the surface is obtained by processing the energy reflection which is sensed by an imaging transducer. Such a vision device can output discrete range data or coordinates, which are in the image form, representing object surface points. The quality of range data acquired by an active sensor is usually better than data by other kind of vision sensors. The conversion from sensed energy information to 3D range data can be faster as well.

Time-of-flight based and triangulation based range sensors are two kinds of frequently used active sensors in machine vision. The former usually consists of a laser source and a detector. Range data is calculated by measuring the elapsed time directly or indirectly by the phase shift when the laser pulse is projected to a scene and reflected back by the surfaces of the scene [29, 30, 49]. The latter employs the device geometry and a structured light source to find the geometric relationship between the device and a scene, allowing calculation of the range data [1, 54].

1.3 3D Vision Data Based Geometric Modeling

To obtain geometries of existing objects, 3D active sensors can be used. A vision data based modeling system is related to three spaces, i.e., the real world space, the image space and the modeling space. Figure 1.4 illustrates these three spaces and connections between them. In principle, the surface geometry of an object in the real

world can be described by a mathematical function. In the image space the surface is represented by the range images that are a set of coordinate points.

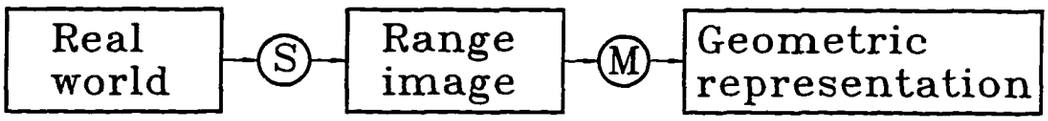


Figure 1.4: Three spaces: from the real world to geometric models

The transformation from the real world to the image space is achieved by a 3D sensor(s) S. A modeler M generates geometric models of the object in the representation space. The algorithm designed for the modeler and the form of the representation to be used for the geometric model are correlated.

A key task for a vision data based modeling approach is to convert the discrete sensed surface points into geometric models in a representation. Different representations handle geometric models in different levels. For instance, in low level a primitive application may only use an image representation where object surfaces are described by the discrete range data; in high level a recognition task may need a symbolic oriented representation; while, in middle level a rendering algorithm may want a surface based representation. To be compatible with discrete range-data based surface points and to provide applications with relatively concise geometric models of objects, an intermediate-level representation is appropriate.

Due to the limited viewing field of a sensor, and to surface complexity and object occlusion, range images for objects must often be acquired from multiple range sensor viewing positions to allow a complete geometry description to be obtained. A priori knowledge of the views to be taken may not always be available. In this

sense, the multiple view range data acquisition is unstructured. Flexible selection of viewing locations is quite important in order to handle objects and workspace with complex surface shapes. In order to avoid any missing data, two adjacent views have to be partially overlapped. The data redundancy caused by view overlaps is inevitable. Therefore, the modeling system needs to perform multiple-view fusion in its modeling processing to merge the data redundancy. Allowing flexible view selection and performing data fusion increases the difficulty of geometric modeling from multiple-view range images.

In summary, a 3D image data based geometric modeling system needs a representation for describing geometric characteristics and requires related algorithms for converting low-level range data into geometric models. In order to generate a complete geometric model for an object, the range data must be acquired from multiple viewing positions to allow coverage of the whole surface of the object. The modeling algorithms should be able to handle and merge the resulting data redundancy.

1.4 Previous Research

This review of the previous research on vision-data based geometric modeling approaches is focused on: 1) representations that are the basis for a modeling system; 2) the research and development for 3D active vision sensors that provide the initial data input for a vision-data based modeling system; and 3) the research on multiple view fusion that is an essential requirement for a vision-data based modeling system.

1.4.1 Representations Used for Design Oriented Modeling Systems

The present representations for geometric modeling used in CAD/CAM and computer graphics tend to be informationally complete, valid, and unambiguous [76]. Among them, the frequently used representations include Constructive Solid Geometry(CSG), Boundary Representation (B-Rep), sweep, and octree. These four representation methods share the common characteristics of decomposing the 3D space that is occupied by an object into sub-level parts (or cells). The decomposition can simplify the complexity of representation and is usually called cell-decomposition [43]. Requicha [53] introduced a general principle for evaluating the appropriateness of representations of solid objects by listing several key properties: validity; completeness; uniqueness; ease of creation; and efficiency in the context of applications. These properties are often used as criteria to evaluate representations for geometric modeling.

The CSG representation is used in most present CAD systems to model 3D objects by representing volumetric information. In a CSG system, primitive solids and Boolean operations are used for constructing objects. A binary tree with end nodes representing the primitives and other nodes recording the logical operations is normally used [20]. The primitives are usually described by analytical equations for fast processing and accurate results. However the capability of describing various geometric shapes is limited due to the restriction of analytical equations.

The B-Rep representation describes objects by representing the boundary vertices, edges, and surfaces of an object. It is more complex than CSG and also more flexible in representing various geometric shapes. This method is based upon the assumptions that an object is a solid and all its boundaries are two-manifolds [20].

A major implementation of the B-Rep representation is the Winded-Edge, in which the graphical and topologic information of vertices and surfaces are organized around the boundary edge of an object. Curved boundaries and surfaces, e.g., represented by B-Spline or NURBS, can also be included.

The sweep representation generates a geometric model using a cross-section curve, a profile curve or guide curve of a line or a curve, and coordinate transformations. The cross-section curve is usually in 2D and described by analytical or parametric equations. The profile curve is used to guide the sweeping and the coordinate transformations are used to generate the discrete swept volume [43]. Sweep rules can be added to alter the shape of the cross-section curve during sweeping.

The octrees representation works by recursively subdividing a cubic cell that encloses an object or part of an object into finite cubic cells and using a tree representation to record the subdivision process and the spatial occupancy enumeration of objects to make the whole internal space addressable. The approach provides easy access to a given spatial point and ensures spatial uniqueness. However, octree representation requires large memory space for data storage and is unable to represent the topologic relationships of an object [6]. An improved method by Brunet [13] allows modeling for some geometric details by introducing vertices and edges to the representation.

Generally, on one hand, the CSG and B-Rep representations can handle both geometric characteristics and topological relationships of an object. This capability is very useful in machine vision for recognition tasks. On the other hand, the two approaches require given topologic relationships to function. This dependency restricts them to modeling only objects with explicit topological relationships.

The sweep and octree representations are volume based, and are concerned with the space occupancy of 3D structures rather than the topological relationships.

Therefore, they can represent objects without requiring topologic relationships. However, their functions are weak in the provision of geometric properties and topological relationships of objects. All schemes reviewed above only deal with constructing object models with closed shapes, i.e., the volume of a model is bounded. It is difficult, even impossible, to model a local part of an object using these schemes.

Conventional geometric modeling is related to designed objects with known geometry as discussed previously. In a design procedure the geometric information required by a representation is gradually derived from abstract to details. For a vision system, the situation is the inverse. Required geometric data needs to be extracted from low level raw image data.

1.4.2 Representations Used for Vision Data Based Systems

In general, most of representations used by vision data based modeling systems are similar to traditional ones. The major task for developing such a geometric modeling system is to convert discrete image data into relatively high level information in order to fit a certain representation.

The algorithms to perform the task can be very complicated and lack robustness, if the representation used requires very high level information, e.g., both geometric and topological data. While high-level representations need very complicated information extraction approaches and lack robustness, low-level ones can increase the algorithm robustness but may reduce the capability of representing details of the geometries included. In practice, the tradeoff has to be considered.

To fully investigate various representations available, the following review covers representations used for computer vision systems using either active or passive imaging sensors. All of these systems need to convert low level image data into

geometric information consistent with a representation.

To represent the acquired 3D vision data of unknown objects, a few machine vision based geometric modeling schemes have been introduced. These schemes use 3D vision data of observed scenes as input to construct geometric models, and normally require a complete observation. They are mostly application-oriented.

Generalized cylinder(GC) representation, a similar scheme to sweep is commonly used as a recognition-oriented object description in machine vision [7]. The conventional sweep uses regular and constant shapes as section shape. The GC scheme may have irregular section shapes in each cross section plane.

Superquadric description is a representation often used for recognition systems. Superquadrics are a family of parametric primitives that are mathematical extensions of quadric surfaces [73]. This representation ignores geometric details by describing primary shapes of objects to reduce difficulties in recognition.

Geons are also used to represent 3D objects [47, 48]. Geons are a set of volumetric primitives which can be composed to form different shapes. A range image of the object is processed to find edges that correspond to the constituent parts of the object. By decomposing these edges and recognizing the individual constituent parts, the geons for these parts and the graph for the spatial connection of parts can be found.

Kumar *et al.* [37] used a hyperquadric representation which is a surface defined by the set of points (x, y, z) satisfying the equation

$$\sum_{i=1}^{n} |A_i x + B_i y + C_i z + D_i|^{\gamma_i} = 1$$

An object is divided into components each of which is represented by a hyperquadric equation. An error-of-fit function was defined to measure the difference between the model and the data. The minimization of the function was used to fit a hyperquadric to range data.

Raja *et al.* [52] proposed a generic part based 3D object representation similar to the geon representation. They used 12 generic primitives. Each primitive has 74 predetermined views for matching the surfaces extracted from a range image by segmentation. This approach ignores geometric details for helping find the match between segmented surfaces and a generic part.

Kweon and Kanade [38] developed a method for generating elevation maps from range data acquired using the ERIM range sensor. The elevation maps were constructed on a regular grid system, the elevation data being the intersections between vertical lines located at the grid points and the surfaces represented by a range image.

Faugeras *et al.* [19] successfully represented the sculptured surfaces of a cast automobile part based on laser range data, using a triangulation facet-oriented polyhedral approximation. Multiple-view structured laser scanning was conducted by putting the object on a rotational table, and by synchronizing the rotation with the image capture (one image per 30° of rotation). Unfortunately, this kind of ideal (structured) multiple-view laser scanning condition is not always applicable.

A cross-section-like representation for the volumetric rendering of Computer Tomography (CT) and Magnetic Resonance (MR) data is utilized in medical imaging systems. The 3D data acquired from a transaxial CT/MR scanner and extracted from the raw images are in slice form. A volume model can be formed by stacking individual CT slices and using linear interpolation between two slices [46]. This approach depends on the well-structured scanning. Similar work also includes organ surface reconstruction using serial microscopic sections in biomedical engineering [41].

With some well-structured scanning approach, a range sensor can directly provide scanned data representing object section contours. Theodoracatos *et al.* [63] developed an active vision data based modeling system. The range sensor projects

a series of parallel laser light planes onto an object and, thus, obtains 3D digitized contour data. Least-squares fitting was applied to the vision data to generate a B-Spline representation with a minimum number of control points.

By taking 2D intensity images of an object in three orthogonal viewing directions, Srinivasan *et al.* [61] extracted the 3D occluding silhouettes of objects. Reconstruction planes were used to render a set of cross-section contours to represent volumetric occupance of the object. The method is limited to objects with simple boundaries since the 3D reconstruction is not robust for objects with curved surfaces.

Applications of close range photogrammetry deal with identification of the geometric properties of an object (such as distances, angles, volumes, sizes and shapes). For railway tunnel wall modeling, Clarke [18] obtained the wall profile data by measuring distance, section by section, from a known position of a measuring system to the tunnel wall surface, and by computing coordinates of profile points according to the measured distances and the known measuring locations. Then reconstruction of the cross-section contour model for the tunnel wall was accomplished by linking these coordinate points.

Cross-section based representations can be used to generate surface models. Chen *et al.* [15] proposed a surface interpolation technique for reconstructing 3D object models from cross-section contours. A series of intermediate sections are generated first. An initial surface model is obtained using spline functions based on these sections. The final surface model is interpolated by applying the quadratic-variation-based algorithm.

Kim *et al.* [36] upgraded the cross-section representation based models to triangulation based surface models. Their method splits each section contour into convex/concave segments and finds the correspondences of segments in adjacent sections. Triangular surfaces are constructed from these adjacent contour segments.

Wei *et al.* [66] converted sliced scan data of an industrial CT scanner to a Q-Mesh representation which is a finite element meshing technique. This mesh representation is based on a deformable grid template approach.

A research review on representations used by vision data based modeling systems for multiple-view fusion is included in Section 1.4.4.

1.4.3 Device Geometries and Range Image Form

The technologies used for obtaining three dimensional structure information have been developed and implemented in many projects for both research and industrial applications. Various sensing principles to measure distances have been investigated by numerous academic researchers and commercial companies. Since active sensors that depend on controlled energy or structured light to measure distances can acquire relatively robust range images, more active sensors are being used in industry and research, than passive sensors that rely on illumination of unstructured light and on image analysis (such as feature matching and statistical processing) to obtain depth information [65].

Since an active sensor usually measures the distance (range) between the sensor and object surface, the sensor is referred to as a range finder or a range sensor. In this dissertation, these terms are used interchangeably. A range sensor associated with dedicated hardware and software is capable of generating a range image of a scene. A range sensor generates a range image that is a 2D (or 1D) numerical array. Each element of the image array records a number to represent a range (or a coordinate value for an affine transformation based sensor [11]). This element is indicated by two subscripts to represent the range measurement direction. Commonly used range sensors are the imaging radar based and triangulation based sensors. The following

review is mainly focused on device geometries of these sensors and the effects imposed on the range data by the device geometries.

Nitzan *et al.* [49] proposed a phase-shift based range finder which has three major functional components - a transmitter, a scanning device, and a receiver. The laser beam emitted by the transmitter is amplitude-modulated with a sinusoidal waveform. This modulated beam is then deflected by the scanner to cover the field of targets. When the beam strikes an object, a portion of the light is reflected back and converted into a sinusoidal current at the modulation frequency. The phase of this current is proportional to the time it takes the light to travel from the instrument to the object and back, and thus provides the range value. While the components of transmitter and receiver handle the signal processing, the scanning unit performing tilting and panning activities governs the device geometry and thus determines the geometric transformations for obtaining surface point coordinates. The initial result r that is measured based on the phase shift is represented in an array of pixels indexed by two scanning activities J and I, respectively. A detailed drawing for the device geometry was given [49] and a mapping function $(x, y, z) = f(J, I, r)$ was derived. The authors also indicated that it is easy to obtain the inverse transformation from x, y and z to J, I and r. Using (J, I, r) image data can significantly simplify the algorithm for detecting jump boundaries, the border between the images of two surfaces, by working on r and $\triangle r$ values only.

Besl [11] discussed the relationship between range image data and coordinate data. Many range image sensors produce range images in the r_{ij} form where the transformation from (i, j, r_{ij}) to (x, y, z) is non-affine. In most of these cases, each ij pair corresponds to a small cone or pyramid of energy centered around a ray that pierces the first object surface it hits as it travels outward from the sensor. In many of these cases, there is almost no uncertainty in the direction of the ray as determined

by the integer i and j indices of the digitized 3D surface points compared to the uncertainty in the measured range r. Thus, the uncertainty in the resulting x, y, z coordinates is dominated by the uncertainty in the r_{ij} values.

The Environmental Research Institute of Michigan (ERIM) developed a range scanner that was used by the Autonomous Land Vehicle project [64]. The ERIM range scanner modulates the power amplitude of a laser beam and measures the phase difference between the reference wave form and the returning signal. Its scanner is composed of by a nodding mirror determining the plane that the beam will be in for any given row of the range image, and a rotating polygon mirror controlling the scanning angle within the plane for any given column of the image. The scanning activities with the nodding and rotating mirrors form a spherical coordinate system (θ, ϕ, ρ), where θ and ϕ are the nodding and rotating angles, respectively, and ρ is the range.

A similar scanning method for a time-of-flight range sensor is found in Matsumoto's research [40]. The laser beam is deflected in two directions with angle β_x and β_y by the mirrors mounted on the pan and tilt scanners. Each range image pixel (L) is indexed by these two angles. The Cartesian coordinates are obtained using the equations

$$x = L\sin(\beta_x) \tag{1.1}$$

$$y = L\cos(\beta_x)\sin(\beta_y) \tag{1.2}$$

$$z = L\cos(\beta_x)\cos(\beta_y) \tag{1.3}$$

Triangulation based range sensors are another class of non-contact vision devices. Such a sensor usually employs a structured light as an edge of the triangle, and uses the line between the surface point hit by the light and the image point (usually generated by a CCD camera) of this surface point and the line between the light

source and the camera as the other two edges. With this triangle the coordinates of the surface point in terms of a sensor-based Cartesian reference system can be calculated.

Araki *et al.* [3] used a cylindrical lens to form an expanded laser beam. This beam was deflected to scan the field of interest at a constant angular interval by means of a rotating mirror. The space coordinates (x_p, y_p, z_p) of the point P on the surface of the target was computed from the positional coordinates $(x', 0, z')$ of its image P' and the deflected angle α of the laser beam, i.e.,

$$\begin{aligned} x_p(x', z', \alpha) &= \frac{(y_f - y_m) + x_m \tan\alpha}{y_f + x' \tan\alpha} x' \\ y_p(x', z', \alpha) &= (x_p - x_m)\tan\alpha + y_m \\ z_p(x', z', \alpha) &= \frac{(y_f - y_m) + x_m \tan\alpha}{y_f + x' \tan\alpha} z' \end{aligned}$$

where z' and α change regularly and independently of scenes, while x' is dependent on scenes. Similar work can be found in [23, 63].

In a research project conducted by Jezouin *et al.* [31] the active triangulation range finding system was composed of a camera, and a laser system generating a plane of light projected on an object placed upon a rotary table. The camera captures the image of the laser light on the object surface in terms of coordinates (x_c, y_c) in the image plane. By calibration, a mapping function represented by a transformation matrix M was found to convert (x_c, y_c, θ) into $R[\theta, h]$, a cylindrical range image, where θ is the rotating angle of the turntable and h the height along the turntable axis. The inverse transformation M^{-1} can be used for estimating measurement errors.

Besides the basic sensor structures based on the triangulation principle, there are many variations for different purposes. Yoshida *et al.* [74] proposed a redundant triangulation-based range measuring system to increase the measuring accuracy by

mounting two cameras aside a laser scanner. A target surface point hit by the laser beam has two images in the CCD cameras A and B individually. The image location of the point is found at (u_A, v) and (u_B, v) in each image plane. In this system, the laser scanning angle is placed by u_A and u_B, when calculating coordinates, e.g., $x = \frac{Df}{u_A+u_B-2u_0}$, where D is the base-line length; f is the focal length; u_0 is the half length of a line of the sensor array; and u_A and u_B are the image positions in the camera A and B, respectively. To solve the occlusion problem that is more serious for triangulation based range finders than other kind of sensors, Yoshimura and Okamoto [75] implemented a dual channel optics using one sensor. Although the geometry becomes complicated, the mapping function $Z = f(X)$ to convert initial range measurement X into the coordinate Z is still available.

Since the triangulation-based measurement requires a long baseline to satisfy accuracy, missing-data can become a serious problem and the devices can be an overly large size. Rioux [54, 55] successfully invented a synchronized scanner for a range finder to eliminate the long-baseline requirement. This range finder uses a position sensor and a rotating laser beam which is scanning synchronized with the rotating mirror for the reflected beam. This structure improves the resolution and the field of view without increasing the baseline. Although the device geometry is more complex than others, the initial range image is still in a numerical array (p) indexed by the scanning angle (θ), and indicates the image position of spot hit by the laser. By the analysis of the device geometry, the mapping function is found

$$\begin{aligned} x(\theta, p) &= dp\left[p + \frac{fl(2l\tan\theta + d)}{(l-f)(d\tan\theta - 2l)}\right]^{-1} \\ z(\theta, p) &= -d\left[p + \frac{p(l-f)}{fl} + \frac{2l\tan\theta + d}{d\tan\theta - 2l}\right]^{-1} \end{aligned}$$

where p is the image position; θ is the scanning angle; f is the focal length; d is the distance between the scanner axis of rotation and the principal point of the lens;

and l is the distance between the common axis of projection and detection and the reference point.

From the above review, it is seen that both imaging-radar and triangulation based range sensors have an initial range image frame, i.e., a numerical array. For all such sensors there exists a transformation function to convert the data from this initial range image frame into a Cartesian system. Each range image pixel is represented by (θ, ϕ, r) for the imaging-radar based and by $(x_{image}, y_{image}, \beta)$ for the triangulation based.

Among these variables, θ, ϕ and β, the scanning angles, and y_{image}, the vertical position in the intensity image, continuously change with constant interval, and are not effected by scenes and noises. Therefore, the values in these variables almost have no errors. However, the variables r, the range, and x_{image}, the horizontal position in the intensity image, are influenced by the scenes and noises, and can have uncertainties in their values, e.g., for range jumps. After the transformation the values in x, y and z coordinates are effected by the uncertainties contributed by r (imaging-radar based sensors) or x_{image} (triangulation based sensors).

1.4.4 Geometric Modeling with Multiple-View Image Fusion

Since any range sensor has a limited view scope and object occlusions are inevitable as well, multiple views from different locations around the observed objects are necessary for generating complete geometric models. The model generation in these multiple-view cases requires the integration (fusion) of image data from different views.

The method of data integration usually is representation oriented. Some representations require the entire data from all views to be processed together

simultaneously for modeling. Other representations allow the image data in each view to be converted individually into a partial model and then merged with a global model.

Noborio *et al.* [50, 35, 28] used object images from multiple views as object contour projections to construct the object model. With the object contours in different views, several polyhedral cones (when the image is considered to be created using central projection) or polyhedral cylinders (when the image is considered to be created using parallel projection) can be built. The object model can then be obtained by finding the intersecting volume of these cones/cylinders. This approach requires each image cover the whole object to get a complete contour.

Wong *et al.* [68] proposed a hypergraph based shape-representing approach using multiple-view range images for the recognition purpose. Geometric elements of an observed object are extracted from range images. The relationships of these elements are represented by a hypergraph. For recognizing the object, this hypergraph is compared with hypergraphes stored in a database for known objects in various orientations. The creation of the database is also based on the same methods for the element extraction and the hypergraph generation methods.

Hebert *et al.* [24] employed a mesh to represent an object. The mesh is derived from deforming a standard spherical mesh until the mesh fits the image data. This method was developed mainly for object recognition.

Chen *et al.* [17] developed inflating balloon method using a triangulation mesh to represent a balloon model. The balloon model is driven by an inflating force toward the object surface, until the mesh elements reach the range data based object surface.

Chen *et al.* [16] used a point based representation to constructing a complete model for an object from multiple-view fusion. The 3D points are represented in a spherical or a cylindrical coordinate system, called the object-centered representation.

The data from each view are transformed into this frame. The data integration is done by simply averaging the overlapped data.

Kawai *et al.* [34] also employed a point based representation to perform multiple-view range image fusion in a global reference frame. Each 3D point is associated with a normal vector for the local surface. The data points belonging to a view are transformed to the global frame and, compared with previous data using the viewing angle and their normal vectors. Visible points are accepted to update the global model. Another point based multiple-view fusion method is described in [39].

Asada *et al.* [4, 5] developed a system to generate height maps from range image sequences. A height map is a $2\frac{1}{2}D$ point based representation. Each range image which initially is in a device frame is transformed to a local height map, and then integrated with a global map by matching geometric properties. Nashashibi *et al.* [44] developed a similar system where the height map is called an elevation map. The global model is based on a $5cm \times 5cm$ grid system. During the multiple-view range data integration the transformed local map is resampled in this grid system.

Kamgar-Parsi *et al.* [32] described an approach for registering multiple overlapping range images to generate sea-floor contour maps. A contour map consists of a set of 2D curves. The approach uses both local and global curve-matching methods to find the location of redundant data caused by overlapped range images. The local matching is based on a modified chain code method. A contour curve is segmented into a group lines. The line orientation is encoded to form the chain code. The global matching is based on minimizing a cost function considering bending and stretching of the range images.

Soucy *et al.* [57, 58, 59, 60] used a triangulation based representation for geometric modeling using multiple range images. The Venn diagram of N range views is built to find range data redundancy. The redundancy is merged by triangle intersection

based erosion, dilation and connection.

Higuichi *et al.* [25, 26] proposed a mesh based 3D modeling system. A local mesh is fit to a set of data points from each view. The curvature of each node of the mesh is mapped to a spherical image for registering multiple views. A global mesh can be built by transforming the data points of a local mesh into a global frame using the registration information.

Stenstrom *et al.* [62] proposed a method for constructing object models from multiple images. The models rely on edges extracted from range images and faces fit with data enclosed by edges. Data integration for constructing incremental models is based on edges and faces.

Parvin *et al.* [51] introduced a B-rep based modeling approach. Each range image is segmented to extract object surfaces visible in the view. An attribute graph is built using the extracted surfaces as the nodes and the surface neighborhood relationships as the links. Such a graph is matched with another one, if the two views represented by their graphs are overlapped. Furthermore, the intersections of the involved surfaces are computed for building B-rep models.

1.5 Objective of This Research

1.5.1 Limitations of Previous Research

The previous research efforts for developing modeling approaches and representations have been discussed in the previous research review sections. Researchers have achieved significant progress in geometric modeling using 3D image data. However, these approaches share one or more of the following limitations.

Some recognition oriented approaches used highly abstract representations for simplifying modeling computation. Representing geometric details is not a major goal for these approaches. Some approaches require complete image data from all aspects around a scene before performing multiple-view based modeling. As well, these approaches cannot generate models for a partial object or for multiple-object based scenes. Many applications require a modeling approach that is able to provide detailed characteristics (e.g., geometric and topological information) of objects without significantly increasing the computational costs. In addition, the data conversion from low level based 3D images to relatively high level modeling representations needs to be facilitated. Furthermore, modeling methods developed with multiple-view fusion did not use representations which are for applications of reverse engineering like rapid prototyping systems. Such applications often use layer or section based geometric representations.

The research reviewed for multiple-view fusion was primarily focused on the development of merging algorithms. The research of the merging algorithms for modeling in a unbounded environment, e.g., the situation encountered by robotic navigation, has not been well investigated. When a working space to be modeled is large, the number of views to obtain images will be large, and therefore the model can involve a huge amount of data. Consequently, data merging between the partial model and the global model becomes difficult in terms of overlap detection. Reduction of the search and detection scope in the model database is an important issue for this modeling problem.

Many applications use a Cartesian system as the reference frame. This is done largely to achieve consistency with both the traditional geometric modeling systems and vision data based modeling systems which provide models in Cartesian frames as well. The range data are converted into geometric models in this Cartesian system.

Ideally, a range image would be stored in an array, where the x and y locations of pixels are indicated by the row and column indices of the array elements (pixel elements) and the z values are stored in these pixel elements. With such an array, any point and its neighborhood can be easily accessed by $z = f(x(i), y(j))$, where $x(i) = x_0 + i\triangle x$ and $y(j) = y_0 + j\triangle y$. In reality, however, a range finder does not directly output range data in this required data format due to the working principle and geometric structure of the range-finding device [10]. Most range finders use a central projection principle with equally spaced angular scanning of associated mirrors. This makes the range image space a non-Cartesian frame which can be a spherical frame or an even more complex frame depending on the particular device in use [4]. Each range image becomes a scattered array, when mapped into a Cartesian reference.

When a scene includes multiple objects and or when object surfaces involve deeply curved features, these scattered data points in the Cartesian frame will cause the following obstacles for handling discrete 3D data points and generating geometric models:

- complex algorithms have to be developed to cope with the sparse array formed by the scattered data since the linear functions, $x(i) = x_0 + i\triangle x$ and $y(j) = y_0 + j\triangle y$, are no longer applicable;
- surface discontinuity detection methods, often based on uniformly distributed data, cannot be used, since it is difficult to identify the difference between nonuniform point intervals due to the central projection based acquisition and point jumps due to occlusions; and
- it is difficult to detect missing data incurred by failure to extract the laser signal from a background image.

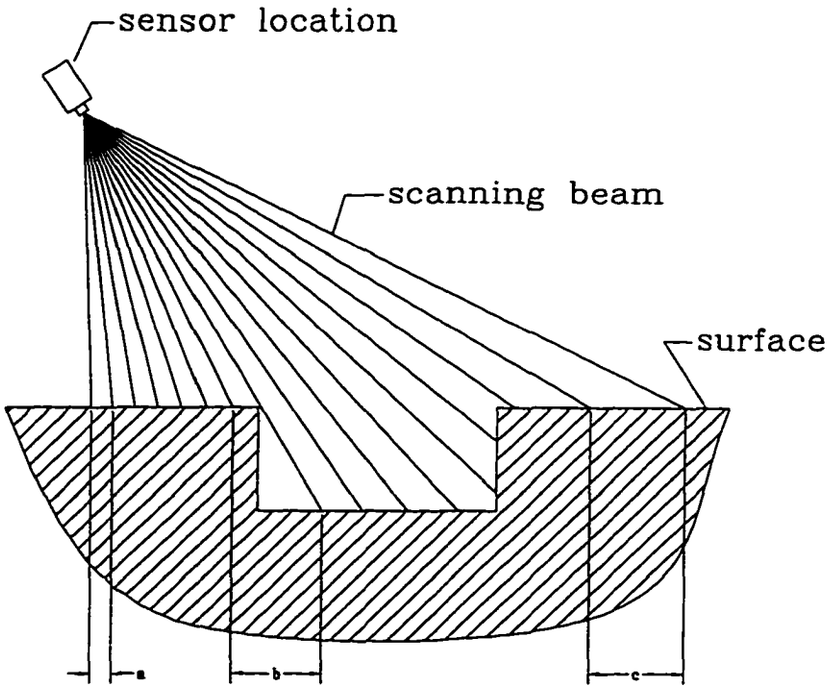


Figure 1.5: Unequal spaced imaging sampling and image shadow.

Surface discontinuity is usually detected by finding nonuniform distributions of acquired data points. However, there exist other possibilities that can make the distribution ununiform which can result in false detection. These potential factors include unequal spaced imaging sampling and image shadow (shown in Figure 1.5), and uneven surface reflection. With an equal scanning angle the sampling space is unequal. The further that the scanning beam reaches, the longer the resulting sampling interval with respect to a Cartesian frame. When surface reflection is very intensive or very weak, or the incident angle of the scanning beam is very large, a range finder may fail to extract range data by missing data.

If a triangulation based range sensor is used, shadow effects may cause the scattered data to be distributed even more irregularly. Figure 1.6 illustrates this problem which is caused by the structure of the sensors where the sensor and the laser are distant due to the base line required for applying the triangulation principle. The

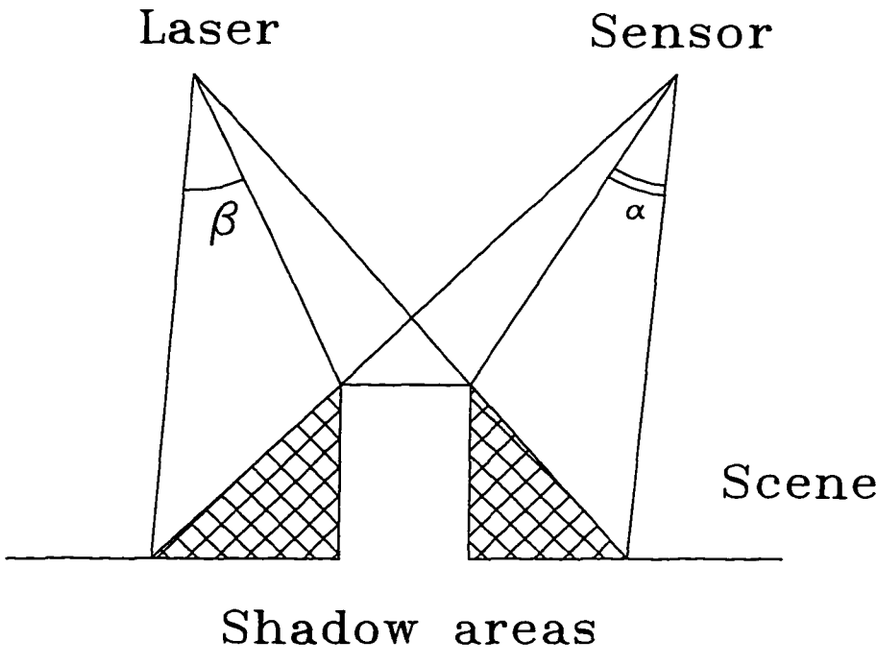


Figure 1.6: Shadow effect of triangulation based range sensors

scene may hide the laser beams in the shadow areas, e.g., corresponding to the sensing angles within the angle α, and/or hide the sensor in other shadow area corresponding to the scanning angles within the angle β, as indicated in Figure 1.6. When the base line is long to improve measurement results, the problem gets worse. Although this phenomenon can be reduced using synchronized scanning where shadow effects are improved significantly compared to simple triangulation, shadow effects are still present and must be addressed [22]. After a sudden change of range value brought about by one of above reasons is transformed to a Cartesian frame, since the change is delivered among three coordinates, a sophisticated detection algorithm able to differentiate such changes from real changes due to surface discontinuity has to be used to provide accurate models.

To avoid the difficulty of handling scanned data in a Cartesian frame, Kweon and Kanade [38] proposed concurrently to this research a method with their 1D elevation

map representation to generate a map in the sensing space of the ERIM sensor.

1.5.2 Research Objectives

The research of this dissertation is focused on:

- Development of a representation scheme which enables the modeling system to provide applications with surface models based on range image data. Objectives of the scheme include reduction of the computational complexity during modeling and facilitation of the conversion from discrete range images to geometric models. In addition, the scheme should be able to represent the model of an object with bounded surface and the model of a scene with unbounded surfaces.

  Surface models based on such a representation should convey both geometric and topological information. Although directly obtaining highly meaningful models from images is difficult, extraction of the topological information about space occupation from the raw range data should be a capability of the algorithm to be developed, e.g., identification of the inner and outer space of an object. This topological information is critical for many industrial applications, such as robotic systems and NC machining, and can also be necessary for upgrading low-level representations to higher-level representations.

- Another important issue is multiple-view fusion that allows dynamic growth of geometric models through partial model generation and model integration. After a partial model for a piece of surface region is generated from a view of image, it should be able to be merged with a global model. This is an image-driven based fusion approach, i.e.: 1) the selection of view location is flexible; 2) one or more partial models can be disconnected with the current global model, if

the related images do not have overlaps with others; and 3) as long as the range images cover the entire surface of an object or a scene, their merged surface model will be complete.

This multiple-view fusion approach should be developed with the ability to detect image overlaps and to merge data redundancy. When a scene has a large unbounded surface, its global model will be large and will have a huge amount of data. Finding potential overlaps has previously relied upon searching for the match of the partial model from the global model. The developed fusion approach should avoid this intensive search for overlap detection to make the relevant computation manageable.

- The research will investigate the possibility of using the characteristics associated with the sensor space for facilitating the computation of cross-section contour modeling. The uniformly distributed data present in the sensor space (device space) may increase the efficiency of the conversion from image data to a model. If the partial modeling is performed in the device frame, the costs for detecting surface discontinuity will be reduced.

- To prove feasibility of the developed system for applications, the system should be tested with multiple types of range sensors including imaging radar based and triangulation based imaging systems. These imaging systems are commonly used in practice and have commercial products available. For generating partial models in a device frame, the device frame for different sensors must be derived.

1.6 Outline of Remaining Chapters

Chapter 2: Development of the Representation –

Different representations handle geometric data in different levels. To describe objects based on discrete range data of surface points and to provide applications with relatively concise and complete geometric models of objects, the criteria for selecting an appropriate representation are primarily considered, including: 1) compatibility with discrete range data; 2) suitability for modeling in the device frame; and 3) capability of facilitating multiple-view fusion and model updating. Based on the evaluation and comparison of different potential representations, the cross-section contour approach in an intermediate level is found to be a preferred representation scheme. This representation is utilized by the developed system. The development of algorithms to satisfy the specific requirements for vision data based geometric modeling and the implementation of a testing of the developed system are discussed in subsequent chapters.

Chapter 3: Partial Model Generation in The Device Frame –

The difficulty of handling range data in a Cartesian frame is caused by the central-projection oriented working mechanism of range sensors. To avoid irregularly distributed data, a new method which takes advantage of the well-distributed data in the range image space is developed. The approach generates contour models in the range image space where the image data is well distributed. The approach transforms the cross-section planes into the range image space generating synthesized plane range images, and detects the intersection contours by comparing the object range images and the synthesized plane range images.

Chapter 4: Implementation of Associated Device Frames –

The implementation of the device-dependent algorithm is closely related to the characteristics of the specific range finders used with the modeling system. These characteristics include the scanning mechanism and device geometry of a range finder. The development of the device-dependent functional components for generating partial models is based upon these characteristics. The implementation of the device frame with individual typical range sensors is necessary to prove the feasibility of the developed algorithm. The device frames of three typical range sensors are derived, including an imaging radar based sensor, the ERIM range finder; and two triangulation based laser scanners, a range-finder developed at the University of Victoria (the SmartEye) and the HYSCAN$^{(R)}$ scanner. The algorithm is tested with these sensors and their associated device frames.

Chapter 5: Global Model Generation with Multiple-View Fusion –

Each range image can describe a partial object surface. After each range image is taken, a partial surface model is constructed and is connected with the previous ones to update the model. This procedure can be expressed by the *OR* boolean operation. Once the multiple range images cover an entire object shape, a complete surface model in a representation form can be achieved.

Chapter 6: Conclusions And Future Research Suggestions –

In this last chapter the new modeling approach and the implementation of the modeling system are summarized. Based on the summary, conclusions are presented. Some suggestions for potential future research are given as well.

Chapter 2

Development of the Representation for the Modeling System

2.1 Overview

A geometric modeling system needs a representation to describe shapes. The selection of the representation is determined by the functions to be performed by the system. Since a range data based modeling system functions in a different way from traditional geometric modeling systems, converting discrete range data to a geometric model rather than forming a model from high-level geometric description, the selection of an appropriate representation has stricter imposed constraints. The major constraint is that the representation should be compatible with range data and suitable for multiple-view fusion. Section 2.2 describes these constraints and requirements of a range vision data based system.

An advanced vision-data based modeling scheme using a high-level representation can provide applications with any required characteristics of modeled objects. Such a modeling scheme will be required to initially obtain sufficient geometric information used by the representation in terms of local features and global characteristics. However, obtaining this global information often requires data integration by multiple-view fusion, a fusion which relies on the support from a modeling system. It is difficult for these two mutually dependent requirements of obtaining initial global information and generating global models to be satisfied simultaneously within the same modeling system.

A primitive modeling system with a low-level representation has very limited capabilities for applications, although a primitive representation can facilitate conversion of the raw image data into geometric models. A modeling system based upon such a low-level representation usually can merely provide some primitive information for subsequent applications. Representation alternatives in different levels are evaluated and compared to select one for the requirements in Section 2.3.

