

In our system, in addition to the 3-D range image, we also get a binary mask image which identifies the valid entries for the object in the scene in the 3-D image from the background.

The software system currently used for the range finder system is provided by K<sup>2</sup>T Inc., Pittsburgh, PA. The system utilizes a calibration cube with calibration marks on three adjacent visible surfaces. The system runs on a Sun SparcStation under X windows.

## **B.2 Data Collection and Processing**

Before each data collection session, the LCRF system is calibrated using the K<sup>2</sup>T software system. In addition, a white flat panel is mounted on the rotary table vertically and two sets of range images are taken at two different rotary table position. 3-D planes are fit to the resulting range images and the rotation center of the rotary table is determined. This, together with the two calibration matrices, is stored and associated to every range image taken after the calibration.

For each object, a set of 8 to 12 wraparound range images are taken first by rotating the rotary table for the needed angles. The rotation angles between successive range images are recorded and used for estimating the initial range image registration transformations together with the rotation center information mentioned above (Chapter 3). In order to cover other parts of an object's surface, we then position the object in a different pose from the one in which the wraparound views are taken. This time another set of wraparound range images may be taken with fewer views. The initial transformation of this set of range images and the first set can be estimated by the operator, or by registering two range images, one from each set, by hand-picking correspondence points and using a standard pose estimation algorithm. The resulting transformation can then be propagated to all range images in one of the sets as the initial transformation for registration. More range images can be taken as needed.

All range images used in this dissertation, except those obtained from NRCC (National Research Council of Canada) (the teapot images in Chapter 3), are used as they are acquired from the range finder without any preprocessing, except that one layer of boundary points (approximately one pixel wide) of the data areas of each image is masked out, because of the high occurrence of outliers on the boundary.

location of the points in the scene relative to the projector. This information, together with the camera and projector calibration information, is subsequently used to compute the actual 3-D coordinates at each image pixel using triangulation.

The LCRF system's calibration consists in two parts, the projector and the camera. The camera defines a 2-D parameter space  $(x_c, y_c)$ , and the projector defines a 1-D (vertical) parameter space  $x_p$ . Using a homogeneous coordinate system, any 3-D point  $(x, y, z)$  can be projected into the camera coordinates and projector coordinates as follows: [79]

$$\begin{bmatrix} h_c x_c \\ h_c y_c \\ h_c \end{bmatrix} = \mathbf{C} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad \begin{bmatrix} h_p x_p \\ h_p \end{bmatrix} = \mathbf{P} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (\text{B.1})$$

where

$$\mathbf{C} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \end{bmatrix}$$

are camera and projector calibration matrices respectively. Once the camera coordinates  $(x_c, y_c)$  and the projector coordinate  $x_p$  (the space code corresponding to the Gray code value of the Gray-coded image at  $(x_c, y_c)$ ) are known, the 3-D coordinates of the corresponding scene location can be computed: [79]

$$\begin{bmatrix} x & y & z \end{bmatrix}^t = \mathbf{Q}^{-1} \mathbf{F} \quad (\text{B.2})$$

where

$$\mathbf{Q} = \begin{bmatrix} C_{11} - C_{31}x_c & C_{12} - C_{32}x_c & C_{13} - C_{33}x_c \\ C_{21} - C_{31}y_c & C_{22} - C_{32}y_c & C_{23} - C_{33}y_c \\ P_{11} - P_{21}x_p & P_{12} - P_{22}x_p & P_{13} - P_{23}x_p \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} C_{34}x_c - C_{14} \\ C_{34}y_c - C_{24} \\ P_{24}x_p - P_{14} \end{bmatrix}$$

This way, a 3-D image in the camera coordinates can be computed:

$$x = x(x_c, y_c), \quad y = y(x_c, y_c), \quad z = z(x_c, y_c)$$

## APPENDIX B

# Experimental Range Finder System

### B.1 System Configuration

Throughout this dissertation, a structure-light based range finder system is used, which is called Liquid Crystal Range Finder (LCRF) after the liquid crystal mask that is used in the system to produce the stripe patterns. The system was designed by Sato and Inokuchi [79] of Osaka University, Osaka, Japan. In the following, we briefly describe the system principles and our system set up for taking the range images used in this dissertation. The LCRF system components are illustrated in Figure B.1. The projector projects 8 patterns with horizontal black-and-white stripes in different widths to simulate the bit-planes of an image which encodes the line numbers of the image with Gray code for space encoding in the vertical direction. The images of the scene illuminated with the stripe patterns are captured with a video camera and the digitized images are assembled into an 8-bit image after binarization. This 8-bit Gray-coded image encodes the spatial

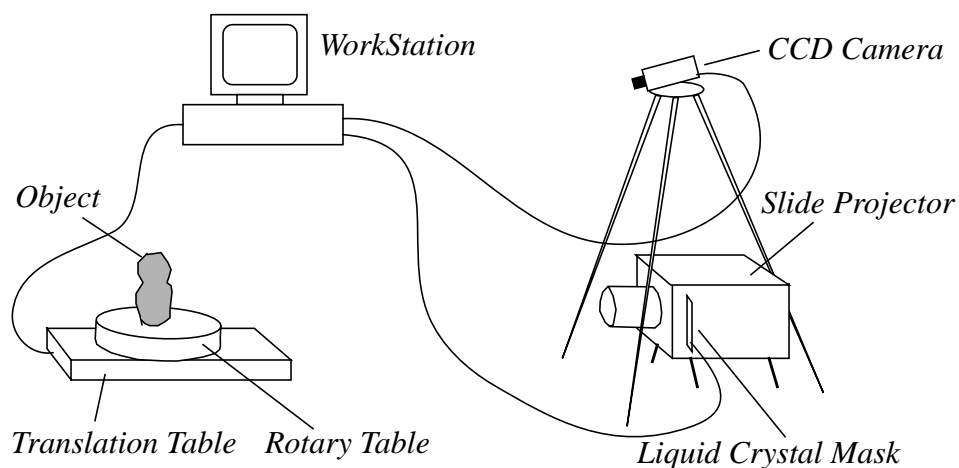


Figure B.1 The components of the LCRF system.

and  $Trace ( )$  is the trace of a square matrix. Now take the partial derivatives of  $e$  against  $\mathbf{v}$  and set  $\frac{\partial e}{\partial \mathbf{v}} = 0$ , we have

$$\frac{\partial e}{\partial \mathbf{v}} = 2\mathbf{H}^t(\mathbf{v}^t\mathbf{H})^t + 2\mathbf{H}(\mathbf{P}^t)^t = 0$$

which gives

$$\mathbf{v} = -(\mathbf{H}^t\mathbf{H})^{-1}\mathbf{H}^t\mathbf{P} \quad (\text{A.11})$$

when the pseudo-inverse  $(\mathbf{H}^t\mathbf{H})^{-1}\mathbf{H}^t$  exists.

Notice that based on the assumption we made in Chapter 3,  $T_\Delta$  will only contain small rotation angles, thus from Equation (3.2) we have:

$$T_\Delta = \begin{bmatrix} 1 & \gamma & -\beta & t_x \\ -\gamma & 1 & \alpha & t_y \\ \beta & -\alpha & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{A.6})$$

when considering  $\sin(x) \approx x$  and  $\cos(x) \approx 1$ , and ignoring terms of order 2 and higher. Substituting Equations (A.3), (A.5) and (A.6) into Equation (A.4) we get:

$$d_s = (A_j, B_j, C_j, D_j) \begin{bmatrix} 1 & \gamma & -\beta & t_x \\ -\gamma & 1 & \alpha & t_y \\ \beta & -\alpha & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} (x_i, y_i, z_i, 1)^t = \mathbf{H}_i \mathbf{v} + P_i \quad (\text{A.7})$$

where  $\mathbf{v} = (\alpha, \beta, \gamma, t_x, t_y, t_z)^t$  is the variable vector, and

$$\mathbf{H}_i = \begin{bmatrix} C_j y_i - B_j z_i \\ A_j z_i - C_j x_i \\ B_j x_i - A_j y_i \\ A_j \\ B_j \\ C_j \end{bmatrix}, \quad P_i = S_j^t \mathbf{p}_i \text{ for } i = 1 \dots N \quad (\text{A.8})$$

Now Equation (A.1) can be written as:

$$\begin{aligned} e &= \sum_N (\mathbf{H}_i \mathbf{v} + P_i)^2 = \text{Trace} [(\mathbf{H}\mathbf{v} + \mathbf{P})(\mathbf{H}\mathbf{v} + \mathbf{P})^t] \\ &= \text{Trace} [\mathbf{H}\mathbf{v}\mathbf{v}^t \mathbf{H}^t + \mathbf{H}\mathbf{v}\mathbf{P}^t + \mathbf{P}\mathbf{v}^t \mathbf{H}^t + \mathbf{P}\mathbf{P}^t] \end{aligned} \quad (\text{A.9})$$

where

$$\mathbf{H} = (\mathbf{H}_1, \mathbf{H}_2, \dots, \mathbf{H}_N)^t, \quad \mathbf{P} = (P_1, P_2, \dots, P_N)^t \quad (\text{A.10})$$

## APPENDIX A

# The Least-Squares Solution for Registration

In Chapter 3 we introduced an iterative registration algorithm based on minimizing a distance measure that reflects the quality of registration (Equation (3.11)). We rewrite this equation here and for convenience reason we drop the superscripts and simply denote  $T^{k-1}\mathbf{p}_i$  as  $\mathbf{p}_i$ , since they are all known and can be precomputed:

$$e = \sum_{i=1}^N d_s^2(T_{\Delta}\mathbf{p}_i, S_j) \quad (\text{A.1})$$

The plane  $S_j$  can be expressed as follows:

$$A_jx + B_jy + C_jz + D_j = \mathbf{S}_j^t \mathbf{x} = 0 \quad (\text{A.2})$$

where

$$\mathbf{S}_j = (A_j, B_j, C_j, D_j)^t \quad (\text{A.3})$$

is the plane coefficient vector, and  $\mathbf{x} = (x, y, z, 1)^t$  is a 3-D point in the homogeneous coordinates. Let  $\mathbf{p}_i = (x_i, y_i, z_i, 1)^t$ , the signed distance function  $d_s$  can be written as follows:

$$d_s = \frac{\mathbf{S}_j^t T_{\Delta} \mathbf{p}_i}{\sqrt{A_i^2 + B_i^2 + C_i^2}} \quad (\text{A.4})$$

If we normalize the plane coefficient vectors, we have

$$\sqrt{A_i^2 + B_i^2 + C_i^2} = 1. \quad (\text{A.5})$$



- [78] M. Vasilescu and D. Terzopoulos. Adaptive Meshes and Shells: Irregular Triangulation, Discontinuities, and Hierarchical Subdivision. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 829–832, Urbana-Champaign, IL, June 1992.
- [79] B. C. Vemuri and J. K. Aggarwal. 3-D Model Construction from Multiple Views Using Range and Intensity Data. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 435–437, 1986.
- [80] Y. F. Wang and J. K. Aggarwal. Integration of Active and Passive Sensing Techniques for Representing Three-Dimensional Objects. *IEEE Transactions on Robotics and Automation*, 5(4):460–471, Aug. 1989.
- [81] P. P. Watson. Computing the N-dimensional Delaunay Triangulation with Application to Voronoi Polytopes. *Computer Journal*, 24(2):167–172, May 1981.
- [82] G. Wyvill, C. McPheeters, and B. Wyvill. Data Structures for Soft Objects. *The Visual Computer*, 2(4):227–234, Aug. 1986.
- [83] M. Zerroug and R. Nevatia. Quasi-Invariant Properties and 3-D Shape Recovery of Non-Straight, Non-Constant Generalized Cylinders. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 96–103, New York, NY, June 1993. IEEE Computer Society Press.
- [84] Z. Zhang. Iterative Point Matching for Registration of Free-form Curves and Surfaces. Technical report, INRIA, Sophia-Antipolis, 1992. Research Report 1658.

- [66] T. M. Sobh, J. Owen, C. Jaynes, M. Dekhil, and T. C. Henderson. Industrial Inspection and Reverse Engineering. In *Proceedings of the 2nd CAD-Based Vision Workshop*, pages 228–235, Champion, PA, Feb. 1994.
- [67] F. Solina and R. Bajcsy. Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):131–147, Feb. 1990.
- [68] M. Soucy and D. Laurendeau. Multi-Resolution Surface Modeling from Range Views. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 348–353, Urbana-Champaign, IL, June 1992.
- [69] F. Stein and G. Medioni. Structural Indexing: Efficient Three Dimensional Object Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 125–146, Feb. 1992.
- [70] G. Taubin. An Improved Algorithm for Algebraic Curve and Surface Fitting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 658–665, Berlin, Germany, May 1993. IEEE Computer Society Press.
- [71] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D. Kriegman. Parameterizing and Fitting Bounded Algebraic Curves and Surfaces. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 103–108, Champaign, IL, June 1992. IEEE Computer Society Press.
- [72] D. Terzopoulos and D. Metaxas. Dynamic 3D Models with Local and Global Deformations: Deformable Superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.
- [73] D. Terzopoulos and M. Vasilescu. Sampling and Reconstruction with Adaptive Meshes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 70–75, Maui, HI, June 1991. IEEE, Computer Society Press.
- [74] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-Seeking Models for 3D Object Reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 269–276, London, England, June 1987.
- [75] V. Torre and T. A. Poggio. On Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):147–163, Mar. 1986.
- [76] F. Ulupinar and R. Nevatia. Recovering Shape from Contour for SHGCs and CGCs. In *Proceedings of the DARPA Image Understanding Workshop*, pages 544–556, Pittsburgh, Pennsylvania, Sept. 1990.
- [77] R. Vaillant and O. Faugeras. Using Extremal Boundaries for 3-D Object Modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:157–173, 1992.

- [52] A. Pentland. Recognition by Parts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 612–620, London, England, June 1987.
- [53] J. Ponce and D. J. Kriegman. On Recognizing and Positioning Curved 3-D Objects from Image Contours. In *Proceedings of the DARPA Image Understanding Workshop*, pages 461–470, Palo Alto, CA, 1989.
- [54] M. Potmesil. Generating Models for Solid Objects by Matching 3D surface Segments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1089–1093, Karlsruhe, West Germany, August 1983.
- [55] M. Potmesil. Generating Octree Models of 3D Objects from Their Silhouettes in a Sequence of Images. *Computer Vision, Graphics, and Image Processing*, 40:1–29, 1987.
- [56] F. Preparata and M. Shamos. *Computational Geometry: An Introduction*. Springer, Berlin, 1985.
- [57] K. Rao and R. Nevatia. Computing Volume Descriptions From Sparse 3-D Data. *International Journal of Computer Vision*, 2(1):33–50, June 1987.
- [58] M. C. Rivara. Algorithms for Refining Triangular Grids Suitable for Adaptive and Multigrid Techniques. *International Journal for Numerical Methods in Engineering*, 20:745–756, 1984.
- [59] H. Rom. *Part Decomposition and Shape Description*. Ph.D. dissertation, University of Southern California, Los Angeles, CA, 1993. (also USC-IRIS Technical Report 93-319).
- [60] H. Rom and G. Medioni. Hierarchical Decomposition and Axial Representation of Shape. In *Proceedings of the DARPA Image Understanding Workshop*, pages 607–613, San Diego, California, January 1992.
- [61] B. Sabata and J. K. Aggarwal. Estimation of Motion From a Pair of Range Images: A Review. *Computer Vision, Graphics, and Image Processing*, 54(3):309–324, Nov. 1991.
- [62] P. Saint-Marc, J.-L. Jezouin, and G. Medioni. A Versatile PC-Based Range Finding System. *IEEE Transactions on Robotics and Automation*, 7(2):250–256, Apr. 1991.
- [63] K. Sato and S. Inokuchi. Range-Imaging System Utilizing Nematic Liquid Crystal Mask. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 657–661, London, England, June 1987.
- [64] F. Schmitt and X. Chen. Fast Segmentation of Range Images into Planar Regions. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 710–711, Maui, Hawaii, June 1991. Computer Society Press.
- [65] W. B. Seales and O. D. Faugeras. Building Three-Dimensional CAD/CAM Models from Image Sequences. In *Proceedings of the 2nd CAD-Based Vision Workshop*, pages 116–123, Champion, PA, Feb. 1994. IEEE Computer Society Press.

- [38] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 259–268, London, England, June 1987. IEEE Computer Society Press.
- [39] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 1:321–331, 1988.
- [40] J. J. Koenderink. What Does the Occluding Contour Tell Us about Solid Shape? *Perception*, 13:321–330, 1984.
- [41] K. N. Kutulakos, W. B. Seales, and C. R. Dyer. Building Global Object Models by Purposive Viewpoint Control. In *Proceedings of the 2nd CAD-Based Vision Workshop*, pages 168–182, Champion, PA, Feb. 1994. IEEE Computer Society Press.
- [42] J. M. Lavest, R. Glachet, M. Dhome, and J. T. Lapreste. Modeling Solids of Revolution by Monocular Vision. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 690–691, Maui, Hawaii, June 1991. Computer Society Press.
- [43] C.-W. Liao and G. Medioni. Surface Approximation of a Cloud of 3D Points. In *Proceedings of the 2nd CAD-Based Vision Workshop*, pages 274–281, Champion, PA, Feb. 1994. IEEE Computer Society Press.
- [44] C.-E. Liedtke, H. Busch, and R. Koch. Shape Adaptation for Modeling of 3D Objects in Natural Scenes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 704–705, Maui, Hawaii, June 1991. Computer Society Press.
- [45] D. G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, 1985.
- [46] T. McInerney and D. Terzopoulos. A Finite Element Model for 3D Shape Reconstruction and Nonrigid Motion Tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 518–523, Berlin, Germany, May 1993.
- [47] P. Meer, D. Mintz, D. Y. Kim, and A. Rosenfeld. Robust Regression Methods for Computer Vision. *International Journal of Computer Vision*, 6(1):59–70, Apr. 1991.
- [48] R. Nevatia and T. O. Binford. Description and Recognition of Complex-Curved Objects. *Artificial Intelligence*, 8:77–98, 1977.
- [49] T. S. Newman and J. Anil K. CAD-Based Inspection of 3D Objects Using Range Images. In *Proceedings of the 2nd CAD-Based Vision Workshop*, pages 236–243, Champion, PA, Feb. 1994.
- [50] B. Parvin and G. Medioni. A Dynamic System for Object Description and Correspondence. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 393–399, Maui, June 1991.
- [51] B. Parvin and G. Medioni. B-rep from Unregistered Multiple Range Images. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 1607–1613, Nice, France, May 1992.

- [26] F. P. Ferrie, J. Lagarde, and P. Whaite. Darboux Frames, Snakes and Super-Quadrics: Geometry from the Bottom-Up. In *Proceedings of the Workshop on Interpretation of 3D Scenes*, pages 170–176, Austin, TX, Nov. 1989. Computer Society Press, Washington, DC.
- [27] L. D. Floriani and E. Puppo. Constrained Delaunay Triangulation for Multiresolution Surface Description. In *Proceedings of the International Conference on Pattern Recognition*, pages 566–569, Rome Italy, Nov. 14-17 1988. Computer Society Press, Washington, DC.
- [28] A. Gross and T. Boulton. Recovery of Generalized Cylinders from a Single Intensity View. In *Proceedings of the DARPA Image Understanding Workshop*, pages 557–564, Pittsburgh, PA, Sept. 1990. Morgan Kaufmann Publishers, Inc.
- [29] A. Guezic. Large Deformable Splines, Crest Lines, and Matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 650–657, Berlin, Germany, May 1993. IEEE Computer Society Press.
- [30] A. Gupta, L. Bogoni, and R. Bajcsy. Quantitative and Qualitative Measures for the Evaluation of the Superquadric Models. In *Proceedings of the Workshop on Interpretation of 3D Scenes*, pages 162–169, Austin, TX, Nov. 1989. Computer Society Press, Washington, DC.
- [31] A. Gutpa. *Surface and Volumetric Segmentation of Complex 3-D Objects Using Parametric Shape Models*. PhD thesis, University of Pennsylvania, June 1991. Technical Report GRASP LAB 267 and MS-CIS-91-45.
- [32] K. Higuchi, H. Delingette, M. Hebert, and K. Ikeuchi. Merging Multiple views Using a Spherical Representation. In *Proceedings of the 2nd CAD-Based Vision Workshop*, pages 124–231, Champion, PA, Feb. 1994. IEEE Computer Society Press.
- [33] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface Reconstruction from Unorganized Points. In *Proceedings of the SIGGRAPH Conference*, volume 26, pages 71–78, Chicago, IL, July 1992. ACM.
- [34] B. K. B. Horn and J. G. Harris. Rigid Body Motion from Range Image Sequences. *CVGIP: Image Understanding*, 53(1), Jan. 1991.
- [35] W.-C. Huang and D. B. Goldgof. Adaptive-Size Meshes for Rigid and Nonrigid Shape Analysis and Synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):611–616, June 1993.
- [36] C. P. Jerian and R. Jain. Structure from Motion-A Critical Analysis of Methods. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):572–588, May/June 1991.
- [37] B. Kamgar-Parsi, J. L. Jones, and A. Rosenfeld. Registration of Multiple Overlapping Range Images: Scenes without Distinctive Features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:857–871, Sept. 1991.

- [13] T. E. Boult and A. D. Gross. On the Recovery of Superellipsoids. In *Proceedings of the DARPA Image Understanding Workshop*, pages 1052–1063, 1988.
- [14] R. A. Brooks. Model-Based Three-Dimensional Interpretations of Two-Dimensional Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):140–150, 1983.
- [15] Y. Chen and G. Medioni. Object Modelling by Registration of Multiple Range Images. *International Journal of Image and Vision Computing*, 10(3):145–155, Apr. 1992.
- [16] Y. Chen and G. Medioni. Surface Level Integration of Multiple Range Images. In *Proceedings of the Workshop on Computer Vision in Space Applications*, Antibes, France, Sept. 1993.
- [17] Y. Chen and G. Medioni. Fitting a Surface to 3-D points Using an Inflating Balloon Model. In *Proceedings of the 2nd CAD-Based Vision Workshop*, pages 266–273, Champion, PA, Feb. 1994. IEEE Computer Society Press.
- [18] Y. Chen and G. Medioni. Surface Description of Complex Objects Using Multiple Range Images. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 153–158, Seattle, WA, June 1994. IEEE Computer Society Press.
- [19] C. H. Chien, Y. B. Sim, and J. K. Aggarwal. Generation of Volume/Surface Octree from Range Data. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 254–260, 1988.
- [20] L. D. Cohen and I. Cohen. Finite-Element Methods for Active Contour Models and balloons for 2-D and 3-D Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1131–1147, 1993.
- [21] H. Delingette, M. Hebert, and K. Ikeuchi. Shape Representation and Image Segmentation Using Deformable Surfaces. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 467–472, Maui, HI, June 1991. IEEE, Computer Society Press.
- [22] H. Delingette, M. Hebert, and K. Ikeuchi. A Spherical Representation for the Recognition of Curved Objects. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 103–112, Berlin, Germany, May 1993. IEEE Computer Society Press.
- [23] T.-J. Fan, G. Medioni, and R. Nevatia. Recognizing 3-D Objects Using Surface Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1140–1157, Nov. 1989.
- [24] G. E. Farin. *Curves And Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, 1990. (2nd Edition).
- [25] O. Faugeras, N. Ayache, and B. Faverjon. A Geometric Matcher for Recognizing and Positioning 3-D Rigid Objects. In *Proceedings of the Conference on Artificial Intelligence and Applications*, pages 218–224, 1984.

## BIBLIOGRAPHY

- [1] N. Ahuja and J. Veenstra. Generating Octrees from Object Silhouettes in Orthographic Views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(2):137–149, 1989.
- [2] S. Barnard and M. Fischler. Computational Stereo. *ACM Computing Surveys*, 14(4):553–572, Dec. 1982.
- [3] A. H. Barr. Superquadrics and Angle Preserving Transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, Jan. 1981.
- [4] P. J. Besl. The Free-Form Surface Matching Problem. In H. Freeman, editor, *Machine Vision for Three-Dimensional Scenes*, pages 25–71. Academic Press, Inc., 1990.
- [5] P. J. Besl and N. D. Mckay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb. 1992.
- [6] B. Bhanu. Representation and Shape Matching of 3-D Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(3):340–351, May 1984.
- [7] I. Biederman. Human Image Understanding: Recent Research and a Theory. *Computer Vision, Graphics, and Image Processing*, 32(1):29–73, 1985.
- [8] T. O. Binford. Visual Perception by Computer. In *IEEE Conference on Systems and Controls*, Miami, Florida, Dec. 1971.
- [9] A. Blake and Cipolla. Robust Estimation of Surface Curvature from Deformation of Apparent Contours. *International Journal of Image and Vision Computing*, 9:107–112, 1991.
- [10] J. D. Boissonnat. Geometric Surfaces for 3-Dimensional Shape Representation. *ACM Transactions on Graphics*, 3(4):266–286, Oct. 1984.
- [11] J. D. Boissonnat. Shape Reconstruction from Planar Cross Sections. *Computer Vision, Graphics, and Image Processing*, 44(1):1–29, Oct. 1988.
- [12] R. C. Bolles and P. Horaud. 3DPO: A Three-Dimensional Part Orientation System. *International Journal of Robotics Research*, 5(3):3–26, 1986.



## 6

# Conclusions and Future Research

In this dissertation, we have identified the major problems in constructing models of real-world 3-D objects from multiple range images, namely the integration and representation of the object surface. We approach these problems by first registering the different views and then combining the view integration with the shape description task. We developed a range image registration algorithm that can find the accurate registration transformation between range image views given coarse estimations. We have shown through examples from real data, that a less constrained least-squares problem leads to a faster convergence for the registration. The resulting registered range image views are then used to fit a dynamic mesh model under an applied inflation force through a physics-based process. In contrast to many deformable models and dynamic mesh models proposed by previous researches, there is no explicit data force involved in this process, which, in conjunction with the mesh subdivision and local mesh adaptation schemes, made our system much less dependent upon a carefully selected initial guess in order to reach the correct result.

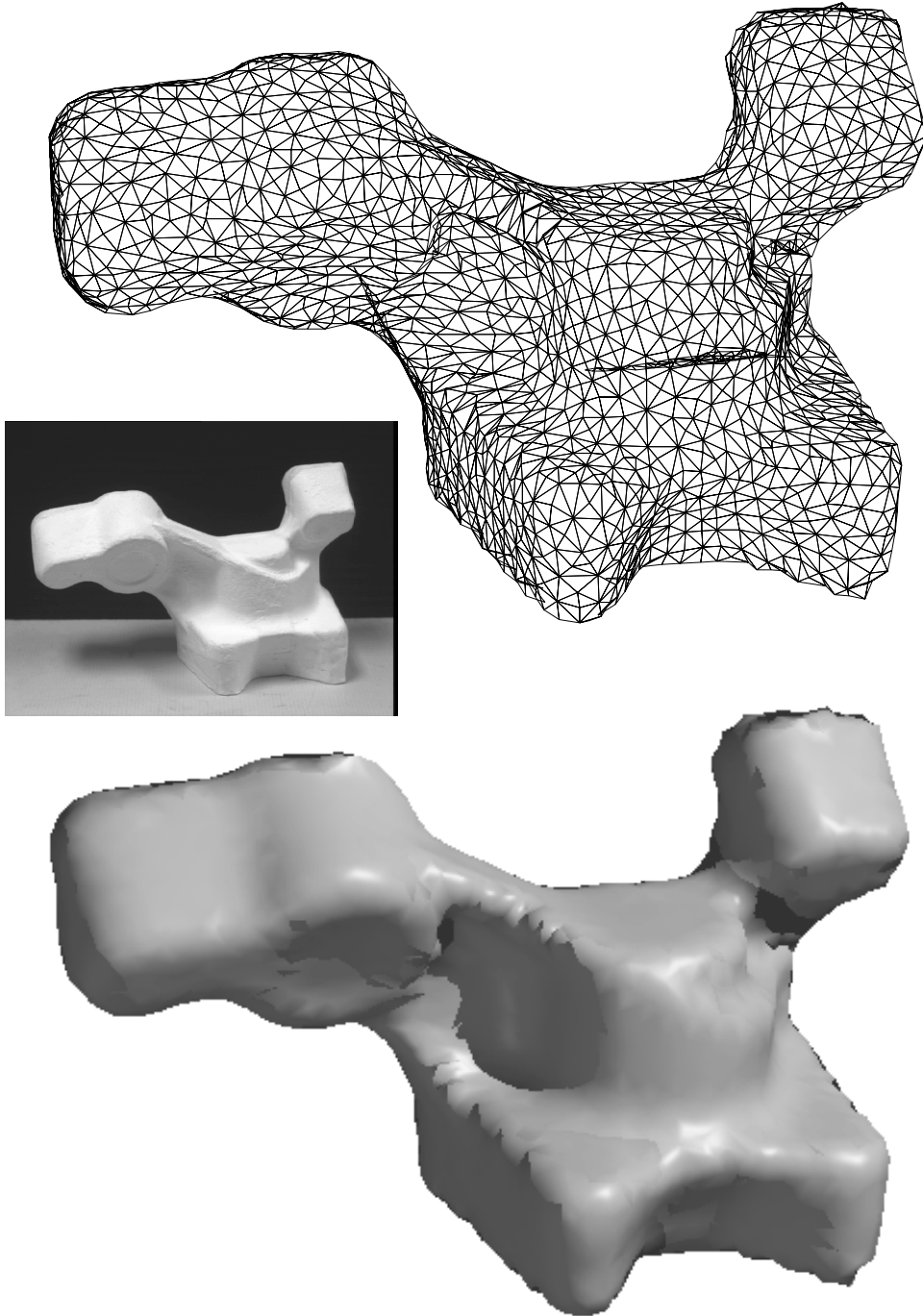
What we have achieved so far is a complete triangulation of the object surface. In order for the resulting model to be useful for visual understanding, recognition and shape learning, a high level description is needed. The first step in achieving such a goal is to construct a smooth shape representation from the triangulation [24]. The next step is to extract certain geometric features, such as the intrinsic curves on the surface, based on the smooth shape representation. These extracted features will help build a high level representation based on constituent parts [7][59].

To extend our current system, one way is to study how to use a different representation from the mesh model that we use so that we can achieve a continuous representation directly [46]. It is also very interesting to investigate how to extend our system to handle objects with holes (genus 1 and up).

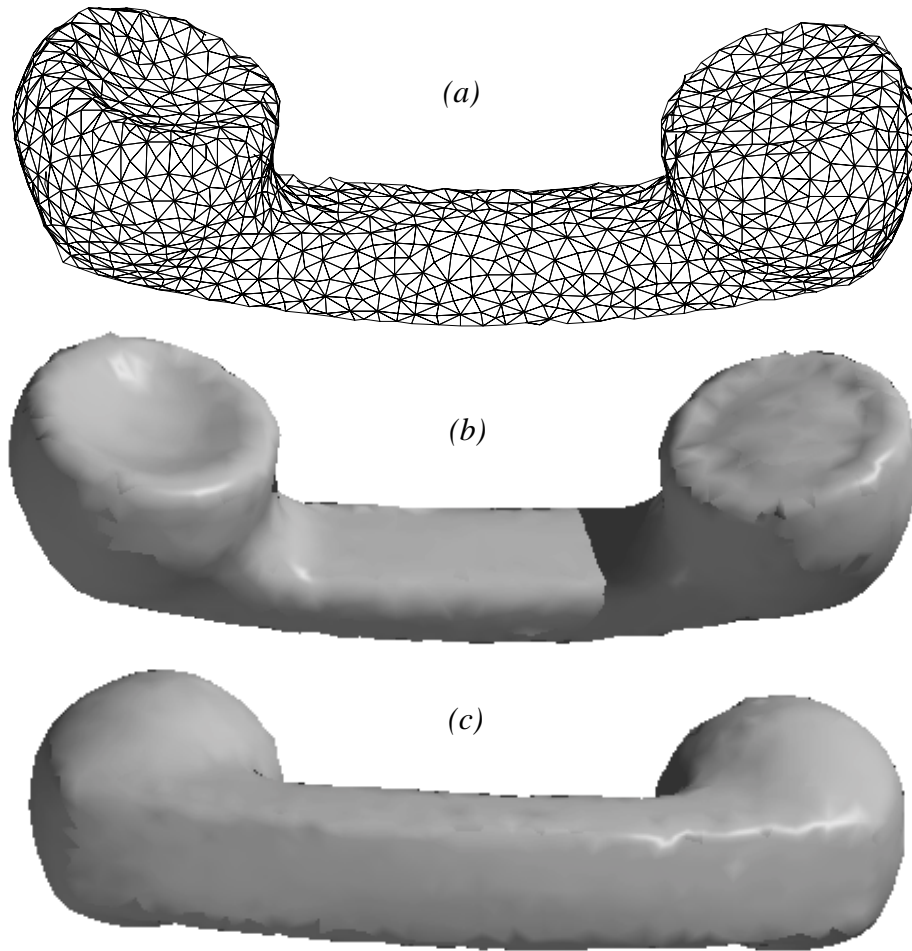


## 5.9 Summary

We have presented a surface description method based on a dynamic balloon model using a triangular mesh with springs attached to the vertices. The balloon model is driven by an applied inflation force towards the object surface from inside of the object, until all the triangles are anchored on the surface. The model is a physically based dynamic model and the implementation of the algorithm is highly parallelizable. Furthermore, our system is not based on global minimization and can allow the model to adapt to local surface shapes based on local measurements. Experiments showed very good results on complex, non-star-shaped object. Some improvements can be made on the resulting model, including using the algorithms presented in Section 4.3 to improve triangle fitting errors, or the method in [68] to merge small triangles into larger ones without affecting the fitting error for constructing a hierarchical representation.



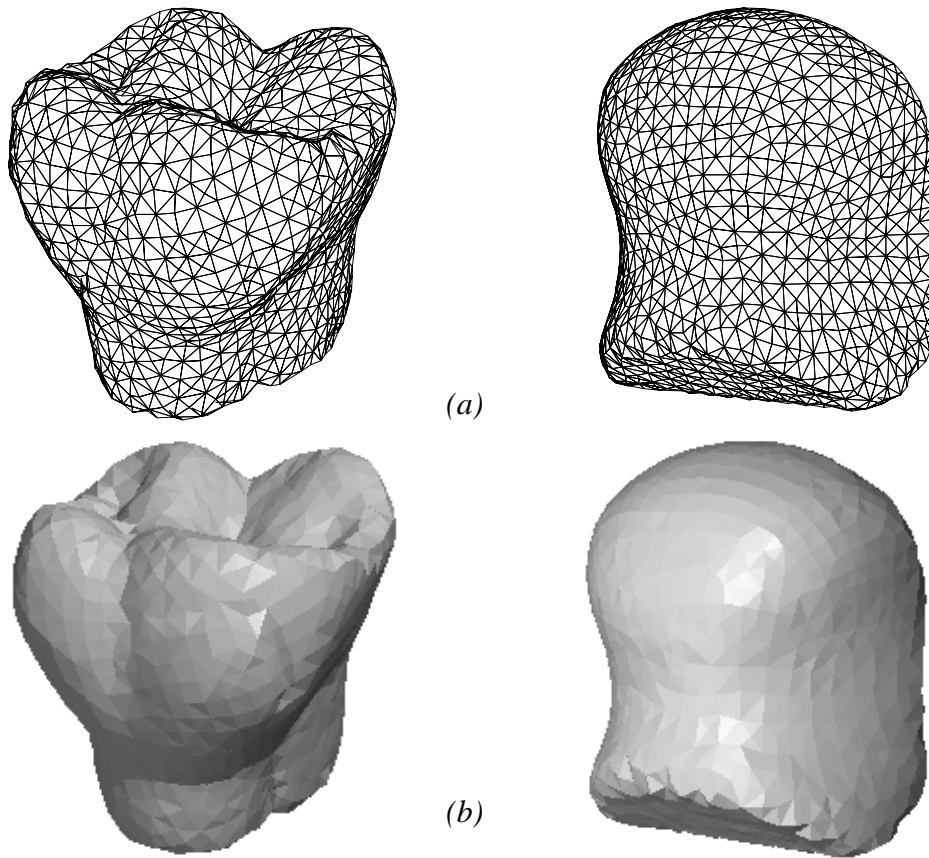
*Figure 5.10 A wireframe and a rendered image of the reconstructed model for the Renault automobile part. The small picture in the middle is the intensity image of the actual object. Note that the wireframe is not produced by a true hidden-line removal algorithm.*



*Figure 5.9 The final balloon model for the Phone: (a) a wireframe, (b) and (c) smoothly shaded images.*

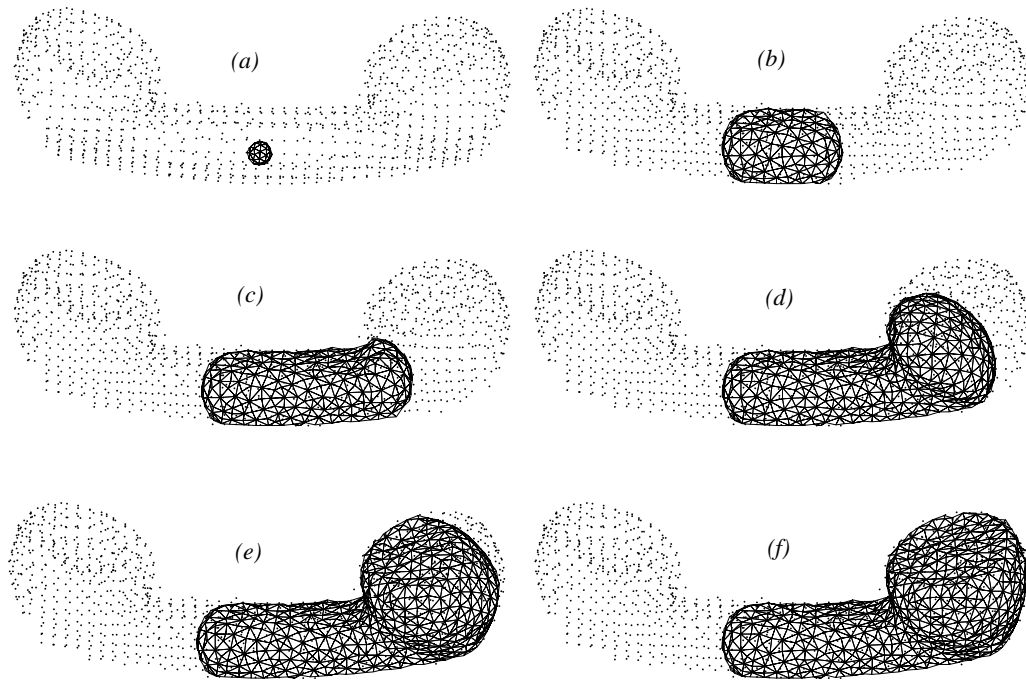
can be solved by examining and identifying local surface changes more closely and adjusting system parameters accordingly in that area.

Comparing the results in this section with the results using the projection-based method in Section 4.4 for the Wood and the Tooth, we can see that, although the results here are better in terms of approximation (0.6 mm vs. 2.0 mm), we do need many more triangles which is necessary for the algorithm itself to maintain the smoothness of the balloon surface. This results in much more computation effort. The tests here only show that the balloon model works on complex as well as simple objects, but it is not necessary for simple objects.



*Figure 5.8 Examples of the balloon model for Wood and Tooth: (a) the wireframes of the obtained balloon models and (b) the rendered shaded images of the models.*

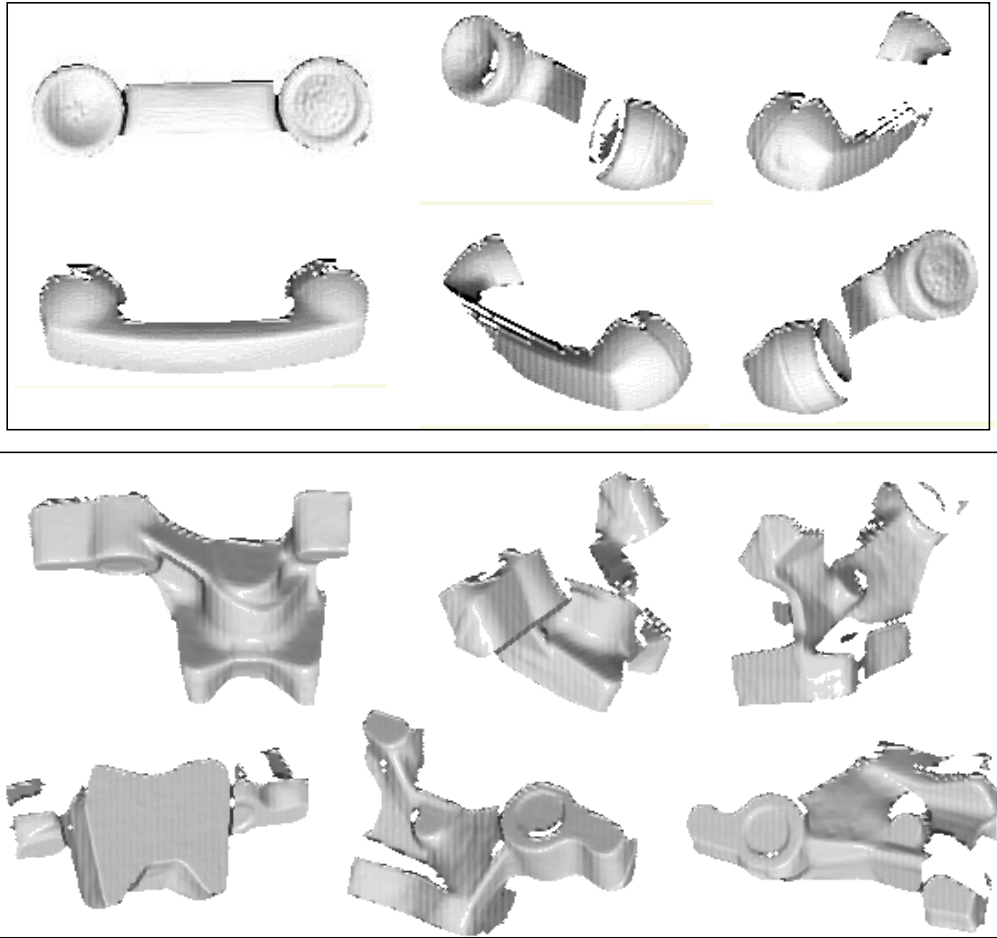
As can be seen from the results presented above, our algorithm works very well on both simple, compact objects and non-star-shaped objects with complex structures. The resulting triangulated model surfaces preserve most of the important geometry feature of the objects with evenly distributed meshes. Our initial meshes are all set in the neighborhood of the center of the objects measuring  $2mm$  and yet our balloons can successfully grow to cover all parts of the object with complex geometric structures such as the Renault part. Also, despite of our effort to acquire many range image views to cover the whole object, the data that we use for the Renault part contain holes on top of both arms, and the resulting mesh was able to interpolate them very well. There is, however, a defect under the right arm of the Renault part, as can be seen in the wireframe drawing. It is a small opening in the mesh that tends to self-intersect which is caused by a small narrow ridge section (about  $8mm$  thick, much smaller than 2 times  $R_t$ , where  $R_t = 10mm$ ). We believe that this



*Figure 5.7 Stages of an inflating balloon inside the Phone, showing the movement of the right front only. The wireframes are superimposed with sample points from the input range images.*

**Table 5.1: Statistics of the test results**

Objects	Views	Triangles	Vertices	Iterations	Time
Wood Blob	16	2864	1434	51	12'24"
Model Tooth	15	2870	1437	60	12'59"
Phone	20	3384	1694		16'17"
Renault	24	5696	2850		32''26'



*Figure 5.6 Sample range images used in constructing the Phone model and Renault model in Figures 5.9 and 5.10, shown here as shaded intensity images.*

Figure 5.10 shows a wireframe and a rendered image of the Renault part, whose real intensity image is shown as an inset. The final model for the Phone has 1694 vertices and 3384 triangles, the Renault part has 2850 vertices and 5696 triangles. Other result statistics can be found in Table 5.1, where the time is measured on a Sun SparcStation-10 running Lucid Common Lisp. The system parameters used can be found in Section 5.6. Both the Phone and the Renault part measure about 200mm across their longest sides. Note that the wireframe drawings presented here are *not* produced using a true hidden-line elimination algorithm, which is the cause of most of the spurious triangles seen in the wireframe drawings.

Another advantage that this computation structure brings us is that we can adaptively adjust system parameters independently for each front based on the information that we gather from the prospective correspondence points of the vertices in the front. For example, if we have detected that the movement of the front is virtually stopped and yet the prospective correspondence points are still certain distance away, this tells us that the preset parameter  $R_f$  in previous section is too large and we should adjust it accordingly.

One example of such adaptation is in handling holes in data. In this case, there exist areas of the object surface that are not covered by any of the input range images, we will not be able to find prospective correspondence points for some of the vertices in the related front. Eventually, when the rest of the vertices in the front have settled down to their correspondence points, we are left with a front for which none of the vertices have a prospective correspondence point. In such situations, the system automatically sets the inflation force to zero ( $k = 0$  in Equation (5.5)), which allows the mesh to reach an equilibrium state that interpolates the surface over the hole.

Another important issue is that of noise. There are two types of noise that may affect our results. One is the noise introduced by the small misalignment among the range images. The other is the outliers from the range images. Our system is very stable with respect to both types of noise. The first one is effectively solved by the weighted sum line-surface intersection algorithm (see Section 3.5) since the misalignment due to registration error causes the actual intersections to form a cluster. The second type of noise usually cause the intersection algorithm on the related range image to fail to converge, in which case it does not contribute to the result of the intersection. Even if the noise does produce a wrong intersection, it can easily be filtered out as an outlier that does not belong to the correct cluster.

## 5.8 Experiment Results and Discussion

Now we present some examples of our balloon model in modeling a model telephone handset (Phone) and an automobile part (Renault) using 20 and 24 range image views respectively, along with examples for two simple object used in Chapter 3. The range images are acquired using the same a Liquid Crystal Range Finder (LCRF) [63], and then registered using the range image registration algorithm described in Chapter 3. Some sample range images used in the experiments are shown in Figure 5.6. Figure 5.7 shows a growing balloon inside the Phone at different stages of the growth. Note that our algorithm works on the fronts in a sequential order, and the figures only show the process of the right front. Figure 5.8 shows two examples of the balloon model in fitting compact objects, the Wood and the Tooth from Figure 3.9 on page 41 and Figure 3.12 on page 44. In Figure 5.9, the final rendered views of the constructed model of the Phone are shown below the wireframe drawing (compare with the real intensity image in Figure 1.1 on page 2).

to  $S_t$ . This also gives us a sample configuration of an ideal front structure when the maximum mesh tension is achieved. Let  $f_{spring-max}$  be the maximum spring force exerted on to a vertex under such conditions. The inflation force needed to overcome the spring force (in order for the vertices to move) is therefore

$$f_{inflate} > f_{spring-max} \quad (5.7)$$

The inflation force is also constrained by Equation (5.3), since once a time step and a maximum displacement per iteration are set, the allowed inflation force should be (considering zero spring force):

$$f_{inflate} \leq \frac{d_{max}}{\Delta t} \quad (5.8)$$

where  $d_{max} = \max(\|\mathbf{x}^{t+\Delta t} - \mathbf{x}^t\|)$  is the maximum displacement. Since a large inflation force tends to dominate the mesh's evolution, which is undesirable, we prefer a smaller one. We choose to use the minimal inflation force as shown in Equation (5.7). We can then compute the needed inflation force amplitude  $k$  according to Equation (5.5).

Now the whole issue comes down to determining  $f_{spring-max}$ ,  $d_{max}$  and the time step  $\Delta t$ . The maximum spring force is determined by the spring natural length  $l_{ij}$  and spring stiffness which are related (Equation (5.4)). In our experiments, we have used  $l_{ij} = 0$  and  $c_{ij} = 4.0$ .  $d_{max}$  and  $\Delta t$  are selected to allow the mesh to evolve smoothly and quickly relative to the object size and complexity. For all the tests in this paper, we have used  $2mm$  and  $0.05$  respectively.

Finally, the user needs to select  $\delta$  and  $R_t$ . For the purpose of simplicity, in our experiments, however, we manually set  $R_t$  and allow a fixed number of  $N$  triangles to fit the sphere with a radius  $R_t$ , which gives a nominal approximation error of about  $0.6mm$  with  $N = 80$  and  $R_t = 10mm$ . These parameters also apply to all the experiments in this chapter.

## 5.7 Adaptive Local Fitting, Holes and Noise

It is worth mentioning that our algorithm presented in Section 5.5 is parallelizable since the computations on each front in the queue  $Q$  are independent of each other. Furthermore, the computation for each vertex within a front is also independent at each iteration.