With proper design an intermediate-level representation based modeling system is able to provide adequate geometric information while keeping good compatibility with the low-level range image data. As well, an appropriate intermediate-level representation can facilitate upgrading to models of higher level to enhance the representation capability. By the evaluation and comparison, the cross-section contour representation is found to be such an ideal representation scheme. The details for the representation is given in Section 2.4. This representation is utilized by the developed system to satisfy the specific requirements for vision data based geometric modeling. The implementation of the cross-section scheme and the structure of the modeling system mainly reflects the essential needs for multiple-view fusion, dynamic growth based modeling and for supplying geometric and topological information as

discussed in Section 2.5.

2.2 Representation Requirements for Multiple-View Range-Image Fusion Based Modeling

2.2.1 World, Sensor and Modeling Spaces

The major task of a vision data based geometric modeling system is to describe the geometry of the real world. From observing an existing object to obtaining its computer based model, two processing stages are required: sensing and modeling stages. These two stages bridge three different spaces: the real world; sensor; and modeling spaces. The sensing activity is associated with the real world and sensor spaces. The modeling processing is related to the sensor and modeling spaces.

In general, the surface (Ψ_{real}) of an object in the real world can be described by

$$\Psi_{real} = \{(x, y, z) \mid F(x, y, z) = 0\}$$

where (x, y, z) is a point on the surface with respect to a global reference, usually a Cartesian frame, and F is a function. As simple example cases, F can be

$$F_{plane}(x, y, z) = Ax + By + Cz + D$$

for a planar surface or

$$F_{quadric}(x, y, z) = Ax^2 + By^2 + Cz^2 + Gxy + Hyz + Izx + Ux + Vy + Wz + D$$

for a quadric surface. Besides using an implicit function F, an alternative form is to use a parametric surface description

$$\Psi_{real} = \{(x, y, z) \mid x = X(u, v), y = Y(u, v), z = Z(u, v)\}$$

where u and v are parametric variables.

In sensor space the surface Ψ_{image} is specified by a set of range image points (pixels)

$$\Psi_{image} = \{\mathbf{d}\}$$

where $\mathbf{d}$ is an image point with respect to a local coordinate reference called the device frame in this dissertation. As well, the surface can be specified by a set of subsets of image points, when the scene is observed from multiple views

$$\Psi_{image} = \{\psi \mid \psi = \{\mathbf{d}\}\}$$

where ψ is one range image from the multiple views. The sensing space is a discrete space. The device frame is device dependent.

From the sensor review in Section 1.4.3, it can be found that the transformation from the real world to the sensing space is generally perspective-transformation oriented for both image radar based and triangulation based range sensors. This transformation F_s is found from the sensor geometry and calibration. The function as a transformation of sensed data from the real world to the sensor spaces is as

$$F_s : \Psi_{real} \longrightarrow \Psi_{image}, \qquad \mathbf{d} = F_s(x, y, z)$$

These image data $\mathbf{d}$ of the surface points for one or more views are stored in one or more arrays in a well-organized structure. However, the original scanned surface points (x, y, z) in the real world are irregularly distributed due to the perspective projection employed by sensors. These points transformed into the modeling space are also irregularly distributed, since both the modeling space and the real world space are generally with respect to Cartesian frames.

In the modeling space the surface description is extracted from the set of data points of the sensing space based surface description. The extraction is a mapping

between the two spaces. Geometric models of objects and/or scenes in the modeling space can be in a discrete, continuous or hybrid form, depending on what kind representation is used to represent the models. Therefore, the Ψ_{model} description will accordingly be in different forms.

2.2.2 Representation Compatibility with Range Image Data

A representation is a scheme used for specifying and storing a model of viewed scene in a computer database. To describe characteristics of surfaces a representation scheme can use various approaches including analytical expressions such as equations for surfaces or curves, nonanalytical methods such as a graph to show relations of surface components, and geometric primitives with simple shapes.

Using multiple approaches can represent more surface details in the models, and provide more flexibility for accessing the surface models for rendering, boundary condition searching, feature extraction and other tasks. In general, representations can be classified into different levels with respect to their capabilities of representing geometric complexity for objects and of providing sufficient information for applications.

An approach for mapping is required in the modeling stage to transform the data from the sensor space into the modeling space. The form of the mapping is specified by the definition of the representation to be used. This mapping approach functions as a modeler which converts the low-level discrete range image data to higher-level models. The complexity of the approach is mainly determined by the representation.

From the application point of view, high-level models which require sophisticated representations are usually desirable. However, the more sophisticated the

representation scheme used, the more difficult it is to implement the mapping approach since the discrete image data does not provide sufficient meaningful information. This fact is a significant difference between traditional modeling systems and vision-image data based modeling systems.

Therefore, if the representation is sophisticated to provide applications with high-level models, the data mapping can become very difficult or even impossible. When models are represented in a relatively low level, the above difficulty can be avoided. Such a modeling system uses simple elements to describe shape of objects. Models in low level are not compact and need a large amount of simple geometric entities. This means that a huge memory space may be required and that little meaningful information may be available.

2.2.3 Representation Suitability for Multiple-View Fusion

Multiple-view fusion is another challenge for vision-data based modeling systems. The vision data (such as a range image) from a view can only provide the initial description for local surfaces of a scene. Among multiple views overlapped image data and/or data gaps may exist, since scanned areas associated with different sensor locations may not be adjacent to each other perfectly, and since object occlusions may exist as well. It is difficult to integrate image data from multiple views in the sensing space, since the data fusion requires a global reference frame.

The data mapping from the sensing space to modeling space is based on individual images. The model generated from each image is a partial model for a local part of the scene. Partial modeling requires the representation to be able to handle a local partial model without prior knowledge about the global model. To obtain a complete model the model integration must be done in the modeling space.

2.2.4 Representation Requirements for a Vision-Data Based Modeling System

From the above description, it can be found that selection of a representation scheme for a vision-data based modeling system is imposed by more constraints than for traditional geometric modeling systems. Such an approach must be capable of coping with the nature of image data during modeling processing. Two important factors which do not affect traditional modeling systems have to be considered by a vision-data based modeling system, including 1) information for the geometric elements which are used by the representation scheme is not available explicitly, and 2) location and topology relationships of representing elements are unknown. The information required has to be extracted and interpreted from multiple images that provide the initial input for the modeling system. The geometric modeling approach for generating models from multiple-view range-image data requires that a representation have:

- Compatibility with discrete range data – Since a range sensor only provides surface image data in low level, a representation is required to be able to directly construct geometric models without more information and without the needs for intensive recognition/interpretation activities. As well, such a representation scheme should provide relatively concise geometric models in relatively high level to yield compact storage needs.

- Suitability for partial modeling – The representation should be able to allow partial modeling and to make partial modeling independent of global modeling. Partial modeling involves the ability to describe both a part of an object with closed surface and a unbounded surrounding scene.

- Capacity for facilitating multiple-view fusion and model updating – The representation must allow a model updating method to merge partial models together, and allow the associated updating computation to be relatively low cost. Model updating includes two steps: 1) detection of redundant data represented by overlapped partial models and 2) merging of the overlaps. To increase measurement reliability and accuracy, a local scene may be scanned by many images in practice. This situation can greatly raise computational burdens. An appropriate representation should be able to avoid this massive computation.

- Adaptability of shape details in range data – The representation should be able to match the shape details that can be determined by a sensor's scanning resolution and viewing distance. When a sensor is close to the object surface, more details will be carried by the range image. Inversely, when a sensor is far away from the object surface, fewer details will be carried. A representation should be capable of meeting this change in detail.

- Capability of rendering object surfaces – The representation should be able to provide the location of any surface point in any reference system with respect to the resolution available in the original image data. Rendering is often needed by applications, such as object recognition by matching rendered data and sensed data, and robot collision detection.

2.3 Representation Alternatives

Different types of representations must be evaluated to find an appropriate representation for compatibility with discrete range images and capability of providing geometric and topological information in relatively high level.

Object shapes can be described using various geometric elements, such as discrete surface points, surface curvatures, surface boundaries, surface equations, volumes and/or object components. These elements reflect object shapes in different levels in terms of various geometric properties. When one or more of these elements are used by a representation scheme, the modeling capability of the representation is primarily determined. These elements can be used individually, as well as combined.

2.3.1 Line based

Line based representations, such as wireframe, are simple schemes. Wireframes are intersections of adjacent surfaces of an object, and can be lines and/or curves. Converting 3D images to this description form requires segmenting each of the images to find C^1 discontinuities for the surface intersections. Model integration is based on merging line/curve sections.

Although wireframe representation is not computationally intensive for manipulating the models, this scheme is not able to lead to models with sufficient geometric information. Usually, surface descriptions are not available in this representation, especially for objects with curved surfaces. As well, topological relationships of surfaces are not represented. This representation is not proper for the required modeling system.

2.3.2 Surface based

This class of representations is able to describe surface shapes of objects, and includes parametric approximation, mesh, sweep and cross-section contour schemes.

B-spline and NURBS are commonly-used parameterized approaches. These representations use polynomials or the algebraic ratio of two polynomials to describe

surface patches. The parameters of the polynomial(s) are determined by some control vertices which can be recursively derived from surface points captured in a range image. The problem to use such a scheme is associated with multiple-view fusion. It is difficult to detect data overlaps and to locate control vertices for connecting surface patches uniformly without using an intermediate representing scheme.

Mesh represents a surface by surface points and spatial grids. Mesh can be used by linking the surface points with a graph to form a polyhedron of three-node or four-node grids. It is flexible for handling complex surfaces and unbounded surfaces. This representation is able to deal with both partial modeling and multiple-view fusion computation. Building a mesh surface model requires setting up the neighborhood relationship of image points for generating the grids. When merging data from different views, finding the relationship is based on the computation of searching adjacent points in three-dimensional space according to the relevant rules. However, such a computation is costly, since the computation has to consider the possible adjacent connection of a point in one view with all points in other views. As well, this computation is performed in 3D.

Cross-section contour and sweep are 2D based surface representation schemes. These schemes use a series of 2D section contours to approximate 3D shapes. With some computation a series of cross-section contours can be formed from object surfaces described by range image points. The compatibility with range data is maintained by computing the intersection between range image data based surface and cross-section planes. Difficulties of handling shape details, data condensation, partial modeling and fusion are reduced due to working in the 2D space of each cross-section plane. Sweep is similar to the cross-section scheme except each section shape is the same as others or changes regularly. Obviously, sweep lacks the flexibility to represent irregular shapes.

2.3.3 Volume based

Volume representations are based on the decomposition of the space occupied by objects. The space is decomposed into cells which usually are cubes. These cubes can be the same or different sizes, and are used to approximate the object volume.

3D voxel array (a digitized space) is such a representation scheme. Objects are depicted by a set of discrete volume pixels (cells) in low level. One voxel is adjacent to its neighbors in the digitized space, otherwise the description is not complete. It is compatible with range data due to the similarity between the digitized space and image space. Converting the raw image data to the relevant model only relies on a mapping. With this scheme the data fusion is easy. However, its memory consumption is significant, especially when a quality resolution is needed; and its digitized model space lacks of flexibility to fit objects. It is also difficult to adjust the digitized space resolution to cope with surface details in range images when varying the distance between sensor and surface. Since it is in low level, the geometric description conveyed in the 3D array cannot easily be utilized. In addition, the computation to manipulate such a model is intensive.

Octree representation is another volume based scheme. The volume of an object is represented by this representation in a discrete form. This scheme encodes the cells by recursively subdividing the volume enclosed by the object surface to form different sizes of cubes for making the space decomposition compact. The recursive spatial subdivision requires, prior to processing, knowledge of dimensions that yield a cube which completely encloses the object to be modeled. This requirement indicates that complete data acquisition must be done before modeling. Therefore, an octree representation precludes the possibility of partial modeling for update based model generation.

2.3.4 Boundary based

B-Rep which utilizes boundaries to describe the geometry of objects is a widely used representation. Models in this representation are concise and in high level, and convey both geometric and topological information. A boundary of an object includes surface facets and surface borders. Linkages between borders and facets are required. Surface facets are usually described by analytical and/or parametric expressions. Borders are represented by the intersections between facets using links. Converting range image data into B-rep based models is computationally intensive. It requires image segmentation for generating surface descriptions and computation for obtaining intersection borders. Although a partial model can be generated based on one view, model integration is very difficult. The integration involves surface and border merging. The required complicated computations can render a geometric modeling system of low reliability.

2.3.5 Object based

Constructive Solid Geometry (CSG) is an advanced representation scheme. This representation describes an object using its component parts (geometric primitives) and boolean operations. The connections of the involved primitives for the boolean computation are expressed by a graph. Most object properties, geometric and topologic, can be represented. Relationships between all parts are explicitly required. These requirements complicate the mapping for converting the original range image into the model within this representation.

2.3.6 Comments

Some schemes commonly used in the area of computer vision, like General Cone (GC), extended gaussian image, geon method and superquadric description, are not considered in the previous subsections. These representations were originally designed for object recognition purposes. They have advantages for facilitating recognition computation, since geometric details can be ignored. However, they lack the capability of describing complex shape details. As well, these representations do not consider flexible sensor locations, multiple viewing observation and unbounded space description. These representations are not adequate for the applications of robotic systems and advanced manufacturing.

Representation schemes, such as wireframe and voxel representations, using simple elements to describe geometries of objects have the flexibility to interface with discrete range image data. These schemes, however, have significant limitations in ability to handle geometric details due to their low level status.

Advanced schemes including boundary-based and object-based schemes have the capability of modeling complicated objects. However, these advanced schemes require complex computations to extract information from raw image data for model generation. Development of such computation algorithms and their robustness are currently research topics in the field of computer vision. In addition, some research [42] showed that these representation schemes would have to rely on an intermediate representation for generation of models from the raw image data prior to upgrading the models into high level.

Intermediate representation schemes have the advantages of: 1) being relatively easy to interface with the raw data, and 2) being able to provide applications with relatively sufficient and high level geometric models.

Of such intermediate representations including cross-section contours, mesh and octree, cross-section contour based representation is better than others in the following aspects [71]:

- Partial modeling – A model can be built based on a single view of range image. Sectional contours allow both closed and unbounded surfaces to be modeled.
- Model updating – The representation allows the concurrent performance of image acquisition and model generation by merging a partial model from an individual view to the existing model. Consequently, massive multiple-view observation becomes applicable without requiring the saving of large amount of raw image data. As well, the computation of the model integration is simplified to a 2D problem in section planes.
- Detail description – The scheme has flexibility to adjust its resolution by changing the number of sections and the number of contour nodes for fitting geometric details and model growth.

The cross-section contour representation is able to represent surfaces, either concave or convex and either open or closed. Cross-section related representations have been used for describing surfaces in robotic navigation, surface reconstruction, NC machining tool path description, coordinate measuring matching probe path generation, and medical imaging systems.

When range images are used to generate cross-section models for existing objects, the data points along each cross-section contour curve must be extracted from each range image. Using these data points, each contour can be formed by linear interpolation, spline fitting or other curve fitting techniques.

Since the cross-section contour scheme is in an intermediate level, the representation has its limitations to handle some special cases, including the

discontinuous surface description between sections and describing surfaces whose surface normal is close to parallel of the section plane normal. However, these limitations can be eliminated or reduced by increasing the section density and by varying the section orientation.

2.4 Cross-Section Based Representation

Considering the requirements for developing an appropriate representation scheme used for this vision-data based geometric modeling system, different representations have been compared. The conclusion is that a surface based intermediate-level representation is most suitable. Based on the modeling requirements, the cross-section representation has more advantages than other surface based representations in terms of image data compatibility, description of shape details and complex surfaces, and computational efficiency of multiple-view fusion.

2.4.1 The Fundamentals of the Cross-Section Representation

Cross-section representation uses 2D surface contours lying on a series of planes to represent object shapes. The contour will be called a cross-section curve; and the plane a cross-section plane (or plane). Cross-section planes are arranged along a guide curve with the planes perpendicular to the guide curve. If the guide curve is a straight line, the planes are a group of parallel planes.

A simple object represented by the cross-sections is illustrated in Figure 2.1 where the cross-section curves, the cross-section planes and the guide curve are shown.

The cross-section curve corresponds to the intersection curve of the cross-section plane and the surface of the observed object and/or the surface of the workspace environment. The points of a range image can be used to generate a piecewise-plane based description for the surface covered by the image. Each piece of plane is bounded by three or four adjacent image points. The original description of cross-section curves is generated by computing the intersection lines between a cross-section plane and the piecewise planes.

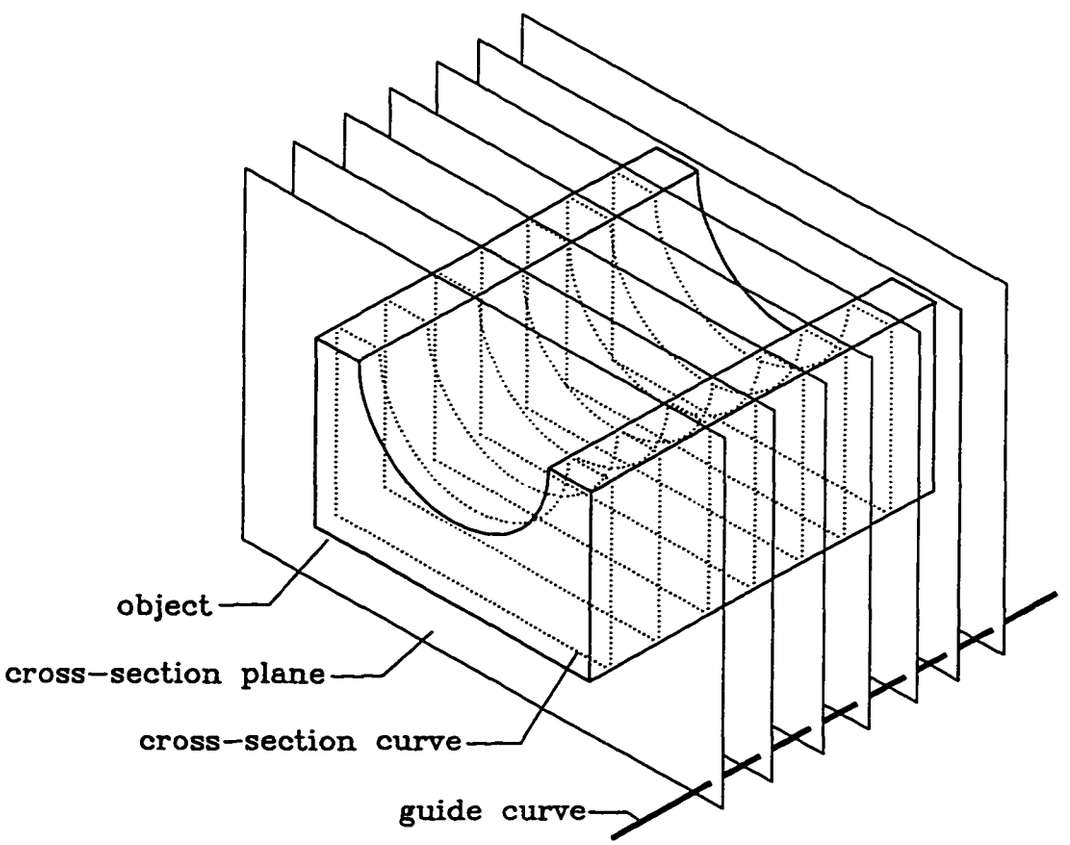


Figure 2.1: The cross-section contour model of a simple object.

The topology of a cross-section contour curve is related to the surface of an object and to the region of a scene to be modeled. If an object surface is enclosed, the

corresponding merged cross-section curves of a complete model will be closed, while if a scene is unbounded, the cross-section curves may be open.

This research is focused on finding and representing cross-section contours using line segments connecting these intersection points. Other descriptions for contours can be derived from the original, e.g., B-spline functions can be found by computing the control vertices from intersection points on the contour.

A surface based approximation model for object's surface can be obtained by interpolating the interval between two neighboring curves. Since exterior and interior sides of object boundary are identified by the curves, the object volume can be computed. When the object surface changes greatly, surface details can be preserved by decreasing the interval of the planes.

A modeling frame (space) needs to be introduced for using the cross-section based representation to generate surface models Ψ_{model}. This modeling frame is a global 3D Cartesian coordinate system. The selection of the modeling frame can be associated with an application to facilitate the relevant computations for the application.

2.4.2 Model within the Cross-Section Representation

With respect to the modeling frame the guide curve and the cross-section planes are described. In order to present the guide curve a vector function is introduced for facilitating the following discussion. Let the guide curve be expressed by

$$\mathbf{g} = \mathbf{g}(u)$$

where u is a parameter, and $\mathbf{g}$ has three components: g_x, g_y and g_z.

The derivative, with respect to u, $\dot{\mathbf{g}} = (\dot{g}_x, \dot{g}_y, \dot{g}_z)$ of the function $\mathbf{g}$ yields as the expression of the normal vectors of the cross-section planes. Thereafter, a plane is

specified by

$$\dot{\mathbf{g}}_p \bullet (\mathbf{p} - \mathbf{g}_p) = 0$$

where $\mathbf{p}$ and $\mathbf{g}_p$ are two different points on the plane. Without loosing the generality, $\mathbf{g}_p$ can be the intersection point between the guide curve and the plane, i.e., $\mathbf{g}_p = \mathbf{g}(u_p)$. Both of $\mathbf{p}$ and $\mathbf{g}_p$ are positional vectors with respect to the global Cartesian modeling frame. These vectors and the derivative $\dot{\mathbf{g}}_p$ are shown in Figure 2.2.

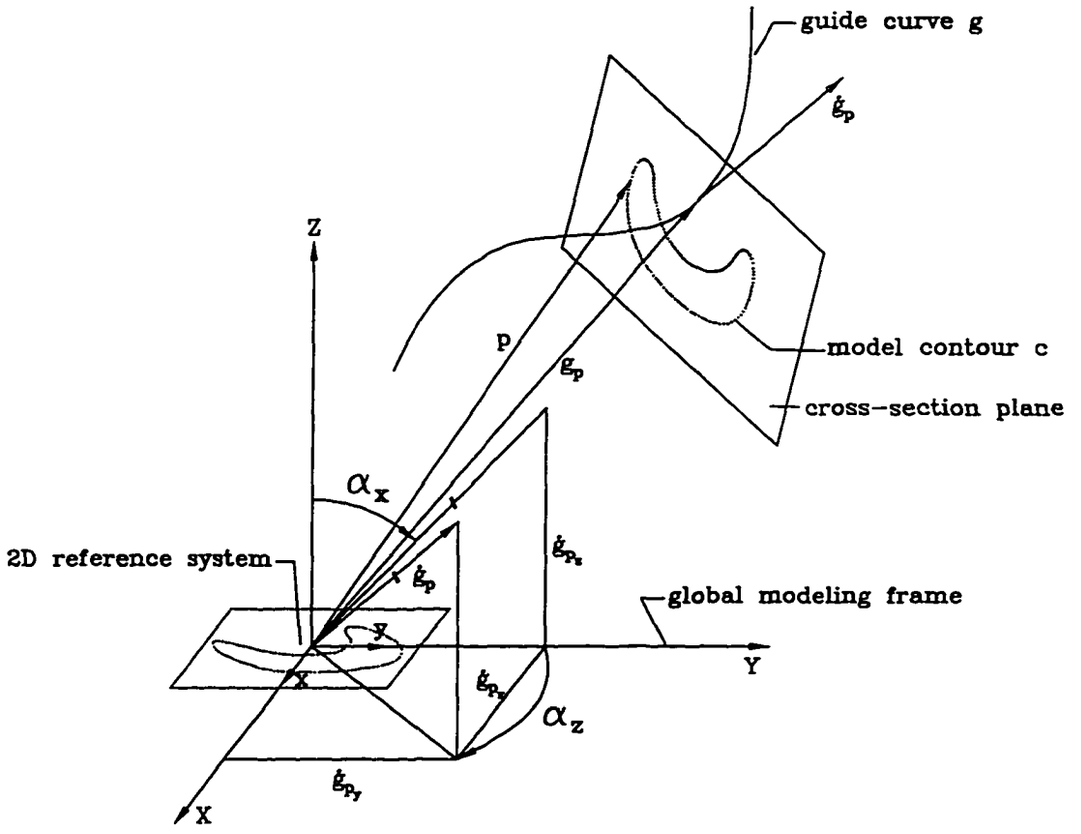


Figure 2.2: Geometric relationship between the plane based reference system and the global modeling frame.

For reducing the complexity of the algorithm for merging data from multiple views, each cross-section contour needs to be described in a 2D reference system, that is,

a plane-based local frame associated with the involved cross-section plane. This 2D reference system is coincident with the $X-0-Y$ coordinate plane of the global modeling frame, i.e., the plane normal with this reference system is coincident with Z axis.

The transformation between this local reference frame and the global modeling frame is determined by $\dot{\mathbf{g}}_p$ and $\mathbf{g}_p$. The point specified by $\mathbf{g}_p$ can be used as the origin of the 2D local frame. All cross-section contours $\mathbf{c}_p$ generated in the plane are expressed based on this origin. The expression $\mathbf{c}$ for these contours in the global modeling frame can be found

$$\mathbf{c} = \mathbf{c}_p[T] + \mathbf{g}_p$$

The rotational matrix $[T]$ and the vector $\mathbf{g}_p$ rotates and translates $\mathbf{c}_p$ from the 2D reference system to the global modeling frame. The matrix $[T]$ includes two rotation steps: 1) rotating by α_x about X axis and 2) by α_z about Z axis

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_x & \sin\alpha_x & 0 \\ 0 & -\sin\alpha_x & \cos\alpha_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha_z & \sin\alpha_z & 0 & 0 \\ -\sin\alpha_z & \cos\alpha_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $cos\alpha_x = \frac{\dot{g}_{p_x}}{|\dot{g}_p|}$, $sin\alpha_x = \frac{|\dot{g}_{p_{xy}}|}{|\dot{g}_p|}$, $cos\alpha_z = \frac{\dot{g}_{p_y}}{|\dot{g}_{p_{xy}}|}$ and $sin\alpha_z = \frac{\dot{g}_{p_x}}{|\dot{g}_{p_{xy}}|}$, and $|\dot{g}_{p_{xy}}| = \sqrt{\dot{g}_{p_x}^2 + \dot{g}_{p_y}^2}$.

From the individual cross-section curve model $\mathbf{c}$, the corresponding surface model Ψ_{model} with respect to the global modeling frame can be derived

$$\Psi_{model} = \{\{\mathbf{c}\} \mid \text{in all cross-section planes}\} \tag{2.1}$$

In each cross-section plane the contour curve may have multiple sections, if some multiple-view range images are not connected with others. Therefore, a set of cross-section contour curves in each plane is indicated in the expression (2.1). For the same reason, the cross-section plane series may not be a continuous series. There may be multiple sub-sets of the planes along the guide curve.

2.4.3 Topological Information Provided by the Representation

The surfaces of objects split the whole modeling space into two parts: the interior volume and exterior space. If no more information is given, a cross-section contour model provides the description of the surface only without indicating the space separation. In applications knowing this separation is sometimes important, especially for automatic decision making like collision detection. To provide applications with a more useful description about objects the topological information carried by the models must be implemented for indicating the above space difference.

A cross-section contour is the intersection between an object surface and a cross-section plane. The contour divides the plane region into two parts. The part of the plane which is on one side of the contour curve is inside the object, and the other part of the plane which is on the opposite side of the contour is outside the object. Therefore, on each cross-section plane in the near proximity of the contour curve there are two half spaces which are in the interior and exterior object volumes, respectively.

If a cross-section contour curve can be used as the boundary to identify the different spaces, the required information will become available. Based on this fact, a curve direction is defined as follows: if one stands on the positive side of the plane normal of a cross-section plane and walks along a contour curve in the curve direction,

the left-hand side should be inside the object and the right-hand side be outside the object. This definition guarantees that the desired topological information can be provided by the specific curve direction.

2.5 Implementation of The Representation and System

2.5.1 The Modeling System with the Representation

A prototyping system for range data based geometric modeling, with multiple-view image fusion, is developed through this research. The developed system takes range images as the original input and generates cross-section contour based geometric models. The modeling system is designed for either working by stand-alone or for cooperating with another application system (e.g., a CAD/CAM system or a robotics system) as a built-in subsystem.

The system includes three components: a data base for storing geometric models; a modeler for generating models from range images; and an interface to AutoCAD (a commercial CAD system) for displaying and manipulating models, and/or for the further applications. In addition, this research has also implemented a range imaging system including three different range finders for testing the functionalities of the modeler. Figure 2.3 shows the structure and data flow of the modeling system with these components.

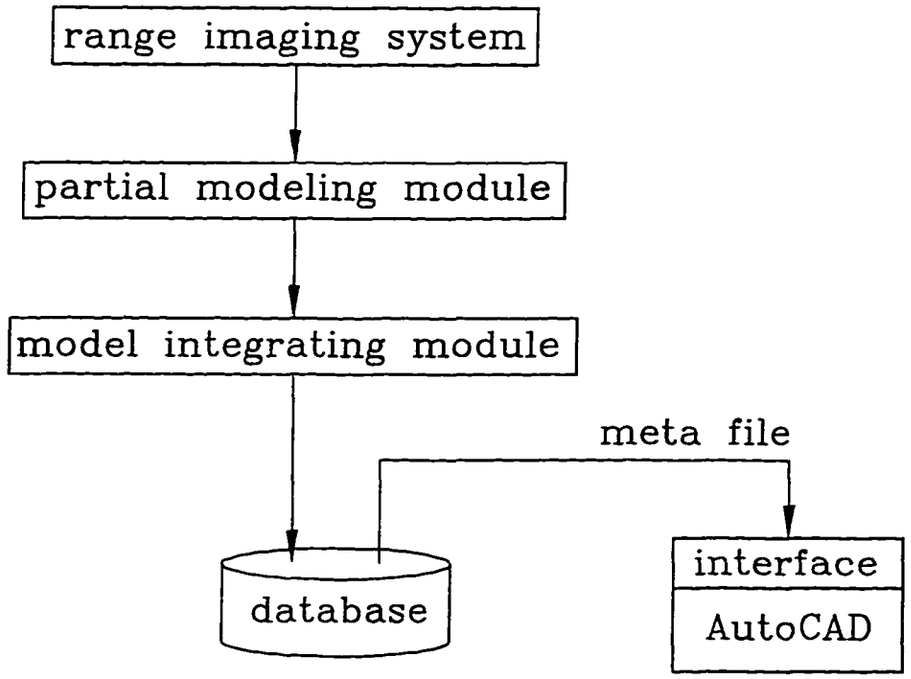


Figure 2.3: The range image based geometric modeling system

2.5.2 The Database

The database stores and maintains models generated by the modeler. The database has the data source which implements the cross-section contour representation and the database driver functions that increases, inserts, deletes and retrieves contour nodes and curve sections.

The data source is composed of components in three levels corresponding to three important types of modeling elements of the cross-section scheme, i.e., cross-section planes, contour curve sections and intersection contour points, shown in Figure 2.4. These levels form a tree structure. At the beginning of a modeling task, there is only the root available. This structure, i.e., the model, will dynamically grow during the taking of multiple views of the scene, so that the planes will gradually cover the whole scene and the curve sections eventually describe all surfaces.

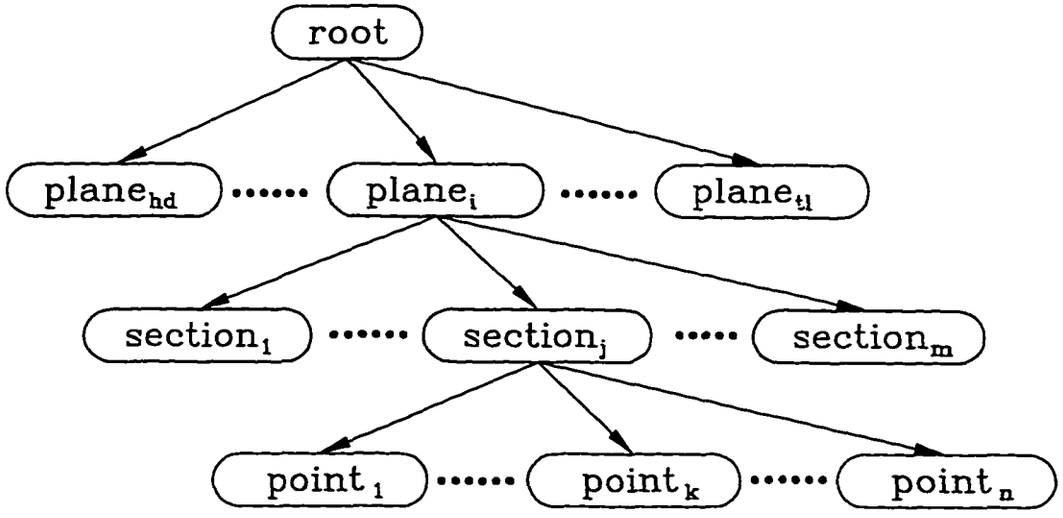


Figure 2.4: The structure of the database in three levels.

In the bottom level, contour points are stored as the leaf nodes of the tree. In the middle level, contour sections are recorded. A section can be an individual contour curve representing the intersection of the object surface and the cross-section plane, or it can be a temporary segment of a curve section due to insufficient image data. After multiple-views, range images gradually cover the whole surface and the temporary sections being merged together will construct a single curve. Under the root it is the plane level. During the partial modeling one plane is involved each time for the computation of intersection. The interval between planes is adjustable to ensure the requirement for resolution. The partial modeling results in a sub-tree with a plane node and some child (section) and grandchild (point) nodes. The integration from a partial model to the global model means that such a sub-tree will be merged into the global tree. Each plane node and section node also includes data for the enclosure box which indicates the minimum and maximum x and y values of all contour points in the plane or in the section. This information is useful for model integration, and is dynamically updated when the global model is growing.

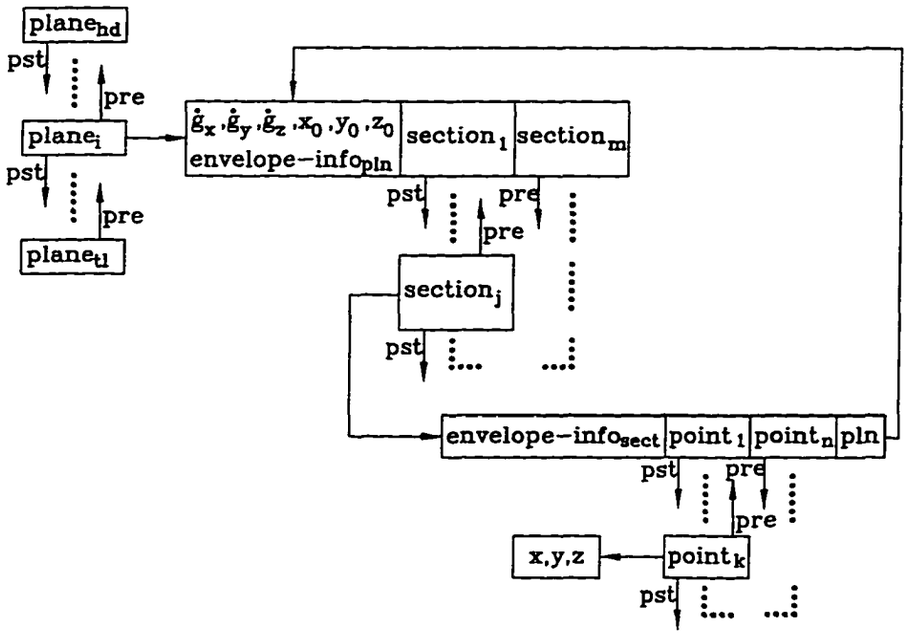


Figure 2.5: The implementation of cross-section representation.

Figure 2.5 illustrates the details of the database structure, including the linkages of nodes in the same level and in different levels and including the geometric information. Since the partial modeling needs to find planes which may have intersections with the local surface scanned by a view, searching the plane nodes is necessary. For facilitating the search, all nodes in the plane level are linked by a bi-linkage: pointer *pre* and *pst*. The data, $\dot{g}_x$, $\dot{g}_y$, $\dot{g}_z$, x_0, y_0, and z_0, used to transform model curves between the plane based local reference frame and the global modeling frame is bounded within the node. Similarly, the forward and backward searches are often required during the merging of models for multiple-view fusion. Therefore, the bi-linkage pointers are used for sections and contour points as well. The envelope information in the plane and section nodes is used to reduce the search scope during the multiple-view fusion.

The cross-section contour scheme is able to provide the topological information about the interior and exterior space divided by object surfaces. The two sides are

indicated by the direction of cross-section curves. This direction is implemented by the backward linkage of the contour points in the bottom level. Let the current point be $\mathbf{p}_i = (x_i, y_i)$. Along the curve direction, the next point pointed to by the pointer *pst* of the node $\mathbf{p}_i$ will be $\mathbf{p}_{i+1} = (x_{i+1}, y_{i+1})$. The vector $\mathbf{p}_{i+1} - \mathbf{p}_i$ gives a linear equation: $(y_{i+1} - y_i)x + (x_i - x_{i+1})y + (x_{i+1}y_i - x_i y_{i+1}) = 0$. Any point (x, y), if it is on the right hand side of the vector, will make the left side of the equation larger than zero, otherwise, smaller than zero. Therefore, when a point makes the result positive (negative), it is outside (inside) of the volume bounded by the object surface.

2.5.3 The Modeler and Interface

The modeler converts each range image into a cross-section based model and merges this model with the models previously generated. The conversion and integration is performed by the subsystem for partial modeling and for model updating, respectively. The modeling system with this research is based on the known location of each view with respect to the global modeling frame. This condition is satisfied in many applications, such as in manufacturing and in robotic systems with a fixed base reference frame. The modeling procedure is described by the following steps:

- After a range image is acquired, a maximum viewing field (a volume) defined by scanning activities and range values is estimated. The volume is then transformed into the global frame for finding the involved cross-section planes which may have the intersection with this volume.

- A partial model is generated by obtaining the intersection curves between the surface scanned in the image and the involved planes. Since range image data in the device frame associated with a range imaging sensor are uniformly distributed, the partial modeling is performed in the device frame. Therefore,

each plane is first transformed into the device frame to form a range image of the plane. Intersection of these images and the sensed range image yields the partial cross-section model corresponding to the sensed range image. Second, the image cross-section model is transformed back to the global frame. This device frame based partial modeling approach eliminates the transformation for massive amount of image data.

- Finally, the partial model generated from an image is merged into the current global model for implementing multiple-view fusion. The cross-section curve sections of the partial model newly generated on the involved cross-section plane are checked to find any overlaps with the current sections (global model) generated previously in the planes. If no overlap exists for a section on a cross-section plane, the new section is inserted in the middle level in the database, otherwise, a method for merging overlaps is applied to integrate the new section with the current section(s).

This vision data based modeling approach can be implemented in a present CAD/CAM system for enhancing the system with the functions provided by the new approach and for utilizing the graphic and model-manipulating functions provided by the present system. As an example, this system is integrated with the AutoCAD system which provides the API interface.

Chapter 3

Partial Model Generation in The Device Frame

3.1 Overview

The cross-section contour representation is able to represent surfaces, either concave or convex and either open or closed. The merit of this representation scheme, in comparison to other surface methods, is the low computational complexity due to conversion of the 3D space problem into a 2D space problem. The modeling system presented in this work generates complete cross-section contour based surface models in two stages: first, converting an individual range image into a partial model of local surface details; and second, merging this partial model updating a global model. This chapter will discuss the algorithm for the device-frame based partial modeling. Section 3.2 gives a general description of the associated algorithm.

A cross-section contour is a curve lying on the object surface, and is formed by the intersection between the object surface and an auxiliary surface which is a plane

(cross-section plane). The generation of cross-section curves can be conducted in the Cartesian modeling frame. However, the range data are non-uniformly distributed in a Cartesian frame, and the processing will be complex. Therefore in this work, cross-section curve generation in the device frame is introduced to preserve the uniform distribution of the surface points (raw range data) in a 2D array.

In applications, most range finders are either imaging radar based or triangulation based. Their range measuring methods are related to the central projection principle using equally spaced angular scanning of associated mirrors. This makes the range image space a non-Cartesian frame which can be a spherical frame or an even more complex frame depending on the particular device in use. The device frame is addressed in Section 3.3.

Partial model generation in the device frame will be shown to require transformation of the cross-section planes into the device frame in order to find the intersections (cross-section curves) of these planes and the object surfaces represented by the range data. When range images are used to generate cross-section models for observed scenes, the image data along each cross-section contour curve must be extracted from each range image. Using these data points, each cross-section contour can be formed by interpolation. Details on cross-section contour point extraction is found in Section 3.4. Sections 3.5 and 3.6 discuss linking these points to form the sub-pixel based partial model curves. Besides the geometric solution, this modeling approach is also able to provide some topological information for applications as discussed in Section 3.7.

3.2 The Algorithm Components for Generating Partial Models in the Device Frame

This section provides an overall description of the algorithm developed for generating contour partial models of an observed scene in the device frame. Technical details for individual functional components are given in subsequent sections. The imaging device based space is determined by the scanning method and the geometric structure of the range sensor. In this chapter, the device frame is discussed as a generic device space without considering a specific device geometry. Specific range sensors are discussed and modeled in Chapter 4.

In general, the acquired range data are arranged in an image array uniformly with respect to constant increments of the scanning variables. These data points can be converted into a Cartesian frame using transformation by a corresponding formula or lookup table derived from the analysis of the device structure and/or calibration. These transformations are nonlinear and device-dependent. The data points, after transformation to a Cartesian frame, are not in an equally distributed grid system, since the central projection based nonlinear transformation cannot provide the uniform data distribution.

To avoid the irregularly distributed data presented in a Cartesian frame, a new approach for generating contour models is developed, which takes advantage of the well-distributed data in the range image space. The approach transforms the cross-section planes into the range image space, generating synthesized plane range images. The intersection contours are detected by comparing the object range images and the synthesized plane range images. This algorithm includes several functional components that are shown in Figure 3.1.

The function of these components are described in the following items:

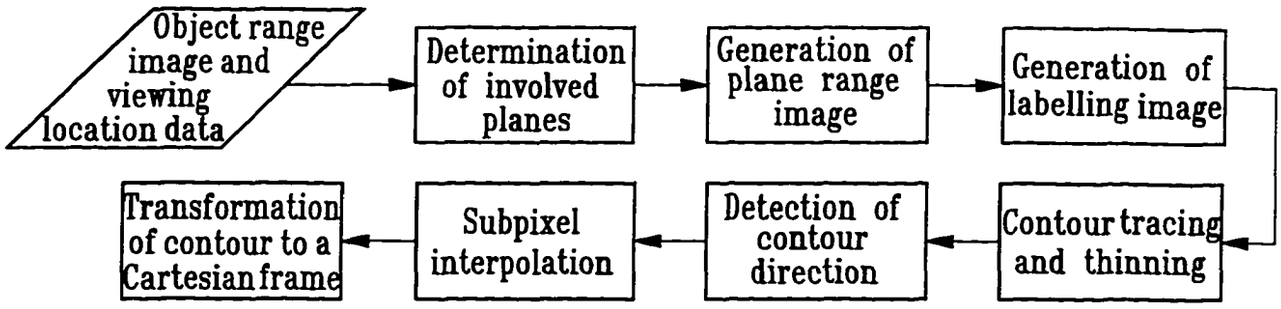


Figure 3.1: The flow chart of the algorithm

- With a range image only some of cross-section planes can have intersections with the image data. To determine the involved cross-section planes and to generate synthesized cross-section plane range images, an estimation of the cross-section planes that potentially intersect with the object surface represented by the range image must be computed. The plane range image for each involved plane can then be generated. The estimation of potential involvement and image generation for the planes is related to an analysis of the range image and to the device geometry. Since these two functional components are device dependent, they will be presented in the next chapter, where the device frames for several typical range finders are investigated.

- In the device frame a labelling image is generated for detecting and locating the intersection between observed surfaces and a cross-section plane.

- Since the result of the intersection detection is range data distributed along a contour curve in the device frame, these pixels based data points need to be linked. Finding neighborhood relationships of the points using tracing and thinning methods facilitates the linking to form the curve. Each contour curve is assigned a curve direction that is used to indicate the exterior/interior volume of an object. Subpixel interpolation is used to render accurate curve solutions.

At the last stage, the obtained cross-section contour curve is transformed from the device frame to a global Cartesian frame for the model integration discussed in Chapter 5.

All of the above components work on data within the device frame. Therefore, before going to each part for explaining the algorithm details, it is necessary to have a more accurate definition for a device frame and to look into the relationship between the device frame and a Cartesian frame.

3.3 Device Frame and Transformation

A device frame is device dependent and is determined by the specific range finder whose scanning activity may be performed by angular and/or linear movements. For example, a scanner may have one or two angular scanning movements, one angular and/or one linear movement and so on. The scanning activities and range measurement govern the number of dimensions of the device frame. Digitized scanning and range values make a device frame measurable. In general, a device frame $\mathcal{D}$ is a subset of $\Re^n$, i.e., $\mathcal{D} \subset \Re^n$, and

$$\mathcal{D} = \{\mathbf{d} \mid \mathbf{d} = (d_1, d_2, ..., d_n)\}$$

where $d_i (i = 1, 2, ..., n)$ are scanning variables corresponding to scanning movements and the range measurement. Related to each range finder a corresponding mapping f exists for transforming the range images from the device frame to a Cartesian frame

$$f : \mathcal{D} \longrightarrow \mathcal{C}, \qquad \mathbf{c} = f(\mathbf{d})$$

where $\mathcal{C} \subset \Re^3$ is a Cartesian frame

$$\mathcal{C} = \{\mathbf{c} \mid \mathbf{c} = (x, y, z)\}$$

The design of a range finder guarantees that f will be such a mapping function that the condition $f(\mathbf{d}_i) = f(\mathbf{d}_j) \Rightarrow \mathbf{d}_i = \mathbf{d}_j$ is satisfied, i.e., there is a one-to-one corresponding relationship between acquired data points and surface points. Hence, an inverse mapping function $F = f^{-1}$ is available [21], i.e.,

$$F : \mathcal{C} \longrightarrow \mathcal{D}, \qquad \mathbf{d} = F(\mathbf{c})$$

Due to a one-to-one mapping between $\mathcal{D}$ and $\mathcal{C}$, both $\mathcal{D}$ and $\mathcal{C}$ have the same dimensions, i.e., $\mathcal{D} \subset \Re^3$. Of the three scanning variables one is for the range measurement and the other two are for scanning activities. Some range finders have only one scanning movement, such that

$$\mathcal{D} = \{\mathbf{d} | \mathbf{d} = (d_1, d_2)\} \subset \Re^2$$

and in the Cartesian frame only two coordinates are involved, e.g., x and y,

$$\mathcal{C} = \{\mathbf{c} | \mathbf{c} = (x, y)\} \subset \Re^2$$

and z will be a constant. Obviously, F is also device dependent. In Chapter 4 the mapping function and the inverse mapping function associated with different device geometries will be formulated.

Since uniform scanning increments are commonly used, a device frame can be easily digitized into a grid system. Consequently, a device frame can be represented by an array. If a range sensor employs two scanning activities, this array has two dimensions (for one scanning activity the array is of a single dimension.). The row and column of the array represent the two scanning variables of the device frame. The size of a row and column represents the number of scanning steps. A range related value d_3 acquired at the step i for a scanning activity d_1 and at the step j for d_2 is stored in the array element indicated by (i, j). As well, the value for d_{1_i} and d_{2_j} can be derived by

$$d_{1_i} = d_{1_0} + i\Delta d_1$$

and

$$d_{2_j} = d_{2_0} + j\triangle d_2$$

where d_{1_0} and d_{2_0} are the initial values.

In a device frame the range data are uniformly distributed in the frame plane defined by d_1 and d_2. To find a range jump in an image due to an occlusion, surface discontinuity or unusual reflection, only d_3 (the range related value) needs to be checked. However, if this range jump is transformed to a Cartesian frame, the detection has to be applied to three dimensions.

The interpretation of each row and column of this array is dependent upon the scanning mechanism, i.e., angular and/or linear movement. A range sensor may use different principles and approaches to measure the range between the sensor and the surface of an object. Usually, the data directly acquired is not the direct range, but is a measurement related to the range. The device geometry and sensing principle determine the method to convert the acquired measurement data to the range values.

The converting function can be non-linear. The function is monotonic. If the acquired data is increasing (or decreasing), the derived range value will be increasing (or decreasing) accordingly. These monotonic and monotropic characteristics are very useful for detecting the intersection of a cross-section plane and the surface of an object in the device frame (i.e., the 2D array). Details on intersection detection can be found in the next section.

3.4 Detection for Intersection

3.4.1 The Principle of the Contour Detection

A cross-section contour is the intersection of an object surface and a cross-section plane. In the device frame the surface and the plane are in discrete form (image pixels). No analytical computation is applicable. The algorithm for finding intersections must rely on a series of pixel based operations. For the intersection computation both the object surface and the cross-section plane need to be represented in the device frame. While the object surface is already presented in its range image (referred to as an object range image) acquired by a sensor, the cross-section plane (referred to as a plane range image) needs to be transformed from a Cartesian frame to the device frame using the inverse mapping function F.

For convenience in the following algorithm description, without losing the generalization of the algorithm to be used with other types of range finders, a device frame determined by a range finder with two scanning movements is assumed. With this assumption $\mathbf{d}$ is a three-tuple (θ, ϕ, r). The variables θ and ϕ represent the two scanning variables and r is the range measurement variable. For such a scanner, an object range image is described by range values $r_{obj}(\theta, \phi)$. During scanning the range image is stored as a 2D array. In the remainder of this chapter the pairs, (i, j) and (θ_i, ϕ_j), are used alternatively for discussion convenience. The notation (θ_i, ϕ_j) or (i, j) corresponds to scan angles $\theta_i = \theta_0 + i(\triangle\theta)$ and $\phi_j = \phi_0 + j(\triangle\phi)$. It is assumed that the guide curve is a line. Therefore, all cross-section planes will have the same plane normal

$$ax + by + cz + d = 0 \tag{3.1}$$

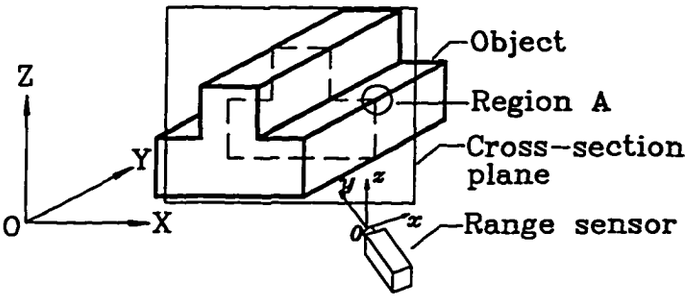
From Eq. (3.1), a series of the cross-section planes is represented by $ax + by +$

$cz + n\triangle d = 0$ ($n = 0, \pm 1, \pm 2, \cdots$), where $\triangle d$ is the interval between two cross-section planes. The inverse mapping function F is applied to each of these cross-section planes to synthesize a plane range image $r_{pln}(\theta, \phi)$ in the device frame, having the same image array size and the same scanning variables as the object range image $r_{obj}(\theta, \phi)$.

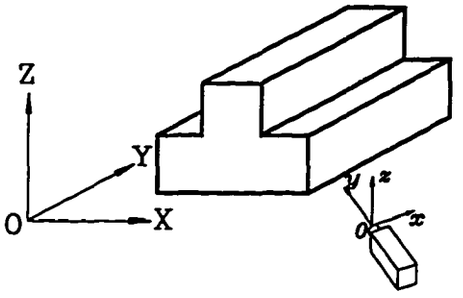
Since a range image covers a limited 3D scene in the Cartesian frame, the surface in the image-covered scene has the intersections with n planes in a domain $[N_1, N_2]$. This is a finite subset of the set of cross-section planes, bounded between $ax + by + cz + N_1\triangle d = 0$ and $ax + by + cz + N_2\triangle d = 0$. The planes outside this domain $[N_1, N_2]$ have no intersections with the surface. However, it is very difficult to obtain the exact domain $[N_1, N_2]$. Only an estimation of the domain $[n_1, n_2]$ can be made by analyzing each range image with the particular device frame. This device-dependent estimation is discussed for particular sensors in Chapter 4.

From the object and plane range images, the algorithm is able to find the intersections in the device frame for obtaining the cross-section contour curves. As illustrated in Figure 3.2 (a), it can be assumed that the plane and the object co-exist in the same 3D scene. When the object range image $r_{obj}(\theta, \phi)$ is acquired, since the assumed plane is transparent, the range sensor only perceives the object surface, see Figure 3.2 (b) and (d). At the same sensor location, a pseudo sensor is assumed existing to acquire the plane image. When the "sensor" perceives the plane, the object becomes transparent, i.e., plane image $r_{pln}(\theta, \phi)$ is computationally synthesized using the inverse mapping function F, see Figure 3.2 (c) and (e). It should be emphasized that the plane range image is not produced by a physical sensor, but by an algorithm.

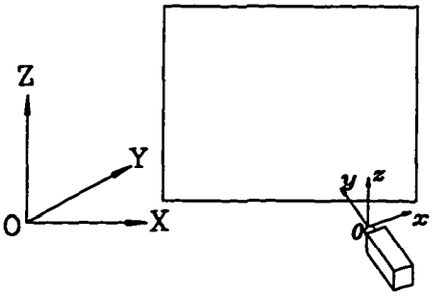
In principle, the intersection curve between two surfaces belongs to both involved surfaces. For both the object image and the plane image the range values for each point along the intersection curve should be the same.



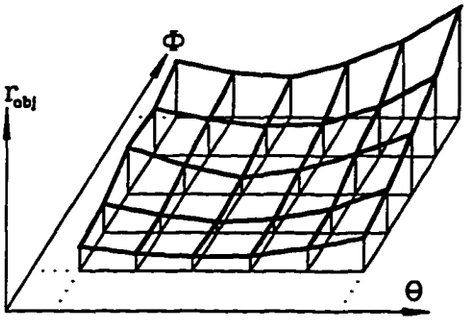
a) Assume that the object and plane exist in 3D space



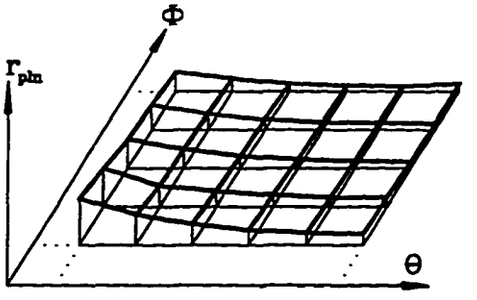
b) Perceive the object only



c) Perceive the plane only



d) Object range image in region A



e) Plane range image in region A

Figure 3.2: Generation of the object and cross-section plane range images

Since the object's interior and exterior volumes divide 3D space into two half spaces, near the intersection a part of the cross-section plane is in the interior object volume and another part is in the exterior volume. When the laser beam reaches the intersecting area, the beam will hit the two different entities one of which is the object surface and the other the plane in two sides of the intersection, respectively. On one side the beam reaches the object surface first and then the plane. On the other side the order is reversed.

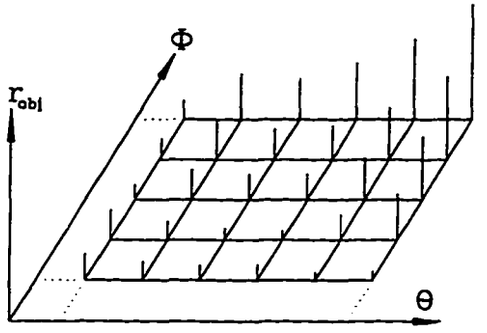
In the device frame the range value (i.e., the distance from the sensor to the surface point hit by the laser) for the object surface is larger (or smaller) than the value for the plane on one side of the intersection, and is smaller (or larger) on the other side. This numeric relationship still exists for the range related data (d_3) acquired and represented in the device frame. This relationship is guaranteed by the monotonic function between the range value and the acquired data. Let the following symbols r_{obj} and r_{pln} be the range for the surface and plane, and $d_{3_{obj}}$ and $d_{3_{pln}}$ be the acquired data for the surface and plane. Let t be the monotonic and monotropic function to make $r_{obj} = t(d_{3_{obj}})$ and $r_{pln} = t(d_{3_{pln}})$. The monotonic and monotropic function assures: if and only if $d_{3_{obj}} > d_{3_{pln}}$, then $r_{obj} > r_{pln}$; and vice versa.

3.4.2 Labelling Image and Intersection Pixels

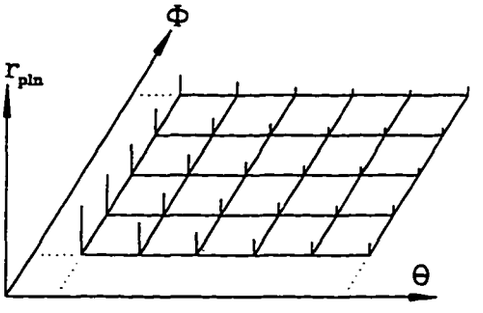
By comparing the range values a labelling image can be constructed for identifying the pixels near the intersection area. The labelling image space is defined as

$$\Omega = \{\theta_0, \cdots, \theta_m\} \times \{\phi_0, \cdots, \phi_n\} \tag{3.2}$$

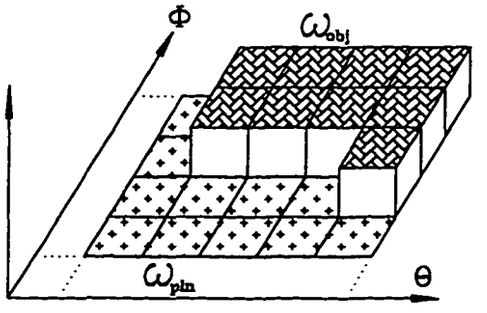
where m and n define the image size whose size is the same as the size of the object and plane images.



a) Object range image



b) Plane range image



c) Labeling image

Figure 3.3: Generation of a labelling image

This space is partitioned into two complement subspaces ω_{obj} and ω_{pln} as shown in Figure 3.3 (c), where ω_{obj} corresponds to the area with larger range values for the object than range values of the plane and ω_{pln} to the area with larger range values for the plane than the object. These spaces are obtained by comparing the object image shown in Figure 3.3 (a) and the plane image shown in Figure 3.3 (b) pixel by pixel from the left to the right and from the top to the bottom.

$$\omega_{obj} = \{(\theta_a, \phi_b) \mid (\theta_a, \phi_b) \in \Omega;\ r_{pln}(\theta_a, \phi_b) < r_{obj}(\theta_a, \phi_b);\ Label(\theta_a, \phi_b) = 1\} \quad (3.3)$$

and

$$\omega_{pln} = \{(\theta_c, \phi_d) \mid (\theta_c, \phi_d) \in \Omega;\ r_{pln}(\theta_c, \phi_d) \geq r_{obj}(\theta_c, \phi_d);\ Label(\theta_c, \phi_d) = 0\} \quad (3.4)$$

Thus,

$$\omega_{obj} \bigcup \omega_{pln} = \Omega, \quad \text{and} \quad \omega_{obj} \bigcap \omega_{pln} = \emptyset$$

Using this labelling image the intersection pixels can be simply identified and encoded. The algorithm scans the labelling image to check the label difference along the row and column directions, respectively. An intersection of a plane and the object surface only exists on the border connecting ω_{obj} and ω_{pln}. For a pixel shown in Figure 3.4, if the adjacent labels in the column direction are different, the pixel is assigned to $Code(\theta_i, \phi_j) = 1$; if in the row direction, assigned to $Code(\theta_i, \phi_j) = 2$; if in both directions, assigned to $Code(\theta_i, \phi_j) = 3$; and otherwise assigned to $Code(\theta_i, \phi_j) = 0$.

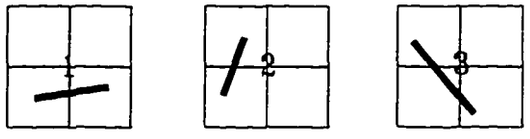


Figure 3.4: Codes used to encode intersection pixels in the labelling image

However, a border connecting ω_{obj} and ω_{pln} does not necessarily mean that an intersection exists. A range step jump caused by object occlusion can also lead to a labelling difference in the two sides of the jump. To distinguish an intersection from a range jump, an additional condition where $|r_{obj}(\theta, \phi) - r_{pln}(\theta, \phi)|$ must be less than a threshold value is checked on the two sides of the border. In the device frame, the additional inspection is only required for the range values (i.e., in only one dimension). However, in a Cartesian frame, the additional inspection has to be done in three directions. This fact makes the computation for generating cross-section models in the device frame simplified compared with the computations required in a Cartesian frame.

An example for the detected contour pixels with the relevant codes is shown in Figure 3.5 where those pixels without an indicated code have code values of 0. Although the contours are explicitly identified by coding, they are still in discrete pixel form. In the next section, a method to convert the contour pixels into contour curves is presented.

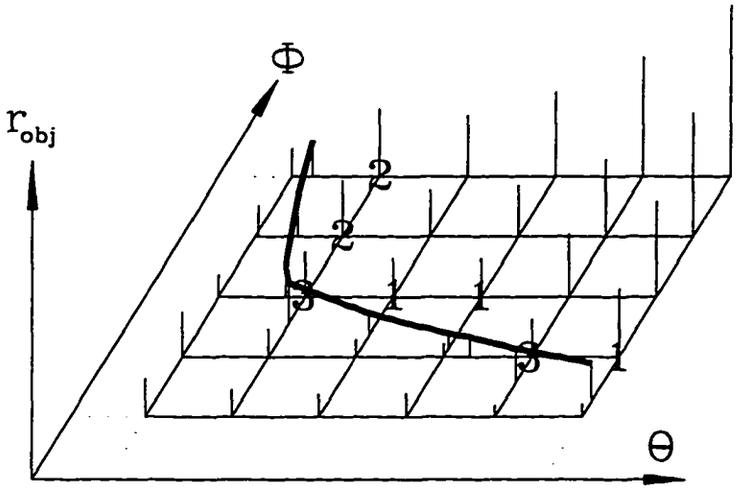


Figure 3.5: Interpolated intersection contour in the device frame

3.5 Linking Intersection Pixels

3.5.1 Contour Tracing

The conversion of contour/edge pixels into vectors is a primary requirement for image processing. This conversion, also referred to as edge tracing or border following, is important for high level image analysis from the low level of discrete image pixels. The developed contour tracing method constructs a directed graph with the nodes of image pixels located along a contour and with the directed edges to link the nodes and to convey the topological information. By following the graph and accessing each node, the geometric contour trace can be explicitly obtained. The directed graph is set up with directed edges such that interior and exterior spaces of an object can be identified. This is useful application oriented topological information.

The tracing method starts from a pixel first found for a new contour curve. The next neighbor pixel is searched to connect to the previously linked pixel, until no more neighboring pixel can be found. During the linking of the contour pixels, the algorithm needs to handle two potential problems. First, contour pixel gaps where contour pixels are not adjacent may exist and form a discontinuous contour. Second, a contour curve may not be represented by a unit-pixel connected skeleton and a "thick curve" can exit. The reason for a gap and/or a thick curve can be caused by image noise or near tangency of the object surface and cross-section plane. Therefore, extensive search and thinning approaches are required.

A local-window based eight-direction search is commonly used in tracing methods [14, 33, 56]. This eight-direction search method locates the window centre at a contour pixel (i, j). To find the next adjacent contour pixel, the method checks the horizontal, vertical and diagonal neighbor pixels $(i + \Delta i, j + \Delta j)$ with a unit pixel interval, i.e., Δi and $\Delta j = -1, 0, 1$. Contour gaps with 1- to 4-pixel distance are assumed by this

research according to the statistics of the gap existence in the experimental images. Handling wider gaps requires a further extensive search. Therefore, the search scope has to be larger than a unit pixel interval around the current contour pixel. Since the local window is enlarged, the eight-direction search is insufficient.

An extensive search window is used. This local window is not a square. In the window, the neighboring pixels around a current pixel (i.e., the window centre) are visited in a search sequence. To increase the smoothness of linked curves and reduce the searching time, this sequence has been determined by considering the code of the current pixel, the potential curve orientation at the current pixel, and the rule of closest-neighbor-visited-first. In most cases, the pixels along a smooth contour curve should have a consistent code. The next curve pixel is expected to have the same code as the current one. When a curve changes its direction, the likelihood for the direction change is also related to the code of the current pixel. The rule of closest-neighbor-visited-first is necessary to find the nearest neighbor pixel and to guarantee every contour pixel to be found.

Figure 3.6 gives the details of the first several steps for the searching sequence with the current pixel $Code(\theta_i, \phi_j) = 1$. In each search step $\triangle i$ and $\triangle j$ are added to the current pixel position to locate the next pixel. The numbers $\triangle i$ and $\triangle j$ for each step are assigned based upon the closest-neighbor-visited-first rule. The search starts from the central area of the window, and gradually goes to the peripheral area. In addition, the expected code is checked with this newly located pixel, e.g., in step 1 the expected code is 1, and in step 3 it is 2.

The search sequence is implemented by a lookup table where the relative location ($\triangle i$ and $\triangle j$) can be found. Once an expected code is encountered at a pixel in the window, a new contour pixel is located. The window center is moved to this pixel for the further search.

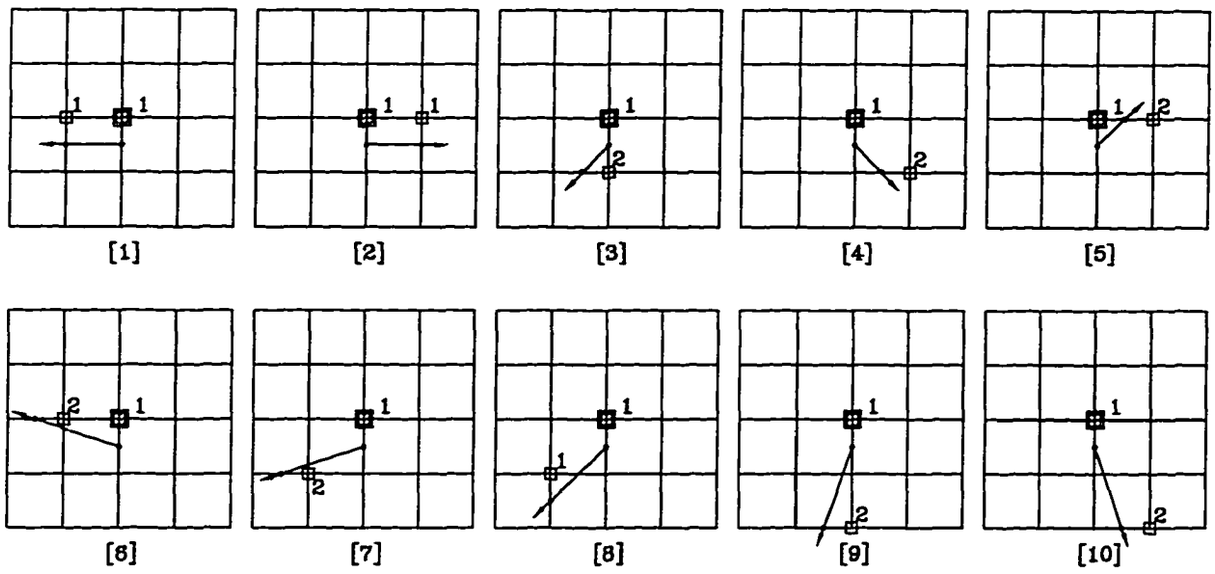


Figure 3.6: An example for the search sequence with the expected code for $Code(\theta_i, \phi_j) = 1$

3.5.2 Curve Thinning

Another problem encountered was non-unit pixel connectivity of contours, i.e., the width of the contour was more than one pixel. Hence, a thinning approach needs to be applied. To date various thinning algorithms have been developed by other researchers. Each thinning algorithm seems to have suitability within a particular application area. For example, thinning methods used for optical characters, cell or gene patterns, and fingerprints were reported in [2], [8] and [27], respectively. The key point for such approaches is to find the skeleton of patterns. A thinning approach based on the LF method proposed by [8] for thinning fingerprint patterns has been modified for the device-frame based contour generation. A device-frame based cross-section contour has no forks, and is similar to and simpler than a fingerprint pattern.

The LF algorithm used successive size-changeable windows to follow a fingerprint

curve with a step d by horizontally or vertically moving the window. Along the moving direction, the algorithm follows the "thick" line border on the two sides making the window centralized at the middle point of the two sides. Accordingly, the trace of the window's centre is the thinned skeleton of the line. If the step d is bigger, windows will be larger and the algorithm will run faster. However, larger windows may lead to the skeletons not being on the medial axis of the thick curve. The LF algorithm is simple, and has a fast processing time.

Since the thinning problem encountered in this research is not very complicated, e.g., no small loops or forks exist, and since "thick" contour curves were found to be at most 3- to 4-pixels wide in the experiments, the LF method is simplified. Assume that the current contour skeleton pixel is located at (i, j), and that the linking window finds a neighbor contour pixel at $(i + \Delta i, j + \Delta j)$. The modified thinning algorithm searches the adjacent pixels horizontally and vertically in the scope defined by $(i+\Delta i, j+\Delta j+\delta j)$ and $(i+\Delta i+\delta i, j+\Delta j)$ with δi and $\delta j = -4, -3, -2, -1, 0, 1, 2, 3$ and 4. Let the most left contour pixel be at $(i + \Delta i, j_l)$, the most right contour pixel at $(i + \Delta i, j_r)$, the most top contour pixel at $(i_t, j + \Delta j)$ and the most bottom contour pixel at $(i_b, j + \Delta j)$. Then, let (i_h, j_h) be $(i + \Delta i, (j_l + j_r)/2)$ and (i_v, j_v) be $((i_t + i_b)/2, j + \Delta j)$. The next contour skeleton pixel is at

$$(i', j') = \begin{cases} (i_h, j_h) & \text{if } \|(i_h, j_h) - (i, j)\| < \|(i_v, j_v) - (i, j)\| \\ (i_v, j_v) & \text{otherwise} \end{cases}$$

3.6 Subpixel Contour Solution

After tracing and thinning, the pixels along a contour curve can be obtained in a directed graph. It is very possible that an actual intersection occurs at a subpixel place, since the scanning is a discrete step-by-step activity. This means that the

real cross-section contour curve exists between the scanning angle θ_i and θ_{i+1} and/or between ϕ_j and ϕ_{j+1}. Figure 3.5 shows the subpixel based cross-section curve in the device frame.

Interpolation based methods are often used to obtain subpixel solutions. An interpolated surface can be used to calculate the intersection with the cross-section plane. The surface based approach is accurate and able to provide cross-section curves directly. Such or related surface interpolation methods have been used in many other research projects. This research has developed another method which is based on the curve interpolation and the curve intersection. This method simplifies the computation for the applications that do not require the high accuracy solution.

Near the intersection region the scanned points along a scanned direction, e.g., $(r_{obj_{i+k,j}}, \theta_{i+k}, \phi_j)$ $(k = \pm 1, \pm 2, \cdots)$, are interpolated for a curve – scanning curve. Since this curve is on the surface being observed, the intersection between this curve and the plane is a point belonging to the cross-section contour. The intersection computation is simplified to that of finding the intersection between a curve and a plane. This computation can be performed either in the global Cartesian frame by transforming the points used for the interpolated curve to this frame; or in the device frame, if the analytical form of the inverse mapping function F is known. The analytical F is required to obtain an analytical expression for the plane in the device frame. An alternative method is to interpolate curves for the scanned surface points and the synthesized plane image points in the device frame.

The selection of image points to participate in the interpolation for a scanning curve is dependent on the code obtained in the label image. The code at each contour skeleton pixel indicates, as shown in Figure 3.4, that the subpixel solution occurs inbetween (i, j) and $(i-1, j)$, if $Code(\theta_i, \phi_j) = 1$; between (i, j) and $(i, j-1)$, if $Code(\theta_i, \phi_j) = 2$; or both, if $Code(\theta_i, \phi_j) = 3$.

The generic description for selecting the involved image points and for interpolating a scanning curve is given below. Let s represent an interpolation function. In the device frame the interpolated result is $\mathbf{d} = s(\mathbf{d}_1, ..., \mathbf{d}_n)$, and in the Cartesian frame the result is $\mathbf{c} = s(\mathbf{c}_1, ..., \mathbf{c}_n)$, where $\mathbf{d}_k = (r_k, \theta_k, \phi_k)$ $(k = 1, ..., n)$, $\mathbf{c}_k = f(\mathbf{d}_k) = (x_k, y_k, z_k)$ $(k = 1, ..., n)$ and n is determined by the order of the interpolation function. To simplify the computation, the labelling image code is used to select points for the interpolation.

When $\mathbf{Code}(\theta_\mathbf{i}, \phi_\mathbf{j}) = \mathbf{1}$, θ_k of $\mathbf{d}_k$ is a constant. The points are selected along the ϕ direction. The series ϕ_k is

$$(\phi_{j-\frac{n}{2}}, ..., \phi_{j-1}, \phi_j, \phi_{j+1}, ..., \phi_{j+\frac{n}{2}-1})$$

When $\mathbf{Code}(\theta_\mathbf{i}, \phi_\mathbf{j}) = \mathbf{2}$, ϕ_k of $\mathbf{d}_k$ is a constant. The points are selected along the θ direction. The series θ_k is

$$(\theta_{i-\frac{n}{2}}, ..., \theta_{i-1}, \theta_i, \theta_{i+1}, ..., \theta_{i+\frac{n}{2}-1})$$

When $\mathbf{Code}(\theta_\mathbf{i}, \phi_\mathbf{j}) = \mathbf{3}$, the interpolations are individually performed with the cases of constant θ_k and constant ϕ_k.

In the global Cartesian frame, the subpixel based solution can be found by solving the intersection of the curve and the plane

$$\mathbf{c} = s(\mathbf{c}_1, ..., \mathbf{c}_n)$$
$$ax + by + cz + d = 0$$

for x, y and z.

In the device frame, the cross-section solution can be found by solving intersection between these two interpolated curves from the surface image and plane image, respectively

$$\begin{aligned} \mathbf{d}_{obj}(r,\theta,\phi) &= s(\mathbf{d}_{obj_1},...,\mathbf{d}_{obj_n}) \\ \mathbf{d}_{pln}(r,\theta,\phi) &= s(\mathbf{d}_{pln_1},...,\mathbf{d}_{pln_n}) \end{aligned}$$

for r, and θ or ϕ. The intersection point then needs to be transformed to the Cartesian frame.

In the following, three different linear interpolation based methods for subpixel solutions are given. One of them is applied in the Cartesian frame, and the other two of them in the device frame.

Cartesian-Frame Based Curve-Plane Method:

For the linear interpolation two points should be found from the surface image. The selection of the two points is dependent on the labelling image code. After the transformation they are represented by (x_1, y_1, z_1) and (x_2, y_2, z_2) in the global modeling frame. The intersection point is found by solving the linear and a plane equations

$$\frac{x_2 - x}{x_2 - x_1} = \frac{y_2 - y}{y_2 - y_1} = \frac{z_2 - z}{z_2 - z_1}$$

$$ax + by + cz + d = 0$$

The generic solution without considering the special cases is

$$\begin{aligned} x &= \frac{-(by_1 + cz_1 + d)x_2 + (by_2 + cz_2 + d)x_1}{a(x_2 - x_1) + b(y_2 - y_1) + c(z_2 - z_1)} \\ y &= \frac{-(x_2 - x)(y_2 - y_1)}{x_2 - x_1} + y_2 \\ z &= \frac{-(x_2 - x)(z_2 - z_1)}{x_2 - x_1} + z_2 \end{aligned}$$

This method is universal and compatible with any type of range finder, but has to transform two image points into the Cartesian frame.

Device-Frame Based Curve-Plane Method:

This method is similar to the previous one except applied in the device frame. The scanning curve on the surface is interpolated in the device frame. Since the analytical plane equation in the device frame is needed, the method requires the analytical form of the inverse mapping function F. Since the method is device dependent, its application with the ERIM range finder, as an example, is given. The detailed description for this device can be found in the next chapter.

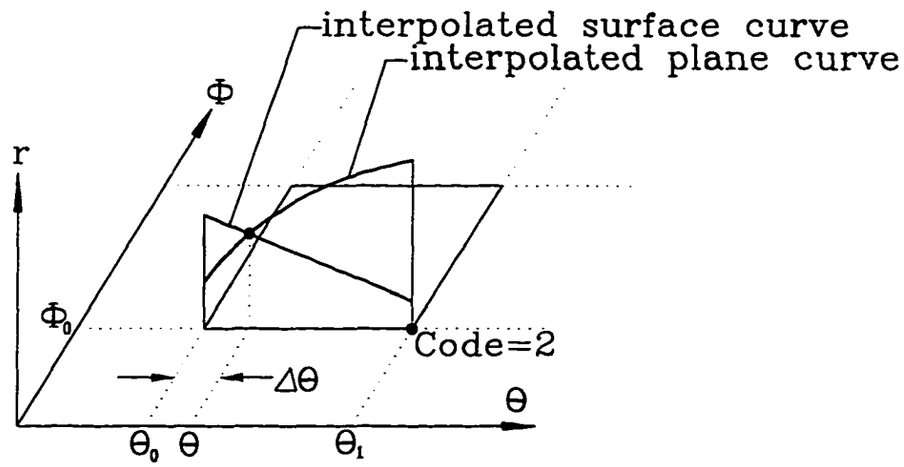


Figure 3.7: The intersection point between the interpolated surface curve and the plane in the device frame.