### Algorithm 5.3

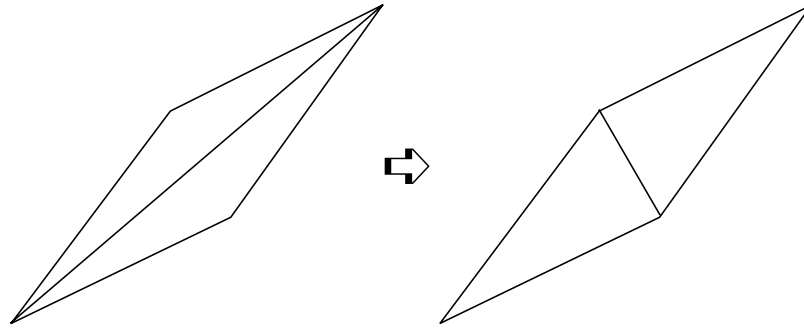
Let all the triangles on the initial mesh be front  $F_0$  and push it into the front queue  $Q$ , then until queue is empty, do the followings repeatedly:

- 1)  $F \leftarrow$  top of the queue  $Q$ , pop the queue.
- 2) Subdivide the triangles in  $F$  if appropriate (see next section.)
- 3) For each vertex  $V_i \in F$ , whose 3-D coordinates at time  $t$  is  $\mathbf{v}_i^t$ , do
  - a) compute the internal force  $\mathbf{g}_i$  and external force  $\mathbf{f}_i = \mathbf{h}_i$  based on Equations (5.4) and (5.5).
  - b) compute the new vertex location  $\mathbf{v}_i^{t+\Delta t}$  for the current iteration according to Equation (5.3).
  - c) compute prospective correspondence point of  $\mathbf{v}_i$ , which is the intersection  $\mathbf{w}_i$  of the surface and the ray through  $\mathbf{v}_i$  and in the direction of the mesh normal at  $\mathbf{v}_i$ .
  - d) if  $\|\mathbf{v}_i^{t+\Delta t} - \mathbf{v}_i^t\| > \|\mathbf{w}_i - \mathbf{v}_i^t\|$ , then  $\mathbf{v}_i^{t+\Delta t} \leftarrow \mathbf{w}_i$  and mark vertex  $V_i$  anchored.
- 4) For each  $V_i \in F$ , update its position with the corresponding new positions  $\mathbf{v}_i^{t+\Delta t}$ .
- 5) Discard triangles from  $F$  that have thus become anchored.
- 6) if  $F = \{\emptyset\}$  then go to 1.
- 7) recompute connected triangle regions in  $F$  and push them into  $Q$ . Go to 1.

The intersection point  $\mathbf{w}_i$  in step (3)(c) is the closest intersection to the mesh surface. If there exist multiple intersections for the ray with many range image views, the weighted average (Section 3.5, Equation (3.15)) of the closest cluster of the intersections is used.

## 5.6 Setting Up the Parameters

In our current implementation, triangles that have areas larger than a threshold  $S_t$  are subdivided at each iteration, since the mesh elements can not be too large for surfaces with certain maximum curvature.  $S_t$  is directly related to the precision of the fit of the final mesh to the input surface data. Assuming that our goal is to approximate the object surface to have a triangle fitting error  $\delta$  for surfaces with maximum curvature of  $1/R_t$ ,  $S_t$  can be easily computed by tessellating a unit sphere of radius  $R_t$  with equilateral (or near equilateral) triangles of sizes smaller or equal



*Figure 5.5 Rearranging the local configuration to eliminate long and thin triangles.*

### **5.4.2 Local Mesh Adjustment**

The above algorithm works very well under most circumstances, but degenerate triangles that are long and thin may still occur. These triangles are undesirable since they do not represent local surface shape well and are often the cause of self-intersections of the mesh surface. Currently, we use a simple algorithm that checks for pairs of such triangles and rearrange the triangle configuration locally. After each subdivision, we check for triangles that are thin and long, and if two such triangles share an edge that is the longest for both triangles, then we simply switch the cross edge as shown in Figure 5.5.

## **5.5 Description of the Algorithm**

In this section we give a brief description of the entire algorithm of our approach. A discussion on how to set the system parameters will follow. We assume that registered range image views of the object to be modeled are available, although we believe the algorithm can be adapted to other types of 3-D input.

We start with selecting an initial point inside the object and constructing an icosahedron shell (Section 4.2.1) at this location. The selection process is currently done by hand and the size of the shell should be small enough so that it is completely inside the object. Since the algorithm does not depend on the actual location of the initial shell, as long as it is inside the object, an alternative to manually selecting the initial position is to choose a smooth patch in any range image and place a very small initial shell under the surface patch. The system algorithm can be described as follows.

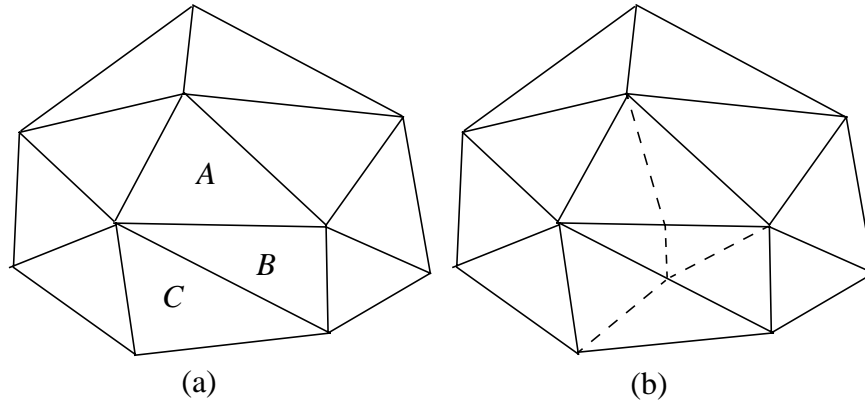


Figure 5.4 Subdivision of triangle mesh: (a) before  $A$  is subdivided, (b) after  $A$  is subdivided and subdivision is propagated to both  $B$  and  $C$ .

- 5) If  $R_{k+1} = \{0\}$ , stop, the subdivision is done. Else, set  $k \leftarrow k + 1$  and go to step 3.

This subdivision algorithm has the feature that the subdivision will only be propagated towards large triangles from the longest edge of a subdivided triangle. It is also proven that the resulting triangles' smallest inner angle is lower-bounded by half of the smallest inner angle of the original triangles [58].

This algorithm, however, does not guarantee that the triangles on the boundary areas of a front conform with the rest of the triangles in the triangulation. Hence, after the algorithm is finished, we must bisect the affected non-conforming triangles accordingly. Thus we have:

### Algorithm 5.2

- 1) Carry out Algorithm 5.1 on the set of triangles  $\tau_0 \subset \tau_{f_0}$ .
- 2) For each non-conforming triangle  $T \notin \tau_{f_0}$  but connected to  $\tau_{f_0}$ , bisect  $T$  by its non-conforming edge.

An example of this algorithm is shown in Figure 5.4, where triangle  $A$  is to be subdivided and  $C$  does not belong to the region (front). As can be seen from the figure, the subdivision is propagated to  $B$  and finally  $C$  is bisected to make the triangles at the region boundary conforming (step (2) above).

surface to adapt to the local object surface geometry without affecting other parts of the mesh surface.

Before introducing the details of the triangle subdivision, we first define some terms. A vertex is said to be *anchored* if it has reached the object surface and has been marked as such. A triangle is said to be anchored if all of its vertices are anchored. At any time in the mesh growing process, the triangles in the mesh can be classified into anchored triangle regions, consisting of anchored triangles, and unanchored triangle regions, consisting of movable triangles called *front*. Each front is a connected component of triangles, in which two triangles are said to be connected if and only if they share an edge.

### 5.4.1 Adaptive Triangle Mesh Subdivision

Triangle subdivision is carried out only on the front, since anchored triangles are not allowed to move. This allows the triangular mesh to adapt to the object surface better without globally adjusting the position of all vertices. A good subdivision scheme would be one that yields an evenly distributed mesh and produces few degenerate (i.e. long and thin) triangles. The algorithm that we use in this paper first selects a set of triangles that needs to be subdivided through bisection. Then after these triangles are bisected on their longest edges, adjacent triangles are also bisected or trisected to make the triangles *conforming*, the state that a pair of neighboring triangles either meet at the a vertex or share an entire edge. In our implementation, only those triangles that exceed certain size limit are subdivided first. The algorithm presented below is adapted from Algorithm 2 (local) in [58], which is developed for refining triangular meshes for finite element analysis.

#### Algorithm 5.1

Let  $\tau_{f_0}$  be the set of the triangles from a given front, and  $\tau_0 \subset \tau_{f_0}$  is the selected set of triangles to be subdivided.

- 1) Bisect  $T$  by its longest edge, for each  $T \in \tau_0$ .
- 2) Find  $R_1 \subset \tau_0$  the set of nonconforming triangles generated in step 1. Set  $k \leftarrow 1$ .
- 3) For each  $T \in R_k$  with non-conforming point  $P \in T$  (mid-point on the non-conforming edge): (a) bisect  $T$  by the longest edge; (b) if  $P$  is not on the longest edge of the  $T$ , then join  $P$  with the midpoint of the longest edge.
- 4) Let  $\tau_0^k$  be the triangulation generated in step 3. Find  $R_{k+1} \subset \tau_0^k$  the set of nonconforming triangles generated in step 3.

where  $k$  is the amplitude of the force and  $\hat{\mathbf{n}}_i$  is the direction normal to the local model surface. In our implementation, the normal at a triangle vertex is estimated from the vector sum of the normal vectors of the surrounding triangles:

$$\hat{\mathbf{n}}_i = \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|}, \quad \mathbf{n}_i = \sum \frac{(\mathbf{n}_{ij} + \mathbf{n}'_{ij})}{\|(\mathbf{n}_{ij} + \mathbf{n}'_{ij})\|}. \quad (5.6)$$

where  $\mathbf{n}_{ij}$  is the direction normal to the  $j$ th triangle  $T_j \in \{T_j\}$  surrounding the vertex, and  $\mathbf{n}'_{ij}$  is the direction normal to triangle  $T'_j$  that is the neighbor to  $T_j$  but  $T'_j \notin \{T_j\}$ . This estimation is more stable than what we get when only the triangles in  $\{T_j\}$  are used.

## 5.4 Subdivision and Adaptation of the Triangular Mesh

In our simulated physical system, during the process of the growth of the mesh model, the mesh triangles become bigger in size, and tensions due to the spring force also build up, which will eventually stop the movement of the mesh, and the inflation force and the spring tension enter an equilibrium state. This is not desirable since we do not consider forces from the input data, so an equilibrium state does not mean a good fit. In order to keep the balloon growing, we can keep the inflation force unchanged (which actually means to keep on inflating) and at the same time reduce the spring tension by subdividing the triangles in the mesh into smaller triangles. Alternatively, we can increase the inflation force and allow the spring tension to increase. But increasing the inflation force also has the side effect of increasing the maximum displacement. As can be seen from Equation (5.3), the spring force  $\mathbf{g}_i$  acts as a balance force to the inflation force  $\mathbf{f}_i = \mathbf{h}_i$  (assuming a convex local structure), thus the maximum displacement is directly related to the inflation force once a time step is chosen. So we choose not to increase the inflation force, but to subdivide the mesh instead.

The purpose of subdividing triangles is twofold. Once a triangle is subdivided, the sides of the triangles become shorter and if we keep the natural length and stiffness of the springs constant, the spring tension is reduced. Also, subdividing the triangles helps maintain an evenly distributed mesh. Subdividing triangles in certain region, as will be discussed later, also allows the mesh

Because of the nonlinear nature of the forces  $\mathbf{g}_i$  and  $\mathbf{f}_i$  involved, the systems of ordinary differential equations (5.1) can be solved using explicit numerical integration [73].

A dynamic system described by Equation (5.1) will reach an equilibrium state when both  $\dot{\mathbf{x}}_i$  and  $\ddot{\mathbf{x}}_i$  become 0, which can take a very long time since it is usually an exponential process. A simplified system can be obtained if we make  $m_i = 0$ , and  $r_i = 1$  for all  $i$ , in which case Equation (5.1) reduces to

$$\dot{\mathbf{x}} = \mathbf{f}_i - \mathbf{g}_i, \quad i = 1 \dots N \quad (5.2)$$

The reason for making this simplification is that our goal for the dynamic system is for the elements in the mesh to reach the object surface and we will stop them without considering whether the system is in an equilibrium state or not. We also do not intend to have a special treatment for any elements in the mesh at this time, so all  $r_i$  should be equal, in which case we can normalize the parameters so that  $r_i = 1$ . The set of first-order differential equations in Equation (5.2) has a very simple explicit integration form as follows:

$$\mathbf{x}^{t+\Delta t} = (\mathbf{f}_i^t - \mathbf{g}_i^t)\Delta t + \mathbf{x}^t \quad (5.3)$$

### 5.3.3 Spring Force and Inflation Force

The spring force exerted on vertex  $i$  by the spring linking vertex  $i$  and  $j$  can be expressed as [73]:

$$\mathbf{s}_{ij} = \frac{c_{ij}e_{ij}}{\|\mathbf{r}_{ij}\|}\mathbf{r}_{ij} \quad (5.4)$$

where  $c_{ij}$  is the stiffness of the spring,  $e_{ij} = \|\mathbf{r}_{ij}\| - l_{ij}$  is the spring deformation,  $\mathbf{r}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ ,  $\|\ \|\$  is the vector length operator and  $l_{ij}$  is the natural length of the spring. The total spring force  $\mathbf{g}_i$  a vertex receives is the vector sum of spring forces from all springs attached to it. The inflation force a vertex receives takes the form of:

$$\mathbf{h}_i = k\hat{\mathbf{n}}_i \quad (5.5)$$

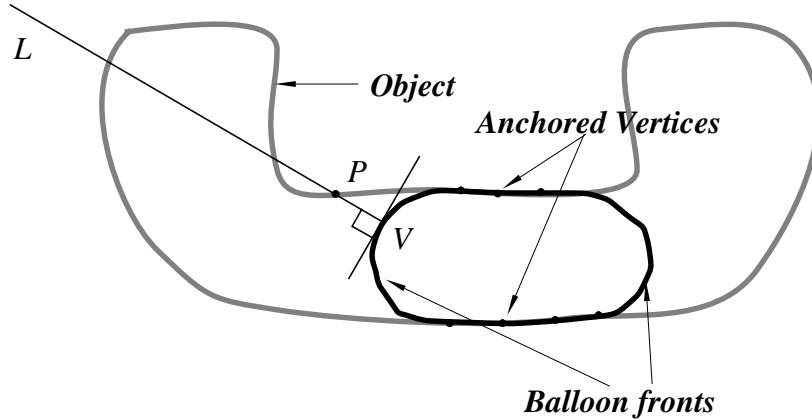


Figure 5.3 Line-surface intersection: searching for a corresponding point in the normal direction.

neighborhood of a mesh element is only in the direction normal to the mesh surface. So it is only natural to look for corresponding point from the object surface in the direction of the normal, which will also change in the process of inflation. In our implementation, this is done by computing the prospective corresponding point  $P$  (Figure 5.3), the closest intersection between a ray in the normal direction and the object surface represented by range images. Details of the algorithm can be found in Section 3.3.4. Once the intersection is found, the distance from the mesh surface to the intersection can be used as a measure of whether the mesh has reached the object surface.

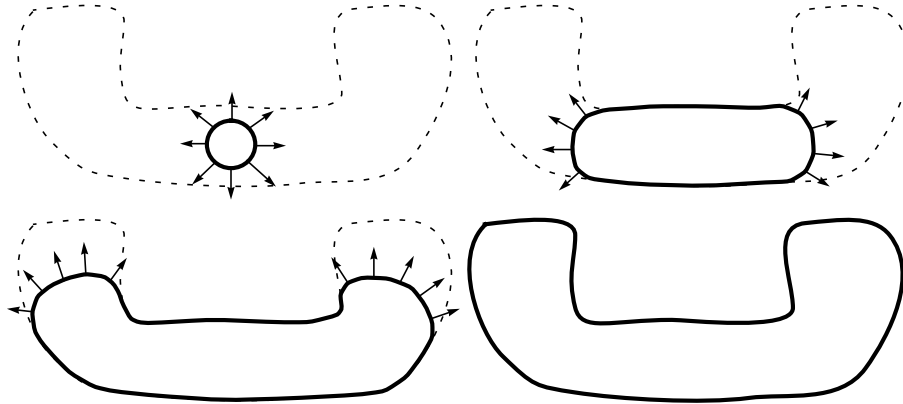
When there are holes in the input data (parts of the object surface not covered by the input data), we will not be able to find the intersections as described above. In such cases, there are no prospective correspondence points for the affected vertices and thus there is no reason to apply inflation force further more. This kind of decision, however, should not be made locally for each vertex point. We will talk more about handling holes later in Section 5.7.

### 5.3.2 A Simplified Dynamic Model

The motion of any element  $i$  on the model surface can be described by the following motion equation [73]:

$$m_i \ddot{\mathbf{x}}_i + r_i \dot{\mathbf{x}}_i + \mathbf{g}_i = \mathbf{f}_i, \quad i = 1 \dots N \quad (5.1)$$

where  $\mathbf{x}_i$  is the location of the element,  $\dot{\mathbf{x}}_i$  and  $\ddot{\mathbf{x}}_i$  are the first and second derivatives with respect to time,  $m_i$  represents the mass,  $r_i$  is the damping coefficient,  $\mathbf{g}_i$  is the sum of internal forces from neighboring elements due to spring attachments and  $\mathbf{f}_i$  is the external force exerted on the element.



*Figure 5.2 The inflating balloon model as illustrated in a 2-D case (from left to right and from top to bottom): the initial state, intermediate states and the final state.*

the surface of the object. This seems to be bad from the viewpoint of a fitting process, which tries to minimize some distance measure between the object surface and the model. But this is important in our case because our main concern is to find a correct mapping between the mesh elements and the object surface. By not using any attraction force from the surface data it allows us to avoid incorrect mappings (which is a similar situation to the local minima in energy minimization approaches) without depending on a very close initial guess.

### **5.3.1 The Correspondence Problem**

So far we have not discussed how to test whether the mesh has reached the surface of the object. This is the key difference between our approach and other dynamic model systems or energy minimization systems. In order to test whether a vertex has reached the object surface, we must measure the distance between the mesh surface and some point on the object surface. Ideally this point on the object surface should be the corresponding point of the vertex, which is not possible before the vertex reaches the surface. Previous researchers have used the closest point on the object surface to a mesh element as an alternative, but it may provide incorrect information. In [46], the distance from the data points on the surface to the nearest model point is used instead, which is an improvement over the above approach. This approach, however, is not practical when there is a large number of surface sample points from the object, as in our case.

In our approach, we look for potential corresponding points only in the direction normal to the mesh surface. This is the best knowledge locally available to the points on the mesh surface at any time during the mesh's growing process, because the equivalent mesh surface movement in the

In the following sections, we first review some of the related previous work and then present our balloon model in detail. Section 5.3 describes the model and how it works, and Section 5.3 defines the dynamics of the system. Section 5.4 explains the adaptive mesh subdivision scheme. In Section 5.5, we give an algorithmic description of our system. Sections 5.6 and 5.7 discuss how the system parameters are set and how to adapt the parameters locally. We present experimental results in Section 5.8 and a summary in Section 5.9.

## 5.2 Related Work

Dynamic mesh models have been proposed by previous researchers for shape description. Terzopoulos and Vasilescu [73] introduced a shape reconstruction algorithm using a dynamic mesh that can dynamically adjust its parameters to adapt to the input data. They also extended this approach by introducing an attraction force from 3-D inputs for shape description [78]. Delingette *et al.* [21] proposed a deformable model with internal smoothness energy and external forces from both the input data and features. There are other deformable model approaches that differ in representation schemes of the model and in the approaches to solving the system (see [20] and [46]).

The main difference between our method and those used by previous researchers is that we do not explicitly introduce a data force into our model, as will be discussed in detail in the following sections. Our model is purely driven by an inflation force introduced inside the balloon. Balloon models have been used by Cohen and Cohen [20] and McInerney and Terzopoulos [46], but in their approaches, the introduced balloon force is used to assist the global energy minimization model to converge to the desired results and to overcome some noise in the volume data, and a careful balance must be maintained between the balloon force and other forces.

## 5.3 The Inflating Balloon Model

Our balloon model is represented by a mesh of triangular patches with vertices linked to their neighbors by virtual springs. The initial triangulated shell is generated from an icosahedron (see Section 4.2.1). When placed inside the object and under the applied inflation force to the vertices, the shell grows in size as the vertices move in the directions normal to the mesh surface, maintaining a spherical shape, until one or more vertices reaches the object surface. During the process of inflation, the triangles may be subdivided adaptively, which also creates new vertices. Once it reaches the surface, a vertex is considered fixed to the surface and thus can no longer move freely. The remaining vertices will keep moving until reaching the surface (Figure 5.2). As can be seen from this process, the movement of the vertices is not influenced directly by any force from

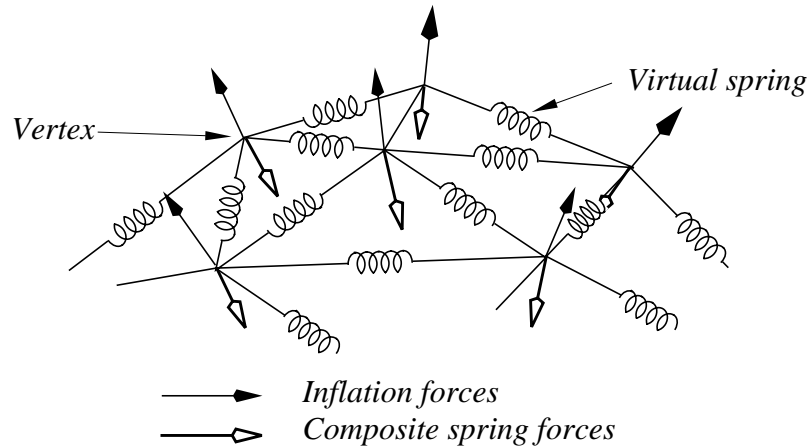


Figure 5.1 Dynamic surface model illustrated.

drives the vertices towards the object surface until they “land” on it. This process is similar to that of blowing up a balloon placed inside a hollow object until it fits the shell of the object. Thus the goal of mapping the model to the object surface is achieved through the physics of a growing balloon in a very natural way.

Our system is not based on global minimization methods, and it can make decisions based on local information about the shape of the object surface. As a result, our system is much less dependent upon the initial state that we selected. As the mesh expands and the vertices start to reach the object surface, the entire mesh surface will gradually be subdivided into pieces of connected triangular regions. This allows us to treat the surfaces based on local surface properties by tailoring the parameters, and possibly strategy, of the system in dealing with each region separately. One such example is that it allows us to handle the case when there are holes in the input data.

Another aspect of our system is its role in data integration. Most of the previous research make use of *scattered* 3-D points or a *single range image* as input. Very few (e.g. [68] and see also Section 2.2) try to use *multiple* densely sampled range images. As has been discussed in Chapter 1, integration can not be done without an appropriate representation. For non-star-shaped objects, it is more difficult since the surface points of an object can not be simply mapped onto, e.g., a unit sphere, which would be sufficient for star-shaped objects (see Chapter 3). If we simply gather all the range images and treat the data as a loud of scattered 3-D point, as some previous researchers do, we throw away useful surface neighbor information that is present in the range images. Therefore we choose to use triangulation as representation, since we believe that triangulation is a proper level for integration purpose.