The inverse mapping for the ERIM sensor is: $x(r,\theta,\phi) = r\cos\phi\cos\theta$, $y(r,\theta,\phi) = r\cos\phi\sin\theta$ and $z(r,\phi) = r\sin\phi$. A cross-section plane in the device frame is represented by

$$ar\cos\phi\cos\theta + br\cos\phi\sin\theta + cr\sin\phi + d = 0 \tag{3.5}$$

The scanned surface curve is represented by a linear equation

$$r = \begin{cases} A\theta + B & \text{if } Code = 2 \text{ or } Code = 3 \\ A\phi + B & \text{if } Code = 1 \text{ or } Code = 3 \end{cases} \tag{3.6}$$

As shown in Figure 3.7, the intersection solution can be found by solving Eq. (3.5) and (3.6). When $Code = 2$, as an example, ϕ is a constant ϕ_0; the two points are in (θ_0, ϕ_0) and (θ_1, ϕ_0); and $r = r_0 + \delta r$, and $\theta = \theta_0 + \delta\theta$. The goal is to find δr and $\delta\theta$.

Using $r_0 + \delta r$ and $\theta_0 + \delta\theta$ to replace r and θ in Eq.(3.5) and (3.6), the following can be obtained

$$[A(\theta_0 + \delta\theta) + B]\,[a\cos\phi_0\cos(\theta_0 + \delta\theta) + b\cos\phi_0\sin(\theta_0 + \delta\theta) + c\sin\phi_0] + d = 0$$

Due to $\delta\theta \ll 1$ degree, $\cos\delta\theta \approx 1$ and $\sin\delta\theta \approx \delta\theta$. Therefore, the above equation becomes

$$[A\delta\theta + (A\theta_0 + B)]$$
$$[(b\cos\phi_0\sin\theta_0 - a\cos\phi_0\sin\theta_0)\delta\theta + (a\cos\phi_0\cos\theta_0 + b\cos\phi_0\sin\theta_0 + c\sin\phi_0)]$$
$$+d \approx 0$$

This equation is equivalent to $l\delta\theta^2 + m\delta\theta + n = 0$. It has two solutions, one of which yields a θ value inbetween θ_0 and θ_1 and another which yields a θ value beyond them. The first one is meaningful. The variable r can then be solved. Similarly, in the case where $Code = 1$, θ is a constant θ_0, and ϕ needs to be solved. If $Code = 3$, two intersection points can be found by applying the above approach for both θ and ϕ.

This method is able to generate accurate results, since the expression for planes is accurate in the device frame. In addition, with this method, only the intersection point needs to be transformed into the Cartesian frame, but not two points required by the previous method.

Device-Frame Based Curve-Curve Method:

This is the simplest method and compatible with any device frame. The two lines are interpolated from the surface image and plane image, respectively. If ϕ is a constant

(i.e., $Code = 1$), the intersection point is found by solving two linear equations

$$\frac{r - r_{s_0}}{r_{s_1} - r_{s_0}} = \frac{\theta - \theta_0}{\theta_1 - \theta_0}$$

$$\frac{r - r_{p_0}}{r_{p_1} - r_{p_0}} = \frac{\theta - \theta_0}{\theta_1 - \theta_0}$$

where (r_{s_0}, θ_0) and (r_{s_1}, θ_1) are the two points from the surface image, and (r_{p_0}, θ_0) and (r_{p_1}, θ_1) from the plane image. Similarly, if θ is a constant (i.e., $Code = 2$), r and ϕ need to be solved. If $Code = 3$, this method needs to be applied twice for r and θ and r and ϕ, respectively.

3.7 Extraction of Topological Information

By going through the above three steps, i.e., contour following, thinning and interpolating, a cross-section contour curve represented by a set of linked intersection points in the device frame is obtained. To provide applications with important topological information about objects in a scene, this research uses a contour curve direction to indicate the interior and exterior object volumes. This direction is represented by the directed edges of the corresponding graph.

A contour divides the plane region into two parts. A part of the plane which is on one side of the contour curve is inside the object, and the other part of the plane which is on the opposite side of the contour is outside the object. Based on this fact, a curve direction is defined as follows: by walking along the curve on the positive cross-section plane normal side, shown in Figure 3.8, the left-hand side should be inside the object and the right-hand side on the outside the object. This definition guarantees that the desired topological information can be provided by the specified curve direction.

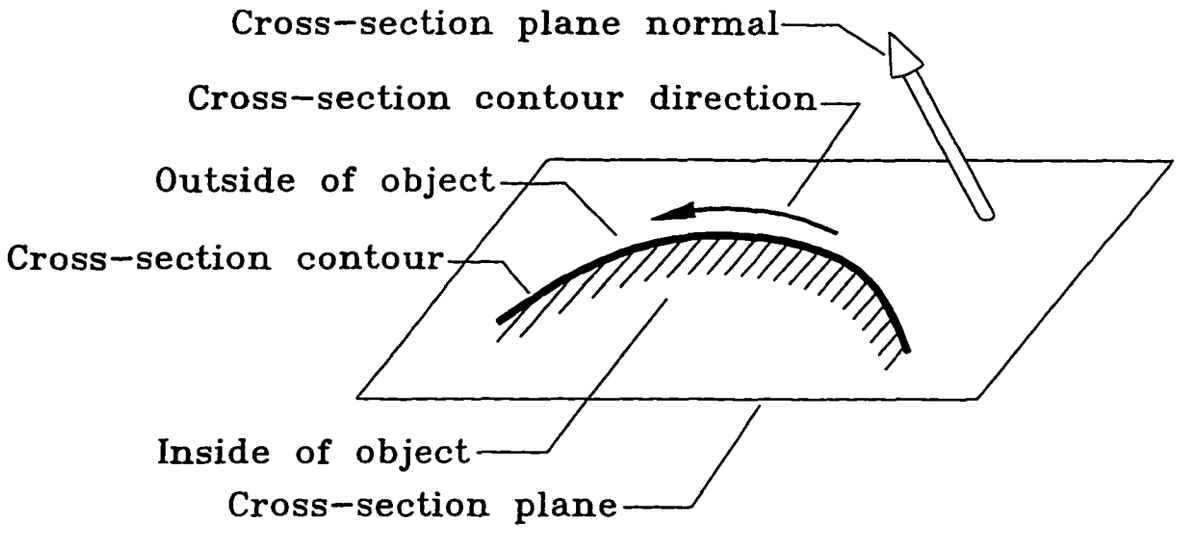


Figure 3.8: Definition of the curve direction

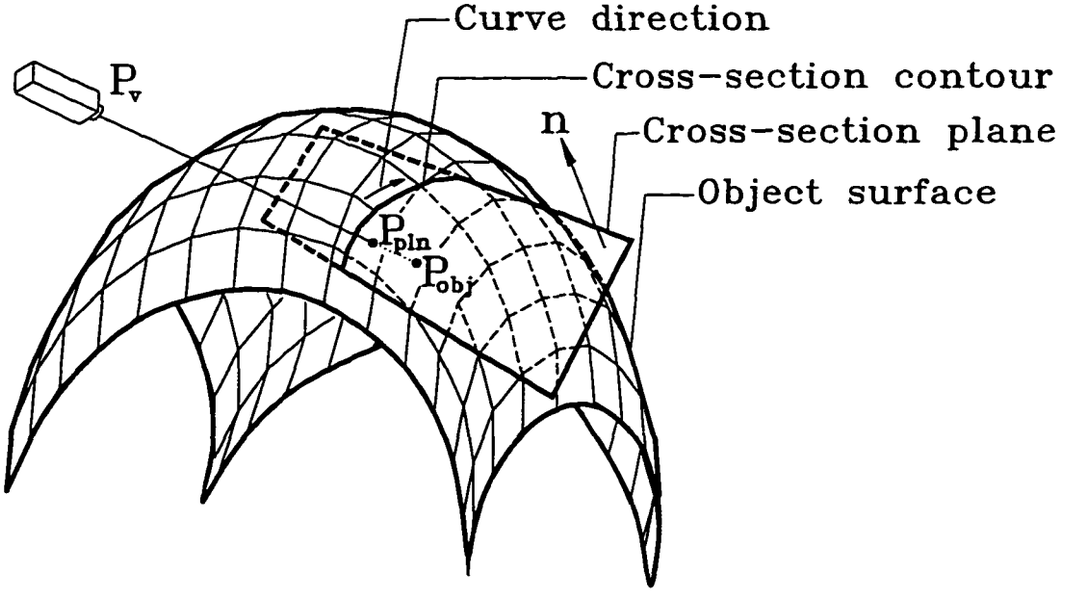


Figure 3.9: The principle to determine the contour curve direction ($\mathbf{n} = (a, b, c)$, $r_{obj} = \overline{P_v P_{obj}}$, and $r_{pln} = \overline{P_v P_{pln}}$)

The directed graph is used to implement this definition for the curve direction. Initially, the curve direction is randomly determined by the curve following algorithm, and is not necessarily consistent with the definition. Thus, an inspection is made to check whether the direction is right. If the current curve direction is not correct, the linkage direction is reversed as detailed below. The inspection algorithm includes several steps,

- a) Let (x_v, y_v, z_v) be the viewing position of the range sensor. Check to find the side of the cross-section plane that the range finder is located. If the *sign* of $ax_v + by_v + cz_v + d > 0$, then the range finder is on the positive side of the plane, otherwise, the range sensor is on the negative side.

- b) Locate the right-hand side pixels (i_{right}, j_{right}) about the current contour pixel $(i_{current}, j_{current})$. Let the previous contour pixel be at (i_{last}, j_{last}), the procedure is based on its contour pixel code.

  if $Code(i_{current}, j_{current}) = 2 \wedge j_{current} < j_{last}$, then
  $i_{right} = i_{current} - 1$;
  $j_{right} = j_{current}$;
  else if $Code(i_{current}, j_{current}) = 2 \wedge j_{current} > j_{last}$, then
  $i_{right} = i_{current} + 1$;
  $j_{right} = j_{current}$;
  else if $Code(i_{current}, j_{current}) = 1 \wedge i_{current} < i_{last}$, then
  $j_{right} = j_{current} + 1$;
  $i_{right} = i_{current}$;
  else if $Code(i_{current}, j_{current}) = 1 \wedge i_{current} > i_{last}$, then
  $j_{right} = j_{current} - 1$;
  $i_{right} = i_{current}$;

else if $Code(i_{current}, j_{current}) = 3$ then

i_{right} and j_{right} are determined by the rules

with the conditions used with $Code(i_{current}, j_{current}) = 1$

and $Code(i_{current}, j_{current}) = 2$;

- c) Compare the range values of the object and plane range image pixels in the position (i_{right}, j_{right}) found by step b. Determine whether or not the right-hand side of the plane is outside the object volume based on the current contour direction. If the object range values are larger than the plane range values, the range finder "sees" the plane first and the object surface second, as shown in Figure 3.9. In this case, the right-hand side of the plane is outside the object; otherwise it is inside the object.

- d) If the range finder is on the positive side of the cross-section plane and the right-hand side of the plane is outside the object, or if the range finder is on the negative side of the cross-section plane and the right-hand side of the plane is inside the object, then the graph direction is correct; otherwise the direction is incorrect and needs to be changed by relinking the graph nodes.

Chapter 4

Implementation of Sensing Systems and Associated Device Frames

4.1 Overview

The underlying principle and method of the new approach for generating partial cross-section contour models from range images was discussed in the previous chapter. The approach is based on rendering cross-section contours in the device frame where the advantages of well organized range data can be exploited. The functional modules for generating labelling images, tracing and thinning contours, detecting contour directions, and obtaining subpixel based intersection contour solutions are primarily device independent. However, the estimation of cross-section planes which have potential intersections with the surface covered by an image, the generation of plane range images, and the transformation of contours from the device frame to a Cartesian

frame are device dependent.

The implementation of these device-dependent algorithm components is closely related to the characteristics of the specific range finder used with the modeling system. These characteristics include the scanning mechanism and the device geometry of the range finder. The implementation of the device frame with individual typical range sensors and the development of the device-dependent functional components for generating partial models is discussed in this chapter.

Range finders are usually classified as an imaging radar type or a triangulation type according to their sensing principles. The contour generation algorithm presented in this dissertation has been implemented for three different range finders, one imaging radar based the other two triangulation based. The discussion in this chapter includes the experimental and programming work for implementing the sensors, and the analysis for deriving the relevant device frames. In addition, the forward transformation from the device frame to the Cartesian frame, and the inverse transformation are discussed as well for the considered range finders.

For an imaging radar based sensor, such as the ERIM range finder, the derivation of the device frame and associated algorithm components is addressed in Section 4.2. For a triangulation based range finder, the technical details can be found in Sections 4.3 and 4.4. A simulation system for a typical imaging radar based sensor, the ERIM, is described in Section 4.5. The analytical derivation for a triangulation based laser scanner (SmartEye) built for this research and a calibration lookup table approach for a HYSCAN$^{(R)}$ commercial scanner are investigated in Sections 4.6 and 4.7.

4.2 The Device Frame with An Image Radar Based Range Finder

4.2.1 The Structure of the ERIM Range Finder

An imaging radar is composed of a laser/ultrasound signal transmitter and a signal receiver. Range data is generated by measuring the elapsed time (time-of-flight range finder) or phase shift (phase-shift range finder) of the source beam that is projected onto and reflected back by the surface of a scene. Early research on imaging radar technology was reported by Nitzan and Jarvis [30, 40, 49]. A typical sensor, the ERIM range finder, was developed at the Environmental Research Institute of Michigan for the vision system of an autonomous land vehicle [64]. In the ERIM range finder the projected and reflected beams share the same optical path. The sensing field is formed by the scanner with a rotating mirror and a nodding mirror. The structure of the ERIM range finder is shown in Figure 4.1.

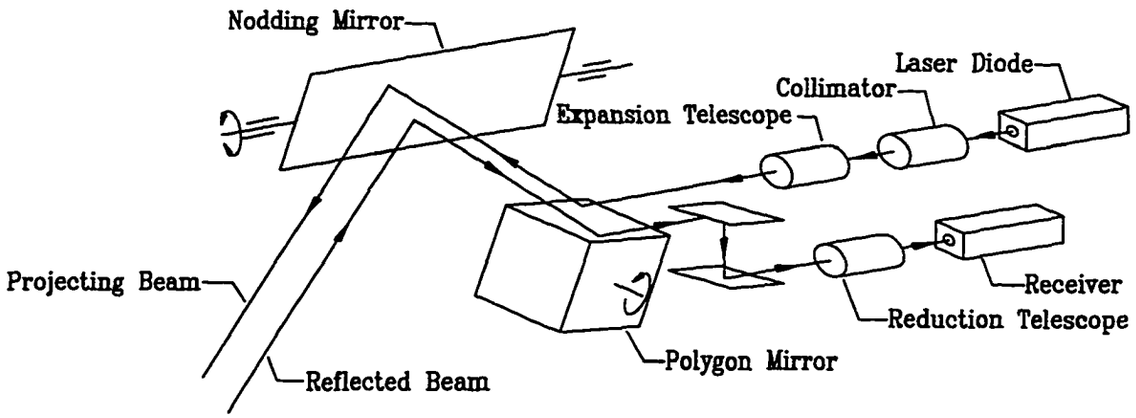


Figure 4.1: Components of the ERIM range finder

A laser beam is generated by a laser diode and is projected onto a polygon mirror

through a collimator and an expansion telescope. The polygon mirror performs horizontal scanning by reflecting the beam onto the nodding mirror. The nodding mirror controls the vertical scanning of a scene. The reflected beam from the object surfaces is received by a receiver through the two mirrors and a similar optical path.

4.2.2 The Device Frame with the ERIM Sensor

An imaging radar based range finder works on a central projection principle. The central projection mechanism actually determines the corresponding device frame. The coordinates of the device frame, as illustrated in Figure 4.2, include two scanning angles and a range. The range for each acquired surface point, indexed by the scanning angles θ and ϕ, is the distance r between the sensor and the surface point. When scanning, the angle changes equally in each scanning direction and the range data is recorded in a rectangular array referred to as a depth map or range image. Each element of this 2D range image array can be addressed by the horizontal and vertical scanning angles. The coordinates of surface points in the Cartesian frame can be found by transforming the obtained range data from the device frame to the Cartesian frame.

The mapping of a projected point P on the object, $f : \mathcal{D} \longrightarrow \mathcal{C}$, from the device frame to the Cartesian frame can be found by

$$x(r, \theta, \phi) = r\cos(\phi)\cos(\theta) \tag{4.1}$$

$$y(r, \theta, \phi) = r\cos(\phi)\sin(\theta) \tag{4.2}$$

$$z(r, \phi) = r\sin(\phi) \tag{4.3}$$

The inverse mapping function $F : \mathcal{C} \longrightarrow \mathcal{D}$ can be obtained from Eqs. (4.1), (4.2) and (4.3) as

$$r(x, y, z) = \sqrt{x^2 + y^2 + z^2} \tag{4.4}$$

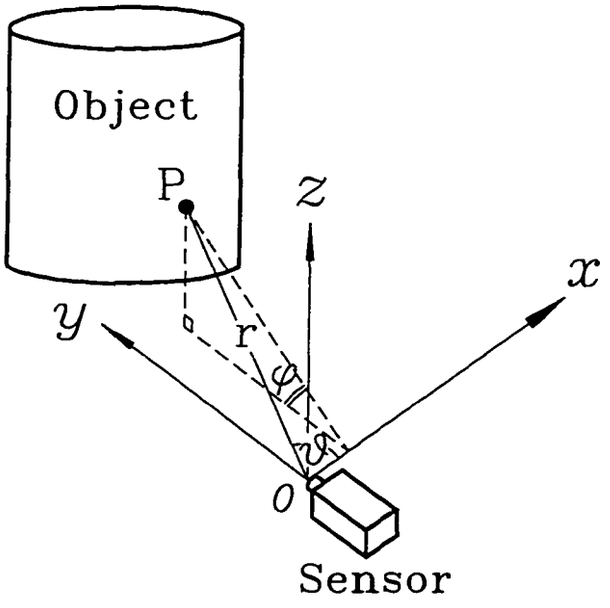


Figure 4.2: The device frame of the ERIM range finder

$$\phi(x,y,z) = \sin^{-1}\frac{z}{\sqrt{x^2+y^2+z^2}} \tag{4.5}$$

$$\theta(x,y,z) = \sin^{-1}\frac{y}{\sqrt{x^2+y^2}} \tag{4.6}$$

The inverse mapping function F can be applied to the cross-section plane described by Eq. (3.1). The range image representation of a cross-section plane in the device frame is obtained by

$$r(\theta,\phi) = \frac{-d}{a\cos(\phi)\cos(\theta) + b\cos(\phi)\sin(\theta) + c\sin(\phi)} \tag{4.7}$$

The plane range image is generated with the same subscripts (corresponding to two scanning angles) and the same array size as the object range image. By comparing the range value r of the object surface and the cross-section plane in the device frame, using the rule given by the expressions (3.3) and (3.4), the intersection pixels can be found.

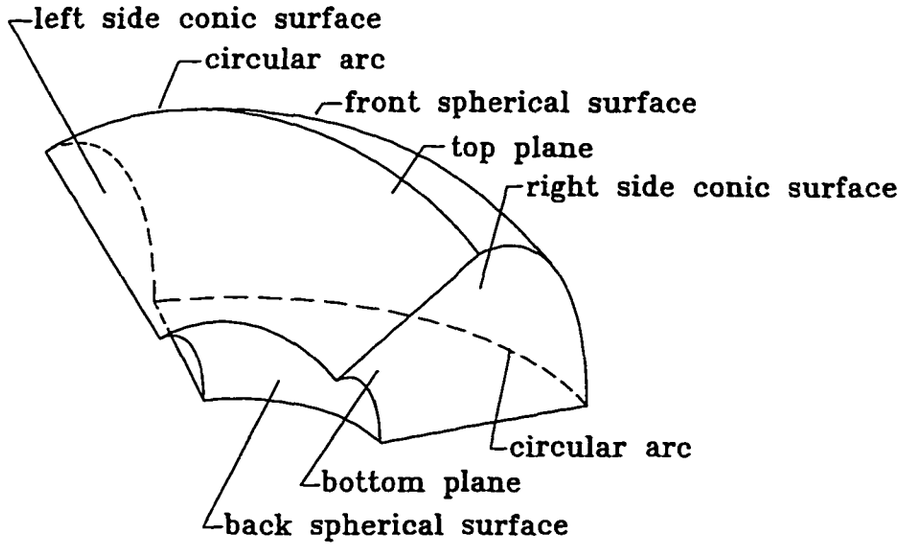


Figure 4.3: Sensing volume of an imaging radar

4.2.3 Computational Advantage of the Device-Frame Based Partial Model Generation

The difficulty of handling sparse data in a Cartesian frame is avoided due to using the device frame based partial modeling. In addition, the computational costs to generate partial cross-section models in the device frame can be lower in comparison to generating partial models with a Cartesian reference frame. This computational advantage can be clearly seen if the computational costs are compared for a general case of modeling, where the guide curve is not restricted to being a straight line.

In the following analysis it is assumed that: the range image is a $m \times n$ array; there are on average l cross-section planes involved in each image; there are on average p intersection points along the cross-section contour(s) in each cross-section plane; and C_{int} is the cost of intersection computation.

Generating a curve section in the Cartesian modeling frame requires the following main stages: 1) transforming surface points from the device frame to the device

based Cartesian frame, followed by transformation to the global Cartesian frame; 2) computing maximum and minimum X, Y and Z to find the number of the involved planes; 3) testing intersection existence; and 4) computing the intersection. These steps need the following computation for *if*, *addition*, *multiplication* and *intersection* operations [1]

$6m \cdot n + 8l \cdot m \cdot n$	(if)
$9m \cdot n + 12l \cdot m \cdot n$	(addition)
$14m \cdot n + 12l \cdot m \cdot n$	(multiplication)
$l \cdot p \cdot C_{int}$	(intersection)

The curve section generation in the device frame requires the following main stages: 1) computing maximum and minimum range values to find the number of the involved planes; 2) transforming the involved planes from the Cartesian frame to the device frame; 3) generating curve sections using a scheme similar to the one mentioned above; and 4) transforming generated curve sections from the device frame to the modeling frame. These steps need the following computations:[1]

$2m \cdot n + 8l \cdot m \cdot n$	(if)
$9l \cdot p + 2l \cdot m \cdot n$	(addition)
$14l \cdot p + 4l \cdot m \cdot n + 3l \cdot m$	(multiplication)
$l \cdot p \cdot C_{int}$	(intersection)

From the above analysis, the computational costs of *if* and *intersection* operations are approximately the same for the curve section generation in both the Cartesian and the device frames. However, the costs of *multiplication* and *addition* operations with the device frame are much lower than the costs associated with the

[1]The detailed computational analysis for both approaches is given in Appendix A.

Cartesian frame, i.e.,

$$14l \cdot p + (4l \cdot m \cdot n + 3l \cdot m) < 14l \cdot p + 7l \cdot m \cdot n \ll 14m \cdot n + 12l \cdot m \cdot n \quad \text{(multiplication)}$$
$$9l \cdot p + 2l \cdot m \cdot n \ll 9m \cdot n + 12l \cdot m \cdot n \quad \text{(addition)}$$

since $l \cdot p \ll m \cdot n$ due to $l \ll m$(and n) normally.

This result comes from two advantages associated with the approach: 1) only intersection curves need to be transformed from the device frame to the Cartesian frame instead of all range image data; and 2) the involved cross-section planes can be transformed from the Cartesian frame to the device frame using an analytical formula.

4.2.4 The Other Device-Dependent Issues

With the inverse function F a cross-section plane can be transformed from a Cartesian frame to the device frame. However, due to the limited sensing field of a sensor, only a part of scene can be covered by the scanning. Hence, only some of the cross-section planes of the whole family of planes will be involved in the intersection with surfaces in the sensing field. An estimation for potentially involved planes is required. This will increase the efficiency of the algorithm and reduce computational costs significantly. The estimation is based on computing the maximum sensing volume of each range image.

For an imaging radar the sensing volume is enclosed by two spherical surfaces (one at the front and one at the back), by two conic surfaces (one at the right and one at the left side) and by two planes (one at the top and one at the bottom of the volume). The sensing volume is shown in Figure 4.3. The plane and conic boundary surfaces are formed by the scanning activity, and remain the same for different views. The radius of the front and back spherical boundaries can be determined by the maximum

and minimum range values of a range image. The two radii are image independent. For a coarse estimation the radii can be determined by the maximum and minimum range sensing scope for which the range finder is designed and/or calibrated. For a fine estimation the radii can be obtained by searching for the maximum and minimum range values in the acquired range image. The spherical centres are coincident with the sensor location.

When a range finder is located in a Cartesian frame, its scanning volume is accordingly located as well. To find the cross-section planes that may intersect with the object surfaces enclosed by the sensing volume, two planes which are tangent with and enclose the sensing volume need to be found. Since the back spherical boundary and two side conic boundaries are concave surfaces, they are not effective for the estimation. Therefore, the front spherical boundary surface and eight corner vertices are only used to find the two most external planes. Eight planes with the same normal as the cross-section planes are located at the eight vertices, respectively. One more plane with the same plane normal is located to be tangent with the front spherical boundary. Another two planes with the same normal are located to be tangent with the circular arcs between the front spherical surface and the top and bottom planer boundary surfaces. The two required planes are the most external ones of these eleven planes. The involved cross-section planes should be between these two most external planes.

4.3 The Device Frame of a Prototyping Sensor

4.3.1 The Structure of the SmartEye Sensor

Active triangulation based range finders are another class of commonly used 3D sensors. Many related methods and systems have been reported in [3, 23, 31, 54, 72]. Most of these systems employ a 1D or 2D sensor (e.g., a CCD camera), a scanner and a laser. A triangulation based range finder usually projects a structured laser beam, such as a point or line, onto object surfaces and captures the reflected energy to obtain the image. Using a simple threshold or edge detection technology, the image position of the laser beam on the surfaces can be detected. The coordinates of the surface points stroked by the laser beam are computed using the optical triangulation principle, the known device geometry and the image positions of the points.

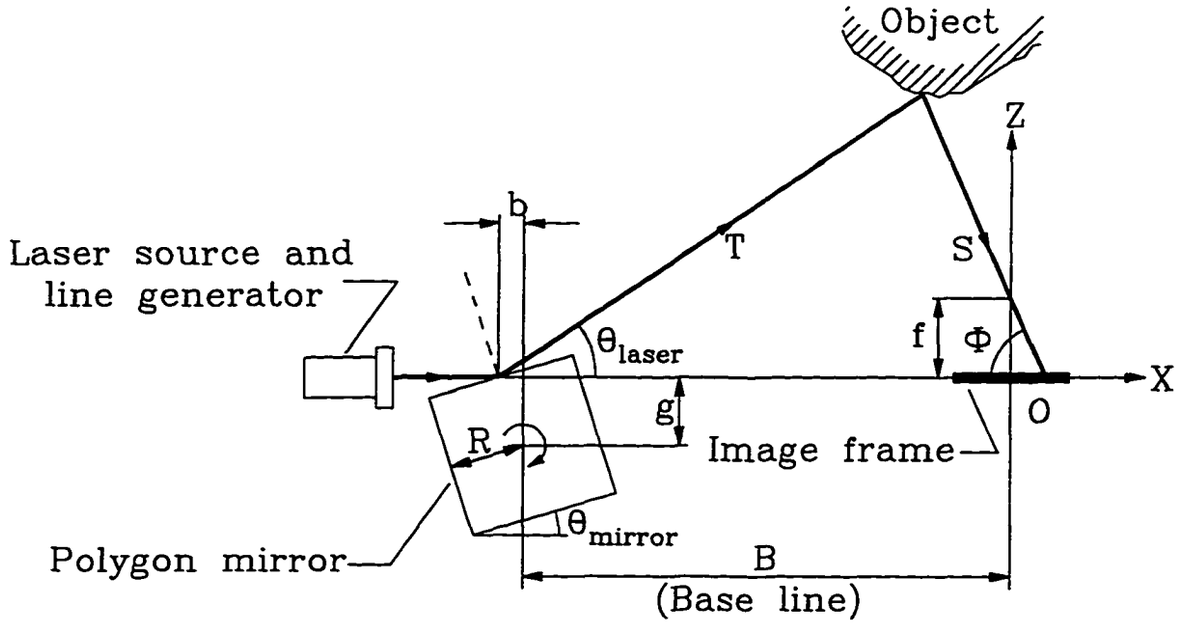


Figure 4.4: Components of a triangulation based range finder (SmartEye)

A triangulation based range finder, named SmartEye, has been prototyped for

this research to test and verify the device-frame based partial model generation. The structure of this range finder is shown in Figure 4.4 which is a section view of the device. A line-structured beam is formed by a line generation lens. This line-structured laser beam T is projected by a rotating mirror with a scanning angle θ_{laser} for the horizontal scanning of the scanner. The reflected laser light S, after hitting object surface, is caught by a 2D CCD camera.

4.3.2 The Device Frame of the SmartEye Sensor

The device geometry and associated coordinate frames are illustrated in Figure 4.5. There are two frames, including $X-Y-Z$ that is the local Cartesian frame associated with the sensor and $x_{frame} - y_{frame}$ that is a 2D frame attached with the CCD array. All captured laser light spots on the object surface are stored in the array. The coordinates of each range data point are related to the location of its image point in the $x_{frame} - y_{frame}$ frame and the scanning angle. The device frame for the SmartEye or other similar triangulation-based range finders can be represented by a 2D array which is indexed by y_{frame} and θ_{laser}. In each element of the array x_{frame} is stored.

The mapping function is derived in the following steps. For convenience to describe transforming an image point to a 3D data point, an intermediate 2D frame, $x_{plane} - y_{plane}$, is introduced. Mapping from the $x_{frame} - y_{frame}$ frame to the $x_{plane} - y_{plane}$ frame and then to the $X - Y - Z$ frame are two $one - to - one$ mappings. The first transformation from $x_{frame} - y_{frame}$ to $x_{plane} - y_{plane}$ is very straightforward and found by

$$x_{plane} = (x_{center} - x_{frame}) aspect\_ratio_x \tag{4.8}$$

$$y_{plane} = (y_{center} - y_{frame}) aspect\_ratio_y \tag{4.9}$$

where x_{center} and y_{center} represent the center of the image frame, and $aspect\_ratio_x$ and

$aspect\_ratio_y$ are the interval distances between two CCD cells in the horizontal and vertical directions, respectively. Based upon the law of sines the following relationship can be found

$$\frac{\sin(\theta_{laser})}{S} = \frac{\sin(180° - \theta_{laser} - \phi)}{B + b + x_{plane}}$$

where $B + b$, called the base line, is the distance from the laser to the image centre. The distance S can be expressed as

$$S = \frac{(B + b + x_{plane})\sin(\theta_{laser})}{\sin(\theta_{laser} + \phi)} \tag{4.10}$$

where the angle ϕ is described as

$$\phi = \tan^{-1}(\frac{f}{x_{plane}}) \tag{4.11}$$

and

$$b = \frac{g\sin^2(\theta_{mirror}) - g + R\cos(\theta_{mirror})}{\sin(\theta_{mirror})\cos(\theta_{mirror})}$$

where $\theta_{mirror} = \frac{1}{2}\theta_{laser}$; g is the distance between the axis of the laser and the center line of the rotating mirror; and f is the focal length. The second transformation from $x_{plane} - y_{plane}$ to $X - Y - Z$ can be found to be

$$x(x_{frame}, y_{frame}, \theta_{laser}) = (1 - \frac{S}{\sqrt{x_{plane}^2 + f^2}})x_{plane} \tag{4.12}$$

$$y(x_{frame}, y_{frame}, \theta_{laser}) = (1 - \frac{S}{\sqrt{x_{plane}^2 + f^2}})y_{plane} \tag{4.13}$$

$$z(x_{frame}, \theta_{laser}) = \frac{S}{\sqrt{x_{plane}^2 + f^2}}f \tag{4.14}$$

From Eqs. (4.12) (4.13) and (4.14) the coordinates of a surface point with respect to the frame $X - Y - Z$ can be obtained.

For the SmartEye sensor the device frame is defined by the image position x_{frame} and y_{frame} of a data point stroked by the laser, and the scanning angle θ_{laser}. Of these

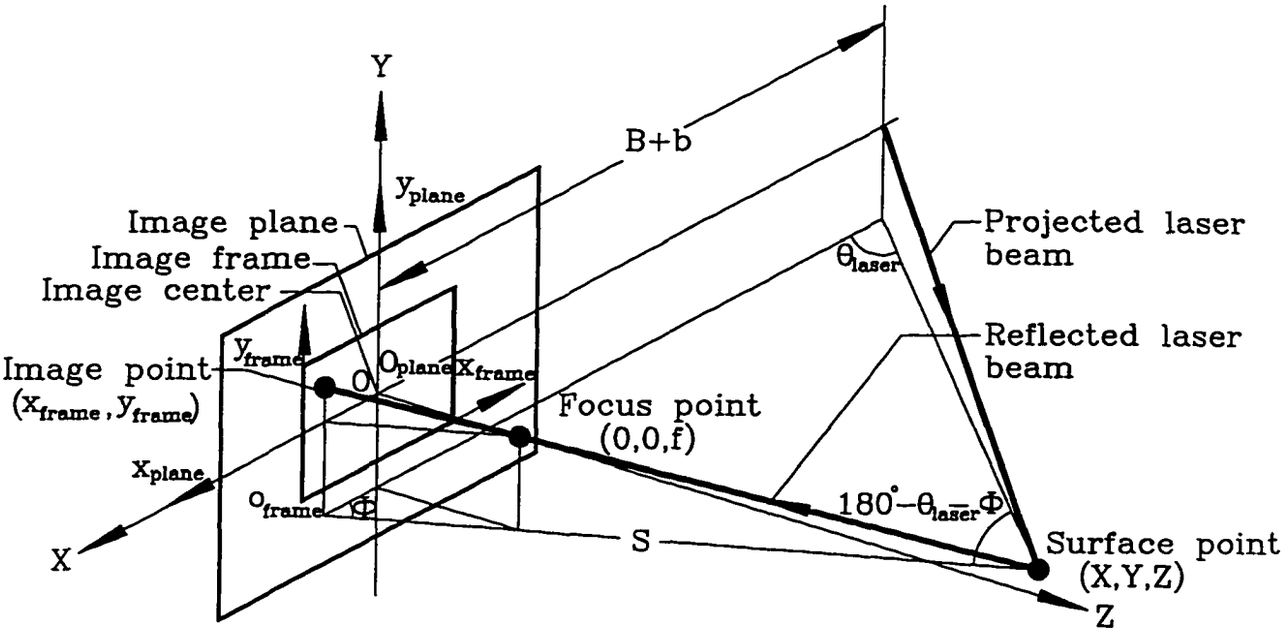


Figure 4.5: The device frame of the triangulation based range finder (SmartEye)

three frame variables, y_{frame} and θ_{laser} are view independent. They change regularly from y_{frame_0} to y_{frame_n} (n is the vertical size of the CCD array) and from θ_{laser_0} to θ_{laser_m} (m is the angular scanning resolution) in each range image. The values of the variable x_{frame} are related to the ranges between the range finder and the object surface points. Equations (4.8), (4.9), (4.10), (4.11), (4.12), (4.13) and (4.14) are used to derive the mapping function $f : \mathcal{D} \longrightarrow \mathcal{C}$ from the device frame to the Cartesian frame for this range finder.

Required by the partial modeling algorithm, range images of all involved cross-section planes must be generated in this device frame. Therefore, the inverse mapping function $F : \mathcal{C} \longrightarrow \mathcal{D}$ needs to be applied for finding x_{frame} values corresponding to all scanning angles θ_{laser} and CCD array rows y_{frame} for the involved cross-section planes.

The difficulty to derive an explicit analytical formula from Eqs. (4.8) to (4.11) for

the inverse mapping function $F : \mathcal{C} \longrightarrow \mathcal{D}$ is easy to see, since the variable x_{frame}, y_{frame} and θ_{laser} are coupled with each other by the nonlinear relationships. Although some approximation method can be applied, it will increase the computational complexity.

To find the inverse mapping function for generating the plane range image without using the approximation, another approach has been investigated. The analytical inverse mapping function is developed by the analysis of the device geometry instead of deriving from the mapping function directly.

In Figure 4.5, the projected laser beam can be found in the laser plane which is generated by the line generator and reflected by the rotating mirror. This laser plane is parallel with the Y-axis, has the angle θ_{laser} about the $X - 0 - Y$ plane, and is described by the plane equation

$$\tan(\theta_{laser})x - z + \tan(\theta_{laser})B_b = 0 \tag{4.15}$$

where $B_b = B + b$ represents the base line distance. This plane has an intersection line with each of the involved cross-section planes expressed by Eq. (3.1). (This plane and the intersection are not depicted in the figure). In principle, the intersection line has its image pixels in different rows of the CCD array. If the CCD camera is described as a pinhole model, a plane family can also be found. This plane family is determined by the focal point and each of the CCD array rows that are on the $X-0-Y$ plane and that are parallel to X-axis. Each row has a Y value $y_{row} = (y_{center} - y_{frame})aspect\_ratio_y$. This plane family is expressed as

$$fy + y_{row}z - y_{row}f = 0 \tag{4.16}$$

By solving the three plane equations, (3.1), (4.15) and (4.16), a number of intersection points for each of involved cross-section planes can be obtained in the

$X - Y - Z$ space. All of these points are on a cross-section plane, are hit by the laser light, and have their images in the row y_{frame} of the CCD array. Since such a point, the focal center and the image pixel of the point should be co-linear (forming a line), the x_{frame} value of the image pixel for this point can be obtained. By solving the intersection of the line and the image plane described by $z = 0$, the X value equivalent to x_{plane} of the image pixel can be calculated, and furthermore, x_{frame} can be obtained using Eq. (4.8).

When y_{frame} continuously varies, the plane described by Eq. (4.16) tilts, and thereafter, the image pixels located in the different CCD image rows for the intersection line between the laser plane and the cross-section plane are obtained. As well, when θ_{laser} varies, a set of the intersections between the cross-section plane and the laser planes determined by θ_{laser} are obtained. Therefore, the range image of a cross-section plane is determined by obtaining x_{frame} for the points of this plane with varying θ_{laser} and y_{frame}.

4.3.3 The Other Associated Issues

The labelling image space for the SmartEye range finder, based on the expression (3.2), is specifically defined by

$$\Omega = \{y_{frame_0}, \cdots, y_{frame_n}\} \times \{\theta_{laser_0}, \cdots, \theta_{laser_m}\}$$

The comparison rules given by expressions (3.3) and (3.4) are accordingly found as

$$\begin{aligned}
\omega_{obj} &= \{(y_{frame_a}, \theta_{laser_b}) \mid (y_{frame_a}, \theta_{laser_b}) \in \Omega; \\
&\quad x_{frame_{pln}}(y_{frame_a}, \theta_{laser_b}) < x_{frame_{obj}}(y_{frame_a}, \theta_{laser_b}); \\
&\quad Label(y_{frame_a}, \theta_{laser_b}) = 1\} \\
\omega_{pln} &= \{(y_{frame_c}, \theta_{laser_d}) \mid (y_{frame_c}, \theta_{laser_d}) \in \Omega; \\
&\quad x_{frame_{pln}}(y_{frame_c}, \theta_{laser_d}) \geq x_{frame_{obj}}(y_{frame_c}, \theta_{laser_d}); \\
&\quad Label(y_{frame_c}, \theta_{laser_d}) = 0\}
\end{aligned}$$

allowing pixels representing the intersection to be found as discussed in Section 3.4.

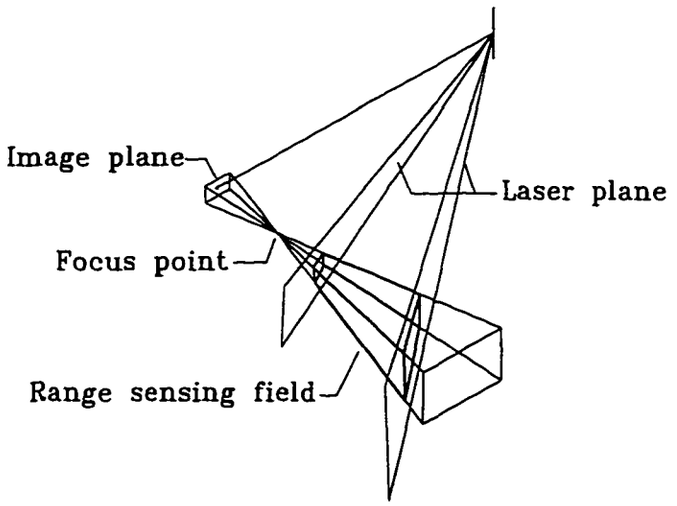


Figure 4.6: Range sensing volume of the SmartEye range finder

Since a pinhole model is used for the CCD camera, the sensing field of the camera is a prism. This prism is truncated by two laser planes at the beginning scanning angle, and the ending angle, as shown in Figure 4.6. The estimation of the maximum involved cross-section planes can be found for each range image, by transforming the eight corner points of the truncated prism to the global Cartesian frame.

4.4 The Device Frame of the Hyscan Scanner

To enhance the capability of the developed vision data based geometric modeling system by diversifying the compatibility with multiple devices, the HYSCAN$^{(R)}$ laser scanner has been investigated. This is a commercial product of HYMARC LTD., and is a triangulation based synchronized laser range finder. The synchronized laser range scanner was initially developed and patented by Rioux at the NRC of Canada [54]. The HYSCAN$^{(R)}$ range finder has a one-dimensional scanner with angular scanning, and measures two coordinates in a Cartesian system, e.g., x and z (or u and v as defined by the scanner).

4.4.1 The Device Frame of the HYSCAN Sensor

In this synchronized range finder the device frame is comprised of two variables: the image position p of the laser spot and the scanning angle θ which is equally changed. Although an analytical relationship between (p,θ) and (x,z) exists, an accurate transformation from the device frame into the Cartesian frame is hard to achieve due to various errors introduced by distortions of the lens and other factors. This commercial product uses a lookup table to perform the transformation. Therefore the mapping function $f : \mathcal{D} \longrightarrow \mathcal{C}$ used for this research is implemented by the lookup table. The Cartesian coordinates of a scanned point can be obtained from p and θ using the table.

To generate the range image for a cross-section plane in the device frame, another lookup table which performs as the inverse mapping function $F : \mathcal{C} \longrightarrow \mathcal{D}$ is required to convert Cartesian coordinates to p and θ. This second lookup table is designed with a number of columns and rows. The columns represent laser beams with different scanning angles, the number of which is equal to the number of angular scanning

steps. The rows correspond to a group of horizontal lines in the scanner based local Cartesian frame. These horizontal lines are obtained by scanning a set of horizontal planes during generation of the lookup table. (Section 4.7 contains further details.) The columns and rows provide the device frame with a regular grid system. The angular scanning lines and horizontal lines digitize the sensing field into an irregular grid system in the Cartesian frame, as shown in Figure 4.7. The denser the grid system utilizes, the more accurate the inverse mapping function is.

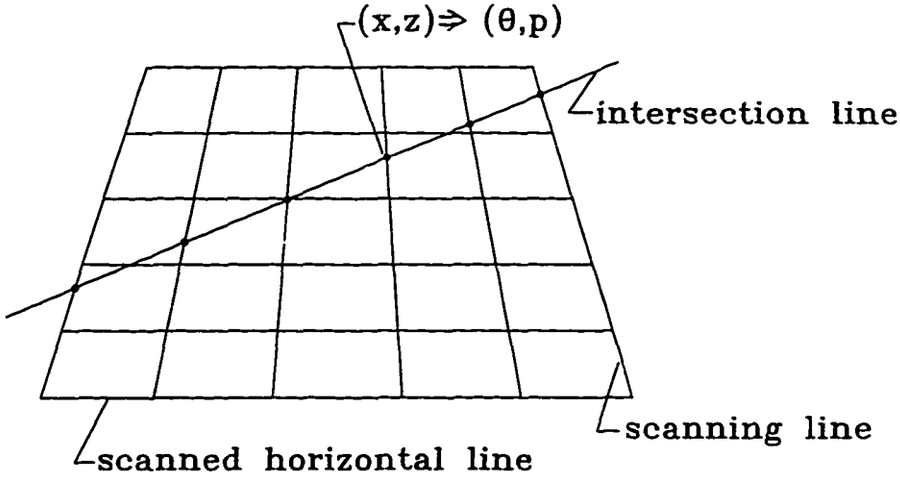


Figure 4.7: Digitized irregular grids for deriving p values

The angular scanning lines and the horizontal lines represented in the lookup table can be individually described by

$$a_s x + c_s z + d_s = 0 \tag{4.17}$$

and

$$a_h x + c_h z + d_h = 0 \tag{4.18}$$

where a_s, c_s, d_s, a_h, c_h, and d_h are determined during generation of the lookup table. Each angular scanning line is indexed by a scanning angle in the second lookup table.

The range image pixels of a cross-section plane can be obtained by solving the following linear equations, as shown in Figure 4.8. The scanning plane formed by the angular scanning is expressed by $y = y_0$. By replacing y of the equation (3.1) with $y = y_0$ the scanned line in the cross-section plane is

$$ax + cz + (by_0 + d) = 0 \tag{4.19}$$

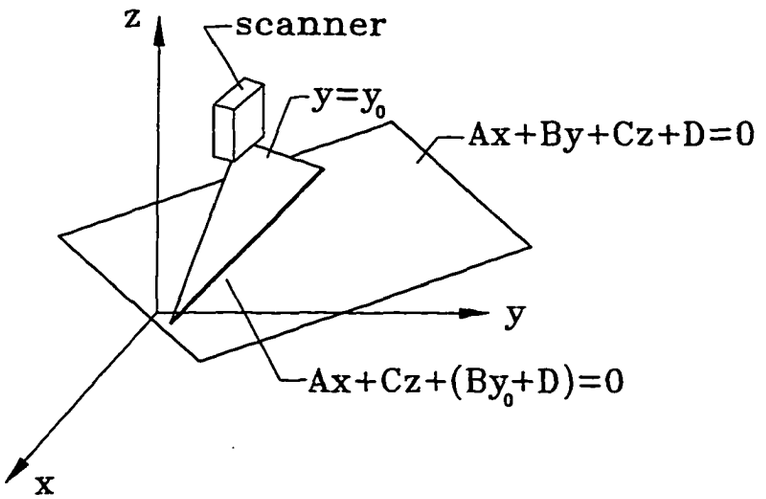


Figure 4.8: Intersection of the scanning plane and a cross-section plane

The intersection point of (4.19) and (4.17) is calculated for all scanning angles. With each θ the associated p value can be found from the lookup table. From Figure 4.7 the intersection point must be located in between two horizontal lines represented by Eq. (4.18). Therefore, the p value can be obtained from the interpolation of the p values associated with these two lines.

If a linear moving device like a coordinate measuring machine is used with the scanner, for example moving along y direction, a frame of range image can be obtained, indexed by θ and y. The object range image and the plane range image are expressed as $p_{obj}(\theta, y)$ and $p_{pln}(\theta, y)$. The cross-section contour solution in the device frame can then be found by comparing these two range images pixel by pixel.

When this scanner is used with a linear moving device, its sensing field is also a prism that is simpler to the prism shown in Figure 4.6. This prism is truncated by maximum and minimum sensing depth planes which are perpendicular to the principal axis of the device. Therefore, it is easy to find the maximum involved cross-section planes.

In the following sections, the experimental work for simulating or implementing the sensors mentioned above is presented with test results.

4.5 Simulation of the ERIM Sensor and the Testing Results

In order to provide a flexible testbed for the development of the vision data based system, a simulation system for the ERIM range-finding device has been developed. The purposes of this work are: 1) to provide the research with simulated object images of various shapes viewed from various positions and orientations for testing the system; 2) to eliminate the affect of image noise for facilitating research; and 3) to eliminate the hardware costs of a time-of-flight based range-finding device [69, 70]. The simulation system is composed of a B-rep based solid modeling system and an R-buffer based pseudo range image generation system by simulating the geometry of the ERIM sensor.

4.5.1 Solid Modeling for Generating Scenes

To model scenes a versatile solid modeling tool which has the basic functions for solid modeling and the interface for obtaining models generated by other CAD systems was developed. The geometric models of objects can be completely constructed by the

developed system, or can be created using the boundary data of 3D models generated by another CAD/CAM system. The bridge to connect the developed system and other systems relies on data files. The data structure and the supported Boolean operations of the system allow a large variety of 3D shapes. The elementary objects are solid primitives such as a cube, a pyramid, a cylinder and a truncated cone, in addition to some objects with curved surfaces like deformed $2\frac{1}{2}$D objects. These objects and their combinations are primarily sufficient for constructing scenes for the research purpose. The functional modules for the modeling system are illustrated in Figure 4.9.

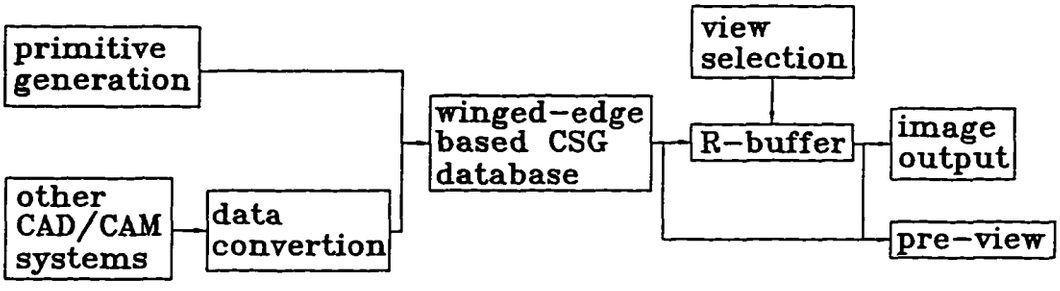


Figure 4.9: Functional modules of the simulation system

Combinations of solid objects are conducted using *OR* Boolean operators. The operands of this Boolean operator can be two primitive solids and/or combined solids. The results are represented by a Constructive Solid Geometry (CSG) tree [20].

4.5.2 Application of the Winged-Edge Data Structure

The geometric models are described in the form of polyhedrons. A general polyhedron in the developed simulation system has two characteristics: i) the bottom and top surfaces share similar shapes; and ii) the two surfaces are parallel. Each of these surfaces is enclosed by a polygon. The top and bottom polygons have the same

number of edges. This general polyhedron is capable of representing objects with planer and curved surfaces. The developed simulation system can generate regular shapes for the top and bottom polygons. Sophisticated shapes can be obtained from other CAD systems through a data file.

The Winged-Edge data structure is used in this system for describing each primitive. This structure [9, 67] is widely utilized with boundary representation (B-rep) due to its capabilities for representing both various geometric shapes and local topological relations of objects.

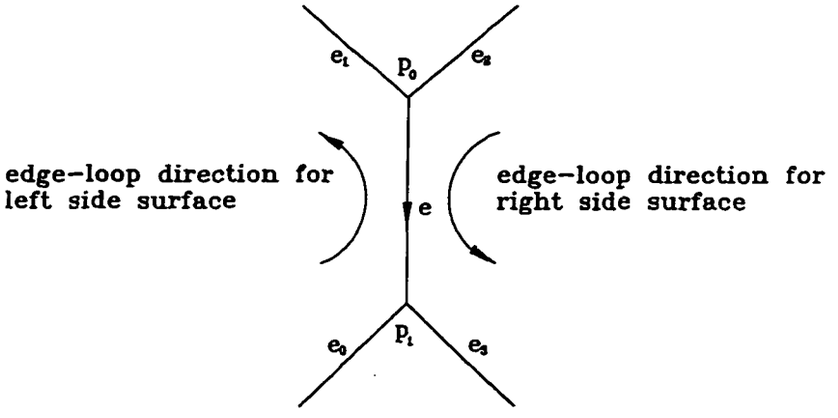


Figure 4.10: A unit of the Winged-Edge structure

An edge is an intersection of two adjacent surfaces. In this data structure, the boundary surfaces of an object are described by edges, vertices and surfaces equations. A represented object has a vertex list, a plane list, an edge list, an edge loop list and a surface list. In the Winged-Edge data structure, edges are the most important components, since they link vertices and surfaces together according to the topological relations of an object. A unit of such a structure is illustrated in Figure 4.10. An edge has two vertices p_0 and p_1, a direction, left and right neighbor surfaces, and four conjunctive edges. A Winged-Edge unit uses pointers to link these elements, and sets up the topological connections among vertices, edges and surfaces.

4.5.3 Synthesizing Pseudo Range Images Using R-Buffer

An imaging radar based sensor usually uses two mirrors to project a laser beam onto the surface of a scene. The range is the distance measured between the laser hit point and the sensor. The scanning activity is controlled by the rotation angles of the nodding mirror and the polygon mirror. This scanning style forms a spherical coordinate system. The synthesized pseudo range image should reflect this scanning activity.

Generating a range image is a conversion from geometric surface models to discrete image data, which is similar to the process for hidden surface removal performed by a computer graphics system. Of various methods for hidden surface removal, the Z-buffer algorithm that finds visible surfaces by detecting the depth (or z coordinate) from the viewing plane to a point in a scene [20] is frequently used. This algorithm was originally used with parallel projection in computer graphic systems.

In this work, the z-buffer approach is modified to work with the spherical coordinate system, to cope with the central projection based scanning accomplished by the range-finding sensor. The modified approach is named the R-buffer approach. To identify the visible points on the surfaces of a scene composed of solid objects, the distance from a view point to a surface point is detected.

The R-buffer approach consists of a matrix, each element of which is indexed by the polygon mirror angle (column) and the nodding mirror angle (row). Therefore, each element of the matrix corresponds to a "light beam" with a fixed orientation. During generation of the pseudo range-data the beam scans through the surfaces of all objects. When the beam strikes a surface, the range from the central view point to the hit point is calculated. If this range is less than the previously calculated range (associated with another surface) the newly calculated range is saved into the buffer

replacing the previous value. When complete, each element of the matrix contains the range value, r, for the nearest surface point along the beam direction. The R-buffer matrix represents the device frame for the ERIM range finder, and makes the mapping function discussed in the previous chapter applicable.

4.5.4 Testing Results with the Simulated Data

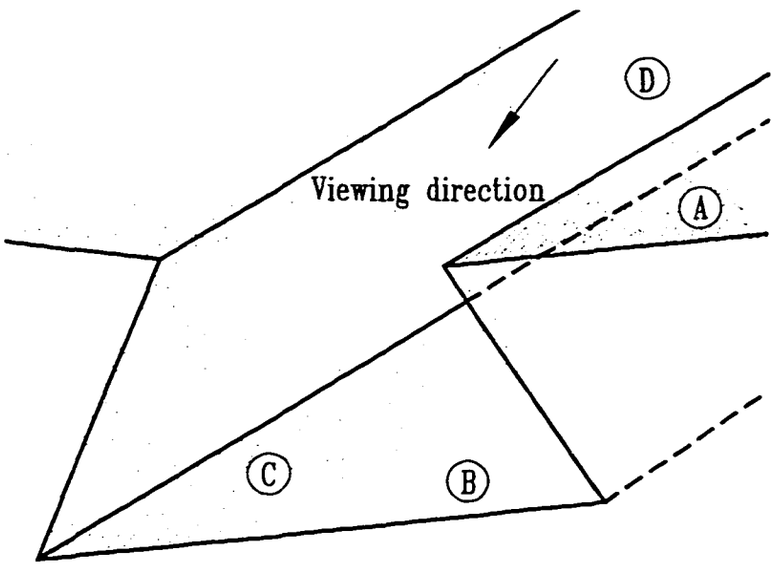


Figure 4.11: Scanned points synthesized by the software simulating an image radar based range finder

Tests were conducted for the software system which simulates the working principle of the image radar based ERIM range finder. Figure 4.11 shows the simulated surface and the scanned points. The scanned points are distributed irregularly on the surface due to central projection based scanning mechanism. Due to the scene occlusion along with the viewing direction, there is a gap (region B) between region A and region C. This gap brings a jump of coordinate values of the scanned points, and provide invalid data for computing the contour model. The jump is very significant in vertical

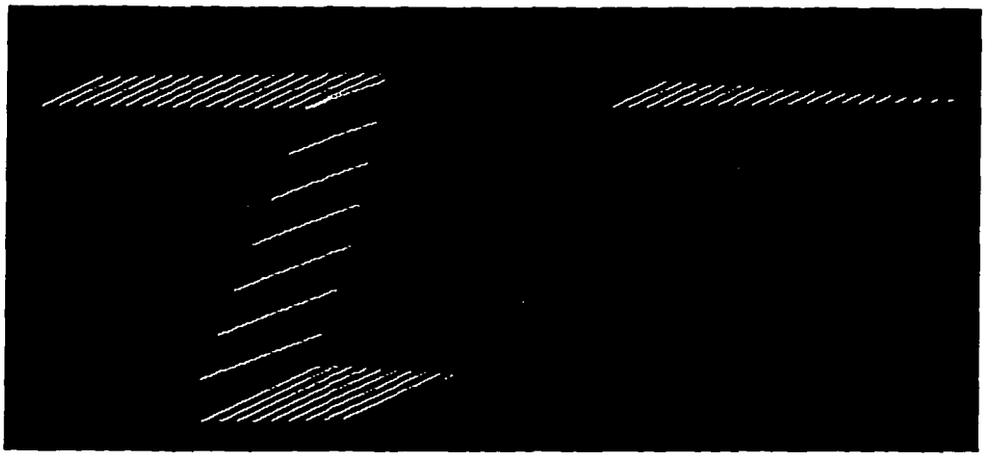


Figure 4.12: The cross-section contour model from the above range image

direction, but not in the other two directions. In some coordinate directions this jump is similar to coordinate jumps caused by the sparsely spaced points in region D where the scanning direction is angularly far away from the cardinal optical center of the range finder. If the generation of the cross-section contour models is considered in a Cartesian frame, the different jumps have to be identified by distinguishing the gaps due to occlusions from the gaps with the sparse data of the region D in the three coordinate directions. In the device frame, the algorithm presented in this work is able to simply detect the gap by directly comparing range values of the object surface and the cross-section plane, allowing the model shown in Figure 4.12 to be generated.

4.6 Implementation of the SmartEye Scanner and the Experimental Results

For this research project the SmartEye range finder, a prototype of a triangulation based sensor, was built with the image processing and computer vision research group

of the Department of Mechanical Engineering at the University of Victoria. Since a triangulation based range finder system has a straightforward structure and is easy to implement, the applications of this kind of device have been found in many reports and publications. The development of the device frame for such a sensor can provide this modeling system with the capability to be compatible with other related research projects and applications.

4.6.1 Prototyping the SmartEye Range Finder

The hardware of the sensor system includes: an AEROTECH OEM5R HeNe laser head with 5.0 (mW) of output power, a straight-line generator lens with 60° of fan angle, a PULNiX TM-7 CCD camera with a 768(H) × 494(V) array size and a VME bus based imaging grabber. As well, the sensor system was designed with a scanner including a stepper motor with 200 steps per revolution, a rotating polygon mirror and a simple mechanical transmission system. Since the camera had not been well calibrated at the moment when this experiment was performed, the internal parameters of the CCD camera, including the CCD array cell size $aspect\_ratio_x$, $aspect\_ratio_y$ and focal length f, were estimated.

Based on Eq. (4.12), (4.13) and (4.14) the coordinate value x_{frame} and y_{frame} to locate the laser spots in $x_{frame} - y_{frame}$ (intensive image) frame should be provided. Since the laser light on the surface of a scene is captured by the CCD camera with the background, a method for extracting the laser light from the background has to be used. When a scene is exposed in illumination such as indoor lights, the extraction of the laser spots becomes difficult since the image intensity difference between the laser and the background is reduced.

Figure 4.13 shows the intensity image data in a row of the image array where the

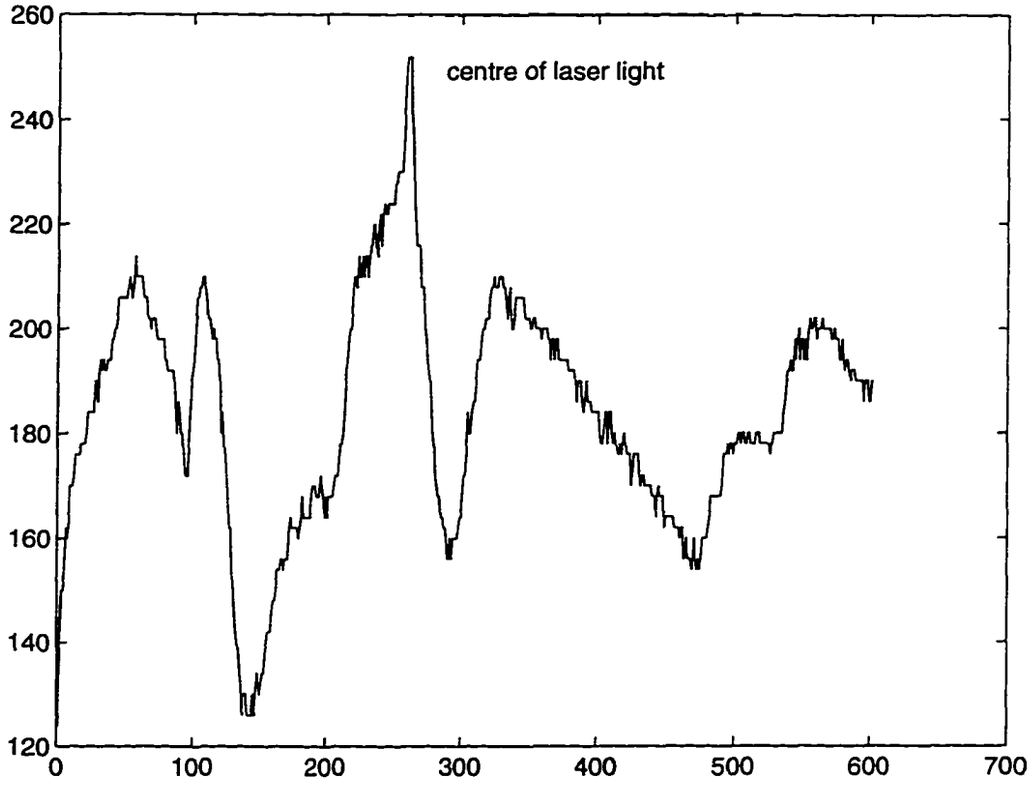


Figure 4.13: Intensity image data of a row

horizontal axis represents image locations and the vertical axis indicates the intensity. The intensity data pattern for the laser spots can be found. Due to the laser beam width and the reflection effect the laser image is not a one-pixel-wide pattern. By assuming the pattern is an isosceles triangle described by a width w and a height h, the pattern at x_{frame_0} can be approximately formulated by

$$intensity_{laser} = \begin{cases} intensity_0 + \frac{2h}{w}(x_{frame} - x_{frame_0}) + h & -\frac{w}{2} \leq x_{frame} - x_{frame_0} \leq 0 \\ intensity_0 - \frac{2h}{w}(x_{frame} - x_{frame_0}) + h & 0 \leq x_{frame} - x_{frame_0} \leq \frac{w}{2} \end{cases} \tag{4.20}$$

The place where the best match of this function with a local intensity data curve occurs in each row is the solution for the laser image location. This is an issue of finding the location of the peak of correlation of two given functions. For obtaining

this goal a local-window based 1D filter was developed. In this local window the 1D array contains the discrete numbers sampled from Eq. (4.20). The window width w is the array dimension. The operations to locate the laser spots using the window are described in the following:

Searching for the correlation peak:

The local window moves from the left-hand side to the right-hand side of each row in the intensity image. When the window is located in a place, the intensity values of the image pixels covered by the window are subtracted by the intensity value of the most left one of these pixels for eliminating the DC value. The absolute value of intensity difference between each image pixel and the corresponding window pixel is then calculated. If the window is very close to the laser spot region, each difference will be small. The summation of these differences is also small. Therefore, a minimum summation in the current row indicates that the location of the laser spots region is found. Of these image pixels covered by the local window the pixel with a local maximum intensity value is located. This pixel is considered as representing the center of the laser light on the surface.

Post processing:

After the laser image pixel with the local maximum intensity value is located in all rows, the laser image curve is found in the image frame. Since a peak pixel is detected locally in each row, some detection failures may exist due to image noise, surface discontinuities and shadows. Some of these pixels may not truly represent the center of the laser light from a global point of view. To smooth the curve a median filter can be applied along the curve direction.

4.6.2 Experimental Results

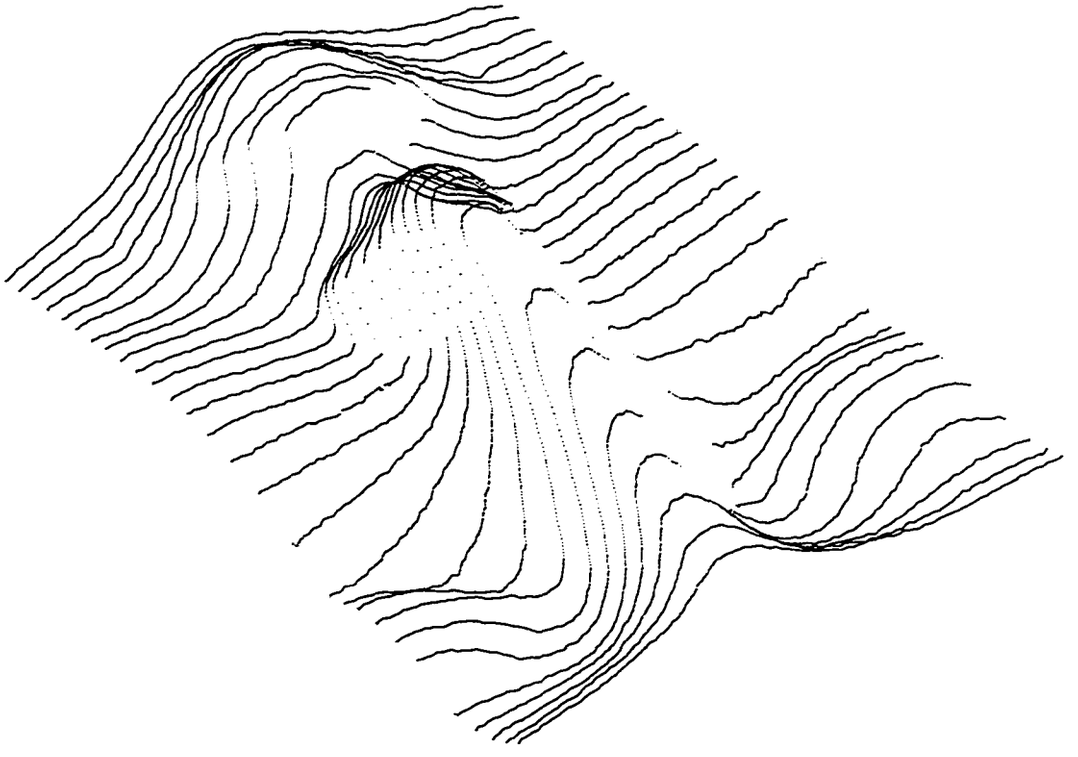


Figure 4.14: Scanned points by the SmartEye range finder

With a structured-light based range finder, the problem of irregularly distributed data due to the structured light and the problem of missing data due to a long base line become more serious. This irregularly distributed and missing data is evident in the scanned points illustrated in Figure 4.14. In the example, the laser light is projected onto a human face mask with the equally spaced scanning angles. The scanned points are distributed along each light curve in a very unequally spaced pattern. Occlusions occur in two places, between the nose and the mouth and on a side of the nose. However, in the device frame described by the variable y_{frame} and θ_{laser} of the range finder, only x_{frame} values are affected by the occlusions. This situation facilitates the algorithm to find the data discontinuity. The 3D plot of the generated contour model is depicted in Figure 4.15.

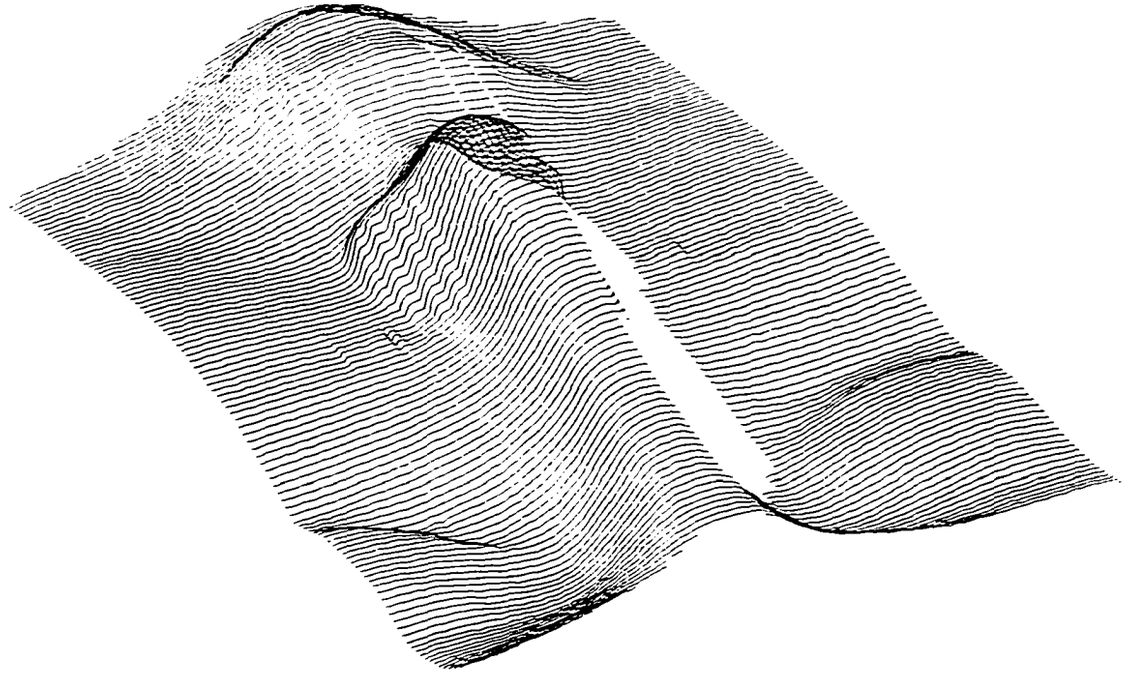


Figure 4.15: The cross-section contour model from the SmartEye range image

4.7 The Hyscan$^{(R)}$ Device Frame Lookup-Table and the Experimental Results

The experiment for verifying and testing the device-frame based model algorithm generation using the Hyscan scanner was supported by the CAD/CAM research group of the Department of Mechanical Engineering at the University of Victoria. Since the Hyscan scanner is one of the commonly used commercial range finder products, this experiment demonstrates the potential application of the developed vision data based modeling system. Since the Hyscan scanner uses a lookup table for the mapping function, the development of the device-frame based modeling can further increase the system compatibility. The work involved with this device is to develop a lookup table based device frame.

4.7.1 The Scanning System

The Hycan scanner is the device with one-dimensional scanning. The CAD/CAM research group uses the scanner mounted on a CMM machine. The CMM machine effectively adds another scanning dimension, to obtain a frame of range data. The whole system is shown as Figure 4.16. In terms of its scanning plane the scanner can be mounted in either the $x - z$ or $y - z$ coordinate plane with a tilt angle about the y or x axis.

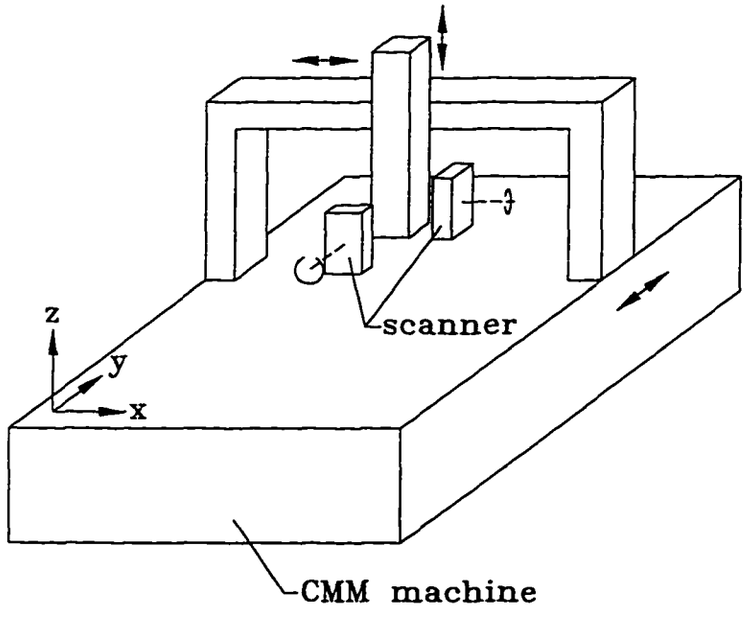


Figure 4.16: The measuring system with the Hyscan scanner and a CMM machine

The sensing depth of the scanner is approximately about 90 mm and a minimum measuring range of 110 mm. With the largest depth the sensing field is about 100 mm wide. All of these parameters are based on experiments associated with this research, they are not from the scanner manual. The system utilizes three coordinate systems, i.e., the device frame, a local 2D Cartesian frame and a global 3D Cartesian frame. The first two systems are based upon the scanner. The last one is based on the CMM machine. The transformation parameters from the local frame to the global frame

are obtained from the alignment (calibration) operation.

The alignment includes scanning the table surface of the CMM machine to determine the tilt angle of the scanner and scanning a sphere to locate the scanner with respect to the global frame. Without losing generalization for the approach to obtain merged global models, this modeling system uses the above global reference frame as the global modeling frame. However, the generation of the lookup tables for the mapping functions is only related to the device frame and the local Cartesian frame. Consequently, the alignment is not required.

4.7.2 Generating the Lookup Tables

In the device frame of the Hyscan scanner, the device frame variables include p and θ. In the local Cartesian frame, the frame variables are represented by u and v. The mapping function f is already implemented in the lookup table by the vendor. Another lookup table for the inverse mapping function F must be derived for generating image data for cross-section planes. In principle, this second table using (u,v) to search for (p,θ) can be derived from the first one using (p, θ) to search for (u,v). However, since the Hyscan scanner is a commercial product, the existing lookup table used by the sensor is not accessible to users. Therefore, a method was developed to generate lookup tables for both of the mapping functions without direct access to the Hyscan table.

Both of the lookup tables need to digitize the sensing field as shown in Figure 4.7 The methodology for generation of the lookup tables includes the following steps:

Acquiring Data:

To digitize the sensing field a group of parallel horizontal lines are obtained by scanning the table surface of the CMM machine with multiple sensing depths in the

sensing range. These parallel lines are not equally spaced within this experiment. The acquired data includes p, θ, u and v values which are represented in Figure 4.17 and 4.18. Since the scanner is not exactly perpendicular to the table surface, the scanned lines are not symmetrical to the cardinal center of the scanner. Since this experiment is for testing and verifying the approach, the sensing field is not digitized with a high density of horizontal lines.

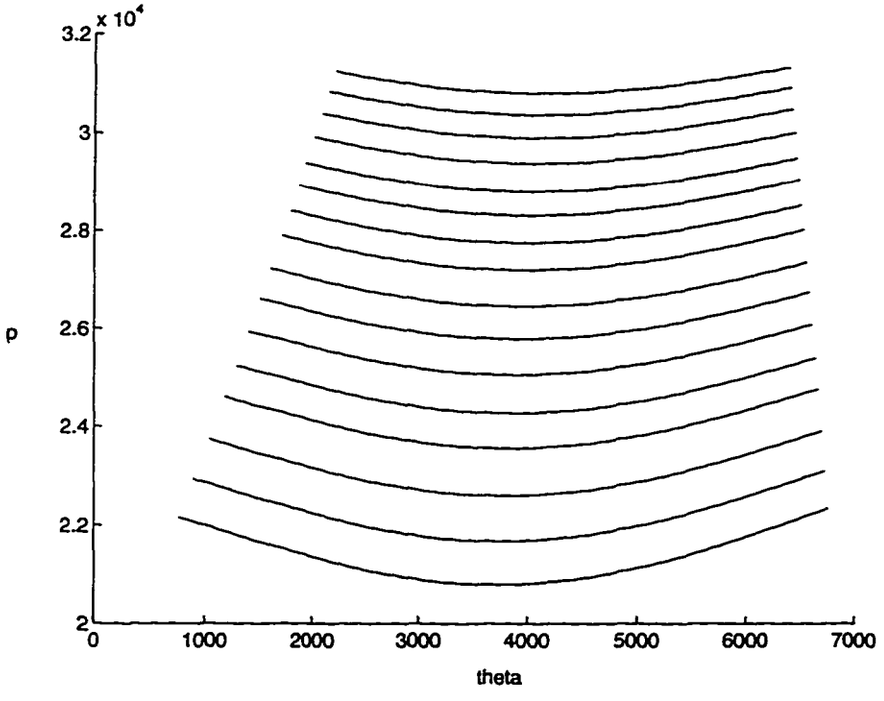


Figure 4.17: The acquired p/θ data for digitizing the sensing field

From each scanned point the value of u and v and the value of p and θ can be obtained from the output of the scanner at the same time.