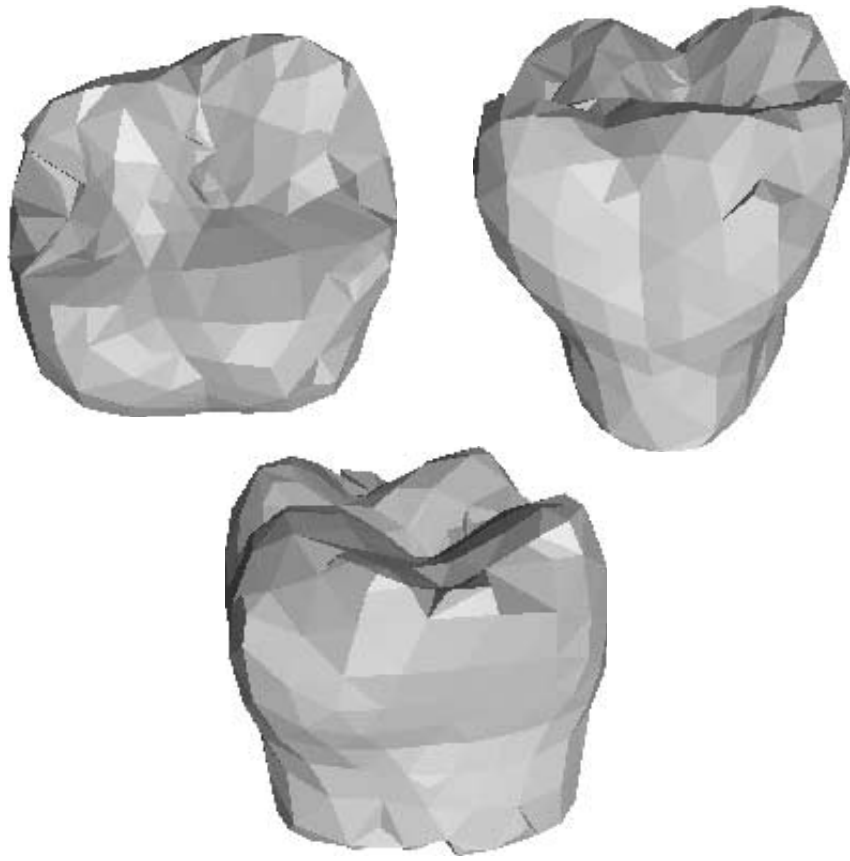
# 5

## Modeling Complex Objects

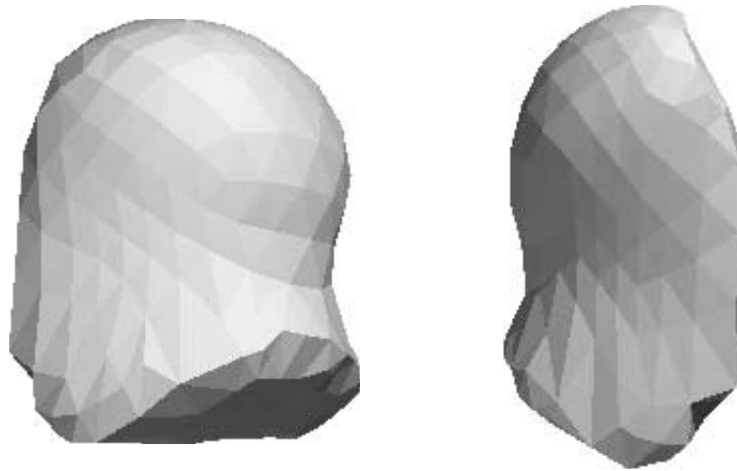
### 5.1 Introduction

Surface description using 3-D input can be described as fitting of a chosen representation (parameterized model surface, or simply model) to the input data points. This process can be formalized in a number of ways involving the minimization of a system functional that represents the fit of the model to the input data explicitly or implicitly. Another very important aspect of such a system is to construct a *mapping* or *correspondence* between the surface of an object and the structure of the model. This mapping exists because the surface of the model and the surface of the object are topologically equivalent, considering genus zero type of objects. Therefore there exists a one-to-one mapping between the model structures and the object surface elements. Previous researchers have studied such mappings in a variety ways using different representation schemes and model fitting methods. Examples of these approaches include the dynamic system using energy minimization in [21] and the dynamic mesh in [73] and [78]. The drawback of these approaches is that they must rely on an initial guess of the model structure that is relatively close to the shape of the object. The reason is that, in the absence of mapping or correspondence information, some other approximations have to be used to associate the points in the model and those in the input so that the model will converge to the desired result under the attraction force between the associated correspondence points. An example of such an approach is the nearest data point to the model surface used in [78]. Such an approximation will have problems in cases where there are data points that are not the closest to their true corresponding points of the model, and thus inevitably lead the system towards a suboptimal situation (local minimum). That is why those approaches can only deal with mainly star-shaped objects.

In this chapter we present a new approach for surface description using a dynamic balloon model represented by a triangular mesh. We start with a small triangulated shell placed inside the object and apply a uniform inflation force on all vertices in the direction normal to the shell's surface. The vertices are also linked to their neighboring vertices through virtual springs to simulate the surface tension to keep the shell smooth (Figure 5.1). The applied inflation force



*Figure 4.8 Three rendered views of the constructed triangulation of the object “Tooth”.*



*Figure 4.7 Two rendered views of the constructed triangulation of the object “Wood”.*

seconds yielding 872 triangles. Note that the original range images have a resolution of about 1 to 2 *mm* with a comparable registration error.

## **4.5 Summary**

In this chapter, we have presented a new method for constructing an integrated surface description from multiple range images based on surface triangulation. This method combines the surface description and data integration through an effective evaluation of triangle approximation error using an integrated error measure. Projecting a triangulated shell onto the surface of the object is simple and very efficient in acquiring triangulated surface models compared to systems that employ global optimization techniques (see [67], [74], [21] and [72]), because the initial projection is only done once and the subsequent iterations are only carried out at locations that have not satisfied the approximation error. This technique, however, still has the limitation that it can only handle compact objects (the object needs to be star-shaped), though the dynamic subdivision of triangles and reprojection of the triangles alleviated the situation to some extent.

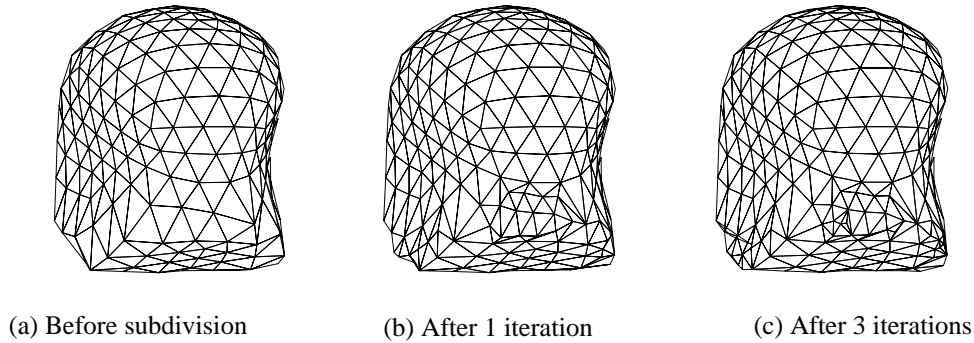


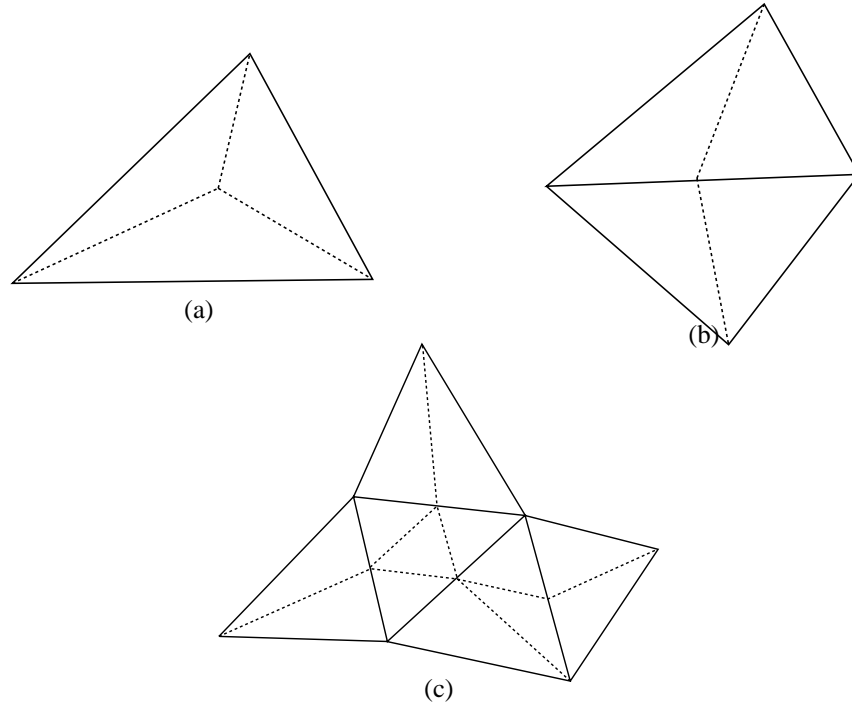
Figure 4.6 *Iterative triangle subdivision*

To improve the fit of the triangles to the object surface, the newly created triangles must be re-mapped onto the object surface. This is done by projecting the triangle vertices in the direction of the surface normal vectors of the object in that location, which can be approximated by the average normal vector of the surrounding triangles of the vertex, weighted by the areas of those triangles. The difference between this projection and the initial projection lies in the direction in which the projection is done.

This division-and-projection process continues until all the triangles have an approximation error within the desired threshold.

#### 4.4 Experimental Results

We now show some test results of the triangle subdivision and approximation. For the tests, the range images from the wood blob (Figure 3.9 on page 41) and the model tooth (Figure 3.12 on page 44) were used. Figure 4.6 shows the wire frames of the triangle subdivision process. Figure 4.6 (a) shows the triangulated shell projected onto the object surface. Figure 4.6 (b) shows the shell after one subdivision iteration with approximation error threshold set to  $2mm$  (see Section 4.3 and Equation (4.5)). Figure 4.6 (c) is the final result after three iterations with all triangles having approximation errors under  $2mm$ . As can be expected, most of the subdivisions take place around the ridge line in the lower part of the object. Figure 4.7 and Figure 4.8 show some rendered intensity images of the “Wood” and the “Tooth” using the derived triangulation models from various view points. In both cases, the initial icosahedrons are subdivided with  $N=5$  and the center of the fitting ellipsoidal are used as the center of the projection. Our test programs are written in Lisp. On a Sun SparcStation 2, with Sun Common Lisp, the test on the “Wood” completed in 3 minutes and 18 seconds, resulting a total of 649 triangles, while the “Tooth” took 3 minutes and 34



*Figure 4.5 Triangles are subdivided in groups with the same type of approximation errors. The dashed lines define the new triangles.*

### 4.3.2 Triangle Subdivision and Reprojection

Once the approximation errors for all projected triangles are evaluated, those triangles whose approximation error  $E$  does not meet the predefined approximation threshold must be subdivided and re-mapped. Our subdivision algorithm tries to subdivide triangles without altering the neighborhood adjacency property of the triangulation, i.e., each triangle must have 3 neighbors that each shares an edge with it. This regularity makes it easy for higher order surface reconstruction in the future. For this reason, we group triangles with the same type of approximation error (i.e., triangles with  $\bar{e}$  of the same sign and their  $|\bar{e}|$  exceeding a certain given threshold) into connected components. Here, two triangles are connected when they share an edge. The subdivision method is illustrated in Figure 4.5, categorized by the size of the connected components. More sophisticated subdivision schemes can also be used and will be discussed in Chapter 5. A simple rule here is that, except for the components that contain only one triangle, triangles with one neighboring triangle in the connected component will be divided into two, those with two neighbors three, and those with three neighbors four.

Let  $Q_i = \{q_{ij}\}$  be the set of 3-D points in  $V_i$  covered by  $T'_i$ , and  $e_{ij}$  be the signed Euclidean distance of  $q_{ij}$  to the plane containing  $T$ . We can compute the mean and standard deviation of  $e_{ij}$  as follows:

$$\sigma_i = \sqrt{\frac{\sum_{q_{ij} \in Q_i} (e_{ij} - \bar{e}_i)^2}{n_i}}, \quad \bar{e}_i = \frac{\sum_{q_{ij} \in Q_i} e_{ij}}{n_i} \quad (4.1)$$

where  $n_i = \|Q_i\|$  is the cardinality of the set  $Q_i$ . Then we adopt a majority vote scheme to effectively eliminate the bad views from being considered in the final approximation error estimation for that triangle. That is, we compute a subset  $G \subseteq \{Q_i, i= 1, 2 \dots m\}$ , such that the absolute value of the difference of the mean approximation error  $\bar{e}_i$  between any pair in  $G$  is below certain threshold:

$$G = \{Q_i \mid \forall Q_k \in G, |\bar{e}_i - \bar{e}_k| < \varepsilon\} \quad (4.2)$$

and that the cardinality of  $G$  is at least half of the original set:

$$\|G\| \geq 0.5 \left\| \{Q_i\} \right\| \quad (4.3)$$

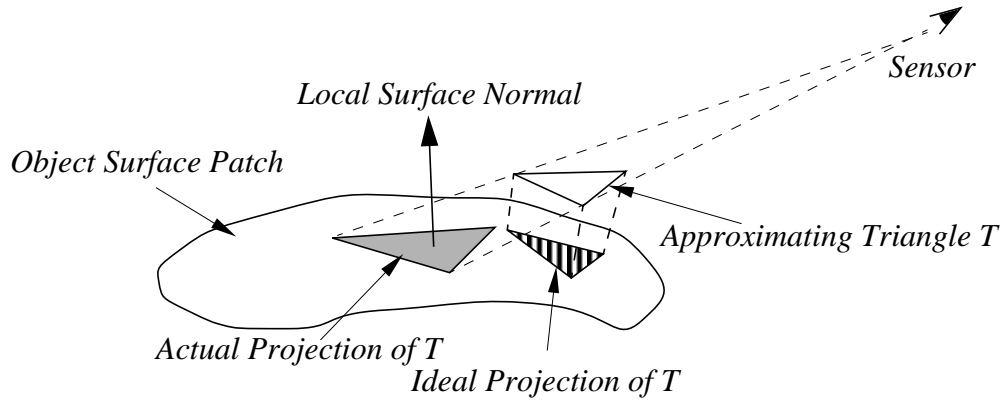
The value of  $\varepsilon$  can be set based on the range image resolution and the statistics of the registration errors from the registration process (see Chapter 3). Once  $G$  is selected, we recompute the mean and standard deviation of the approximation error for all the points in  $G$  for triangle  $T$ . Let the results be  $\bar{e}$  and  $\sigma$  respectively. In the case when  $G$  is an empty set, or  $G$  is a minority group, then we simply select the  $Q_i$  that has the least absolute  $\bar{e}_i$ :

$$G = \{Q_i \mid |\bar{e}_i| = \min_{Q_k \in \{Q_j\}} (|\bar{e}_k|) \} \quad (4.4)$$

The overall approximation measure is defined as

$$E = \bar{e} + \text{sign}(\bar{e})\alpha\sigma \quad (4.5)$$

where  $\text{sign}()$  returns 1 or -1 depending on whether its argument is larger or smaller than 0.  $\alpha$  is a constant that reflects the noise present in the data. The approximation measure  $E$  in the equation above is used to simulate the maximum approximation error for a triangle. We have found  $\alpha=2$  to be a good choice, which can eliminate most of the outliers in the data.



*Figure 4.4 The problem in defining data points in evaluating triangle approximation error. The data points in the shaded area are not appropriate because of the oblique viewing angle of the sensor relative to the surface patch. The hashed triangle is preferred but cannot be obtained.*

the local surface is at an oblique angle, however, the data points so collected may no longer be meaningful to evaluating the approximation error, as shown in Figure 4.4. The ideal projection should be one with projection direction roughly normal to the local surface, but it is difficult to compute. To collect data points that are meaningfully associated with a triangle, we further project the points collected based on the method above onto the plane of the triangle orthographically. If the projection of a point still stays inside of the triangle then it is considered in the approximation evaluation. We call this set of 3-D points from all views for a triangle  $T$  the covered points of the triangle.

The second problem in the approximation error evaluation is to define a measure for the approximation error. In triangulating single view range images, the maximum Euclidean distance of the covered points to the plane containing the triangle can be used as such a measure [64]. But this will not work when there are multiple views. The reason is that although the range image views are well registered in general (see Chapter 3), some range images or a section of them might deviate from the rest. There can also be regions that contain contaminated data. A simple maximum distance measure does not truly reflect the status of the approximation in such situations. Therefore we have adopted a statistical approach which allows us to eliminate the bad data points and bad range image views for each approximating triangle.

Let  $T'_i$  be the projected 2-D triangle of  $T$  for range image view  $V_i$ ,  $i = 1, 2, \dots, m$ , where  $m$  is the total number of range images that have at least one vertex of  $T$  projected in them.

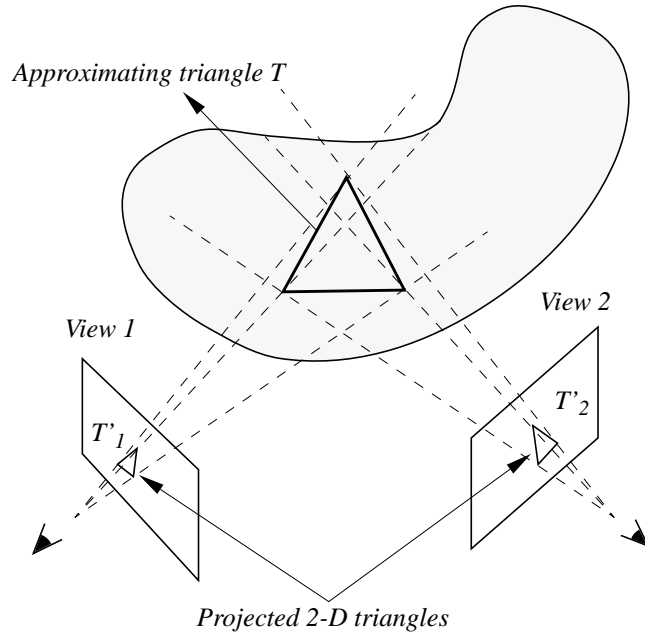


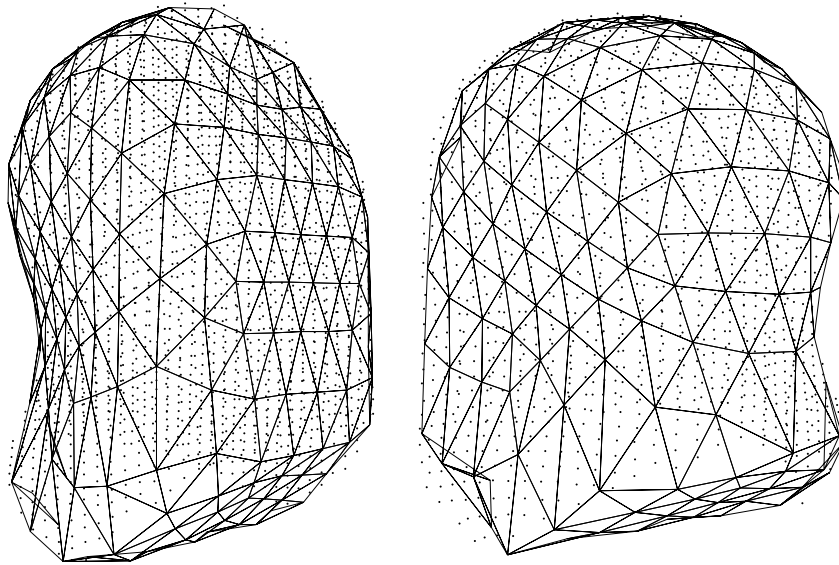
Figure 4.3 A 3-D triangle projected into range image parameter space.

### 4.3.1 Triangular Patch Approximation Error Evaluation

To evaluate the approximation error of a triangle  $T$  to the input range data, we first need to define the set of data points that need to be considered in evaluating the approximation error for the triangle. Naturally we do not need to consider those range images in which none of the triangle vertices have projections. For the range images in which at least one of the vertices has projections, we project (not to be confused with the projection mentioned early) the triangle  $T$  onto the range image parameter space to obtain a 2-D triangle  $T'$ , as though the triangle had been in the place when the range images were taken (see Figure 4.3).

In the simplest case of a Cartesian range image, where the depth value  $z$  is indexed by the  $x$  and  $y$  coordinates as  $z = f(x, y)$ ,  $T'$  is simply the orthographic projection of  $T$ . In general, however, the projection of  $T$  involves in a transformation from 3-D world coordinates into the sensor coordinates. Such a transformation is usually available from the calibration of the range finder system (see [63] and APPENDIX B for details). Once the projected 2-D triangle for each view is obtained, all range image points in that view that fall inside of the 2-D triangle can be considered in evaluating the approximation error.

This method works well when the triangles are close to the underlying surface represented by the range images, or when the local surface patch is not too oblique relative the sensor. When



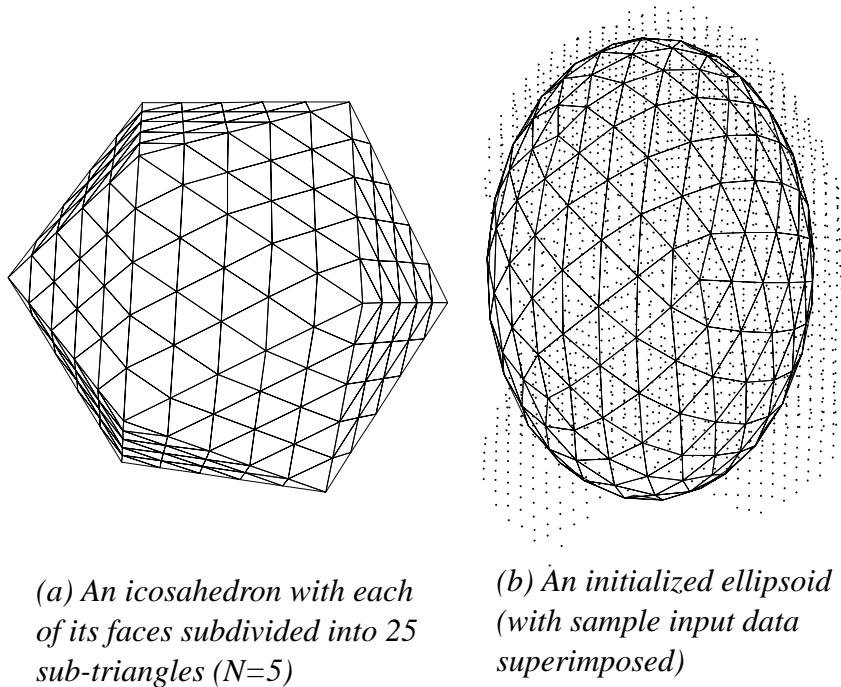
*Figure 4.2 Result of projecting the ellipsoidal shell onto the surface of an object. Shown with the wire frames are sample data points from the range image views of the object.*

itself on the ellipsoid. Thus an initial triangulated shell which is close to the object surface is desirable. An example of the result of the projection is shown in Figure 4.2.

Since our input is in the form of range images from many viewpoints, there are overlaps in the surface areas that each range image covers. This results in multiple projection points on the range images when a vertex is projected using the method described above. Assuming that there is only one intersection between the projection ray and the actual object surface, we need to combine the set of intersections obtained for the range images from different views into one. We have used the same weighted average method as in Section 3.5 (see Equation (3.15)), which takes into account the reliability of the input data.

### **4.3 Approximation via Triangle Subdivision**

When the triangulated shell is projected onto the object surface, each triangle may or may not approximate the covered surface well. In this section, we discuss how to evaluate the approximate error of the triangles, especially in the context of data integration, and how to improve the approximation error by subdividing the triangles.



*Figure 4.1 Initializing the ellipsoidal shell from an icosahedron with its faces subdivided.*

#### 4.2.1 Initializing and Projecting the Triangulated Shell

Our goal is to map a triangulated shell onto the surface of an object. We start with an icosahedron and divide each of its triangular faces into  $N \times N$  sub-triangles [21] (Figure 4.1(a)), where  $N$  is the order of initial subdivision. The triangulated shell is then re-initialized into an ellipsoidal shell of approximately the size of the object (Figure 4.1 (b)), based on the distribution information of the input data. This is done by estimating the three eigenvalues of the covariance matrix of the sample data points from the range images and by constructing an ellipsoid with axes in the direction of the eigenvectors and axes length in proportion to the eigenvalues.

The triangulated ellipsoidal shell is then projected onto the object surface by projecting the vertices, from the center of the projection (which is the center of the ellipsoid in our case). To find the projection for a vertex, a ray (the line of projection) is constructed from the center of the projection through the initial position of the vertex and the intersection between the ray and the surface is computed using the iterative algorithm presented in Section 3.3.4. The starting point for this intersection algorithm should be relatively close to the intersection point, which also means close to the surface. In our case the starting point is chosen to be the initial position of the vertex

## 4

# Surface Level Integration: Simple Objects

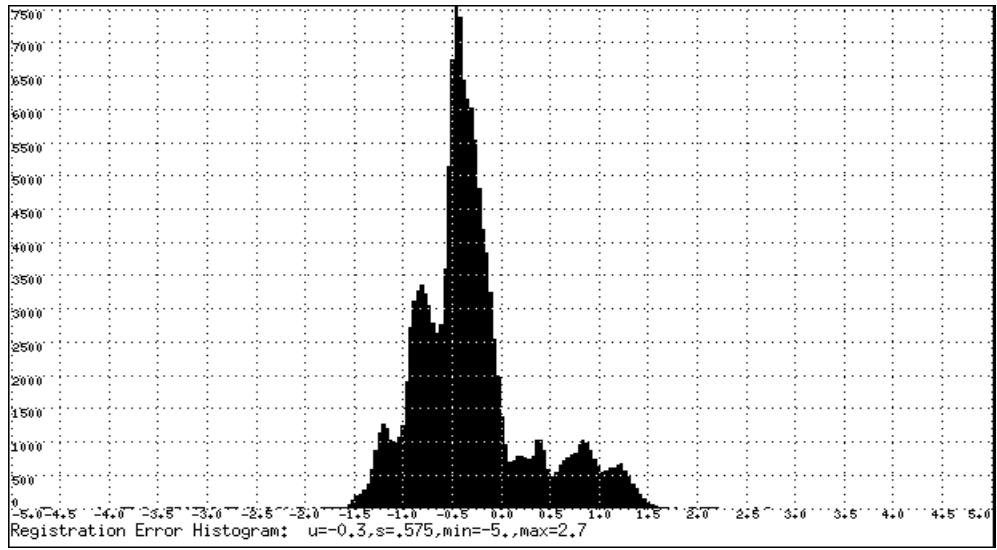
### 4.1 Introduction

In this chapter, we present an algorithm that builds a triangular mesh that approximates the object surface directly from multiple range image views, as opposed to integrating the data first as we have done in Chapter 3. The algorithm is based on a simple projection of the vertices of an initial mesh onto the object surface followed by a dynamic subdivision scheme. We also introduce an approximation error estimation scheme that reflects the integration role of our triangular mesh surface model. The main motivation for this research is that if the object we are dealing with are known to be not very complex, we can use a much simpler algorithm than the ones reported in the literature, which are much more complex yet have only been tested on simple objects.

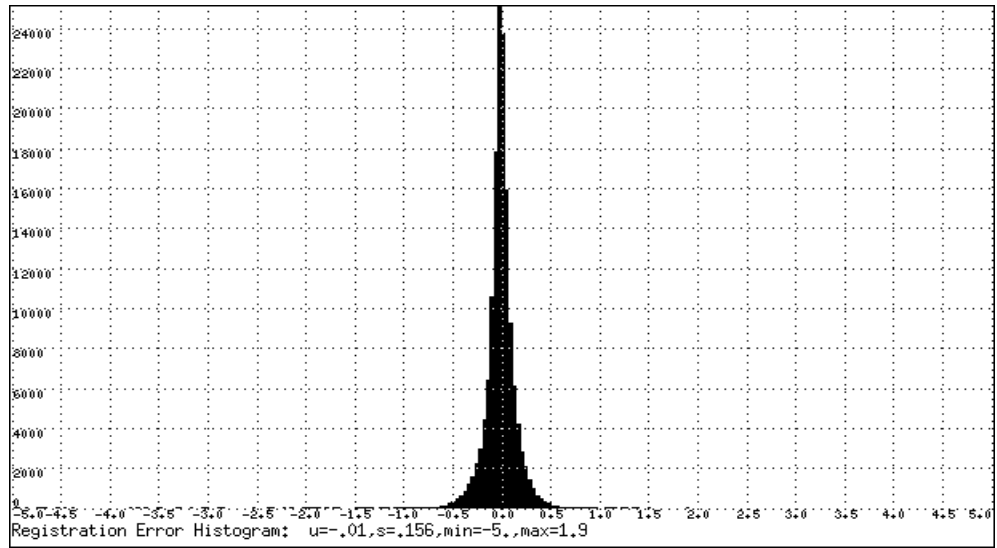
### 4.2 Mapping a Triangulated Shell by Projection

In our new method, a triangulated shell is first initialized around the object. Then the triangles are mapped onto the object surface by projecting them in the radial direction from the center of the projection. The projection is obtained by computing the intersections of the lines of projection and the object surface represented by the range images. Our input is a set of range image views of an object registered using the registration algorithm described in Chapter 3. Although registered, the individual views might not align exactly everywhere, which reflects some defects present in the raw data and some misalignment due to noise and other defects. We also assume that the range images are dense enough for evaluating surface properties such as surface normal vectors using neighborhood fitting. We first present the details of the method and then discuss the advantages and disadvantages of the method.



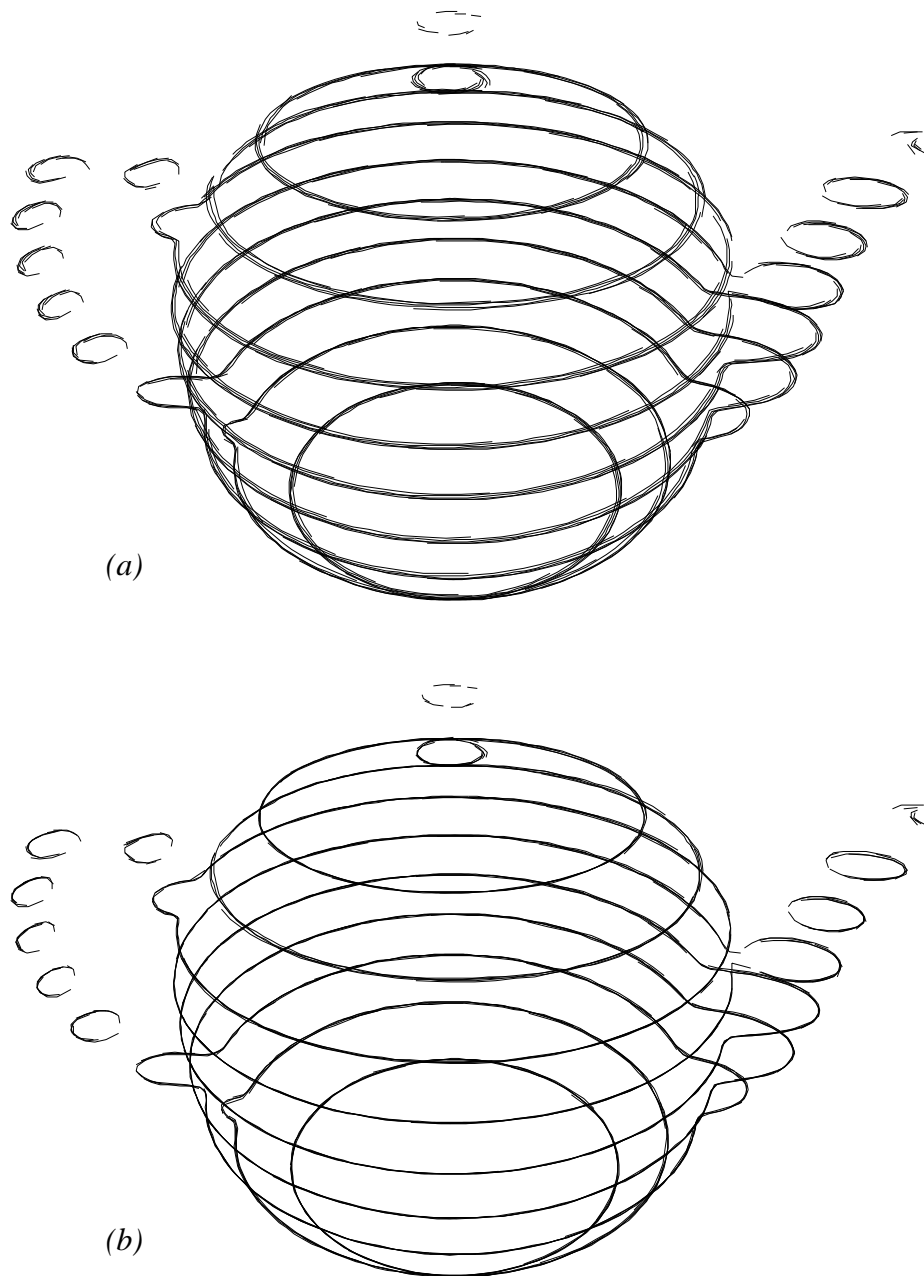


(a)



(b)

Figure 3.16 Registration error distribution for the teapot range images before (a) and after (b) registration with the global registration scheme.



*Figure 3.15 Range image slices of a teapot before (a) and after (b) registration with the global registration scheme. The initial positions of the roughly registered slices (a) are computed from the system calibration information.*



Figure 3.14 Sample range images from the teapot.

(standard deviation is  $0.575\text{ mm}$ ) and the registered ones have a nice Gaussian-like distribution (standard deviation is  $0.156\text{ mm}$ ).

### 3.6 Summary

In this chapter, we presented a range image registration algorithm that can find the registration transformation between the range image views of an object based on coarse estimates. An extension to this algorithm has been successfully used to model simple, compact objects with complex surface structures. This is the foundation for constructing models for more complex (non-star-shaped) objects.

Yet our registration algorithm can still be improved in the following aspects. The first is to reduce the dependency on the initial transformation information. This can be done by using certain object recognition or pose estimation schemes (e.g., [69]) to estimate the initial transformation. We can also use multi-resolution range images to enhance the ability of the algorithm to converge to the global minimum. The second is to find a systematic way of selecting control points for the registration (Section 3.3.5), so that the control points are well distributed to allow the best possible registration result. The third is to apply certain robust estimation techniques [47] to make the algorithm more stable to noisy input data.

where  $\mathbf{q}_i$  is the intersection of  $l$  with  $Q_i$ ,  $\hat{\mathbf{n}}_{q_i}$  is the unit vector normal to  $Q_i$  at  $\mathbf{q}_i$ ,  $\mathbf{n}_s$  is the vector pointing towards the sensor and  $a_i$  is a binary number depending on the intersection of  $l$  with  $Q_i$ , and

$$a_i = \begin{cases} 1, & \text{if the intersection exists} \\ 0, & \text{if there is no intersection} \end{cases}$$

The reason behind taking a weighted sum of the intersection points is that the sensor measurement of the position of a surface point is less reliable if the local surface is facing away from the sensor. The weight  $w_i$  is a reflection of this heuristic. In case of a Cartesian range image with no information about the sensor, one can simply take  $\mathbf{n}_s = (0, 0, 1)^t$ . In general, this information is available from the range image sensor setup and calibration. Likewise, we can also estimate the tangent plane at the estimated intersection point with its normal vector being:

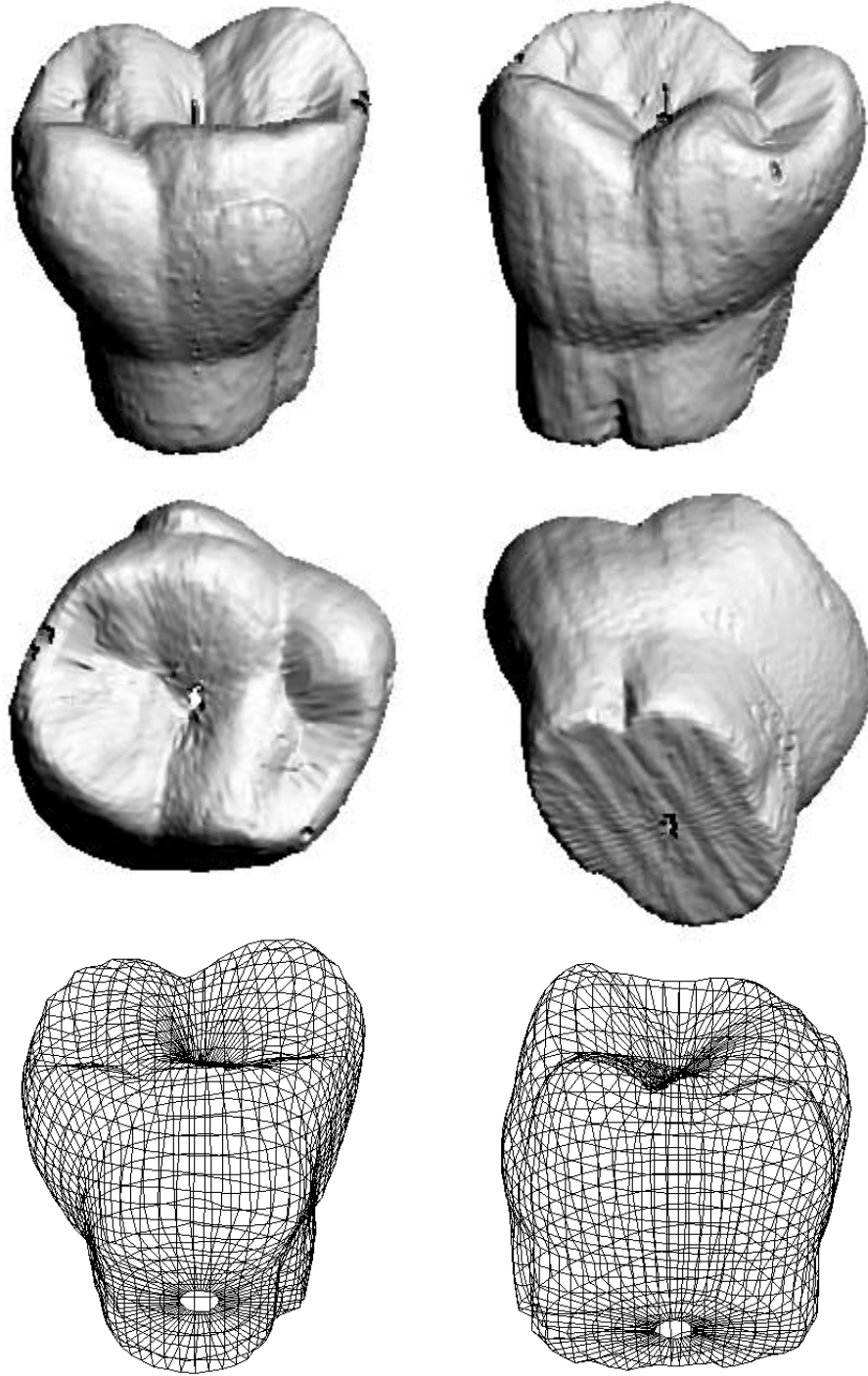
$$\hat{\mathbf{n}}_q = \frac{\mathbf{n}_q}{\|\mathbf{n}_q\|} \quad (3.16)$$

where

$$\mathbf{n}_q = \frac{\sum_{i=1}^m a_i w_i \hat{\mathbf{n}}_{q_i}}{\sum_{i=1}^m a_i w_i}$$

and  $w_i$  is defined in Equation (3.15).

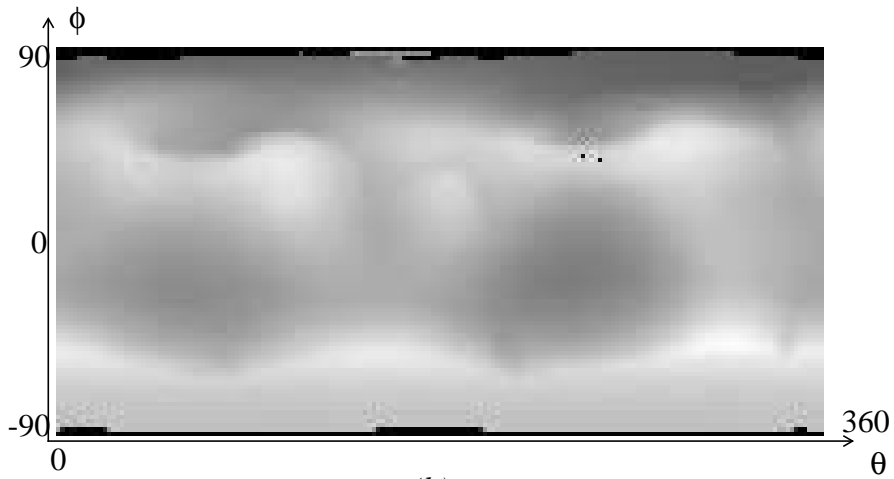
An example of this global registration is shown in Figure 3.14 and Figure 3.15 for a set of 12 range images of a teapot. Figure 3.14 shows some of the range images and Figure 3.15 shows the wireframes by linking sample data (a few horizontal slices) from each range image. In Figure 3.15 (a), the unregistered range images were in their initial positions computed from the calibration of the system when the range images were taken (the center of the rotary table and the amount of rotation are given by the calibration). Figure 3.15(b) shows the same wireframe after the registration. The error histograms of the registration are shown in Figure 3.16. The registration errors are computed using the same method as in Section 3.3.6, but for all possible pairs of range images. The histograms shown are for the total error distribution. As can be seen that the unregistered range images have scattered error distribution with multiple peaks and biased mean



*Figure 3.13 The rendered images (top and middle rows) and the wireframe drawings (bottom row) of the constructed model for the tooth.*

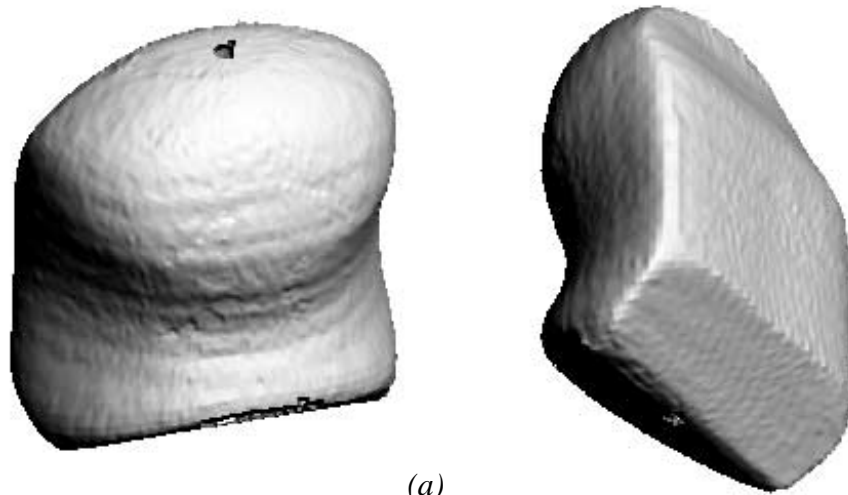


(a)

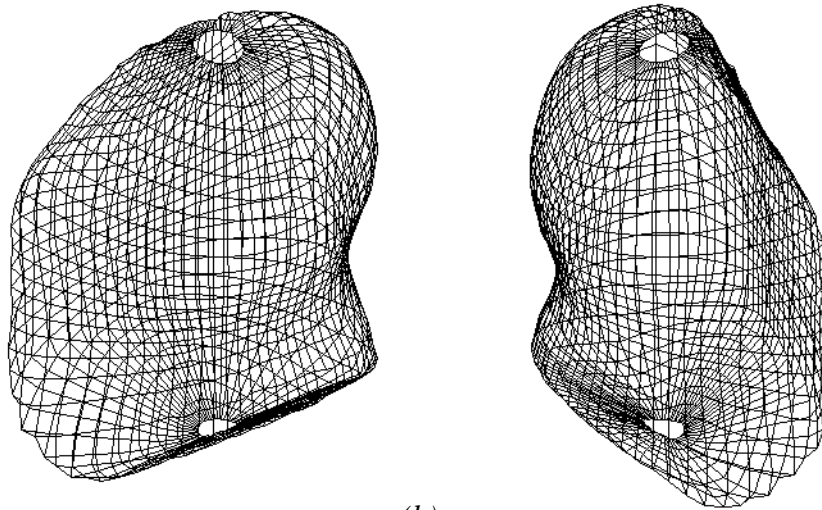


(b)

*Figure 3.12 The original model plaster tooth (a) and the constructed model (b) in spherical image form.*



(a)



(b)

*Figure 3.11 The rendered images (a) and the wireframe drawings (b) of the constructed model of the wood blob.*

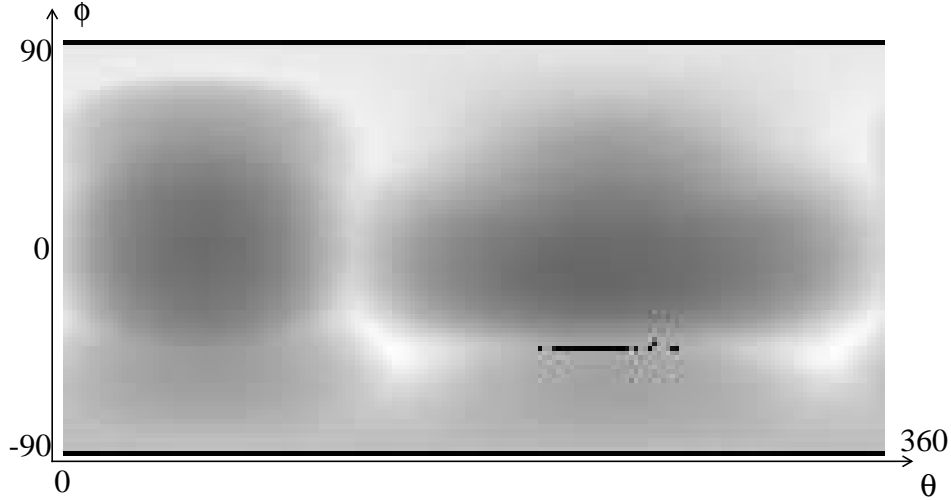


Figure 3.10 Constructed model for the wood blob in spherical coordinate image for the wood blob in Figure 3.9, where the horizontal and vertical axes correspond to the  $\theta$  and  $\phi$  angles respectively, while the image value encodes the radius  $r$ .

this is done and how the line-surface intersection is handled when there exist multiple range images that overlap.

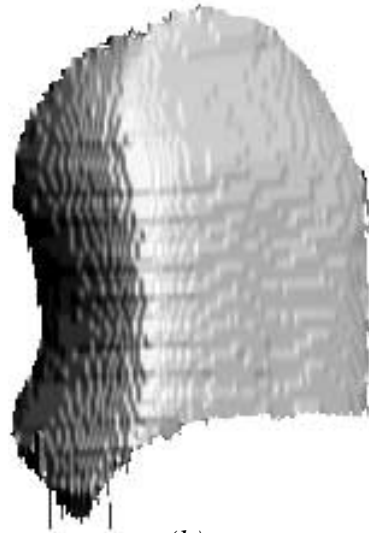
The basic idea of global registration is the same as that for the simple object case. When a new range image view is considered, we try to register it with all the previously registered views so far to reduce possible error accumulations. But instead of carrying out registration with the integrated previous views, we need to do it with a set of registered range images. As can be seen from Algorithm 3.1, the relevant part of this change in input form is step (2)(a), in which the intersection of a line with the range image and the tangent plane at the intersection are sought. A straightforward extension to the algorithm is to compute all the intersections of the line with all the range images we are registering it with. Let  $\{Q_i\}, i = 1, \dots, m$ , be the set of range images already

registered and  $l$  be the line in consideration. The intersection of  $l$  with  $\{Q_i\}$  can be written as

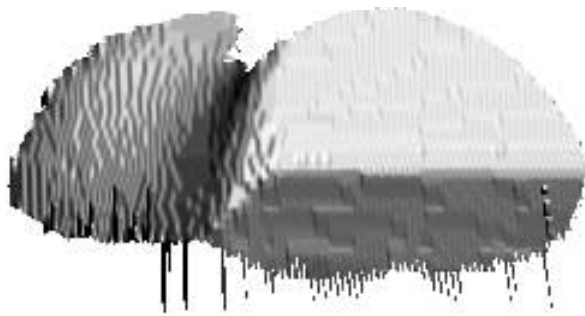
$$\mathbf{q} = \frac{\sum_{i=1}^m a_i w_i \mathbf{q}_i}{\sum_{i=1}^m a_i w_i}, \quad w_i = \hat{\mathbf{n}}_{q_i} \bullet \mathbf{n}_s \quad (3.15)$$



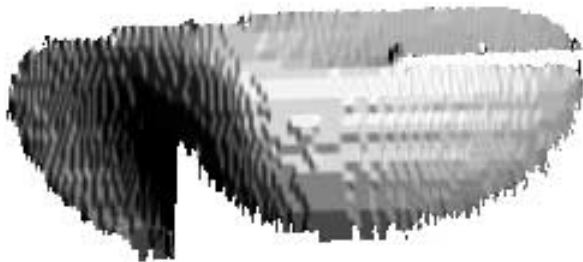
(a)



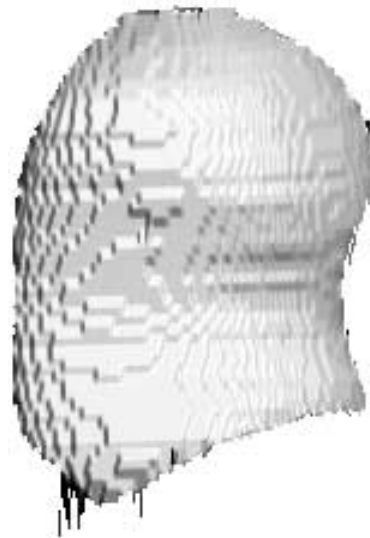
(b)



(c)



(d)



(e)

*Figure 3.9 The wood blob and four of its range images used for model construction.*

### 3.4.3 Results and Discussion

We present results for two different objects. One is a free-form shaped wood blob (“Wood”), and the other is a plaster model of a tooth (“Tooth”). Both objects are free-form objects with few distinct surface features. Thus, matching detected surface features for registration would prove very difficult. The surface structures of these objects (especially the model tooth) are very special in that it is very difficult to define a reliable segmentation to achieve high level description. In Figure 3.9, the original wood blob intensity image and some of the range images (shaded) used in the modeling process are shown. The integrated results are shown in Figure 3.10 as a spherical coordinate map, and in Figure 3.11 as rendered intensity images and wireframe plots. In Figure 3.12 the results for the model tooth are shown in spherical image along with the intensity image of the original object. The rendered images and wireframe drawings of the model are shown in Figure 3.13. In these examples, we have used 8 side views and 6 to 8 top and bottom views for each object with  $45^\circ$  of rotation angle between successive side views for simplicity. From the results, we can see that, although the original range images (as shown in Figure 3.9) exhibit heavy quantization noise in the acquisition process, the final integrated models are generally smooth, which shows the quality of the registration. Although the averaging effect of the integration of multiple range images has contributed in some extent, the shape of the object surfaces in the ridge and valley areas have been presented very well, which shows the good performance of the registration algorithm, since, otherwise, the data from different views would not “fit” together and the ridges and valleys on the object surface would have been blurred. The spherical coordinate maps shown in Figure 3.10 and Figure 3.12 contain a few small uncovered areas, especially in the pole areas. This is due to the fact that the poles are special points for our representation and to the process of mapping range images into spherical maps. Other small uncovered areas can be solved by a more careful selection the individual views in taking range images.