Generating the Table for the Mapping Function:

Generating the lookup table for the mapping function is straightforward. The scanned data is stored in an array, each column of which is indexed by the scanning

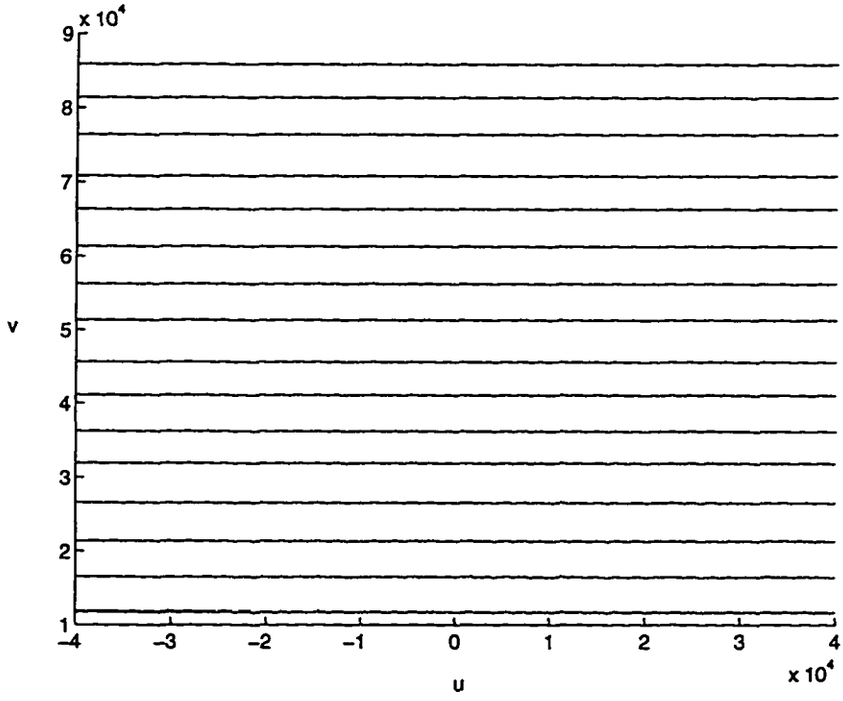


Figure 4.18: The acquired u/v data for digitizing the sensing field

angle, and each row of which represents a scanned horizontal line. Each element of this array points to a p value and the corresponding u and v values.

When a pair of p and θ is given for searching for its u and v coordinates, θ is used to locate the relevant column. Along this column the p values in each row are compared with the given p to find a row where the given p is larger than or equal to the stored p and the next row where the given p is smaller than or equal to the stored p. Therefore, the u and v coordinates can be interpolated from the two pairs of u and v values associated with these two rows.

Generating the Table for the Inverse Mapping Function:

The lookup table for the inverse mapping function relies on the digitized sensing field with a grid system formed by the scanning lines and scanned horizontal lines. The grid pattern is ladder-shaped and not symmetrical to the cardinal axis of the scanner. This table is also represented by a 2D array. Its rows and columns have the same meanings as the previous lookup table. Each element of this array stores a pair of u and v and the corresponding p. In addition, each column has a pointer to a linear equation for this scanning line (beam) with the angle indicated by the column. Such an equation is used to intersect with the "scanned" line (see Figure 4.8 in a cross-section plane for the u and v values. With these values, the utilization of this lookup table is also similar to the previous one. The required p value is interpolated instead.

The linear equations for scanning beams are derived from the acquired data. Due to the existing noise a medium filter is applied to the data to increase the accuracy of the table. For a scanning beam the u and v values from two adjacent scanned horizontal lines and in the same column are used to obtain the linear equation parameters a, b and c. Since there are n scanned lines, there are $n-1$ such equations. These a's, b's and c's are sorted, respectively. The one third of all data, in the middle

portion of each series, are extracted. They are further averaged for a, b and c.

4.7.3 The Experimental Results

Figure 4.19 and 4.20 demonstrate the scanned points and the contour model obtained in the device frame.

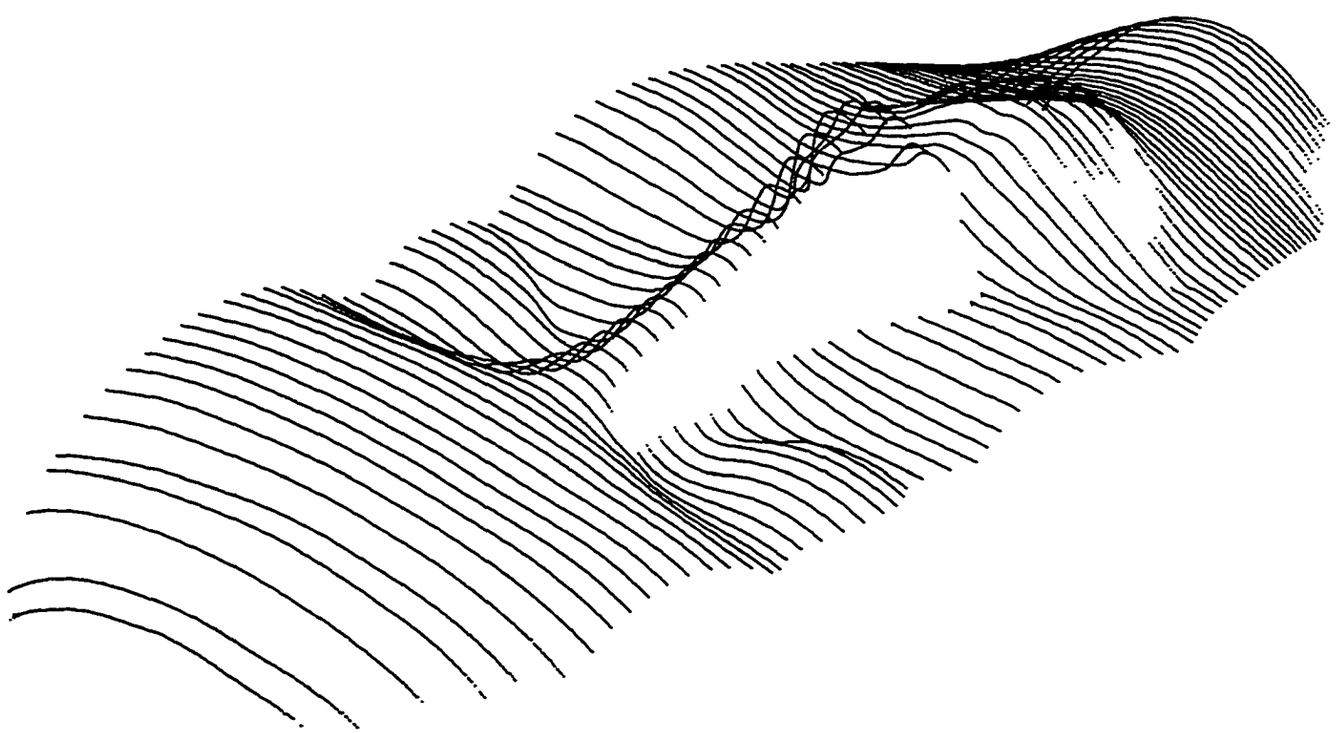


Figure 4.19: Scanned points by the Hyscan range finder

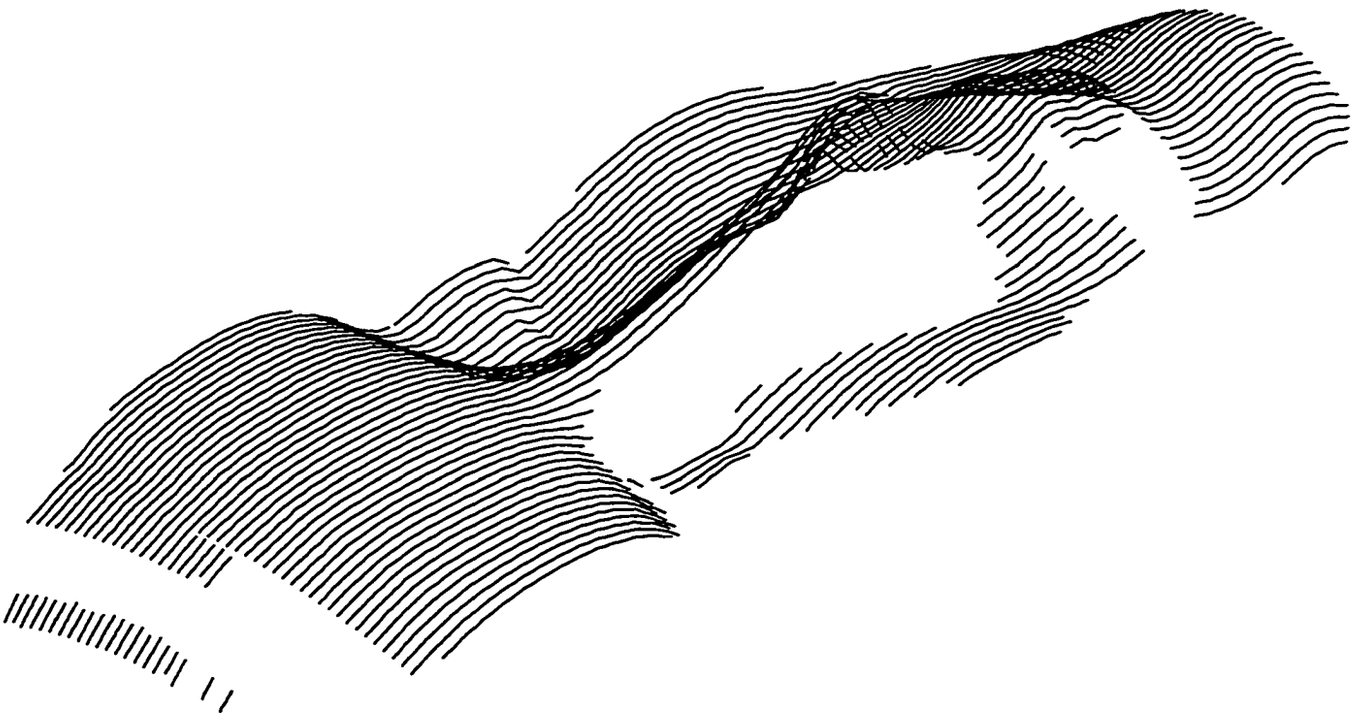


Figure 4.20: The cross-section contour model from the Hyscan range image

Chapter 5

Global Model Generation with Multiple-View Fusion

5.1 Overview

By means of the device frame based modeling approach, a partial surface model is constructed with incomplete cross-section curves in a series of planes. This partial model from a current range image must then be integrated with the present global model generated by previous partial models. The merging activity should blend redundant surface data caused by overlaps of range images. Consequently, the multiple-view fusion is implicitly implemented during the integration process.

In the methodology introduced by this chapter, the model update is accomplished on each involved 2D cross-section plane. The principle of the global model integration is discussed in Section 5.2. On an involved plane the possibility of overlaps between the current partial curve section and the previously generated global curve section(s) are tested (Sections 5.3 and 5.4). If a potential overlap is found and confirmed, the

current partial model curve is connected with the global model curve, integrating the common parts. This is done in two steps, i.e., 1) locating the overlap region along both the global model curve and the partial model curve and 2) integrating overlapped curves (Section 5.5). If the current partial model curve is separated from the current global model curves, this partial model curve will be stored as a separated section with other previous ones for updating the global model. When a complete observation is achieved, the incompleteness between separated curve sections will no longer exist. Two examples for the global model generation are given in Section 5.6.

5.2 Generation of the Global Model

The global geometric model represented by a series of cross-section contours is generated by integrating all partial models together. When a new range image covering a regional area of the surface of a scene is taken, the partial model is generated in the device frame. This partial model is transformed to the global Cartesian modeling frame and merged with the previously generated global model. The generation of the global model is the consequence of the model growing.

The more range images that are acquired the more complete the model of the scene that will be obtained. Since the cross-section based representation is able to handle modeling for either an object with a bounded surface or a scene with a unbounded surface, the definition of a complete model relies on the application. As well, a complete model may require many images from different viewing locations. The selection of a sensor(s) trace of the viewing location would be also application dependent. The model growth itself should not need a specific viewing path or sequence.

5.2.1 Principle of the Model Integration

The cross-section based models (ψ_{model}) described in the global Cartesian frame represent the intersection of cross-section planes and surfaces of objects. The generation of a global model Ψ_{model} is based on the integration of individual partial models. The integration operation can be expressed by the boolean union computation

$$\Psi_{model_n} = \Psi_{model_{n-1}} \bigcup \psi_{model_n} \qquad n = 1, 2, ... \tag{5.1}$$

where, Ψ_{model_n} is the updated object surface model, i.e., the present global model; $\Psi_{model_{n-1}}$ is the previous global object surface model generated before the n^{th} range image is taken; and ψ_{model_n} is the partial surface model generated from the n^{th} range image. Initially, when $n = 1$, no previous global model exists, i.e., $\Psi_{model_0} = \emptyset$.

When a current range image and a previous one cover common areas of the surface of a scene, overlaps (data redundancy) of the model curves will be present. Some common curve sections in both the global model and the current partial model will exist. Corresponding to this redundancy, the following boolean expression will become true

$$\Psi_{model_{n-1}} \bigcap \psi_{model_n} \neq \emptyset \tag{5.2}$$

In extreme cases of overlapping existence, the current partial model is entirely redundant to the previous global model or the previous global model becomes entirely redundant to the current partial model, i.e., ψ_{model_n},

$$\Psi_{model_{n-1}} \bigcap \psi_{model_n} = \Psi_{model_{n-1}} \tag{5.3}$$

or vice versa

$$\Psi_{model_{n-1}} \bigcap \psi_{model_n} = \psi_{model_n} \tag{5.4}$$

When the current range image does not cover any surface area of a scene which has been shot by previous image(s), gaps between the global model and the partial model will exist. If the acquisition of multiple-view range images is well structured, ψ_{model_n} may be perfectly adjacent with $\Psi_{model_{n-1}}$. In these cases, the following expression will be true

$$\Psi_{model_{n-1}} \bigcap \psi_{model_n} = \emptyset \tag{5.5}$$

The above expressions give a general explanation for the cases that may be met during the model integration. Since the cross-section contour based models are constructed within the basic elements including the cross-section planes and contour curves, as described by $\Psi_{model} = \{\{\mathbf{c}\} \mid \text{in a cross-section plane}\}$ from Eq. (2.1), the union operation is performed in each of the involved planes. The two operands of this boolean function are the cross-section curves representing the global model and the partial model, respectively.

During the model integration with a partial model, the overlaps may occur in multiple places, e.g., Area_1 and Area_2 as shown in Figure 5.1. This situation is caused by range data jumps due to the occlusion of scenes. In a cross-section plane the overlaps can be associated with several curve sections, e.g., in Figure 5.1 a cross-section plane with orientation 1 has overlapped curve sections in two places. The overlaps may also exist in several separated groups of the cross-section planes, e.g., the planes with orientation 2 are involved in the overlap in Area_1 and Area_2.

5.2.2 Procedure of the Model Growth

Based on the principle of the model integration, the model growth implemented by the algorithms of this modeling system is associated with the model data update in two levels: the cross-section plane level and the contour curve section level. The

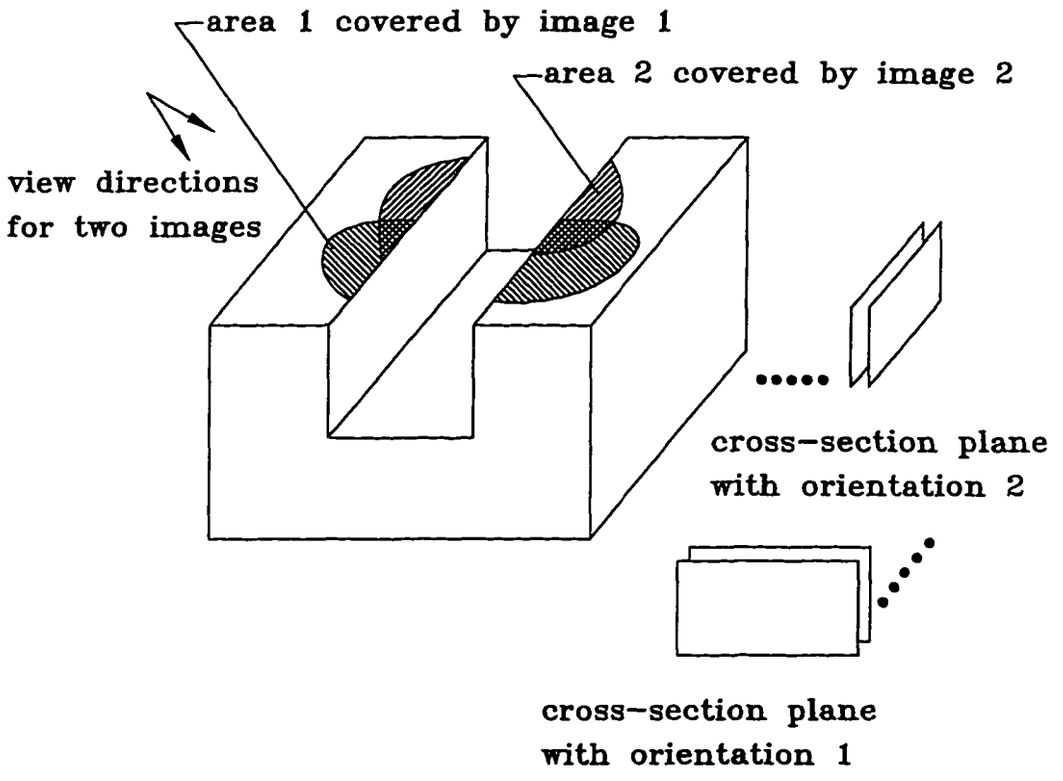


Figure 5.1: Overlaps may occur in multiple places due to occlusion.

procedure carried out by the modeling system for the model growth is given in the following list.

Step 0 – The partial modeling module generates a partial model based on a range image. This partial model involves one or more subsets of cross-section planes in each of which the model is represented by one or more curve sections. The involved planes are initially estimated and later verified during the partial modeling. The details for this step have been discussed in Chapters 3 and 4.

Step 1 – In this step, the procedure is based on three potential cases.

Case 1: If the involved planes are not included in the current database, this range image does not cover any part of a scene that has already been modeled. The plane records for these new planes with the curve sections are inserted in the database, and sorted with other plane records according to the plane locations along the guide curve. This record order will help search for planes in the database to check overlaps with the new-coming partial models. Then, the procedure goes back to Step 0.

Case 2: If the database includes the whole or part of these involved planes, the function to estimate the potential overlaps (see Section 5.3 for details) is applied within each involved plane. If the estimation shows no overlap, the record for the new curve sections are appended after the present curve section records under each the plane record in the database. Then, the procedure goes back to Step 0.

Case 3: If an overlapping possibility exists, the procedure goes to the next step.

Step 2 – The overlapping estimation is further examined. If the overlap does not exist, the database is modified as the *Case 2* in Step 1. If it does exist, the

function to merge the model redundancy (see Section 5.5 for details) is applied within each involved plane. During the merging process the database is updated accordingly. Then, the procedure goes back to Step 0.

5.3 Estimation of Curve Overlap Region

Modeling for the scene covered by a range image may involve many cross-section planes and may generate multiple contour curve sections in each plane. To search for any overlap by directly comparing a newly generated curve section with each of previously generated curve sections in each cross-section plane will be computationally intensive. Note that not all partial model curve sections will have an overlap with the global model curve sections. Furthermore, a partial model curve section does not necessarily have an overlap with all global curve sections. If a screening approach is used to limit the search to an accurate search scope, computation effort can be greatly reduced. This approach should be able to quickly find the partial model curve sections that may have an overlap with the global model curve sections.

The approach used by the developed modeling system for estimating the potential overlaps is based on the principle of maximum enclosure envelope detection. This or similar technology is commonly used in computer graphics and geometric modeling systems. The condition used by the approach to detect the potential overlaps is necessary for overlap existence, but not sufficient. During examination of two curve (partial and global) sections, if the condition is not satisfied, definitely no overlap will occur between curve sections; and if the condition is satisfied, the overlap may happen, but not necessarily. In the second case, further detection to verify the estimation is required.

5.3.1 Maximum Envelope Based Detection

When a partial model is generated, its rectangular maximum enclosure envelope is computed in the plane-based local frame where each cross-section curve is described. A maximum enclosure envelope is described by x_{max}, x_{min}, y_{max} and y_{min} that are maximum/minimum coordinates of a curve section (for a curve section maximum enclosure envelope) or of all curve sections in a plane (for a cross-section plane maximum enclosure envelope) with respect to the 2D Cartesian frame in each involved cross-section plane. A curve section is an individual part of a contour in a plane. Such an envelope is built and attached not only with each curve section for both the partial and global models, but also with each cross-section plane to show the maximum enclosure for all partial and global model curve sections, as shown in Figure 5.2 in a cross-section plane.

Since the global model is always dynamically growing with the new incoming partial models, the envelopes for curves in the global model should be simultaneously updated during the model integration. The envelope for each plane is updated based on the updated curve envelopes in the plane. The following is the procedure of using the maximum enclosure envelope to estimate the potential overlaps. Let H_{g_i} and H_{p_i} represent the plane envelope, and h_{g_s} and h_{p_t} represent the curve envelope with g and p denoting the global and partial model, respectively. Assume the planes involved with a range image are from the plane m to n.

for $i = plane_m$ to $plane_n$
  if *no_plane_envelope_intersection_between_two_planes_exists* then
    simply integrate all partial model curves into the global model;
  else
    for $t = the\_1^{st}\_partial\_model\_curve\_section$ to *the_last_one*

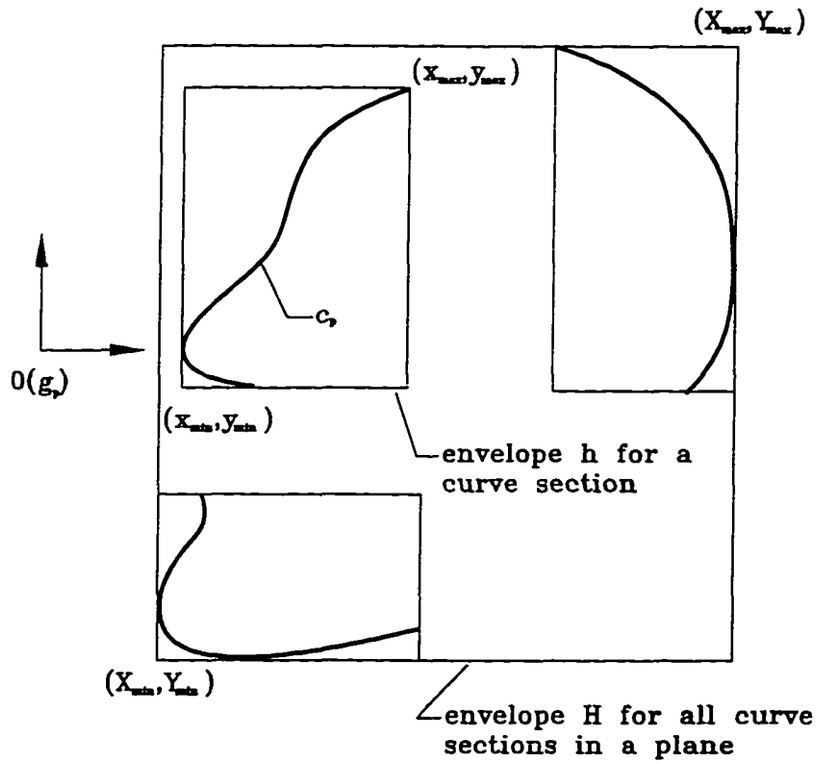


Figure 5.2: A maximum-enclosure envelope.

for s = *the_1st_global_model_curve_section* to *the_last_one*
if *curve_envelope_intersection_exists* then
verify and merge the overlap (see Section 5.5);
update the envelope for this curve section;
remove the tag for the s^{th} *global curve section*
from database (see details below);
end *for_loop (with s)*;
integrate the t^{th} *partial model curve section*
into the global model;
end *for_loop (with t)*;
update the envelope for this plane;
end *for_loop (with i)*;

Since a range image may have overlaps with several other images in different regions, a partial model curve can have overlapped curve sections with multiple global model curves. To handle this situation the fusion algorithm merges the global model curve section into the partial model curve section (not the partial curves merged into the global), and cancels the global model record log in the database. This partial model curve section with the merged global curve will participate in the further overlap detection in the loop with s. After all global model curve sections in the current cross-section plane are checked and merged (if overlaps exist), this partial model curve record is sent to the database as a part of the global model.

5.3.2 Potential Overlap Region

After the overlap estimation algorithm identifies the intersection of two envelopes for the partial model and global model curve sections, a potential curve overlap

region can be found with these two envelopes. The region is the common area of the two rectangular maximum enclosure envelopes. Since a cross-section curve is composed of a set of connected lines obtained from partial modeling, lines enclosed by the rectangular region of the intersected envelopes belonging to the newly generated partial model curve section and the global model curve section respectively, can be found. These lines are the estimation result for the potential curve overlap. This rectangular region can reduce the number of points participating in the computation of the overlap merge.

Figure 5.3 (a), (b) and (c) show three possible intersection patterns of two rectangular envelopes h_1 and h_2. The intersection is only a necessary condition for the existence of curve overlaps, but not sufficient. The intersections between the maximum enclosure envelope h_1 and the envelope in dashed lines in each pattern shows this non-sufficiency. Using the potential overlap region these detection faults can be identified. However, this further identifying method is still unable to handle the worst cases where no true solution exists in this estimation stage. The worst cases happening with the related three envelope intersection patterns are illustrated in Figure 5.4 (a), (b) and (c).

When a surface is close to another surface, their model curves are also close. If the global model includes some curves for a surface, and the partial model includes others for another surface, the estimation method may make a faulty detection. The curve direction, the topological information obtained from the partial modeling and attached with each cross-section curve, can be used to find the fault as shown in Figure 5.5. With a true overlap two involved curve sections must have the same curve direction, since the curve direction is independent from views and is only dependent on the cross-section plane normal.

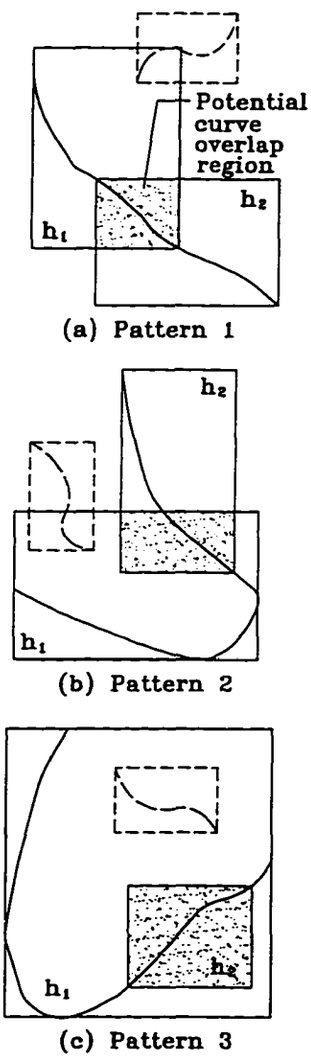
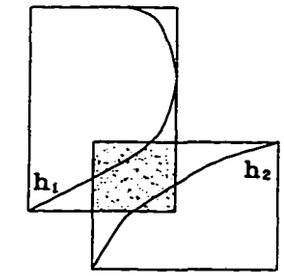
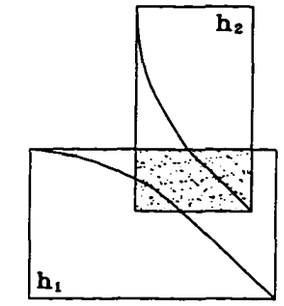


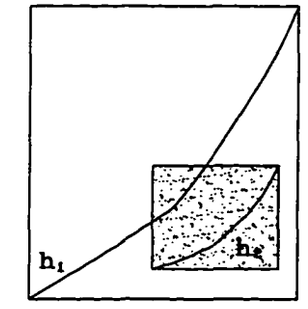
Figure 5.3: The intersection patterns of maximum-enclosure envelopes.



(a) Case 1



(b) Case 2



(c) Case 3

Figure 5.4: The worst cases in curve overlap estimation.

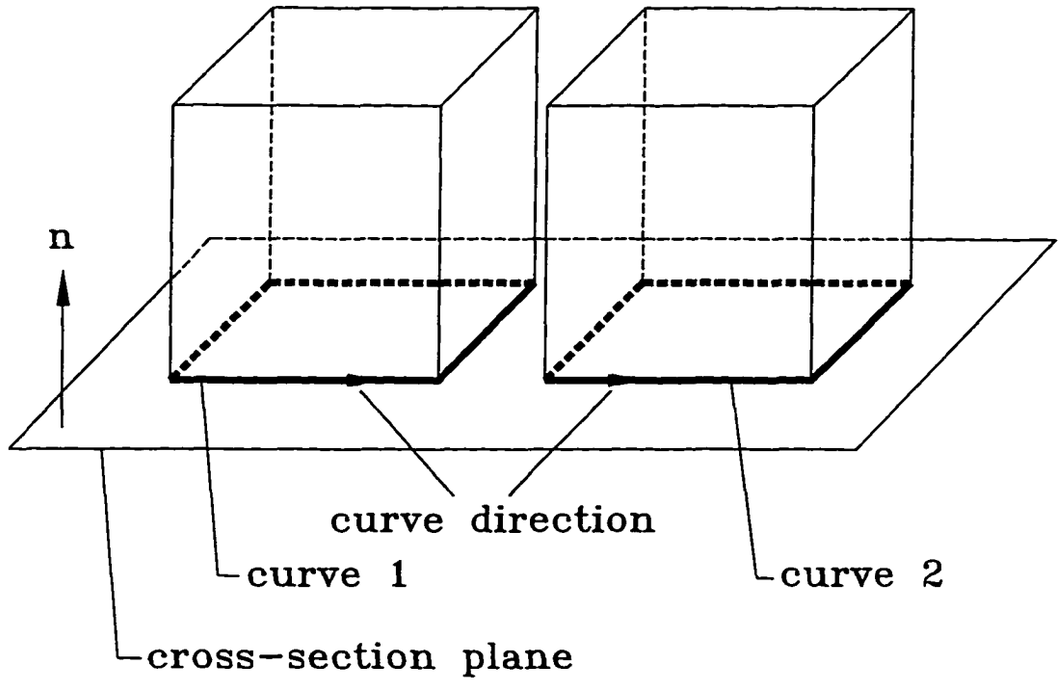


Figure 5.5: Use the curve direction to find the detection fault caused by close surfaces

5.4 Classification of Overlap Patterns

The global model and partial model curve-sections [1] identified by the intersecting region of the two corresponding envelopes can form various overlapping patterns. The patterns are determined by the curve direction and the overlapping location. To facilitate the algorithm design for merging two curves, all possible overlapping patterns should be encountered and be classified.

To find all possible overlapping patterns two steps are used: 1) enumerating all patterns; and 2) using a rule system to examine the possibility of existence of these patterns. The rule system is constructed based on the topology existing in the real world and is described as follows.

[1] In the following discussion the term curve-section means the portion of a curve which is enclosed by the common region of two envelopes.

Rule 1 – No fork "Y" exists in any cross-section curve. Since any object has a volume bounded by its surface, no topology that associates with a fork intersection between a cross-section plane and an object can be found in the real world frame.

Rule 2 – An overlap exits in only one place along either a partial model curve or a global model curve, if these two curves don't form a ring. Otherwise, Rule 1 will be violated.

Rule 3 – At most, two overlaps exist in two places along two involved curves, if these two curves form a ring. Otherwise, Rule 1 will be violated. This rule can be considered as identical to Rule 2, if breaking the ring into two sections anywhere along the curve.

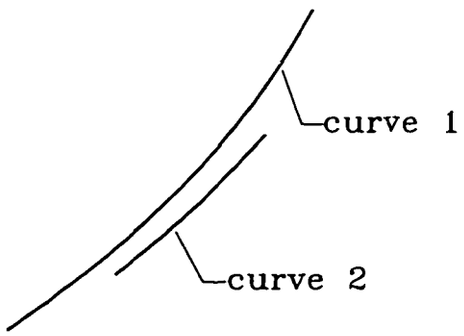


Figure 5.6: The primary overlapping pattern

Based on Rule 2 and Rule 3, a primary overlapping pattern shown in Figure 5.6 is set up. All other pattern variations can be derived from this primary one. Since from the merging point of view the identification (global or partial) of a curve section is not important, a curve will no longer be marked as a global or partial model curve in the following discussion. Based on the primary pattern, for facilitating the pattern enumeration each curve-section is decomposed into three parts. These curve parts

are represented by some symbols [1].

The symbols include lower-case letters for one of the involved curves and upper-case letters for the other. Each of the symbols represents a curve-part, a unenclosed curve-part at the beginning end of a curve section, a unenclosed part at the terminating end, an empty curve-part at the beginning end, and an empty part at the terminating end. Both of the beginning and terminating ends are with respect to the curve direction. An empty curve-part means that no intersection line exists at the beginning/terminating end of a curve section. An empty curve-part is used for facilitating the algorithm design.

a/A : a unenclosed curve-part at the beginning end.

c/C : a unenclosed curve-part at the terminating end.

b/B : an enclosed curve-part.

$\hat{a}/\hat{A}$: an empty curve-part at the beginning end.

$\hat{c}/\hat{C}$: an empty curve-part at the terminating end.

With these symbols a curve-section can be represented as one of the follows:

a b c / A B C
$\hat{a}$ b c / $\hat{A}$ B C
a b $\hat{c}$ / A B $\hat{C}$
$\hat{a}$ b $\hat{c}$ / $\hat{A}$ B $\hat{C}$

[1] In the following discussion the term curve-part means the portion of a curve which is corresponding to the symbol system.

Therefore, there are four possible symbolic strings for a curve-section. When two curve sections are involved in an overlap, one of these symbolic strings for a curve-section can be combined with one of the four strings for another curve-section. Therefore, the enumeration of all combinations of these patterns can be found from

$$\{abc, \hat{a}bc, ab\hat{c}, \hat{a}b\hat{c}\} \times \{ABC, \hat{A}BC, AB\hat{C}, \hat{A}B\hat{C}\}$$

Table 5.1: The checklist for all combinations of curve-part patterns and their validity

	ABC	$\hat{A}$BC	AB$\hat{C}$	$\hat{A}$B$\hat{C}$
abc	N/A (3)	N/A (2)	N/A (1)	VALID
$\hat{a}$bc	N/A (2)	N/A (2)	VALID	VALID
ab$\hat{c}$	N/A (1)	VALID	N/A (1)	VALID
$\hat{a}$b$\hat{c}$	VALID	VALID	VALID	VALID

Table 5.1 provides the details for the enumeration. However, not all of these symbolic-string combinations are valid. Some of them can not exist. The rule system is applied to identify the validity of existences for all combinations. Those invalid patters that violate Rule 1 are marked in the table and classified into three cases which are explained below:

(1) : A fork appears at the beginning end of the merged curve;

(2) : A fork appears at the terminating end of the merged curve;

(3) : A fork appears at the both ends of the merged curve.

All valid combinations can be further classified into five overlapping patters. Figure 5.7 illustrates all valid overlapping patterns: (a) an overlap exists in the beginning end of one curve and the terminating end of another one; (b) an overlap exists in

the whole of two curves; (c) an overlap exists in the whole of one curve and the terminating end of another one; (d) an overlap exists in the whole of one curve and the beginning end of another one; and (e) an overlap exists in the whole of one curve and in the middle of another one. These five patterns will be used for designing the fusion algorithm.

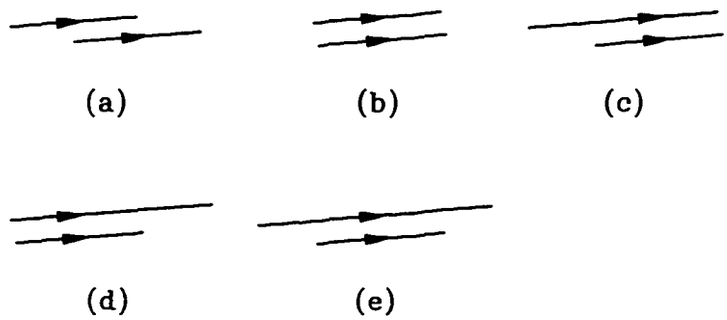


Figure 5.7: All valid overlap patterns

5.5 Merging of Overlapped Curves

5.5.1 Verification of the Overlapping Estimation

With the estimation of a curve overlap the solution for model integration can be found with the reduced curve data enclosed in the region. Since the maximum enclosure envelope has a tolerance that ensures no missing detection for all overlaps, two close curves without overlap may be identified by the detection method. Although the five overlapping patterns are applied during the estimating stage, the result doesn't necessarily mean that a true overlap exists. The estimation needs to be further confirmed.

For convenience to discuss the method for the confirmation and the integration, of two curve-sections enclosed by a common region, one curve section is called distant curve and the other is called basis curve, as shown in Figure 5.8. If the orthogonal projection of the first point (labelled q_1) of a curve is between any two adjacent points ($\overline{p_b p_c}$) of another curve, the former curve is the distant curve and the latter is the basis curve. In the following discussion the line formed by any two adjacent points of a curve is called a curve-unit.

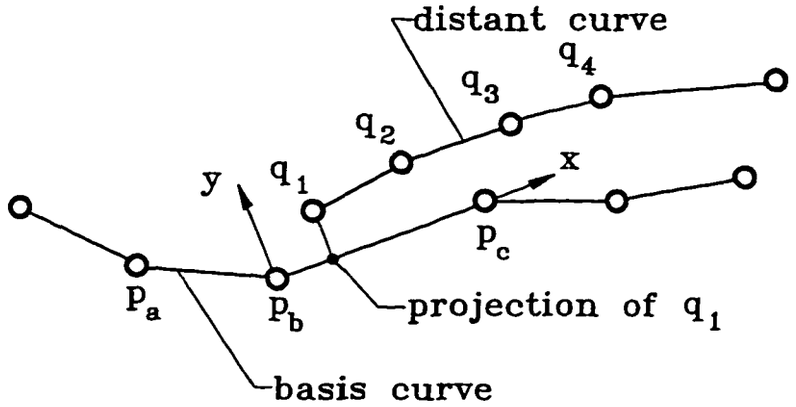


Figure 5.8: Definition of the basis and distant curves

The verification algorithm initially assumes any one of the two curve-sections as a basis curve, and checks if the projection of the first point of another curve-section (thereafter supposed to be a distant curve) is in the first curve-unit of this curve-section. A local 2D coordinate reference system is set up using the first point of the first curve-unit of the assumed basis curve as the origin and the line direction as the positive x-axis. Therefore, the first point becomes $(0, 0)$ and the second one $(x_{end}, 0)$.