## 3.5 Handling Complex Objects

We have presented a complete strategy in constructing low level models for compact objects. This method, however, is clearly not suitable for objects with more complex geometries due to the limitation of the representation schemes used. In order to be able to handle object such as a telephone handset (see Figure 1.1 on page 2), we must use some other representation schemes. Before we commit to certain type of shape representation schemes, however, we believe that the best thing to do is to keep all the range images in their original form, since once registration is done, we can choose what ever representation is appropriate for integration and representation. The question now is what changes we need to make in the registration algorithm in order to achieve a similar global registration effect as in the simple object case. In the followings we will discuss how

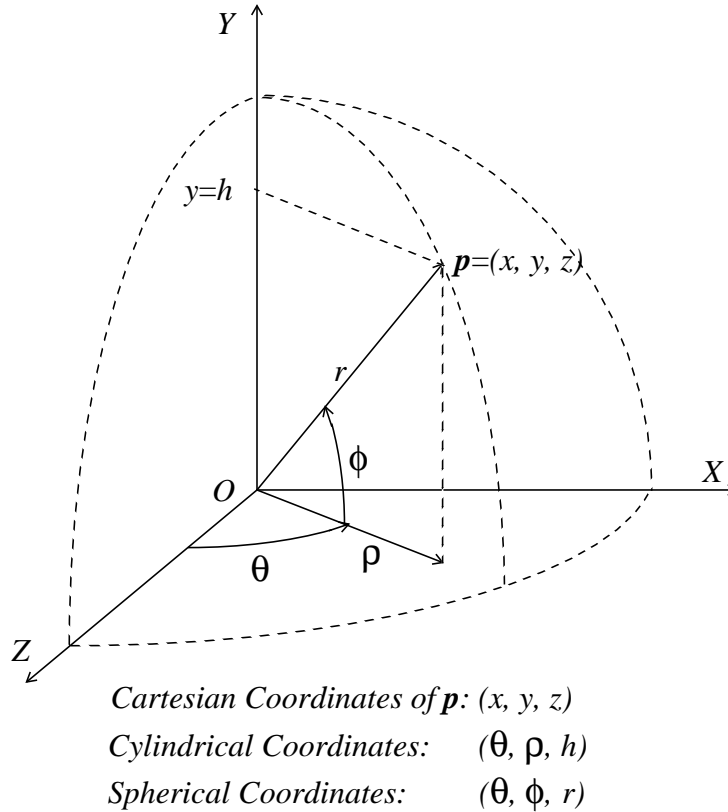


Figure 3.8 Relationship between Cartesian, cylindrical and spherical coordinate systems.

They are then further re-parameterized through interpolation into cylindrical or spherical coordinates (see [62] for the re-parameterization method). Since all data are now in the same parameter space, additional data from different views can be merged into one by simple average in the overlapping areas. The actual implementation also includes decisions about avoiding outliers.

### 3.4.2 Global Registration

For this specific modeling procedure, we can work globally for better overall results. That is, when we merge a current view range image, instead of registering it with only the neighboring views, we register it with the merged data from all previous views to find out the needed transformation. In this way, the information from all the previously merged views is used, and the error accumulation due to successive registration can be reduced.

established, and the fact that the distance measure used is not invariant to the sensor’s viewing direction.

We have noticed that in [84], there is also a pair of human head range images with similar initial rotation between them to ours, but with much less noise in the images. Their closest-point-based algorithm takes 39 iterations. The reason why our algorithm and that of Horn and Harris’ has similar rate of convergence is that in Horn and Harris’ method, the derivatives of the range images are used in their formulation and hence achieved similar effect as our first order approximation.

### 3.4 Integration of Multiple Range Images for Compact Objects

In this section, we present an application of the range image registration algorithm to the modeling of compact objects. In order to get a description of the whole object surface, multiple range images of the object from different vantage points are needed. While putting an object on a rotary table is enough for obtaining a wraparound representation of the object without use of registration, as is done in [79], it is seldom the case that all surface areas can be covered in this manner. But when taking range images from different object poses, the exact spatial relationship between them is lost. Our test goes as follows. We first put the object on rotary table and take a set of 4 to 8 side view range images of the object, and then the object is laid down and the range images of the top and bottom areas of the object are taken. The number of side views depends on the complexity of the object surface structure. For the range images taken from the top and bottom views, we need the approximate relationship between them and those of the side views in order to use our registration algorithm, as assumed in the first part of this paper. This is done by asking the user to estimate the 3 rotation angles, and they do not have to be accurate. Then the registration algorithm is applied to bring the range image of the top and bottom views in registration with those of the side views. The details for merging data are described in the following sections.

#### 3.4.1 Object-Centered Representation

The original range images come as three separate dense images  $x(u, v)$ ,  $y(u, v)$  and  $z(u, v)$  in the parametric space of the sensor. Our goal is to build an object-centered model for a class of compact objects. The object is described in a cylindrical or spherical coordinate system (based on the type object, e.g., for an elongated shape we use cylindrical coordinates) centered in the object (Figure 3.8), hence the term object-centered as opposed to the viewer centered representation used in the original range images. This kind of representation provides us with an intermediate representation towards a higher level description of the object. Once the object coordinate frame has been selected, the range image from each view of the object is transformed to this coordinate frame with transformations from *a priori* knowledge (for side views, e.g.) or from registration.

**Table 3.5: Registration convergence test: translation in Y axis**

Translation in Y axis in mm	2.5	5.0	7.5	10.0	12.5	15.0	17.5	20.0	22.5
Iterations	5	6	6	7	8	8	8	9	9
RMS error	0.56	0.56	0.57	0.57	0.56	0.56	0.57	0.57	0.56
Control pts	278	276	275	276	273	275	272	268	272
Translation in Y axis in mm	-2.5	-5.0	-7.5	-10.0	-12.5	-15.0	-17.5	-20.0	-22.5
Iterations	5	6	7	8	8	9	9	10	10
RMS error	0.57	0.57	0.57	0.57	0.57	0.57	0.56	0.57	0.57
Control pts	278	277	277	277	274	274	274	276	268

**Table 3.6: Registration convergence test: translation in Z axis**

Translation in Z axis in mm	2.5	5.0	7.5	10.0	12.5	15.0	17.5	20.0	22.5
Iterations	4	5	6	6	6	8	7	8	8
RMS error	0.57	0.57	0.57	0.57	0.57	0.56	0.57	0.58	0.58
Control pts	279	278	274	278	277	276	276	274	275
Translation in Z axis in mm	-2.5	-5.0	-7.5	-10.0	-12.5	-15.0	-17.5	-20.0	-22.5
Iterations	4	4	4	6	6	7	5	7	6
RMS error	0.56	0.57	0.56	0.57	0.57	0.57	0.59	0.56	0.59
Control pts	277	273	274	274	275	273	272	271	271

**Table 3.3: Registration convergence test: rotation in Z axis**

Rotation in Z axis in degrees	5	10	15	20	25	30	35	40	45
Iterations	7	8	7	7	12	9	12	14	14
RMS error	0.57	0.57	0.57	0.56	0.57	0.58	0.57	0.57	0.56
Control pts	276	274	272	268	260	260	260	257	261
Rotation in Y axis in degrees	-5	-10	-15	-20	-25	-30	-35	-40	-45
Iterations	6	7	9	8	9	12	12	12	13
RMS error	0.57	0.57	0.57	0.57	0.57	0.57	0.56	0.57	0.56
Control pts	276	273	275	273	271	268	264	266	263

**Table 3.4: Registration convergence test: translation in X axis**

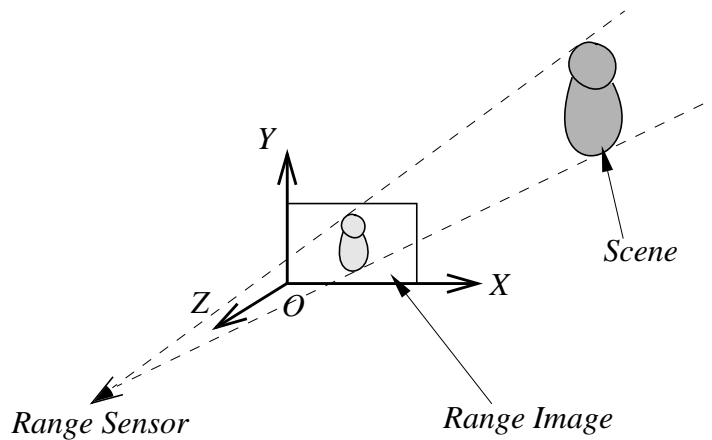
Translation in X axis in mm	2.5	5.0	7.5	10.0	12.5	15.0	17.5	20.0	22.5
Iterations	8	8	8	9	9	10	12	11	13
RMS error	0.57	0.56	0.57	0.57	0.57	0.57	0.57	0.57	0.57
Control pts	279	279	276	276	273	273	267	273	270
Translation in X axis in mm	-2.5	-5.0	-7.5	-10.0	-12.5	-15.0	-17.5	-20.0	-22.5
Iterations	5	8	9	9	9	12	15	NC	NC
RMS error	0.57	0.57	0.57	0.58	0.60	0.57	0.56	N/A	N/A
Control pts	276	273	271	265	273	267	255	N/A	N/A

**Table 3.1: Registration convergence test: rotation in X axis**

Rotation in X axis in degrees	5	10	15	20	25	30	35	40	45
Iterations	4	5	6	7	7	9	12	14	15
RMS error	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57	0.57
Control pts	279	280	279	280	275	270	272	268	273
Rotation in X axis in degrees	-5	-10	-15	-20	-25	-30	-35	-40	-45
Iterations	4	6	7	7	7	10	11	15	NC
RMS error	0.57	0.57	0.55	0.57	0.57	0.57	0.57	0.57	N/A
Control pts	278	279	276	278	277	277	273	267	N/A

**Table 3.2: Registration convergence test: rotation in Y axis**

Rotation in Y axis in degrees	5	10	15	20	25	30	35	40	45
Iterations	6	5	5	7	6	7	7	8	12
RMS error	0.56	0.57	0.57	0.57	0.57	0.56	0.56	0.57	0.59
Control pts	278	280	280	279	279	273	273	271	263
Rotation in Y axis in degrees	-5	-10	-15	-20	-25	-30	-35	-40	-45
Iterations	5	6	5	6	8	8	9	11	11
RMS error	0.57	0.57	0.60	0.60	0.57	0.59	0.59	0.57	0.58
Control pts	277	275	277	274	269	268	268	268	265

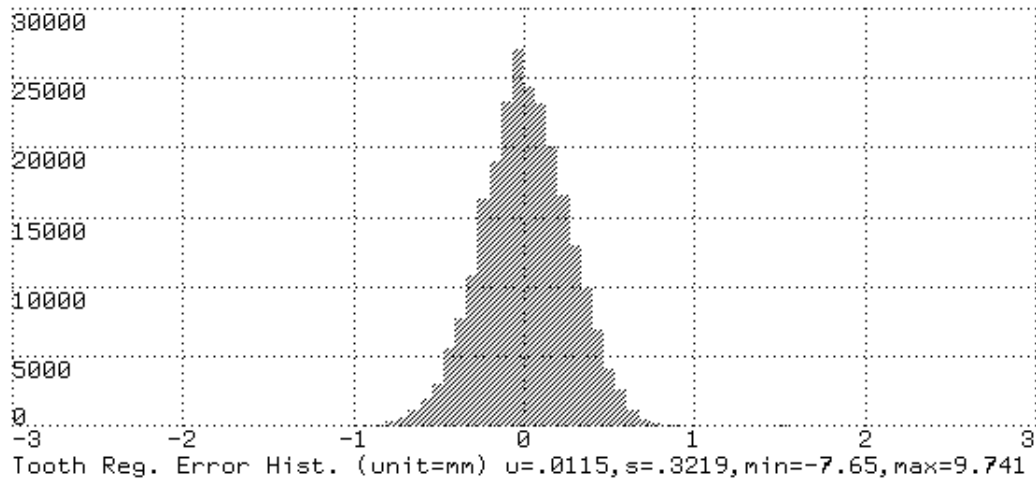


*Figure 3.7 Coordinate system for the range images used in this dissertation.*

points are used in these experiments than in the previous section partly ensures that the RMS errors are good indications of the quality of the registration. As shown in the tables, our algorithm can still converge under very large initial rotation (up to  $40^\circ$ ) and translation (up to  $17.5\text{ mm}$  which is about 40% of the width of the face of the bust shown in Figure 3.1) with a reasonable number of iterations. For those tests that did converge, the quality of the registration seems to be unrelated to the initial states, although it does take more iterations when the initial position is further away from registration.

### **3.3.8 Comparisons and Analyses**

We have also implemented Horn and Harris's algorithm [34] for comparison, with a similar iterative scheme as the one described earlier. The corresponding points are obtained by simply projecting the points from one image onto the second after the initial transformation is applied. We used the least-squares method that is described in the first part of the paper with the same set of control points as used in our algorithm, instead of over the whole image for computation time considerations. Iterative application of the method is needed because the simple projection scheme, by which the algorithm establishes correspondence between points in two range images, does not produce true correspondence. In the examples we tested successfully using our method (including those presented later in this dissertation), Horn and Harris's method produced similar results in a comparable number of iterations, but failed to converge on quite a few examples, due to poor initial transformation estimations. This could be contributed to the way the corresponding points are



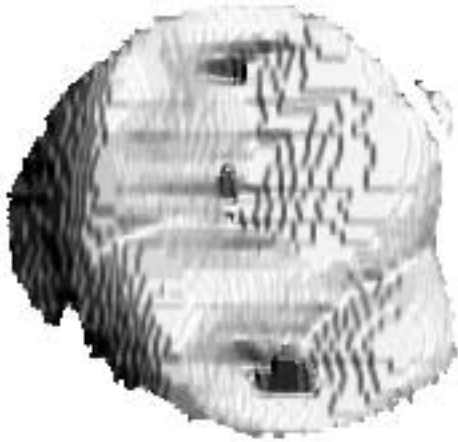
(d) Histogram of the error image.

Figure 3.6 (continued) Registration results for the model tooth.

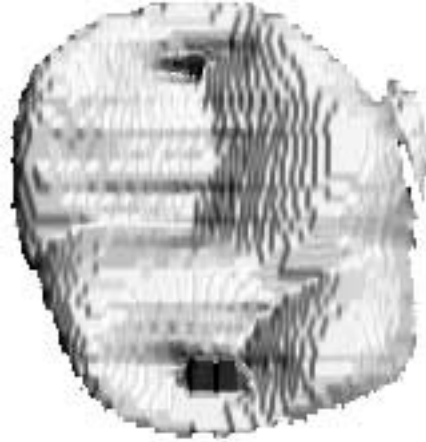
(72 of which found correspondence) and 6 iterations. In both cases,  $\epsilon_e$  in Equation (3.12) was set to 0.01.

### 3.3.7 Stability with Respect to Initial Conditions

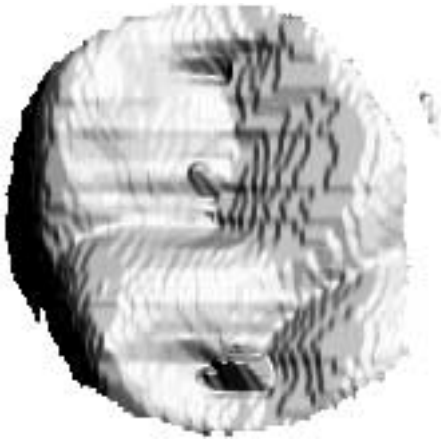
In order to show the ability of the registration algorithm under varying initial conditions, we have conducted the following experiments. We first register the two Mozart range images as described above. Then we manually control the initial position of the first range image relative to the second by rotating or translating the first image from the position in registration. We vary the amount of rotation and translation in both positive and negative directions and in all axes. The coordinate system for the range images in this experiment is such that the X axis is pointing towards the right, the Y axis up and the Z axis out towards observer (Figure 3.7). The center of rotation is located at the center of inertia of the sample data points from the two range images. The results are shown in Table 3.1 through Table 3.6. In these tables, the first and fifth rows show the amount of rotation or translation about the indicated axes or in the indicated directions. The second and sixth rows show the number of iterations the algorithm takes to converge for  $\epsilon_e=0.01$  (table entry NC means non-converging). The third and seventh rows show the residue RMS (root mean squares) errors for the set of control points used, and the fourth and eighth rows show the number of control point used when the registration converges (out of a total of 301 control points used). We have not included in the tables the recovered parameters (rotation and translation) since the six parameters are not independent and it is impossible to list them all. The fact that many more control



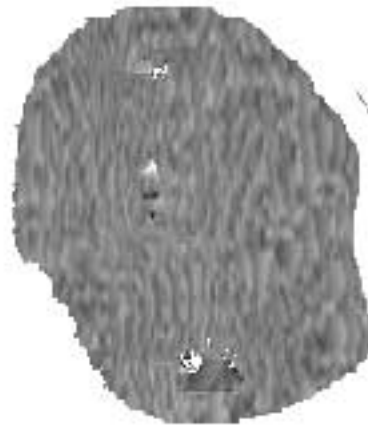
(a) The first view of a model tooth (175×168).



(b) The second view of the model tooth (163×168).



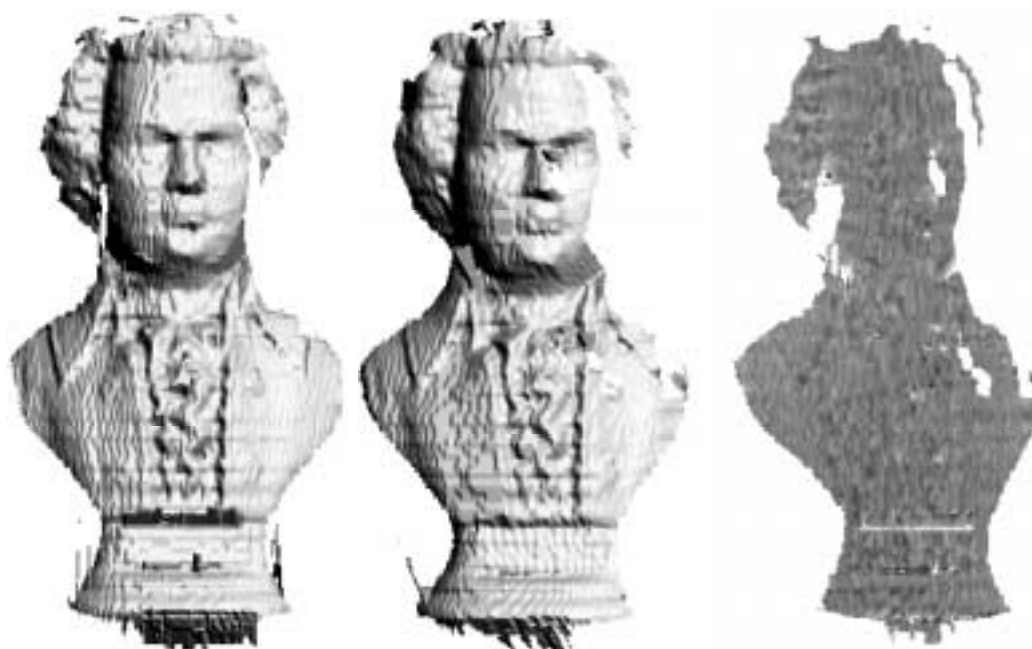
(c) The first view of the tooth after registration.



(d) The registration error image

Figure 3.6 Registration results for the model tooth.  
(Continued on next page.)

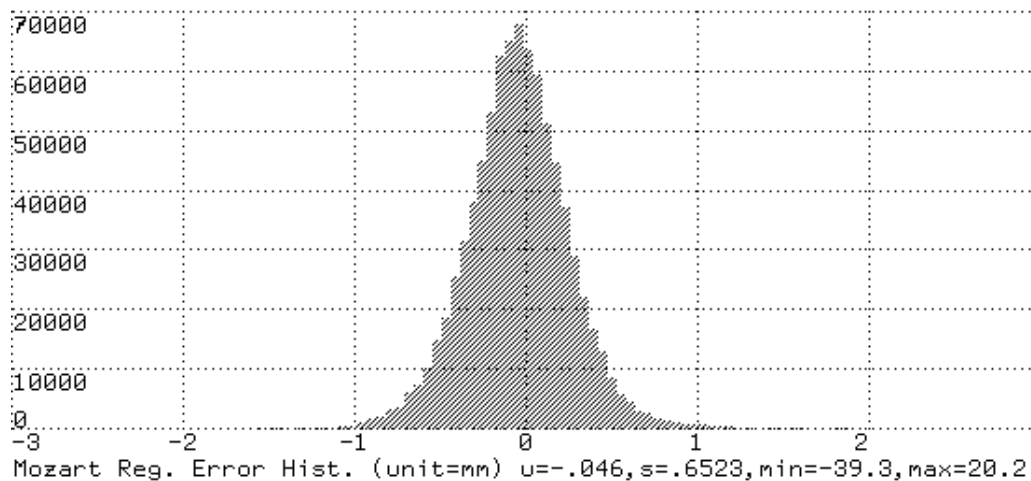
the rotation between the two Mozart images to be  $-15.06^\circ$  while the actual rotation is  $-15^\circ$ . For the second test shown in Figure 3.6, the computed rotation of the two images is  $-19.75^\circ$  while the actual value is  $-20^\circ$ . The running times (elapse time) for the tests on a Symbolics 3620 Lisp Machine for the whole registration processes, from control point selection to transformation output, are as follows. It took 20 seconds for the Mozart image with 82 control points (78 of which found correspondence) and 7 iterations. The tooth image took 15 seconds with 88 control points



(a) The second view of the Mozart bust.

(b) The first view of the Mozart bust after registration

(c) The registration error image.



(d) The histogram of the error image in (c).

Figure 3.5 Registration results for the Mozart range images. (In the histogram figure, “u” stands for “mean”, “s” for “standard deviation” and “min” and “max” for minimal and maximal values respectively.)