The first point of the assumed distant curve is transformed into this local reference system. If $0 \leq x \leq x_{end}$ is true and y is less than a threshold value for this point, the overlap estimation is verified. The threshold value is used to define the maximum distance between two overlapped curve-sections. Otherwise, the algorithm turns to

the next curve-unit of the assumed basis curve-section, sets up the local reference system based on this curve-unit, and checks if the first point of the second curve-section is in this curve-unit. This procedure is repeated until that the estimation is confirmed with the satisfied condition $0 \leq x \leq x_{end}$, or until the last curve-unit of the assumed basis curve is met.

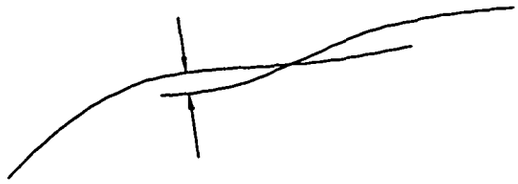
If no verification with the assumed basis/distant curves can be found after reaching the last curve-unit of the assumed basis curve, the algorithm exchanges the assumed basis and distant curves, and follows the same procedure to verify the estimation with the new assumption. If there is still no satisfaction for the condition, the overlap estimation is not true.

5.5.2 Averaging Method

If a true overlap is found, the next step is to use a fusion method to merge the redundancy. During the model integration the fusion method should also be able to handle inaccurate data and errors caused by: 1) local high frequency noises, as shown in Figure 5.9, invoked by the corresponding noises existing in range images; and 2) displacement between a global model curve and a partial model curve due to view location errors.

Selecting a fusion approach is an application oriented issue, and can be based on some criteria for obtaining the merging result emphasizing the smoothness of model curves, simplicity of the fusion method, and reduction of the affect of imaging location errors. The research presented in this work intends to obtain integrated cross-section models using a simple method with smoothing curves and suppressing model errors, although with well-identified overlap redundancy more advanced schemes for fusion can be pursued in the future.

noises in partial models



shift between global/partial model curves due to imaging registration errors

Figure 5.9: Errors and noises existing in the model curves.

An average method has been used in this vision data based modeling system. The method does not rely on sophisticated computation due to not requiring any high order curve fitting. The averaging function, as a low-pass filter, is able to bypass some high frequency noises existing in partial models and is able to find a merged curve between two separated curve-sections for reducing the modeling errors brought by imaging registration errors.

Figure 5.10 shows the fusion principle. With the identified curve-unit $\overline{p_a p_b}$ of the basis curve, a local 2D reference frame is set up in the relevant cross-section plane as mentioned above. Starting from the first point q_1 of the distant curve, each distant curve point is transformed into the local reference frame as (x_i, y_i). Their y values which are the distances of the points to the basis curve-unit are accumulated as $\sum y_i$. This transformation and summation with the points of the distant curve is continued as long as these conditions $0 \leq x_i \leq x_{end}$ and $|y_i| \leq threshold$ are true.

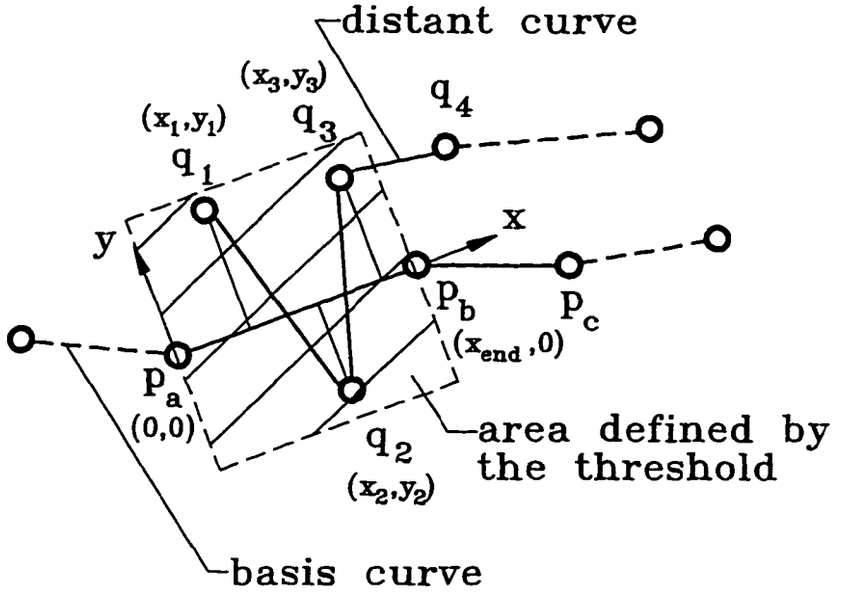
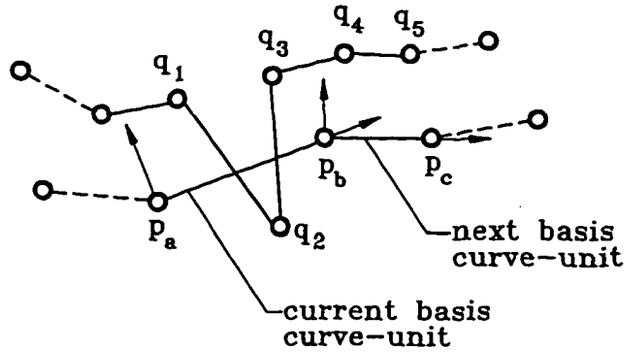


Figure 5.10: Curve integration

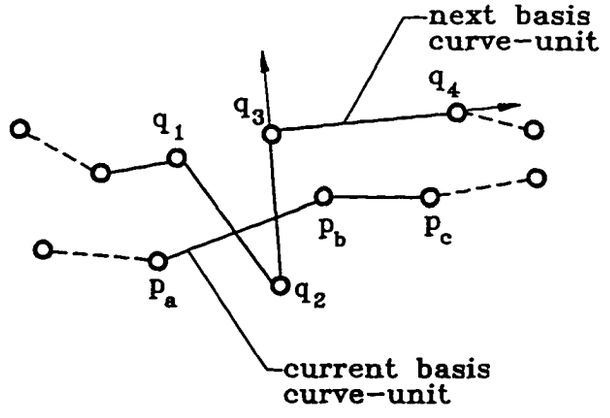
When $x_i > x_{end}$ becomes true, the above computation with the current basis curve-unit is finished. The new point for the integrated model curve is derived as $(\frac{x_{end}}{2}, \frac{\sum y_i}{2n})$, where n means that n points have been involved in the calculation of the summation. The average function $\frac{\sum y_i}{n}$ is a low-pass oriented filter which can smooth the global model curve, while the result reduces the amount of data from n points to 1. As well, $\frac{\sum y_i}{2n}$ reduces the gap between the basis and distant curves by locating the integrated result in the middle of the two curves.

In a normal case, when $x_i > x_{end}$ becomes true, if the condition $|y_i| \leq threshold$ is still satisfied, the algorithm turns to the next basis curve-unit $\overline{p_b p_c}$, as shown in Figure 5.11 (a). With the new curve-unit $\overline{p_b p_c}$ the previous x_{end}, point p_b becomes the origin and p_c becomes the new x_{end}, while a new reference frame is set up based on this new curve-unit. The transformation for the distant-curve point q_4 is re-calculated based upon the new reference.

If the x value of the next point on the distant curve (q_4 in Figure 5.11 (a)) is



(a)



(b)

Figure 5.11: Two possibilities for the fusion with next curve-unit.

not larger than the new x_{end}, the procedure for the transformation and summation is continued. Otherwise, there is no distant-curve point covered by the new basis curve-unit, as shown in Figure 5.11 (b). To continue the fusion procedure the identifications for the two involved model curves are exchanged. The previous basis curve is newly identified as the distant curve, and vice versa. The new basis curve-unit is $\overline{q_3q_4}$. The first distant curve point to be processed is p_b. The above procedure is continued based on the new curve identifications assignment.

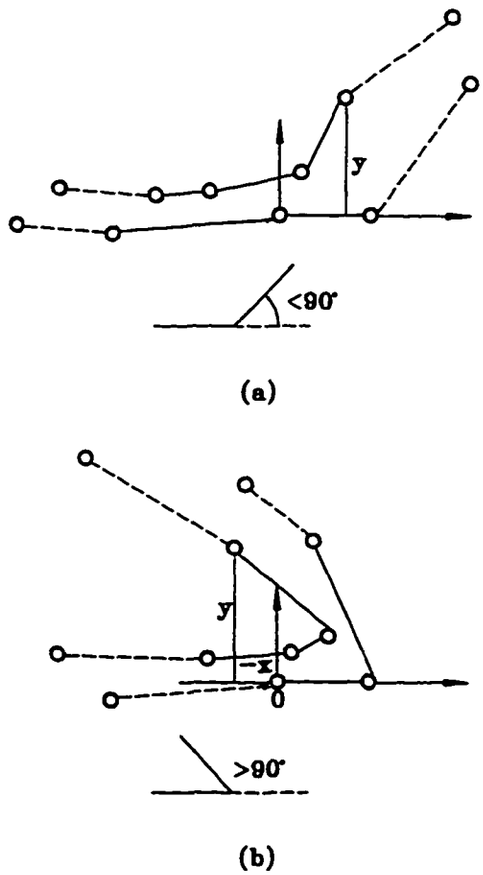


Figure 5.12: Changing curve orientation with sharp corners

In some special cases, the condition $|y_i| \leq threshold$ and/or $x_i \geq 0$ is not satisfied. This is caused either by a sharp corner on a curve or by noises. When the angle at

a corner is less than 90°, the $|y|$ value of a point can be significantly increased, as shown in Figure 5.12 (a). However, noises can also make the partial model curve have dimples represented by increased $|y|$ values. To distinguish a corner from a dimple a look-forward method is used. This method checks the subsequent points along the distant curve. If their $|y|$ values continuously increase, a corner detection can be confirmed. A $|y|$ value decrement after the increment, however, indicates an existence of noises on the curve. When the angle at a corner is greater than 90° shown in Figure 5.12 (b), a $x < 0$ value is met. The look-forward method is used to confirm the corner detection by searching for a continuous decrement for the x values.

5.6 Example Results

5.6.1 An Indoor-Scene Model from Multiple-View Simulated Images

The simulation system for the ERIM sensor (an imaging radar based sensor) was used to synthesize an indoor scene. The scene includes the walls, a table, a chair and a cactus. The pseudo sensor was in turn located in the four corners of the room to generate individual range images of the scene. In these four images the shadow effect is significant due to the scene occlusions. The range images are represented in the intensity images, as shown in Figures 5.13, 5.14, 5.15 and 5.16. Based on each image a partial model is generated. The global cross-section model grows accordingly, the final merged result being illustrated in Figure 5.17. Four views are not enough to cover all aspects of this indoor scene. Therefore, the model gaps caused by view field limit and occlusions can be found on the wall, the table pedestal, the chair and so on.

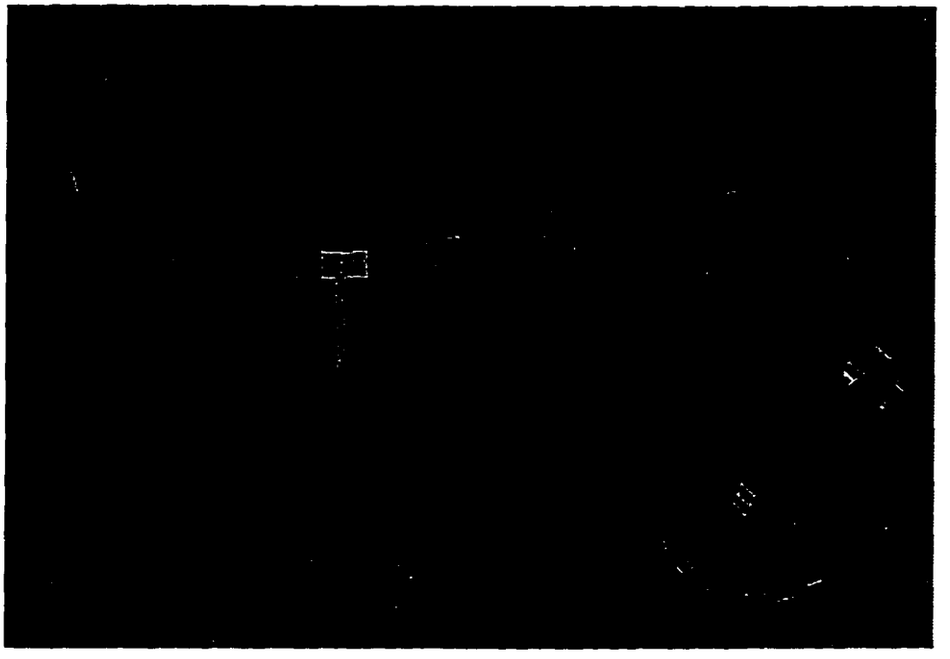


Figure 5.13: A range image from a first sensing location represented by the intensity image.

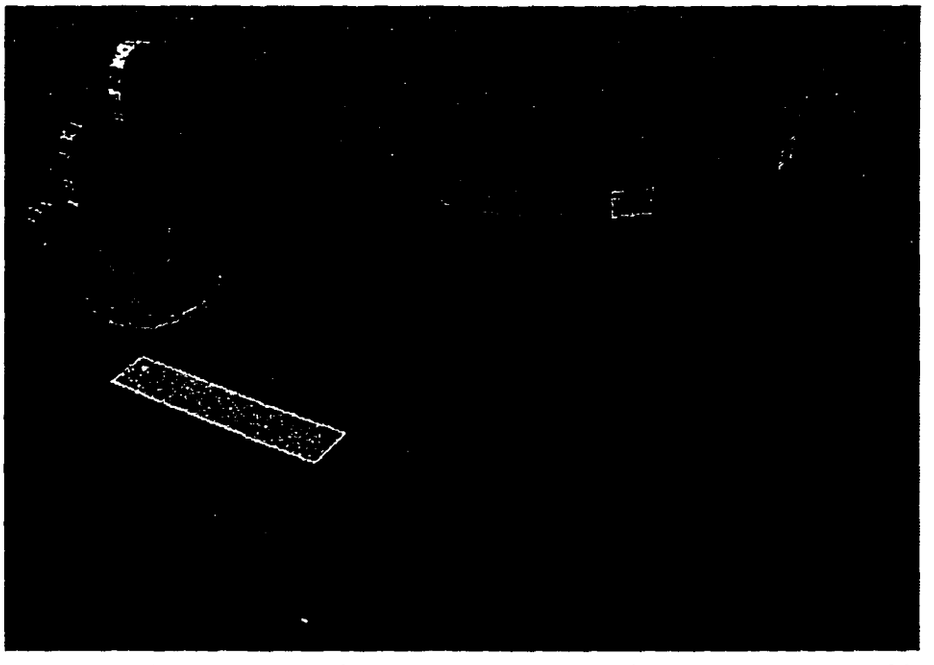


Figure 5.14: A range image from a second sensing location represented by the intensity image.

Figure 5.15: A range image from a third sensing location represented by the intensity image.

Figure 5.16: A range image from a forth sensing location represented by the intensity image.

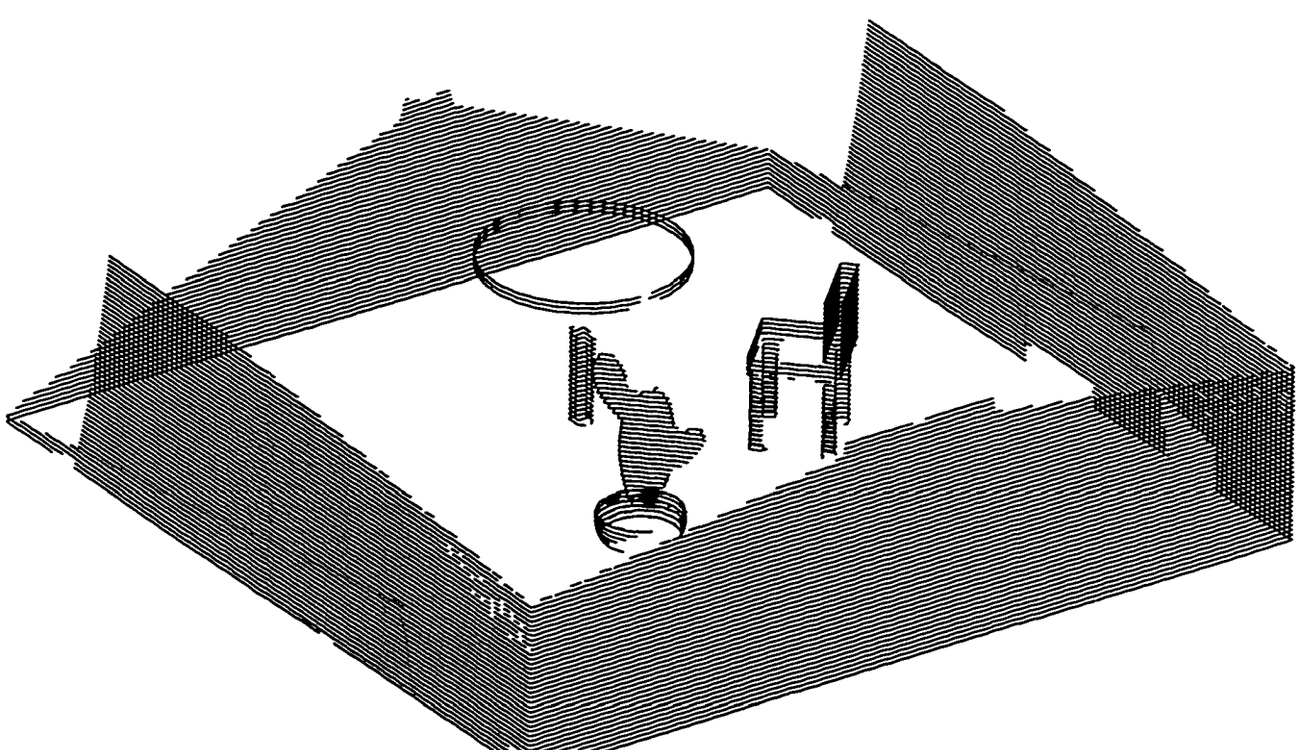


Figure 5.17: The indoor-scene cross-section model based on the four range images

5.6.2 A Face-Mask Model from Multiple-View Hycan Images

This example is associated with modeling a human-face mask. The mask was scanned by the Hyscan scanner (a triangulation based sensor) with multiple views due to the mask area larger than the scanner viewing-scope. Since the mask had an irregular curved surface, the scanner had to be located at different angles to acquire complete data for the surface. The surface points were acquired from range images in six views to test the robustness of the merging algorithm with massive amounts of overlapped data.

The scanner was mounted being parallel with the $X - 0 - Z$ plane of the CMM machine. The location parameters of the scanner were with respect to the machine reference frame $X - Y - Z$ which was used as the global modeling frame as well. Each time when the scanner angle was changed, the scanning system was calibrated by scanning a sphere and the CMM machine table to obtain the location parameters. Figures 5.18, 5.20, 5.22, 5.24, 5.26, 5.28, 5.30 and 5.32 show an intensity representation of the device frame range images of eight views. Figures 5.19, 5.21, 5.23, 5.25, 5.27, 5.29, 5.31 and 5.33 show the corresponding integrated global model growth associated with the images.

Figure 5.18: Scanned surface range image acquired from a first sensing location

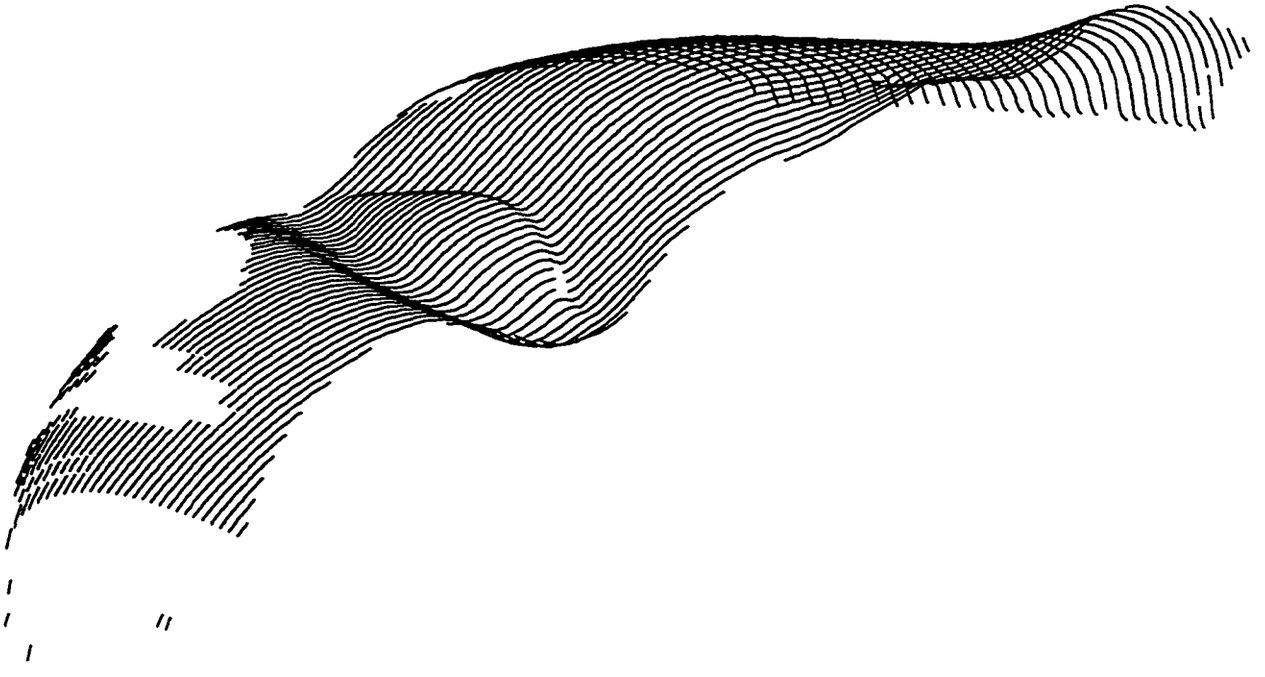


Figure 5.19: The corresponding global model associated with the first image

Figure 5.20: Scanned surface range image acquired from a second sensing location

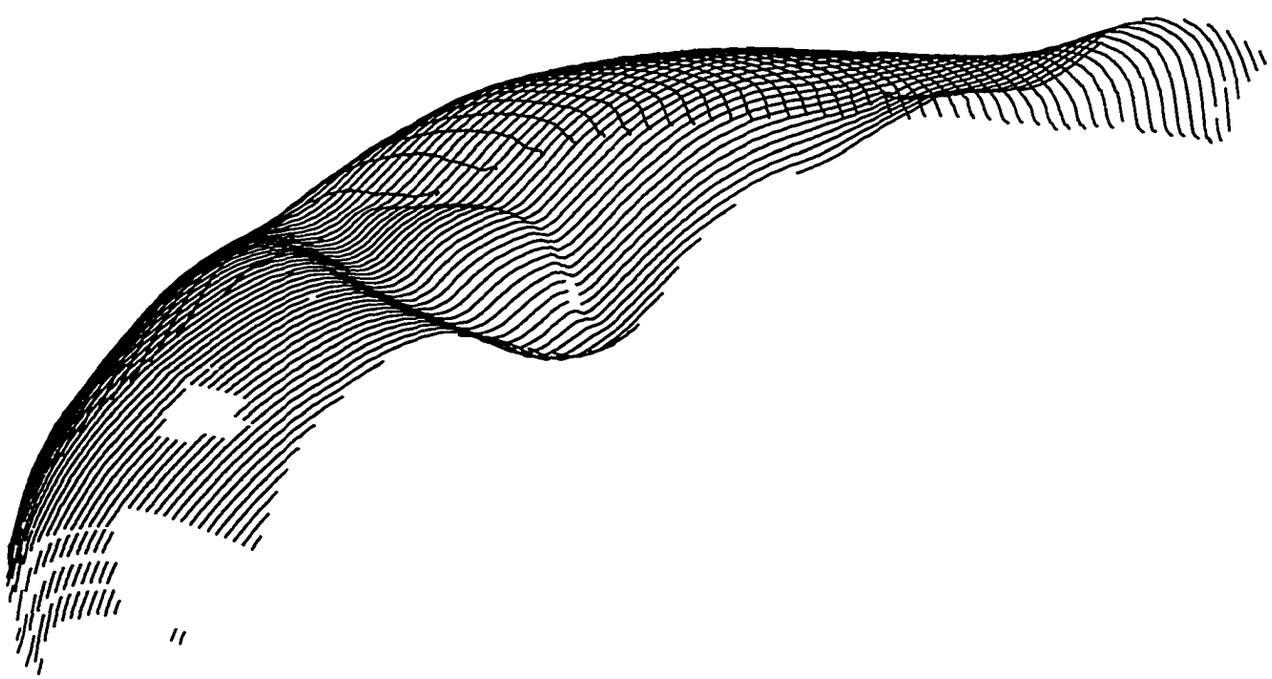


Figure 5.21: The corresponding global model associated with the second image

Figure 5.22: Scanned surface range image acquired from a third sensing location

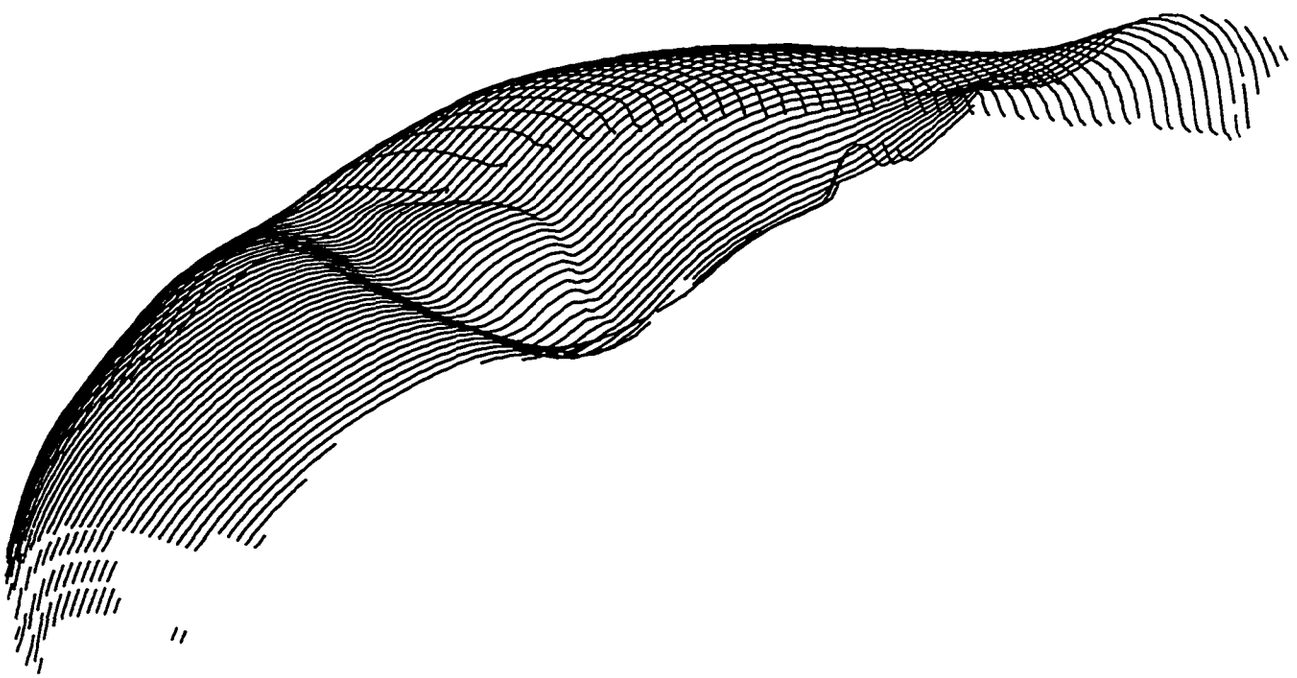


Figure 5.23: The corresponding global model associated with the third image

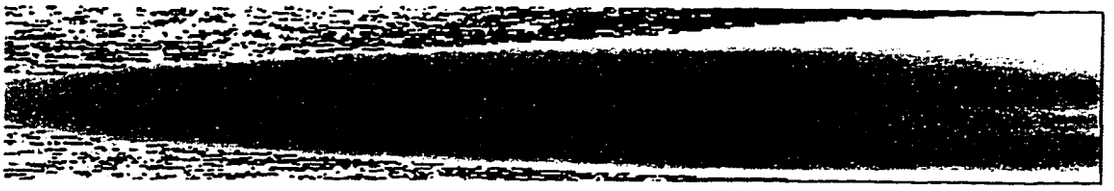


Figure 5.24: Scanned surface range image acquired from a forth sensing location

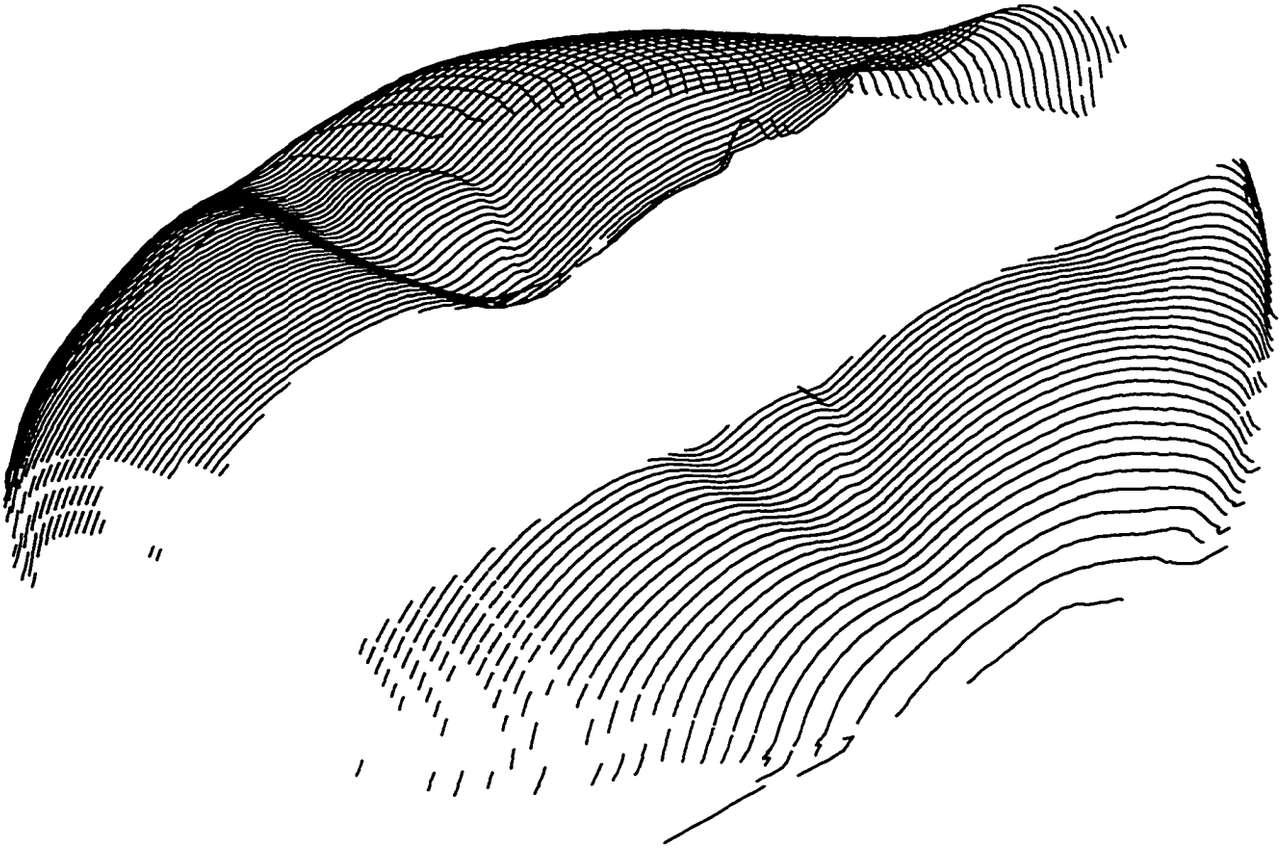


Figure 5.25: The corresponding global model associated with the forth image

Figure 5.26: Scanned surface range image acquired from a fifth sensing location

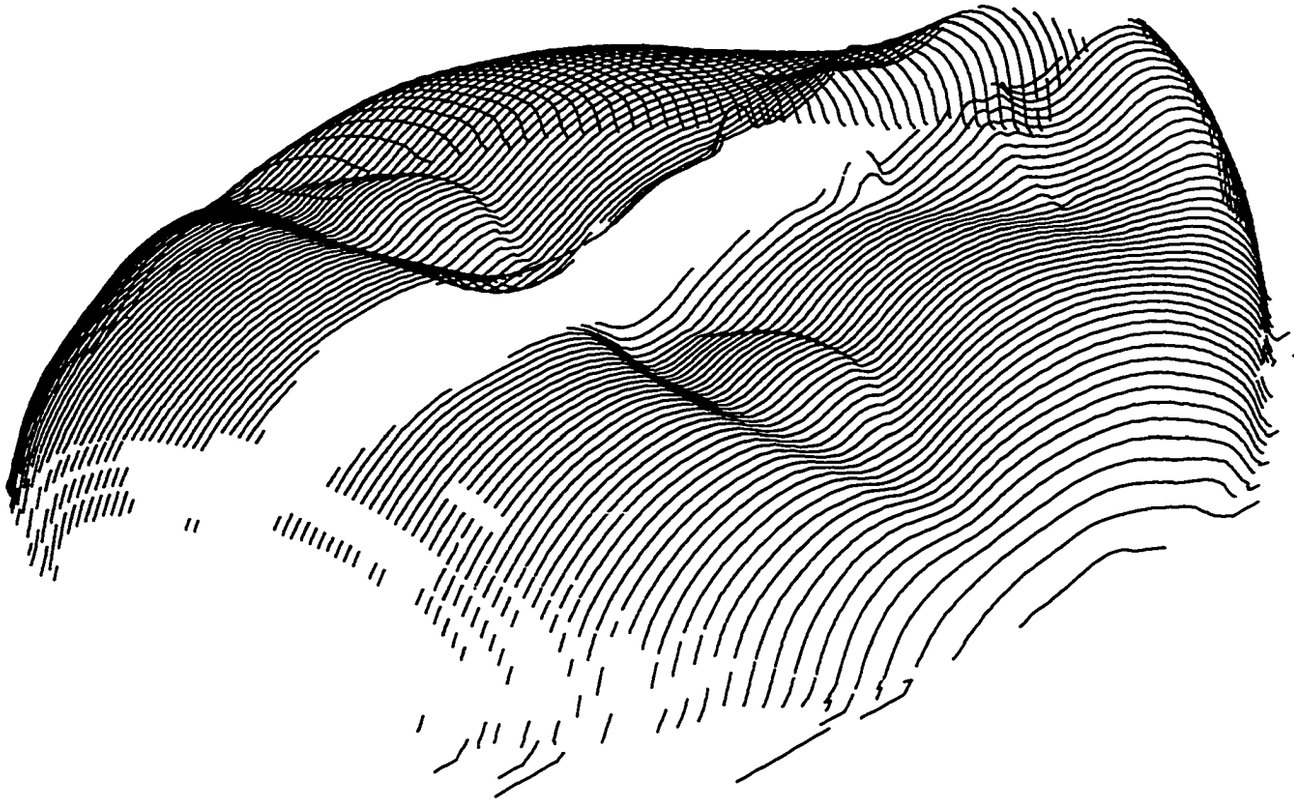


Figure 5.27: The corresponding global model associated with the fifth image

Figure 5.28: Scanned surface range image acquired from a sixth sensing location

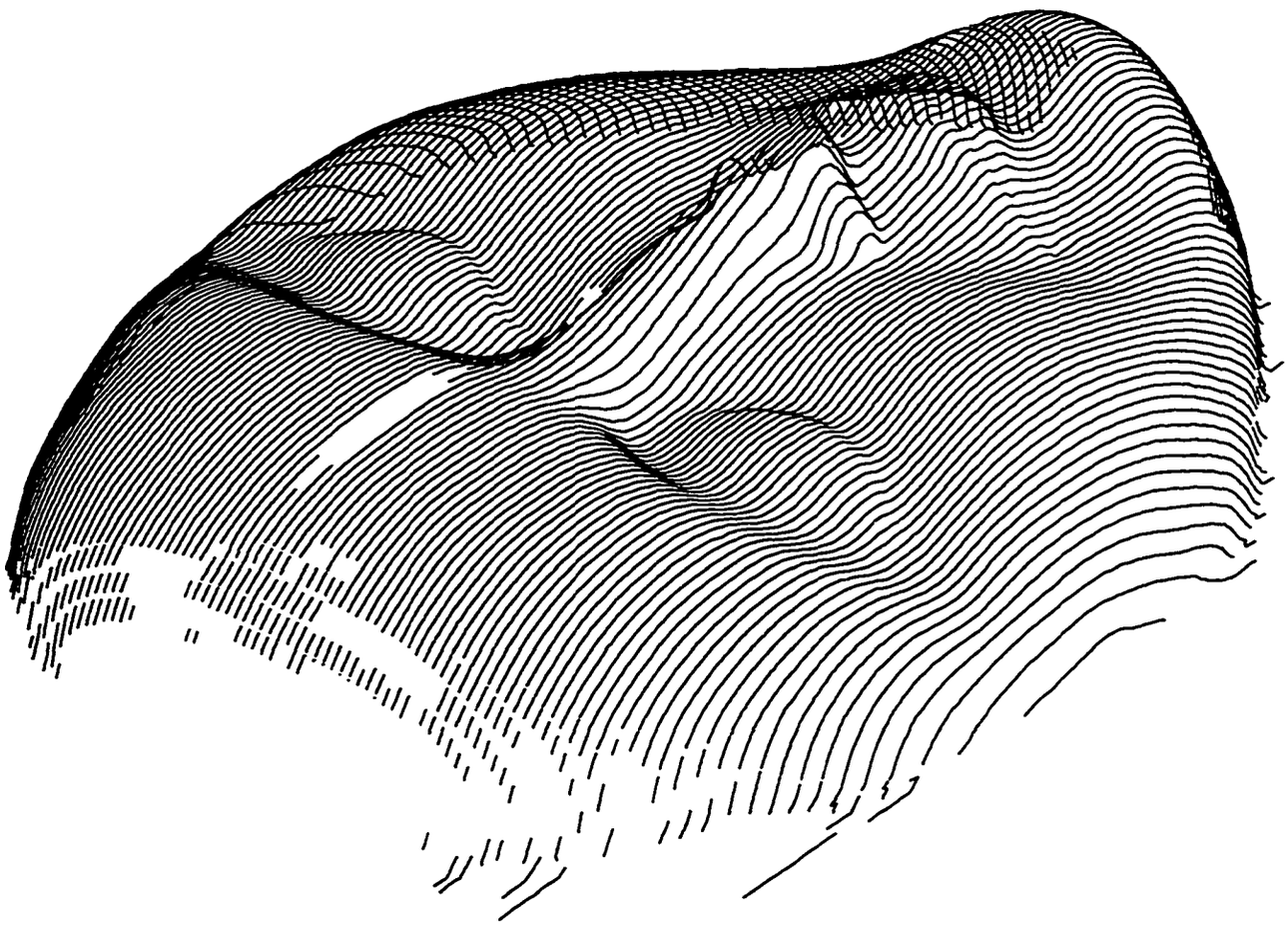


Figure 5.29: The corresponding global model associated with the sixth image

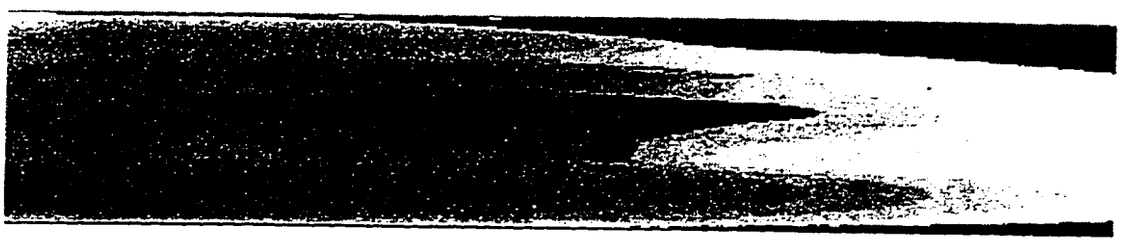


Figure 5.30: Scanned surface range image acquired from a seventh sensing location