*Figure 3.4 The selected control points for Figure 3.1(a).*

images are all 0.5 mm, while the accuracy of the range finder is in the neighborhood of 1 mm. In the first example, we register the two range images shown in the beginning of this chapter in Figure 3.1. Figure 3.5(b) shows the first view of the Mozart bust after the registration is performed, while (c) and (d) show the registration error image and error histogram. The second test is shown in Figure 3.6 for the range images from a model tooth. The range images in the first example contain many detailed patterns and complex surface structures. But due to quantization noise, surface features cannot be detected reliably, thus matching surface features for registration may not produce satisfactory results. On the contrary, the range images in the second example contain few interesting features. Besides, the surface type is very special in that it is very difficult to have a good surface segmentation, and therefore difficult to match high level surface descriptions. Our registration algorithm works very well on these examples, as can be seen from the error images and error histograms in Figure 3.5 and Figure 3.6. They are very comparable to the resolution of the original data. In fact, if the images are smoothed with a small Gaussian filter, an even better registration accuracy can be achieved. From the computed registration transformation, we found

### 3.3.5 Control Point Selection

Next, we discuss the selection of control points on  $P$ . Since our control points do not need to represent meaningful surface features, and we only need them from one surface  $P$ , we can simply pick points on  $P$  on a regular grid, instead of every point on  $P$  to save computation time. However, we do want the control points to be in smooth areas for two reasons. First, the corresponding neighborhoods on surface  $Q$  of the control points will likely be smooth, and, consequently, the outcome of the line-surface intersection algorithm will be more reliable and stable. Second, the surface normal can be computed more reliably. This makes a major difference between our control point selection strategy and most other feature-based matching algorithms which seek points or features that represent certain geometric context for localization purpose. This is because our registration algorithm does not rely on the accurate correspondence of the control points or other features, but uses the constraints from the overall shape of the surfaces. To check the smoothness of a surface in some neighborhood, we can fit a smooth surface function to the neighborhood using a least-squares method and check the residual error of the fit. For simplicity in implementation, we fit a plane. The threshold on the residual error is currently chosen through experiment, and it should not be a problem with some noise estimation procedure on the range data. One example is shown in Figure 3.4 for the chosen control points for the range image in Figure 3.1(a), where the threshold for the residual standard deviation of a  $9 \times 9$  window plane fitting was set to half of a spatial sampling unit. Other issues relating to the selection of control points are the number of the control points and their distribution in the range image. Depending on the type of surface and how much the two range images overlap, the control points needed for a good registration usually range from 50 to several hundred, but more can be used if needed. Since more control points means more computation, we can begin with less control points and increase the number when we come close to convergence. Currently we have not adopted a measure to ensure a good distribution of control points, since it is a complicated issue relating to global surface properties and evaluation criteria. But it is important for a robust registration algorithm.

### 3.3.6 Experiments

We present some examples of the range image registration algorithm. The range images used in the examples are Cartesian images with the background area already identified. As before, all range images are shown as shaded intensity images for display purposes. The actual images all have single-precision (24-bit) floating-point number pixels or  $(x, y, z)$  values (depending on the images). The two range images used in each example are taken under the same conditions, but the objects in the images were rotated 15 to 20 degrees about the  $y$  axis. In these tests, we simply use the identity matrix as the initial transformation. The error images are computed as the distances from the surface of the first image to the second in the surface normal direction of the first surface, once registration has been performed. The spatial resolutions of the range images and the error

surface normal of the tangent plane,  $\mathbf{r}$ , satisfy  $\mathbf{n}_p \bullet \mathbf{r} > 0$ , which simply means that the corresponding surface neighborhoods have to be in approximately the same orientation. Similar to the Newton method for finding roots of a function of type  $y=f(x)$ , the convergence of this algorithm depends on the choice of the starting point, which is the orthographic projection of  $\mathbf{p}$  on  $Q$  along the  $z$  axis, and on the surface characteristics of the neighborhood of the actual intersection point. So, if the neighborhood of  $\mathbf{q}$  is smooth and the assumption about the initial approximate registration is true, then the projection of  $\mathbf{p}$  onto  $Q$  will be close to  $\mathbf{q}$  and we will have a high likelihood to converge. Typically one intersection takes 3 to 5 iterations for the  $\epsilon_d$  specified below. The algorithm will fail when either the projection of  $\mathbf{p}$  is outside of the represented surface by  $Q$  or the surface normal is nearly perpendicular to the  $z$  axis.

The surface normal vectors needed in computing line-surface intersections are estimated using surface fitting [75]. If the range image is represented as a Cartesian image  $f(x,y)$ , we first approximate the first order derivatives  $\frac{\partial f}{\partial x}$  and  $\frac{\partial f}{\partial y}$  using a finite difference mask and then compute the normal as

$$\left(1, 0, \frac{\partial f}{\partial x}\right)^t \times \left(0, 1, \frac{\partial f}{\partial y}\right)^t \quad (3.13)$$

When the range images are specified as three separate dense images of 3-D coordinates  $x(u,v)$ ,  $y(u,v)$  and  $z(u,v)$ , surface normal vectors are computed as follows. The surface fitting technique is applied to each of the  $x$ ,  $y$  and  $z$  images to get partial differential estimates for each of them with respect to the parameters  $u$  and  $v$ . Then the surface normal is computed by

$$\left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right)^t \times \left(\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v}\right)^t \quad (3.14)$$

For a fixed window size in evaluating surface normal vectors, the time complexity for each iteration of this line-surface intersection algorithm is constant. As for the selection of  $\epsilon_d$ , we have set it to the space sampling unit of the range image, which means that we terminate the iteration process when the computed intersection converges to within the neighborhood of a spatial sampling unit. This is because our final goal of computing the intersection point  $\mathbf{q}$  is to find the tangent plane at  $\mathbf{q}$  according to the last section, and the exact location of  $\mathbf{q}$  is not very important. A second order surface fitting and a subpixel intersection algorithm can be used which may provide more accurate results.

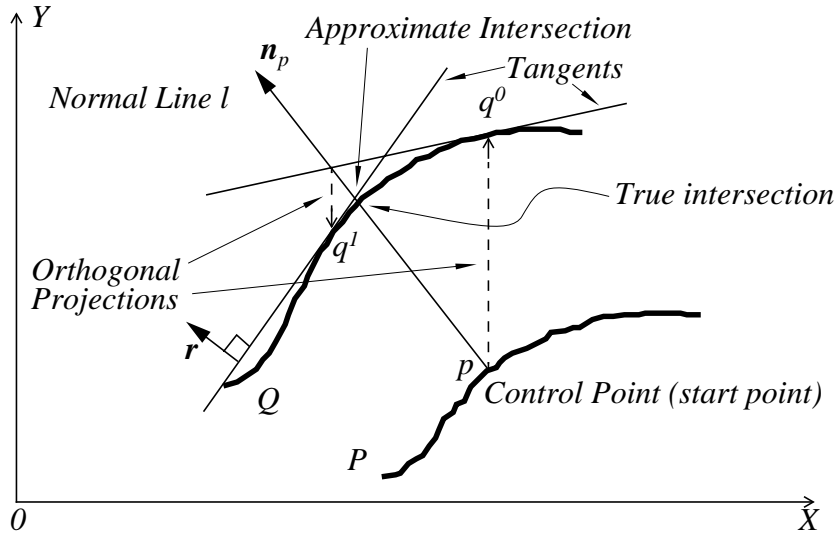


Figure 3.3 Intersecting a line with a digital surface illustrated in a 2-D case.

$Q=Q(x,y)$ , and the value of  $P$  or  $Q$  represents the distance from some reference plane. Under the assumption of approximate initial registration,  $p$  is expected to be in the neighborhood of  $q$ . In the following algorithm the initial prospective intersections are chosen by projecting  $p$  orthographically along the  $z$  axis onto  $Q$ . In the case where the range images are represented in a general parametric form of  $x(u,v)$ ,  $y(u,v)$  and  $z(u,v)$ , this projection can be replaced by a system calibration matrix (see [63] and APPENDIX B) that projects any 3-D point on to the range image parameter space. Our intersection algorithm works as follows (see Figure 3.3 for an illustration of the process for a typical 2-D case):

### Algorithm 3.2

- 1) Let  $p=(x,y,P(x,y))^t$ , be a point on  $P$ , and  $l$  a line normal to  $P$  at  $p$ . Let  $x^0=x$ ,  $y^0=y$ .
- 2) At each iteration  $k$ , for  $k=1, 2, 3 \dots$ , compute  $q^k=(x^k,y^k,z^k)^t$ , the intersection of  $l$  with the tangent plane of  $Q$  at  $(x^{k-1},y^{k-1},Q(x^{k-1},y^{k-1}))^t$ , under a directional constraint (see below).
- 3) We stop when  $\|q^k - q^{k-1}\| \leq \epsilon_d$ , ( $\epsilon_d > 0$ ).

The directional constraint for computing the intersection of line  $l$  and the tangent plane of  $Q$  says that the intersection exists only when the direction of  $l$  (the surface normal vector  $n_p$ ) and the

- a) For each control point  $p_i$ ,
  - Apply  $T^{k-1}$  to both the control point  $p_i$  and the normal  $n_{q_i}$  to get  $q'_i$  and  $n'_{p_i}$ .
  - Find the intersection  $q_i^k$  of surface  $Q$  with the normal line defined by  $p'_i$  and  $n'_{p_i}$  (details in Section 3.3.4).
  - Compute the tangent plane  $S_i^k$  of  $Q$  at  $q'_i$ .
- b) Find the transformation  $T_{\Delta}^k$  that minimizes  $e^k$  in Equation (3.11) with a least-squares method.
- c) Let  ${}^k = T_{\Delta}^k \circ T^{k-1}$ .

The control point selection and line-surface intersection problems will be discussed in the next two sections. The convergence of the procedure is tested by checking

$$\delta \equiv \frac{|e^k - e^{k-1}|}{N^k} \leq \varepsilon_e, \quad (\varepsilon_e > 0) \quad (3.12)$$

where  $\varepsilon_e$  is a threshold set via experiment,  $N^k$  is the actual number of  $p'_i$ 's used, since some of them may not have counter part in  $Q$ . Although we could have checked  $e^k$  directly, since  $\delta$  is much less sensitive to noise while  $e^k$  is a direct reflection of the noise level, a reasonable  $\varepsilon_e$  serves for a variety of range images. The time complexity for each iteration of this algorithm is linear in the number of control points used.

### 3.3.4 Iterative Line-Surface Intersection Algorithm

In this section, we present the algorithm for finding the intersection of a line with a digital surface, which is needed in Equation (3.10). It is also the basis of the line-surface intersection algorithms used through out of this dissertation.

Let  $P$  and  $Q$  be the two views of the surface in consideration, and  $p \in P$ . In implementing the idea of registration from last section, we need to find the intersection of the line

$l = \{ \forall x \in \mathbb{R}^3 \mid n_p \times (q - x) = 0 \}$ , which passes through  $p$  and in the direction of the surface

normal vector  $n_p$  of  $P$  at  $p$ , with surface  $Q$ . Let the intersection point be  $q \in Q$ . Since we do not have an analytical form for surface  $Q$ , we have to use an approximation. Our approach is to find the intersection of the line  $l$  with the tangent plane to  $Q$  in the neighborhood of prospective intersection points on  $Q$  instead (first-order approximation). Thus this is an iterative process and we need to have a prospective intersection to start with. As an example, let us consider the case where  $P$  and  $Q$  are represented in a Cartesian range image form, i.e., we have  $P=P(x,y)$  and

### 3.3.3 Algorithm Description

Assume that we have two surfaces  $P$  and  $Q$ , and an initial transformation  $T_0$  which applies to  $P$ . We rewrite Equation (3.10) in Section 3.3 as follows:

$$e^k = \sum_{i=1}^N d_s^2(T_{\Delta}^k T^{k-1} \mathbf{p}_i, S_i^k) \quad (3.11)$$

where

- $T_{\Delta}^k T^{k-1} = T^k$ , and  $T_{\Delta}^k$  is the transformation to be computed.
- $S_i^k = \{ \forall \mathbf{x} \in \mathfrak{R}^3 \mid \mathbf{n}_{q_i}^k \bullet (\mathbf{q}_i^k - \mathbf{x}) = 0 \}$  is the tangent plane to  $Q$  at  $\mathbf{q}_i^k$ .
- $\mathbf{n}_{q_i}^k$  is the unit vector normal to surface  $Q$  at  $\mathbf{q}_i^k$ .
- $\mathbf{q}_i^k = (T^{k-1} l_i) \cap Q$  is the intersection point of  $Q$  with line  $T^{k-1} l_i$ .
- $l_i = \{ \forall \mathbf{x} \in \mathfrak{R}^3 \mid (\mathbf{p}_i - \mathbf{x}) \times \mathbf{n}_{p_i} = 0 \}$  is the line normal to  $P$  at  $\mathbf{p}_i$ ,
- $\mathbf{p}_i \in P$  is a point on  $P$ .
- $d_s(\cdot)$  is the signed distance from a point to a plane.
- “ $\bullet$ ” and “ $\times$ ” stand for scalar and vector products respectively.

Our registration algorithm is to find the  $T_{\Delta}^k$  which minimizes  $e^k$  in the above equation with a least-squares method. At each iteration,  $T_{\Delta}^k$  updates the overall registration transformation  $T^k$  until the process converges. The algorithm can be described as follows:

#### Algorithm 3.1

- 1) Select a set of control points  $\mathbf{p}_i \in P$  ( $i=1 \dots N$ ) (see Section 3.3.5 for details) and compute the surface normal vectors  $\mathbf{n}_{p_i}$  at those points. Let  $T^0 = T_0$ .
- 2) At each iteration  $k$  for  $k=1, 2, 3 \dots$ , repeat the following until the process converges (see below for explanation).

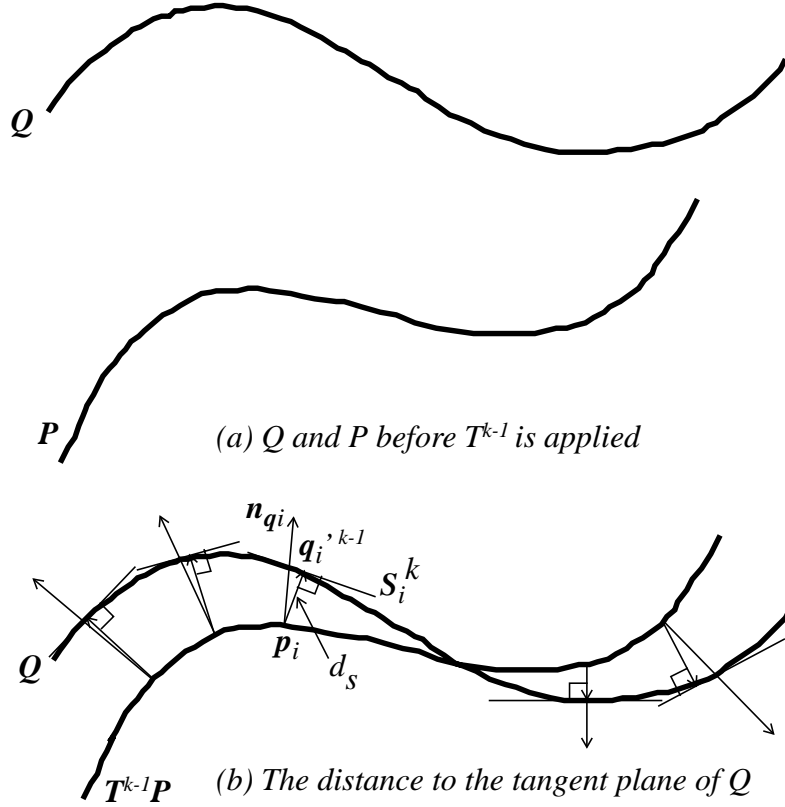


Figure 3.2 Defining the distance measure between 3-D surfaces  $P$  and  $Q$  illustrated in a 2-D case.

where  $d_s(\mathbf{x}, S)$  is the signed distance from a point  $\mathbf{x}$  to a plane  $S$  and  $\mathbf{n}_{q_i}^k$  is the surface normal vector of  $Q$  at  $\mathbf{q}_i^k$ , and “ $\cdot$ ” stands for scalar (dot) product of vectors. This way, we do not work with specific correspondence point pairs, so we can achieve faster convergence. In fact, by minimizing the distance from a point to a plane, we only constrain the direction in which this distance can be reduced. It has two other degrees of freedom (i.e., inside the plane perpendicular to the said direction), along which it can move in accordance with the constraints imposed by other points and planes. Thus, optimization (in our case, minimization of the sum of distances) can be achieved more quickly, which has been verified in our experiments and in comparison with the results of others. This idea is an extension to that used by Lowe [45], who minimized point-to-line distance in his object recognition system SCERPO.

In the following, we first present the complete iterative registration algorithm. Then the line-surface intersection algorithm and the control point selection strategy will be discussed.

With this approach, however, we need to perform the minimization on a surface in discrete format (i.e., a range image) to find  $\mathbf{q}_i^k$ , as is usually the case. If we use an approximation point instead of the  $\mathbf{q}_i^k$  defined in Equation (3.7), the problem becomes easier. Potmesil [54] has used the distance between the surfaces in the direction normal to the first surface as a registration evaluation function. Following this idea, we have

$$e^k = \sum_{i=1}^N \left\| T^k \mathbf{p}_i - \mathbf{q}_i^k \right\|^2, \mathbf{q}_i^k = \left( T^{k-1} l_i \right) \cap Q \quad (3.8)$$

where  $l_i = \{ \forall \mathbf{x} \in \mathfrak{R}^3 \mid (\mathbf{p}_i - \mathbf{x}) \times \mathbf{n}_{\mathbf{p}_i} = 0 \}$  is the line normal to  $P$  at  $\mathbf{p}_i$ ,  $\mathbf{n}_{\mathbf{p}_i}$  is a unit vector normal to  $P$  at  $\mathbf{p}_i$  and  $(Tl_i) \cap Q$  stands for the intersection point of line  $l_i$  (after being transformed by  $T$ ) with surface  $Q$ ,<sup>1</sup> and “ $\times$ ” stands for vector product. The common problem with the above two iterative methods is that, at each iteration, they work with certain pairs of “control points” that do not necessarily correspond to the same points on the actual object surface. The consequence is a slower convergence, since the constraints imposed by different pairs of such control points are mutually incompatible until the state of registration is reached.

Our approach is to approximate  $Q$  using its tangent plane  $S_i$  at  $\mathbf{q}_i$  as shown in Equation (3.6). Equation (3.6) can then be written as:

$$e = \sum_{i=1}^N \left\| T \mathbf{p}_i - \mathbf{q}'_i \right\|^2, \mathbf{q}'_i = \mathbf{q} \mid \min_{\mathbf{q} \in S_i} (\| T \mathbf{p}_i - \mathbf{q} \|) \quad (3.9)$$

where  $S_i$  is the tangent plane of  $Q$  at  $\mathbf{q}_i$ . As mentioned earlier, we do not know where  $\mathbf{q}_i$  is. But if we have an initial  $T^0$  as mentioned above, we can start an iterative process as an approximation. In this case we can use the  $\mathbf{q}_i^k$  defined in Equation (3.8) as an approximation to the  $\mathbf{q}_i$  at each iteration. Since the distance from a point to a plane can be expressed as a linear function of the coordinates of the point, the iterative procedure can be formulated as follows (see Figure 3.2(b)):

$$e^k = \sum_{i=1}^N d_s^2(T^k \mathbf{p}_i, S_i^k) \quad (3.10)$$

with

$$S_i^k = \{ \forall \mathbf{x} \in \mathfrak{R}^3 \mid \mathbf{n}_{\mathbf{q}_i}^k \bullet (\mathbf{q}_i^k - \mathbf{x}) = 0, \mathbf{q}_i^k = \left( T^{k-1} l_i \right) \cap Q$$

---

1. Here we always choose the intersection point closest to  $P$  if there are more than one.

they are approximately registered. The purpose of doing this is twofold. First, we argue that the goal of the surface registration algorithm is to find a finer, more accurate transformation between different descriptions/views of an object surface and that the initial approximate transformation can be obtained through high level matching or through the knowledge of the geometry of the data acquisition setup. Second, since we are not sure that a global minimum for  $D(P, Q)$  can be found in general, a good starting point is very important. In the following sections we will discuss how the ideas in this section are implemented and how we can achieve registration without knowing the function  $f$ .

### 3.3.2 Registration Evaluation Functions

According to our definition of surface registration, if we know a set of  $N$  pairs of corresponding points, called control points, in two views,  $\mathbf{p}_i \in P$  and  $\mathbf{q}_i \in Q$ ,  $i=1\dots N$ , we can easily find the transformation by minimizing the following (see [61] for a review of general methods):

$$e = \sum_{i=1}^N \|\mathbf{T}\mathbf{p}_i - \mathbf{q}_i\|^2 \quad (3.5)$$

when the set size  $N$  is larger than 3. Unfortunately, this point to point correspondence information is difficult to obtain especially for non-structured surfaces, since surface features cannot be detected reliably and located accurately. Another approach to this problem is to minimize the distances from points on one surface to the other. That is, we minimize

$$e = \sum_{i=1}^N \|\mathbf{T}\mathbf{p}_i - \mathbf{q}_i\|^2, \mathbf{q}_i = \mathbf{q} \mid \min_{\mathbf{q} \in Q} (\|\mathbf{T}\mathbf{p}_i - \mathbf{q}\|) \quad (3.6)$$

This follows directly from Equation (3.1), since if  $\min_{\mathbf{q} \in Q} (\|\mathbf{T}\mathbf{p}_i - \mathbf{q}\|) = 0$  for all  $i=1 \dots N$ ,  $e$  in Equation (3.6) will be zero. This is the formalization used by Besl and McKay [5] and Zhang. This is very difficult to implement in our case given the form of input we have (raw range images), since finding  $\mathbf{q}_i$  is an optimization problem itself [5]. Zhang [84] used  $k$ -D tree [56] to speed up the closest-neighbor problem. Besl and McKay [5] assume that an analytical surface representation for  $Q$  is available. Approximations of the algorithm by an iterative method can be used if we know an initial transformation  $T^0$  that brings  $P$  to near registration with  $Q$ . In this case, at each iteration  $k$ , we use the previous value  $T^{k-1}$  to find  $\mathbf{p}_i^k$ :

$$e^k = \sum_{i=1}^N \|\mathbf{T}^k \mathbf{p}_i - \mathbf{q}_i^k\|^2, \mathbf{q}_i^k = \mathbf{q} \mid \min_{\mathbf{q} \in Q} (\|\mathbf{T}^{k-1} \mathbf{p}_i - \mathbf{q}\|) \quad (3.7)$$

### 3.3 Registration as Minimization

#### 3.3.1 Problem Definition

Intuitively, we say that two views of a surface are registered if they coincide when one view is placed at a proper position and orientation relative to the other (here we use the term “view” to mean the rendered representation of the object surface from a specific viewpoint). More precisely, two views of a surface are said to be in registration when *any* pair of points  $(\mathbf{p}, \mathbf{q})$  from the two views representing the same surface point can be aligned by a rigid spatial transformation. That is, there exists a rigid transformation  $T$ , such that

$$\|T\mathbf{p} - \mathbf{q}\| = 0, \forall \mathbf{p} \in P, \exists \mathbf{q} \in Q \quad (3.1)$$

where  $P$  and  $Q$  are two views of the same surface,  $T\mathbf{p}$  is the result of applying transformation  $T$  to  $\mathbf{p}$  and  $T$  is a rigid spatial transformation, which can be expressed in matrix form as follows in homogeneous coordinates:

$$T = T(\alpha, \beta, \gamma, t_x, t_y, t_z) = \begin{bmatrix} c\beta c\gamma & s\alpha s\beta c\gamma + c\alpha s\gamma & -c\alpha s\beta c\gamma + s\alpha s\gamma & t_x \\ -c\beta s\gamma & -s\alpha s\beta s\gamma + c\alpha c\gamma & c\alpha s\beta s\gamma + s\alpha c\gamma & t_y \\ s\beta & -s\alpha c\beta & c\alpha c\beta & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

where  $\alpha$ ,  $\beta$  and  $\gamma$  are rotation angles about the  $x$ ,  $y$  and  $z$  axes respectively,  $t_x$ ,  $t_y$  and  $t_z$  are the translation components, and  $sx$  and  $cx$  are short form of  $\sin(x)$  and  $\cos(x)$  respectively. In general, the transformation  $T$  needed to bring the two views into registration has 6 degrees of freedom. Thus, the task of registration is actually to search for such a transformation in the transformation parameter space, so that Equation (3.1) is satisfied. In practice, the problem can be expressed as solving  $T$  to minimize the following measure:

$$D(P, Q) = \sum \|T\mathbf{p} - f(\mathbf{p})\| \quad (3.3)$$

where  $f$  is a correspondence mapping function:

$$f: P \rightarrow Q \mid \forall \mathbf{p} \in P, f(\mathbf{p}) \in Q \quad (3.4)$$

The difficulty in solving for  $T$  is that the process is highly nonlinear [4] when  $f$  is unknown. Furthermore,  $D(P, Q)$  may not be a convex function of  $T$  in general and there is no guarantee that a global minimum can be reached by an iterative procedure. Potmesil [54] used a heuristic search in the transformation parameter space to match surfaces, but the second problem mentioned above still exists. In fact, there are other ways to help reduce our search effort. Our approach is based on the assumption that an *approximate* transformation between two views is known before hand, i.e.,

algorithm to eliminate gross outliers to make the estimation of the transformation more robust. An initial estimate of the transformation is assumed.