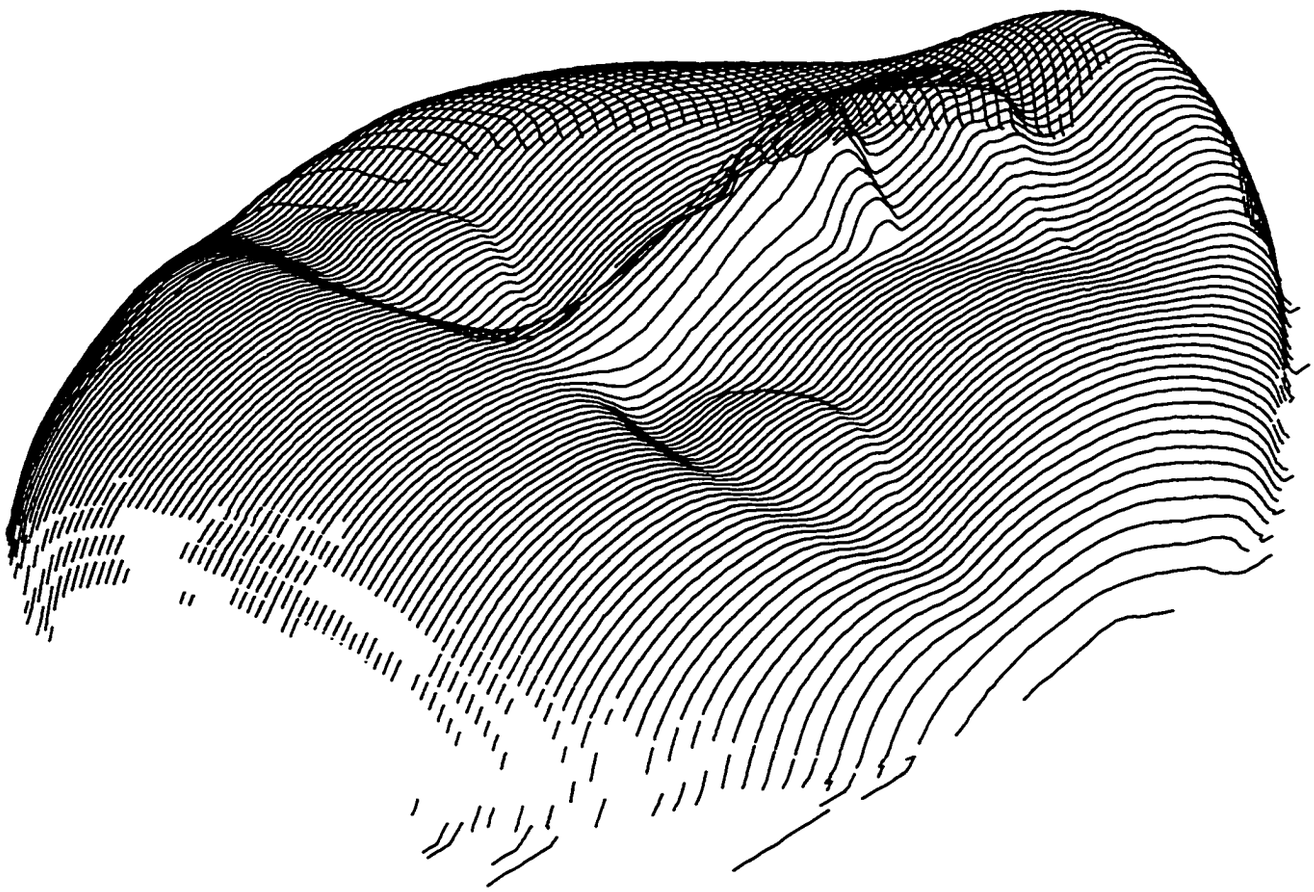


Figure 5.31: The corresponding global model associated with the seventh image

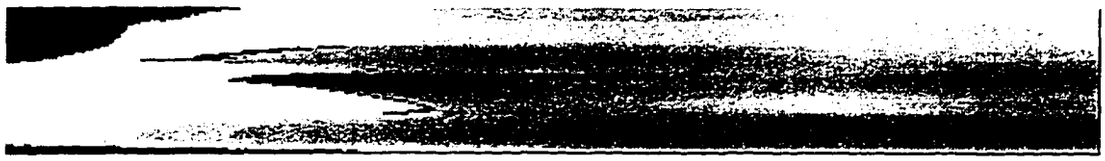


Figure 5.32: Scanned surface range image acquired from an eighth sensing location

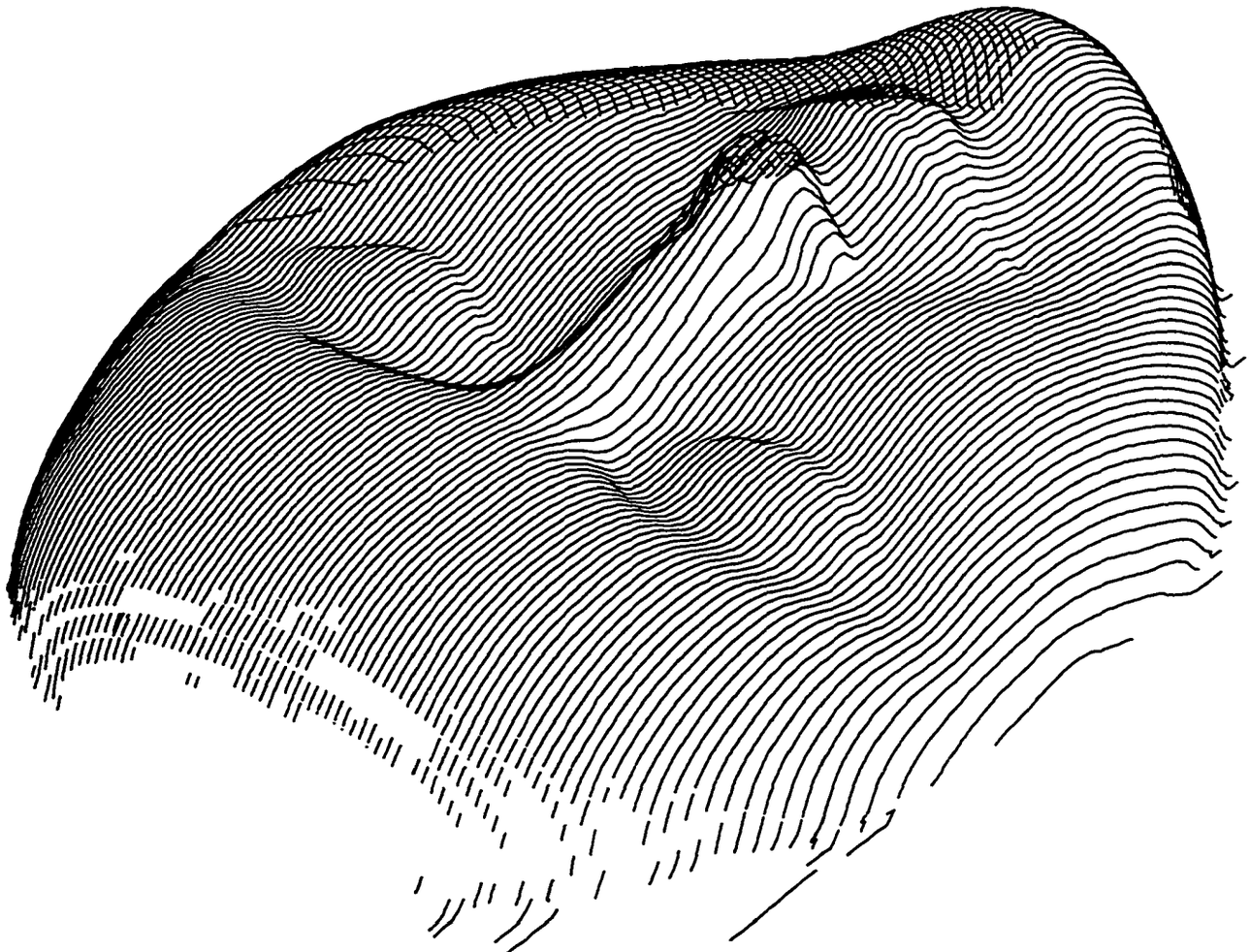


Figure 5.33: The corresponding global model associated with the eighth image

Chapter 6

Conclusions And Future Research Suggestions

6.1 The Research Objectives

The objectives of this research are focused on: developing a new modeling approach and related system based on an appropriate representation allowing dynamic growth of geometric models through partial model generation and model integration; investigating the possibility of using the characteristics associated with the sensor space for facilitating the modeling computation; and testing the modeling approach with multiple types of range sensors.

6.2 Summary of the Completed Research

A new approach to generate geometric models from multiple-view range images for existing objects or scenes was developed within this research. This approach used

a new partial-global modeling method for generating a complete geometric model from multiple-view range images, and employed a cross-section based representation for describing geometric models. Device-frame based generation of partial model was introduced. Global models were generated through sequential model growth and the merge of redundant model data caused by image overlaps. Based upon the approaches, a new range-image based geometric modeling system has been implemented. The modeling method has been verified and tested with three typical range sensors.

Development of the Cross-Section Representation

A geometric modeling system is greatly related to a modeling representation. Representation schemes in different levels have different capabilities to handle geometric and/or topological information carried by models, and require the relevant modeling system to provide input data in different levels as well. A low-level representation cannot provide applications with sufficient data in models. For a vision-data based modeling system the initial input is massive amounts of discrete image data without explicit geometric and/or topological information available. To extract such information from images, for an advanced representation scheme, is very computationally intensive. In addition, a range-data based system requires the representation to be compatible with range data, suitable for partial modeling, capable of multiple-vision fusion and adaptable to geometric details. Therefore, an intermediate-level surface representation is preferred (Chapter 2).

By comparing and evaluating different representation schemes, the cross-section contour based representation was found to be an ideal one. Within the developed modeling system the representation is composed of elements including a guide curve, cross-section planes and cross-section curves. The relationships among these elements are well defined. Without increasing the complexity, this representation was

developed in this research to be capable of handling topological information using a cross-section curve direction for carrying topological information to indicate the exterior/interior volume of an object. The representation was implemented in the data structure of multiple-level trees (Chapter 2).

Development of the Device-Frame Based Partial Modeling

Most range sensors use a central projection principle with equally spaced angular scanning steps of associated mirrors. This makes the range image space a non-Cartesian frame referred to as the device frame in this research. The transformation of range data from a device frame to a Cartesian frame is non-linear. Since the transformed range-data points are irregularly distributed in a Cartesian frame, obstacles for handling these data are caused in terms of accessing the data, and detecting surface discontinuities and missing data.

To avoid the irregularly distributed data present in a Cartesian frame, a new approach for generating partial contour models which takes advantage of the well-distributed data in the device frame was developed. The approach transforms the cross-section planes into the device frame by generating synthesized plane range images. The intersection contours are detected by comparing the object range images and the synthesized plane range images.

In the device frame the algorithm components to generate partial models perform the following functions: estimation of the involved cross-section planes based on the enclosed volume covered by a range image and viewing location, and generation of the synthesized range image for each involved cross-section plane; generation of a labelling image through comparison of the object and plane range values to detect and locate the intersection region; tracing, thinning and linking the intersection pixels; forming a partial model contour curve by applying a subpixel based interpolation method; extracting the topological information the interior/exterior volume of an

object by analyzing the object and plane range values in the intersection region; and transforming the partial model curve into the global Cartesian frame (Chapter 3).

Implementation of the Device-Frames

Since some algorithm components for generating partial models in a device frame are device dependent, the device frames associated with several different sensors were derived. These sensors are commonly used in research or commercial application, and included the imaging-radar based ERIM sensor, and the triangulation based HYSCAN$^{(R)}$ and SmartEye sensors. Associated with the development of the relevant device frames, this research developed: 1) a software simulation system for the ERIM sensor, including sub-systems for geometric modeling and sensor simulation, to provide pseudo range images for facilitating the related research; 2) a prototype of a triangulation-based sensor (SmartEye) including hardware integration and software design; and 3) a transformation lookup table for the HYSCAN scanner developed using data acquisition experiments and data analysis (Chapter 4).

Global Model Generation

The global model generation developed in this research is based upon model growth. After a partial model is generated in the device frame, the partial model is integrated with the present global model generated by previous partial models. Due to scene occlusions and the limited viewing scopes of sensors, multiple-view images are required to acquire sufficient surface data to allow complete modeling. These multiple images may have overlaps which produce a large amount of redundant data. During the global model generation the redundancy must be merge.

The method developed in this research for integration of partial models into the global model is straightforward, if no overlap exists. If the cross-section planes involved with a current range image have never be involved with previous images, no overlaps between the current partial model and the global model exists. Otherwise, a

further overlap estimation is performed using the maximum envelope detection. This estimation is only a necessary condition but not sufficient for the existence of overlaps. A classification method for various potential overlap patterns was developed. Based on the classification result the final verification for overlaps is applied, and then, if confirmed, the overlaps are merged using an averaging approach which is simple and fast.

The global model generation was tested with multiple-view range images generated by the simulation systems for the ERIM sensor and by the experiments with HYSCAN scanner (Chapter 5).

6.3 Conclusions

Cross-Section Contour Representation

By conducting an extensive evaluation for various representation schemes, the research associated with this dissertation has demonstrated that intermediate-level surface based representations are the most suitable for the requirements imposed by multiple-view range image based geometric modeling systems. Such a representation: 1) has relatively easy compatibility with the raw range images and 2) is able to provide applications with sufficient geometric and topological information.

This research has further demonstrated that, besides the above merits, the intermediate level cross-section contour representation has more advanced capabilities including: 1) allowing modeling for both closed and unbounded surfaces; 2) allowing the concurrent performance of image acquisition and model generation; 3) providing the flexibility of selecting view locations; 4) facilitating the merging computation in two-dimensional planes; 5) providing exterior/interior volume based topological

information for applications; and 6) having the flexibility to adjust its resolution for geometric details and model growth. By considering these merits, the developed modeling system adopted the cross-section contour representation for describing geometric models.

Device-Frame Based Partial Modeling

Device-frame based partial cross-section contour modeling, a new concept and approach, has been introduced by this research. A device frame is defined by the coordinates associated with the working principle, scanning activity and the device structure of a range sensor. The new partial modeling approach performs conversion of raw range image data into partial cross-section models in the device frame of range sensors. Using the device frame for partial modeling takes advantage of the uniform distribution of range data in device frames. Uniform range data distribution facilitates the identification of surface discontinuities and the generation of partial models.

Development of Modeling Algorithms

With the cross-section representation and the introduced concept of partial modeling within the device frame, a series of methods and algorithms for the partial or global modeling have been developed. Advantages of the developed methods and algorithms include the following.

The searching algorithm for intersection points in the labeling image is based on the context of the labeling codes of intersection points. This method has enhanced the traditional eight-direction search by enlarging the search scope with fewer search efforts.

Since the extraction of topological information represented by the contour curve direction from range images is simply based on the comparison of range data values between an acquired object image and a generated plane image, the method is easy to implement without using a complex algorithm and increasing the complexity of

the data structure, and the result is robust.

The introduced overlap classification method that finds the valid overlap patterns by applying the rules to all possible patterns enumerated is independent of merging approach, and can be used for detection and verification of overlapping curves, caused by overlapped multiple-view range images, during the global model generation in any similar applications using the cross-section representation.

The averaging based merge approach for multiple-view fusion involves straightforward computations, and is able to merge data overlaps and reduce high-frequency image noises.

System Implementation and Testing Using Typical Sensors

The device frames for several types of commonly used range sensors have been derived. Consequently, the device-frame based partial modeling approach has been tested with these sensors and their associated device frames. The tested sensors work with either imaging radar based or triangulation based sensing principles. The relevant device frames were derived in the form of analytical expressions or as a lookup table. The derivation of these device frames demonstrates that the device-frame based modeling approach is widely applicable with different sensors. The tests with different range sensors have shown the feasibility of the new approach for applications.

6.4 Future Research Suggestions

With the newly developed vision-data based geometric modeling system, this research project provides some fresh topics for further research. In general, the future research should be concentrated on: performing a modeling error analysis for the developed modeling approach; enhancing the modeling approach for applications by considering intelligent view location selection; and continuing development of the

modeling system.

Intelligent Viewing Location Selection

This modeling approach provides subsequent research with a platform for the intelligent view acquisition schemes. Since geometric details in a scene can significantly vary, the adjustment of distance between a sensor and a local surface is necessary for obtaining detailed image data and consequent fine models. In addition, image noise, scene occlusion and/or environmental illumination can affect the correctness or completeness of models. With models provided by this modeling system, an intelligent scheme can be applied to find the required details and/or incompleteness for further image acquisition and refined modeling.

Modeling Error Analysis

Since this research has not investigated the influence of errors on the accuracy of the modeling results, error analysis and error reduction efforts may be important issues in future work. Errors appearing in a global model may reflect several error sources including: image noise, view location registration error and errors associated with partial modeling.

In the device frame, the range image form is exactly the same as the form used by intensive images. The research in the field of image processing has provided various approaches to suppress noises and to improve image quality. Most of these approaches can be applied to a range image to investigate their capability of improving range image quality. Some typical noises can be added in the pseudo range images to examine how the noises may affect modeling results. The view location registration error may cause a curve jump at two ends of overlapped curve sections. To handle such a jump, a curve smoothing method needs to be applied after merging two overlapped cross-section contours.

The errors associated with the partial modeling approach can be identified by

modeling some typical surfaces like planar, cylindrical or spherical surfaces described in pseudo range images. The relevant errors can be found by comparing generated cross-section curves and the analytical surface models used to synthesize the pseudo range images. The errors may change when the orientation of the involved cross-section planes is changed. Finding relationships between types of surfaces and cross-section plane orientations in terms of potential modeling error levels will be useful for improving the modeling approaches.

Simulation System for Various Sensors

The device frames derived for the typical sensors can be used for the development of a simulation system for various range sensors to generate pseudo range images. Such a simulation system would be helpful for a vision-data based system during prototyping, since the range image can be generated for typical objects or scenes, at any viewing orientations and with or without adding noises. Furthermore, when a simulated scene is the same as a real scene, the comparison between the real range image and the simulated one may help find error sources associated with a range sensor.

Advanced Development for the Modeling System

Further development or improvement for this range-data based geometric modeling system can be continued. Further development should focus on: increasing the noise-handling capability for the estimation algorithm to find intersections between an object range image and a plane image; new sub-pixel based interpolation methods to reduce modeling errors; and merging methods to detect for two overlapped model curves which curve is closer to the true surface, and to make the merged curve closer to this curve.

References

[1] Ahlers, R. and Lu, J., "Stereoscopic vision - an application oriented overview," *SPIE Optics, Illumination, and Image Sensing for Machine Vision IV*, Vol. 1194, pp. 298-308, 1989.

[2] Ammann, C. and Sartori-Angus, A., "Fast thinning algorithm for binary images," *Image and Vision Computing*, Vol. 3, No. 2, pp. 71-79, May 1985.

[3] Araki, K., Sato, Y., Tanaka, N. and Fujino, T., "A method for high speed 3-d range measurement and its trial instrumentation," *IEEE 9th International Conference on Pattern Recognition*, Vol. 2, pp. 755-757, Nov. 1988.

[4] Asada, M., "Building a 3-d world model for a mobile robot from sensory data," *IEEE 1988 International Conference on Robotics and Automations*, Vol. 2, pp. 918-923, Apr. 1988.

[5] Asada, M., Kimura, M., Taniguchi, Y. and Shirai, Y., "Dynamic integration of height maps into a 3-d world representation from range image sequences," *International Journal of Computer Vision*, Vol. 9, No. 1, pp. 31-53, 1992.

[6] Baer, A., Eastman, C. and Henrion, M., "Geometric modelling: a survey," *Computer-Aided Design*, 11(5), pp. 253-272, Sep. 1979.

[7] Ballard, D. and Brown, C., *Computer Vision*, Prentice-Hall, Englewood Cliffs, 1982.

[8] Baruch, O., "Line thinning by line following," *Pattern Recognition Letters*, Vol. 8, pp. 271-276, Oct. 1988.

[9] Baumgart, B., "A polyhedron representation for computer vision," *NCC75*, pp. 589-596, 1975.

[10] Besl, P., *Surfaces in Range Image Understanding*, Springer-Verlag, New York, 1988.

[11] Besl, P., "Active optical range imaging sensors," *Advances in Machine Vision*, edited by J. Sanz, Springer-Verlag, pp. 1-63, 1989.

[12] Bradley, C. and Vickers, G. W., "Automated rapid prototyping utilizing laser scanning and free-form machining," *ANNALS of CIPR Manufacturing Technology*, 41(1), pp. 437-440, 1992.

[13] Brunet,P. and Navazo, I., "Solid representation and operation using extended octrees," *ACM Transactions on Graphics*, Vol.9, No.2, pp. 170-197, Apr. 1990.

[14] Chen, B. and Siy, P., "Forward/backward contour tracing with feedback," *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. 9, No. 3, pp. 438-446, May 1987.

[15] Chen, J. and Lin, W., "A new surface interpolation technique for reconstructing 3-d objects from serial cross-sections," *The 9th IEEE International Conference on Pattern Recognition*, Rome, Italy, pp. 1100-1102, Nov. 1988.

[16] Chen, Y. and Medioni, G., "Object modeling by registration of multiple range images," *IEEE International Conference on Robotics and Automation*, pp. 2724-2729, Apr. 1991.

[17] Chen, Y. and Medioni, G., "Description of complex objects from multiple range images using an inflating balloon model," *Computer Vision and Image Understanding*, Vol. 61, No. 3, pp. 325-334, May, 1995.

[18] Clarke, T., "The use of optical triangulation for high speed acquisition of cross sections or profiles of structures," *Photogrammetric Record*, Vol. XIII, No. 76, pp. 523-532, Oct. 1990.

[19] Faugeras, D., Hebert, M., Mussi, P. and Boissonnat, J., "Polyhedral approximation of 3D objects without holes," *Comput. Vision, Graphics, and Image Process.*, Vol. 25, pp. 169-183, 1984.

[20] Foley, J. D., Dam, A. V., Feiner, S.K. and Hughes, J.F., *Computer Graphics Principles and Practice*, Addison-Wesley Publishing Company, 1990.

[21] Gans, D., *Transformations and Geometries*, Meredith Corporation, US, 1969.

[22] Godin, Guy., Roth, G. and Boulanger, P., "Using laser geometric sensing for rapid product development," *SPIE*, 1994.

[23] Harrison, D. and Weir, M., "High-speed triangulation-based 3-d imaging with orthonormal data projections and error detection," *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. 12, No. 4, pp. 409-416, Apr. 1990.

[24] Hebert, M., Ikeuchi, K. and Delingette, H., "A spherical representation for recognition of free-form surfaces," *IEEE PAMI*, Vol. 17, No. 7, pp. 681-690, Nov. 1995.

[25] Higuichi, K., Hebert, M. and Ikeuchi, K., "Building 3-d models from unregistered range images," *CMU-CS-93-214 Technical Report of the Department of Computer Science of Carnegie Mellon University*, 1993.

[26] Higuichi, K., Hebert, M. and Ikeuchi, K., "Building 3-d models from unregistered range images," *Graphical Models and Image Processing*, Vol. 57, No. 4, pp. 315-333, Jul. 1995.

[27] Hilditch, C., "Comparison of thinning algorithms on a parallel processor," *Image and Vision Computing*, Vol. 1, No. 3, pp. 115-132, Aug. 1983.

[28] Hong, T. and Shneier, M.O., "Describing a robot's workspace using a sequence of views from a moving camera," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-7, pp.721-725, 1985.

[29] Jarvis, R., "A perspective on range finding techniques for computer vision," *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. 5, No. 2, pp. 122-139, Mar. 1983.

[30] Jarvis, R., "A laser time-of-flight range scanner for robotic vision," *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. 5, No. 5, pp. 505-512, Sep. 1983.

[31] Jezouin, J. "Building an accurate range finder with off the shelf components," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 195-200, Jun. 1988.

[32] Kamgar-Parsi, B., Jones, J. and Rosenfeld, A., "Registration of multiple overlapping range images: scenes without distinctive features," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 282-290, Jun. 1989.

[33] Kaneko, T., "Line structure extraction from line-drawing images," *Pattern Recognition*, Vol. 25, No. 9, pp. 963-973, 1992.

[34] Kawai, Y., Ueshiba, T., Yoshimi, T. and Oshima, M., "Reconstruction of 3-d objects by integration of multiple range data," *IEEE International Conference on Pattern Recognition*, Vol. A, pp. 154-157, 1992.

[35] Kim, Y. and Aggarwal, J., "Rectangular parallelepiped coding: a volumetric representation of three-dimensional objects," *IEEE J. Robotics Automation*, Vol. RA-2, pp. 127-134, 1986.

[36] Kim, Y. and Luo, R., "Surface reconstruction based on descriptions of cross-sectional contours," *SPIE Sensing and Reconstruction of Three-Dimensional Objects and Scenes*, Vol. 1260, pp. 191-197, Feb. 1990.

[37] Kumar, S., Han, S., Goldgof, D. and Bowyer, K., "On recovering hyperquadrics from range data," *IEEE PAMI*, Vol. 17, No. 11, pp. 1079-1083, Nov. 1995.

[38] Kweon, I. and Kanade, T., "High-resolution terrain map from multiple sensor data," *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. 14, No. 2, pp. 278-292, Feb. 1992.

[39] Masuda, T. and Yokoya, N., "A robust method for registration and segmentation of multiple range images," *Computer Vision and Image Understanding*, Vol. 61, No. 3, pp. 295-307, May, 1995.

[40] Matsumoto, M. "High accuracy laser scanned 3-d range sensor," *IEEE International Workshop on Industrial Applications of Machine Intelligence and Vision (MIV-89)*, Tokyo, pp. 157-162, Apr. 1989.

[41] Mercer, R., McCauley, G. and Anjilvel, S., "Approximation of surfaces in quantitative 3-d reconstructions," *IEEE Trans. on Biomedical Engineering*, Vol. 37, No. 12, pp. 1136-1145, Dec. 1990.

[42] Milroy, M., Bradley, C., Vickers, G. W. and Weir, D., "G^1 continuity of b-spline surface patches in reverse engineering," *Computer-Aided Design*, Vol. 27, No. 6, pp. 471-478, Jun. 1995.

[43] Mortenson, Michael E., *Geometric Modeling*, John Wiley & Sons, Inc., 1985.

[44] Nashashibi, F., Devy, M. and Fillatreau, P., "Indoor scene terrain modeling using multiple range images for autonomous mobile robots," *IEEE International Conference on Robotics and Automation*, pp. 40-46, May 1992.

[45] Newman, T. and Jain, A., "A survey of automated visual inspection," *Computer Vision and Image Understanding*, Vol. 61, No. 2, pp. 231-262, Mar. 1995.

[46] Ney, D., Fishman, E., Magid, D. and Drebin, R., "Volumetric rendering of computed tomography data: principles and techniques," *IEEE Computer Graphics & Applications*, pp. 25-32, Mar. 1990.

[47] Nguyen, Q. and Levine, M., "3-d object representation in range images using geons," *IEEE International Conference on Pattern Recognition*, Vol. A, pp. 149-153, 1992.

[48] Nguyen, Q. and Levine, M., "Representing 3-d object in range images using geons," *Computer Vision and Image Understanding*, Vol. 63, No. 1, pp. 158-168, Jan. 1996.

[49] Nitzan, D., Brain, A. and Duda, R., "The measurement and use of registered reflectance and range data in scene analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 65, No. 2, pp. 206-220, Feb. 1977.

[50] Noborio, H., Fukuda, S. and Arimoto, S., "Construction of the octree approximating three-dimensional objects by using multiple views," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-10, No.6, pp. 769-782, Nov. 1988.

[51] Parvin, B. and Medioni, G., "B-rep from unregistered multiple range images," *IEEE International Conference on Robotics and Automation*, pp. 1602-1607, May 1992.

[52] Raja, N. and Jain, A., "Obtaining generic parts from range images using a multi-view representation," *CVGIP: Image Understanding*, Vol. 60, No. 1, pp. 44-64, Jul. 1994.

[53] Requicha, A., "Representations for rigid solids: theory, methods, and systems," *Computing Surveys*, Vol. 12, No. 4, pp. 437-464, 1988.

[54] Rioux, M., "Laser range finger based on synchronized scanners," *Applied Optics*, Vol. 23, No. 21, pp. 3837-3844, Nov. 1984.

[55] Rioux, M., Blais, F., Beraldin, J. and Boulanger, P., "Range imaging sensors development at nrc laboratories," *IEEE Workshop on Interpretation of 3D Scenes*, pp. 154-160, Nov. 1989.

[56] Rosenfeld, A., "Algorithms for image/vector conversion," *Computer Graphics*, Vol. 12, pp. 135-139, Aug. 1978.

[57] Soucy, M. and Laurendeau, D., "Building a surface model of an object using multiple range views," *SPIE Intelligent Robots and Computer Vision X*, Vol. 1608, pp. 85-96, 1991.

[58] Soucy, M., Croteau, A. and Laurendeau, D., "A multi-resolution surface model for compact representation of range images," *IEEE International Conference on Robotics and Automation*, pp. 1701-1706, May 1992.

[59] Soucy, M. and Laurendeau, D., "Multi-resolution surface modeling from multiple range view," *IEEE Conference ?*, pp. 348-353, 1992.

[60] Soucy, M. and Laurendeau, D., "A general surface approach to the integration of a set of range views," *IEEE PAMI*, Vol. 17, No. 4, pp.344-358, Apr. 1995.

[61] Srinivasan, P., Liang, P. and Hackwood, S., "Computational geometric methods in volumetric intersection for 3-d reconstruction," *IEEE International Conference on Robotics & Automation*, Vol. 1, pp. 190-195, 1989.

[62] Stenstrom, J. and Connolly, C., "Constructing object models from multiple images," *International Journal of Computer Vision*, Vol. 9, No. 3, pp. 185-212, 1992.

[63] Theodoracatos, V. and Calkins, D., "A 3-d vision system model for automatic object surface sensing," *ASME 1992 on Advances in Design Automation*, pp. 191-205, 1992.

[64] Veatch, P. and Davis, L., "Efficient algorithms for obstacle detection using range data," *Computer Vision, Graphics, and Image Processing*, Vol. 50, pp. 50-74, 1990.

[65] Wang, Y. and Aggarwal, J., "An overview of geometric modeling using active sensing," *IEEE Control System Magazine*, Vol. 8, issue 3, pp. 5-13, Jun. 1988.

[66] Wei, C. and Skinner, C., "Finite element mesh generation with computed tomography scan data," *ASME Conference on Computer in Engineering*, Vol. 2, pp. 271-279, 1991.

[67] Weiler, K., "Edge-based data structures for solid modeling in curved-surface environments," *CG & A*, 5(1), pp. 21-40, Jan. 1985.

[68] Wong, A. K. C., Lu, S. and Rioux, M., "Recognition and shape synthesis of 3-d objects based on attributed hypergraphs," *IEEE PAMI*, Vol. 11, No. 3, pp. 279-290, 1989.

[69] Yao, H., Podhorodeski, R., and Dong, Z., "Modeling and simulation of a range-finding device," *Proceeding of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 166-169, May 1991.

[70] Yao, H., Dong, Z. and Podhorodeski, R., "Simulation of range finding devices by geometric modeling," *Proceedings of the ASME Computers in Engineering Conference and Exposition*, pp. 327-332, 1991.

[71] Yao, H., Podhorodeski, R., and Dong, Z., "A cross-section based multiple-view range image fusion approach," *SPIE Sensor Fusion and Aerospace Applications*, Vol. 1956, pp. 212-223, Apr. 1993.

[72] Yao, H., Podhorodeski, R., and Dong, Z., "Object and workspace modeling based on a 3-d vision system," *Chapter 11: Artificial Intelligence in Optimal Design and Manufacturing*, edited by Z. Dong, PTR Prentice Hall, New Jersey, pp. 285-310, 1994.

[73] Yokoya, N., Kaneta, M, and Yamamoto, K., "Recovery of superquadric primitives from a range image using simulated annealing," *IEEE International Conference on Pattern Recognition*, Vol. A, pp. 168-172, 1992.

[74] Yoshida, K. and Hirose, S., "Laser triangulation range finder available under direct sunlight," *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 1702-1708, Apr. 1988.

[75] Yoshimura, K. and Okamoto, S., "A three-dimensional sensor for automatic visual inspection of soldered parts," *IECON'89 15th Annual Conference of IEEE Industrial Electronics Society*, Vol. 3, pp. 562-567, Apr. 1989.

[76] Zeid, I., *CAD/CAM Theory and Practice*, McGraw-Hill, Inc., 1991.

Appendix A

Computational Complexity Analysis for Partial Model Generation with ERIM Range Sensor

The conclusion of Section 4.2.3 that the computational costs associated with device-frame based partial model generation are lower than that with the Cartesian frame is related to the analysis reported in this appendix. For the analysis some expressions are used, which are

Homogeneous coordinate transformation:

$$\begin{bmatrix} X, & Y, & Z, & 1 \end{bmatrix} = \begin{bmatrix} x, & y, & z, & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{A.1}$$

Plane equation:

$$Ax + By + Cz + D = 0 \tag{A.2}$$

Frame transformation equations from a Spherical to a Cartesian frame:

$$\begin{aligned} x(r, \theta, \phi) &= r\cos(\phi)\cos(\theta) \\ y(r, \theta, \phi) &= r\cos(\phi)\sin(\theta) \\ z(r, \phi) &= r\sin(\phi) \end{aligned} \tag{A.3}$$

In addition, it is assumed that: 1) the range image is a $m \times n$ array; 2) there are on average l cross-section planes associated with each image; and 3) there are on average p intersection points for each cross-section curve section.

The curve section generation in the modeling (Cartesian) frame needs a number of *if*, *addition*, and *multiplication* operations in several different stages as follows.

- Transform surface points from the device frame to the device based Cartesian frame using Eq. (A.3):

  $5m \cdot n$ (multiplication)

- Transform surface points from the device based Cartesian to the modeling Cartesian frame according to Eq. (A.1):

  $9m \cdot n$ (multiplication)

  $9m \cdot n$ (addition)

- Compute maximum and minimum X, Y and Z to determine the number of the involved planes:

$$6m \cdot n \quad \text{(if)}$$

- Generate curve sections. A curve section is composed of short pieces of intersection lines. Each short line is an intersection of a cross-section plane and a four-point-defined surface patch. This intersection computation needs testing the point positions related to the plane using Eq. (A.2), interpolating the four-point-defined patch and computing the intersection. The computational costs for the intersection computation associated with the last two steps are indicated by C_{int} due to its dependency on a certain scheme. The costs of these steps are:

$$\begin{aligned} &12l \cdot m \cdot n && \text{(multiplication)} \\ &12l \cdot m \cdot n && \text{(addition)} \\ &8l \cdot m \cdot n && \text{(if)} \\ &l \cdot p \cdot C_{int} && \text{(intersection)} \end{aligned}$$

- The total computational costs of the curve section generation in the modeling frame are:

$$\begin{aligned} &12l \cdot m \cdot n + 14m \cdot n && \text{(multiplication)} \\ &12l \cdot m \cdot n + 9m \cdot n && \text{(addition)} \\ &8l \cdot m \cdot n + 6m \cdot n && \text{(if)} \\ &l \cdot p \cdot C_{int} && \text{(intersection)} \end{aligned}$$

The curve section generation in the device frame needs a number of if, *addition*, and *multiplication* operations in several different stages.

- Compute maximum and minimum range values to determine the number of the involved planes:

$$2m \cdot n \quad \text{(if)}$$

- Transform the involved planes from the modeling frame to the device frame. Since only a limited number of cross-section planes will be involved, the computation of transforming the planes from the modeling frame to the device based Cartesian frame can be ignored. From Eqs. (A.2) and (A.3), transformation of the planes from the device based Cartesian to the device frame can be found. Solving for the cross-section plane range values for known range-sensor mirror angles, θ and ϕ, yields

$$r(\theta,\phi) = \frac{-d}{a\cos(\phi)\cos(\theta) + b\cos(\phi)\sin(\theta) + c\sin(\phi)}$$

  where a, b, c and d are the plane equation coefficients with respect to the device based Cartesian frame. The costs for finding r for $\theta = 1,...,m$ and $\phi = 1,...,n$ are:

$$\begin{aligned} &l \cdot m \cdot (4n+3) && \text{(multiplication)} \\ &2l \cdot m \cdot n && \text{(addition)} \end{aligned}$$

- Generate curve sections using a similar scheme to the one discussed for Cartesian frame based partial modeling. The computational costs are:

$$\begin{aligned} &8l \cdot m \cdot n && \text{(if)} \\ &l \cdot p \cdot C_{int} && \text{(intersection)} \end{aligned}$$

- Transform generated curve sections from the device frame to the modeling frame using Eqs. (A.3) and (A.1):

$$\begin{aligned} &14l \cdot p && \text{(multiplication)} \\ &9l \cdot p && \text{(addition)} \end{aligned}$$

- The total computational costs of the curve section generation in the device

frame are:

$14l \cdot p + 4l \cdot m \cdot n + 3l \cdot m$	(multiplication)
$9l \cdot p + 2l \cdot m \cdot n$	(addition)
$8l \cdot m \cdot n + 2m \cdot n$	(if)
$l \cdot p \cdot C_{int}$	(intersection)