Delingette *et al.* [22] reported an approach for matching two sets of descriptions of an object based on the so-called Simplex Angle Image (SAI) representation. The SAI representation is a semi-regular discrete mesh tessellating the unit sphere, with nodes representing certain surface elements of a genus 0 type object. The mapping between the original object description and the SAI mesh nodes is achieved by deforming an initial mesh to fit the object surface under certain local regularity constraints. The matching is done in rotation first for the two SAIs by minimizing a distance measure between the two SAIs through searching the 3-D rotation space using a similar scheme as Besl and McKay [5]. The true rotation and translation between the two inputs can then be found by mapping the matched mesh elements back to the original inputs to get the true correspondence sets. The difficulty in this approach is in constructing the necessary SAI representation and in guaranteeing its local regularity property especially when only partial object surface data is available from one range image, which directly affects the final result of matching.

Horn and Harris [34] address the problem of estimating the rigid body motion of an observer, given sequentially digitized range image frames of some terrain environment from the observer. They describe a range rate constraint equation and an elevation rate constraint equation. They obtain the 6 motion parameters with a closed form least-squares method under the assumptions that the motion between frames is small and the underlying surface is differentiable. Since this approach is based on minimizing viewer-dependent measures, iteration is still needed to achieve best results and some system error is also expected under noisy conditions.

In contrast to the methods discussed above, where known correspondence between two sets of input is not required beforehand, there is much more work done in motion estimation between two sets of 3-D input (mainly range images) based on features (points, line segments and surface structures) extracted from the input. Sabata and Aggarwal [61] presented an extensive review of this type of methods and we only briefly outline the key points here. Assuming that the correspondence between the feature sets is given, the main point of these methods is to derive the motion parameters (3 rotations and 3 translations) efficiently and accurately. All these methods formulate the problem in terms of minimizing a certain distance measure in one way or another, with various representation schemes for the motion parameters. The solutions can be linear or nonlinear depending on the representations used and other constraints. The main problems of these approaches lie in the fact that the accuracy of the motion parameters are directly related to the localization accuracy of the features themselves, which is prone to noise and far from being solved, and in establishing the correspondence between the two feature sets in the first place.

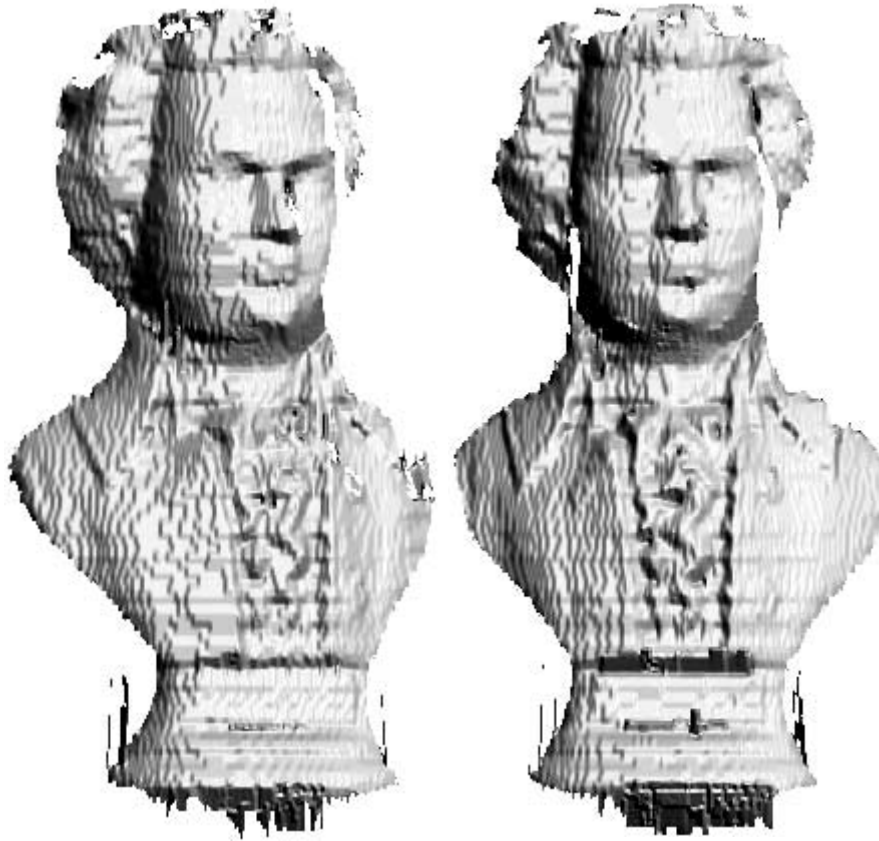
simple, compact objects. Section 3.5 explains the registration strategy for complex objects. A summary is given in Section 3.6.

## 3.2 Previous Work on Registration

As mentioned earlier, to register two sets of 3-D input is to find the transformation or motion between them. So, registration is synonymous to pose estimation, motion estimation and to matching to some extent. In this section, we will use these terms interchangeably unless specified otherwise.

Potmesil [54] developed a system for modeling complete surface of an object by taking multiple range images and then matching (i.e., registering) them through heuristic search in the transformation parameter space. Although his matching technique is quite general, we feel that searching through the huge parameter space, even with some heuristics, is neither computationally tractable nor necessary. Kamgar-Parsi *et al.* [37] developed a range image registration method based on matching 2-D elevation contours for ocean floor mapping. Their method can only handle 2-D transformations between range images to be registered.

Besl [4] presented a unified formulation for the so-called “free-form surface matching” problem and suggested possible ways to it with the conclusion that reliable detection of generic surface points, curves or regions are essential to the matching problem. Besl and McKay [5] presented some implementation of the approaches proposed in [4]. Their approach is formulated as the minimization of the distance from a set of 3-D points, called data point, and a set of corresponding 3-D points (model points) from a model surface iteratively. For each data point, the corresponding model point is defined as the point on the model surface which is closest to the data points, hence the algorithm is called “iterative closest point” or ICP algorithm. They proved that the algorithm must converge to some local minimum. To increase the chance of finding the global minimum, initial guesses of the parameters are selected at equally spaced sample points over the entire 6-D transformation parameter space to search for the global minimum, which is still not guaranteed. The method also needs an efficient algorithm to find the corresponding point for a data point. Another important drawback of this method, as will be discussed later in this chapter, is the use of minimization based on the “corresponding points” (they are actually not true corresponding points), which results in a very slow convergence as reported. Zhang [84] proposed a system similar to that of Besl and McKay’s, in which at each iteration, a modified closest-point algorithm is used to establish correspondence between two sets of 3-D data (for surface matching) and a quaternion-based least-squares algorithm is used to estimate the 3-D transformation between the two sets of input. During each iteration, certain constraints are applied to the closest-point



*Figure 3.1 Two range image views of the Mozart bust shown as shaded intensity images. The image sizes are  $217 \times 422$  and  $221 \times 420$  respectively.*

be called “pose estimation without correspondence”. Our registration method avoids the search through the transformation parameter space by assuming an initial approximate transformation for the registration algorithm, since we believe that this information is available from the range finder setup (e.g. rotation on a rotary table) or through high level feature matching, e.g., [69], [50] and [23]. Figure 3.1 presents two range images taken of a Mozart bust in Cartesian image format (also called depth map or graph surface), which show the type of complex surface that our registration algorithm can handle. For display purposes, the range images are shown in shaded intensity image form computed from floating-point values (the pixel values of the shown images are proportional to the scalar product of the normal to the underlying surface at the point and the directional vector of a predefined light source).

In the following sections, we first review some previous work on range image registration (Section 3.2), then we introduce the registration algorithm and show some test results (Section 3.3). In Sections 3.4 we present a method to register the range image views globally for modeling

## 3

# Range Image Registration

### 3.1 Introduction

As discussed in Chapter 1, if one needs a complete model of an object, the following steps are necessary:

- 1) data acquisition,
- 2) **registration between views**, and integration of the views,
- 3) description.

The goal of registration is to find the spatial transformation between the range images taken of an object at different view points, so that the points found in different range image views (“views” for short hereafter) that represent the same surface point are “aligned”. This is also known as solving the *correspondence* or *pose estimation* problem. Many object recognition systems also solve the problem of obtaining a transformation between the scene and the model for pose estimation and/or as a verification ([69], [50], [23], [12] and [25]). Because all of these systems rely on the correspondence of some features, the accuracy of the obtained transformation depends on the localization of the features (see [61]), which is prone to noise. The registration problem is also important because it belongs to a class of very important problems known as “free-form surface matching” problem (see Besl [4]), which has applications in shape matching, inspection, medical image analysis, terrain modeling and navigation.

In this chapter, we present an iterative algorithm for range image registration. Our algorithm iteratively minimizes a distance measure which measures the closeness between the two views. Unlike most other pose estimation techniques used in the literature, ours does not require point-to-point correspondence, since we minimize the distance from points to planes, which are first-order local approximations of the corresponding surface, and the correspondence is thus achieved implicitly. An added benefit of this is that the convergence of the registration process is faster than those using point-based registration without correspondence (e.g. [84] and [5]). Our method can

## 2.6 Summary

From the above discussion we can conclude that global shape models do not carry enough degrees of freedom for describing shape details. Deformable models and dynamic mesh models, though powerful and flexible, have serious restrictions in formulating the system so that a global optimum is always achieved, and hence must rely on good initial guesses or need human intervention. They also have to address the selection of system parameters. Triangulation methods have been shown to be successful in describing complex shapes but with some restrictions on the sampling of the object surface. Few have addressed the problem of integrating multiple densely sampled range images, where too much input data may pose a serious problem regarding to time and space complexities as well as the issue of noise.

Besides superquadrics as the parametric model, researchers have used implicit algebraic functions for shape description [71]. In addition to the aforementioned drawbacks with other global shape models, this approach also suffers from instability and unboundedness problems [70].

## **2.5 Other Approaches**

There are many other shape description approaches developed by vision researchers that may be used for model construction purposes. But for the following reasons we only give a brief review here, and interested readers should refer to the cited references for details: 1) these approaches are only suitable for specific types of object, 2) they have not so far been demonstrated to be advantageous in describing complete object surface shapes and 3) segmentation is necessary before description is constructed, which is not preferred in our approach as has been discussed in Chapter 1.

### **Generalized Cylinder (GC)**

Shape description using generalized cylinders (also known as generalized cones) was proposed by Binford [8] in the early 70's. A GC is a volumetric shape primitive defined by sweeping a closed planar curve (called cross-section) along a 3-D space curve while changing its shape and size. Since the model is very general, the representation is generally not unique. More constrained versions of GCs are often used (see [48], [14] and [57]). Some form of GCs can be recovered from the contour of the object with certain assumptions on the objects or imaging models (see [53], [28], [42], [76] and [83]). In short, GC is suitable for describing certain regular shapes such as cylinders or cones precisely or approximate shapes that are close to these shapes. It is not for describing any arbitrary shape precisely.

### **Shape from Multiple Intensity Images**

Using multiple intensity images, it is possible to estimate dense, but mostly sparse 3-D information of points of an object in the scene, using either stereo (a pair of images) [2] or motion (a sequence of images of a moving object or camera) [36]. Although the results of the systems based on these methods can be used for shape description purposes, they themselves do not make full description of the shape. Recently, there have been developments in recovering local surface shape descriptions using the apparent contour images of an object (see [40], [77] and [9]). There have also been reports of using these methods for global shape descriptions (see [65] and [41]) using calibrated image sequence and purposive vision. But much work still needs to be done before they can actually be used for constructing models for arbitrary objects.

local surface shape. The system is described by the governing equation of motion, and solved using a time-integration technique for differential equations. In [78], this approach is extended by introducing an attraction force from 3-D inputs and a dynamic mesh subdivision scheme to describe 3-D shape with scattered 3-D points. Huang and Goldgof [35] also proposed a similar system with dynamic nodal addition/deletion for shape description and nonrigid object tracking.

One common problem for the deformable models is that they rely on a carefully selected initial state for the system to converge to the desired solution. This is because these systems rely on the establishment of the correspondence between the elements in the model and those on the object surface, which is dynamically established and can only be approximated when the two are relatively close. Another drawback of these systems is that the system parameters are usually very difficult to define and need to be tuned to suit different objects.

## 2.4 Parametric Models

Parametric models are global shape models, therefore they are usually not good for describing details of the shape of an object unless appropriate segmentation is performed and the resulting parts are described separately.

Superquadric models [3] are one of the commonly used parametric models. One of the problems with fitting superquadrics to 3-D input has been to choose a meaningful and yet easy-to-evaluate distance function, with which a least-squares based minimization can be formulated for the fitting process. This is because a closed-form Euclidean distance measure between a point in space and the model surface does not exist. Boulton and Gross [13] and Gupta *et al.* [30] have studied alternatives to such a distance measure. The simplest one is to use the so-called inside-outside function based on the definition of the superquadric models. The inside-outside function is larger than 1 if a given point  $(x, y, z)^t$  is outside the volume, and smaller than 1 if it is inside. By minimizing the mean-squares value of the difference of this inside-outside function and 1, we can recover the parameters of the superquadric model. The problem with this approach is that the inside-outside function is not uniformly weighted over the entire parameter space. Nevertheless, this works for simple, compact objects.

In order for the superquadrics model to accommodate more complex shapes, Solina and Bajcsy [67] introduced superquadrics models with global deformations, which include bending, tapering and cavity. Another example of introducing deformation to superquadrics is the work of Terzopoulos and Metaxas [72] we discussed in Section 2.3 on page 13. Pentland [52] and Ferrie *et al.* [26] use superquadrics for obtaining segmented description from range data. Gupta [31] also uses superquadrics to segment range images into convex volumes by using multiple representations and hypothesis generation and verification.

functions represent the stiffness and smoothness of the model shape, and the external energy functions represent the mismatch of the model shape to the input data, which is often represented as a potential field around the input data points and is inversely proportional to the distance to the data. 2-D deformable shape models are called “snakes”, after Kass *et al.* [38][39], because of their behavior. Terzopoulos *et al.* [74] used a 3-D version of such deformable models to reconstruct GC-like (Generalized Cylinder, see [8] and [48] and also Section 2.5), symmetry-seeking tube models from object contour information, by combining a deformable tube with a deformable spine. An instance of the model is reconstructed by combining the internal energy constraints of the spine with symmetry-based constraints, plus the external energy constraints that deform the model so that the projection of the model is consistent with the 2-D silhouette in the image. This method relies on a good initial guess of the model, which often requires human interaction.

Delingette *et al.* [21] used a 3-D deformable model to describe the shape of compact objects for modeling and collecting pebble samples using range images. The system combines both data and feature forces to accommodate both local and global deformations at the same time. The feature force comes from geometric features extracted from the input and acts as a global attraction force, which decreases over time. The data force has limited active range and therefore can deform the surface shape locally when the surface comes close to the data. The surface is represented as a discrete sphere initialized at the center of the object and gradually deformed over time under the attraction of the data points and features.

Terzopoulos and Metaxas [72] combined local deformation with the global deformation of a superquadric model (see Section 2.4 for more on this) to recover 3-D shapes. They formulate their approach as a dynamic system which models the physical properties of the object surface in terms of kinematics, dynamics and elasticity. The system is able to capture both global and detailed local shape using such a scheme. A similar deformable model using a finite-element model is used by McInerney and Terzopoulos [46] to describe the shape and track the motion of the left ventricle (LV) from computer tomography (CT) image sequences.

Besides the approach mention above, there are other deformable models using B-splines, which differ in formalization of internal and external energy terms and in implementation schemes (see [29] and [43]).

Another type of deformable shape model is called dynamic mesh. This type of model simulates the model surface with a set of nodes with a certain mass and the neighboring nodes are linked to each other with virtual springs to simulate surface tension. Terzopoulos and Vasilescu [73] introduced a shape reconstruction algorithm using a dynamic mesh that can dynamically adjust its parameters to adapt to the input data. The mesh is represented by a mesh of nodes connected to their neighbors with adjustable springs. The spring stiffness changes based on the properties of the input, e.g., curvatures, which allows the mesh to adapt to the complexity of the

points from the object surface. In the surface-based approach, a set of nearest neighbors of each data point is found, and then a local surface triangulation is constructed and extended iteratively by projecting the neighborhoods into their tangent planes and then constructing triangles in 2-D. In the volume-based approach, the 3-D Delaunay triangulation (3-D volumes instead of triangles, see [81]) of the set of surface sample points is constructed, which forms the convex-hull of the set of points. An iterative elimination algorithm subsequently deletes any resulting tetrahedra that do not belong to the object volume, resulting in a volume representation of the object with its boundary being the surface of the object. However, these approaches require strict dense sampling of the entire object surface, which may be violated in practice, otherwise the result may not correspond to the actual object. Another drawback of these approaches is the lack of consideration of noise. A specialized and more constrained application of the volume-based approach was also reported by Boissonnat [11] for constructing triangulations from a series of 2-D cross sections.

Hoppe *et al.* [33] also proposed a method to construct triangulated description from scattered 3-D points. In their approach, the neighborhood of each 3-D point is used to estimate the tangent plane of the surface at the point and a tree traversal is then carried out to make consistent labeling of the orientation of the tangent planes across the entire surface. The local tangent planes define an implicit surface which is then used to construct a triangulation using the “marching cubes” algorithm (see [82]). Their algorithm is supposed to be able to handle objects of arbitrary topology, but the difficulty in the consistent orientation labeling involving surface normal discontinuities seems to be the biggest problem for it to work in practice.

Soucy and Laurendeau [68] presented a system that builds surface descriptions from multiple registered range images. Their system first compute the intersection among the range image views, from which a hierarchical Venn diagram structure is constructed that divides each range image into disjoint regions that are exclusively shared among a subset of the range images, called canonical subsets (CS). Surface triangulations are then constructed for each CS and then merged to obtain an integrated representation. The system is also claimed to be able to handle objects of arbitrary topology, yet no result for a complete surface model is shown. One major disadvantage of the system is the complexity of the algorithm in constructing the CSs, which is exponential, which prohibits us from considering more than a dozen views. The system also requires segmentation for each range image along depth discontinuities.

## 2.3 Deformable Models

Deformable shape models are introduced by Kass *et al.* [38][39]. The deformable models are formulated by using variational principles with shape constraints. Shape description is done by globally minimizing the internal and external energies of the model shape. The internal energy

Delingette *et al.* [22] proposed a system using the so-called Simplex Angle Image (SAI) representation (after EGA, Extended Gaussian Image), which maps any genus 0 type object surface onto a unit sphere. SAI representations of the range image views of an object are constructed using a deformable model technique [21] with constraints on the mesh shape regularities. The range images are then registered (see Section 3.2 on page 19 for a more detailed discussion) using their SAI representations. A final SAI of the object is constructed using data from all the views after transforming them into a common reference frame. This technique depends on the specific representation, which should still be considered a research problem itself.

So far, the only system that achieves high level description from multiple range image views is that of Parvin and Medioni [51]. They developed a system using unregistered multiple range images to model objects with planar and quadric surfaces. In their approach, both convex and concave edges are extracted and vertices, edges and surfaces are recovered for each range image view to form an attributed graph. Matching is done using the graph representation with a two-level constraint satisfaction network with local and global constraints [50]. A boundary representation (B-rep) of the object model is constructed by merging the attributed graphs and computing the intersection of surface patches.

## 2.2 Triangulation

There are two types of method which are based on surface triangulations for surface descriptions. The first one is based on local surface triangulations and the second explores theories in computational geometry for the construction of triangulations both in 2-D and 3-D.

To triangulate a surface represented in range image means to construct a triangle mesh that approximates the actual surface represented by the range image. Floriani and Puppo [27] introduced an algorithm to construct such a triangulated approximation based on planar Delaunay triangulation (see [56]). A Delaunay triangulation approximation of a range image is a triangulated approximation whose projection on to the image plane is a Delaunay triangulation. Their algorithm first constructs a Delaunay triangulation over the 2-D image domain. The initial triangulation is then iteratively refined by inserting vertices for points that exceed the allowed approximation error threshold. The reason for using the Delaunay triangulation is that it is optimal in the sense of equal-angularity for the resulting set of triangles. Actually, this property is only true for a planar triangulation, but enforcing optimality of the triangulation in 3-D is very difficult. Schmitt and Chen [64] used a similar algorithm to construct triangular approximations of range images for segmentation purposes.

Boissonnat [10] introduced a surface-based triangulation scheme and a volume-based triangulation scheme for construction of geometric models of an object from a set of scattered

representation is then converted into a triangulated surface approximation. This approximation is further refined by examining and reducing the intensity difference between observed and reconstructed images. There have also been efforts trying to build surface models from multiple views directly. Vemuri and Aggarwal [79] use range images taken of an object on a rotary table. Intensity images are also taken at the same time to cover the patterns drawn on the base-plane on the rotary table, and are used to compute the inter-frame rotational transformation by matching the image of the patterns in two adjacent views. The range images are then transformed and integrated into a single object-centered cylindrical image representation. Wang and Aggarwal [80] developed a system that integrates both active and passive sensing techniques. In their system, a structured light pattern is projected onto the object and the image is used to extract the local surface structure and relative depth information. Such information from multiple views is integrated with passively sensed occluding contour from multiple views of the object to constrain the actual 3-D position of the object surface. They used the same technique as Vemuri and Aggarwal to obtain the inter-frame rotational transformations. The resulting surface is represented as a set of cross section contours. The above methods are only able to obtain wraparound data of the object surface, because they rely on the object being rotated on a rotary table.

Potmesil [54] studied the problem of building surface models using general viewpoint range images. A hierarchical quadtree representation of the surface patches is used for the surface segments from each view. The surface segments from all the views are transformed into a common 3-D space by a surface matching procedure and then their overlapping sections are eliminated by a merging procedure. The matching is done by a heuristic search algorithm through the 6-parameter transformation space for a registration transformation, which will spatially “align” two overlapping surface segments. The search is guided by minimizing distances of certain shape features on the surface segments. The match-and-merge process is repeated for all 3-D surface segments until a complete model for the object is generated. Test results for simple compact objects are shown.

Bhanu [6] constructed a surface representation from multiple range images for object recognition. In his system, inter-frame transformations are either known (from object rotation) or are computed by matching special control points on the object. 3-D surface points from individual views are transformed into a single coordinate system, and a 3-point seed algorithm is used to extract points belonging to the same planar faces. These points are then grouped and approximated by using planar polygons. The 3-point seed algorithm used is very inefficient since it does not use the topological information that is present in the range image in computing the polygonal approximation of the surface, which results in a high time complexity ( $O(n^2 \log n)$ ), where  $n$  is the number of data points in the input.

## 2

# Object Modeling: A Review

In this chapter, we review some of the previous researches on shape description and object modeling related to our own. The emphasis is on describing the complete shape of an object from one or more sets of input. Range image registration techniques are reviewed in Chapter 3. The review is conducted in the following sections: low level (Section 2.1), triangulation based method (Section 2.2), deformable models (Section 2.3) and other approaches (Section 2.5).

### 2.1 Object Modeling Using Multiple Views: Data Integration

Some previous research efforts have used multiple views for object modeling. These methods actually achieve low to mid-level descriptions and thus can be better categorized as integration methods than description methods. Two issues need to be solved in integration. One is the inter-frame transformation between the views, since the data from different views are often stored in different coordinate frames. The second is the representation for the integration. Most of the methods reviewed below either assume the inter-frame transformations to be known, or try to recover them through various methods. The output of the system is represented either by volume octrees or a low to mid-level surface descriptions.

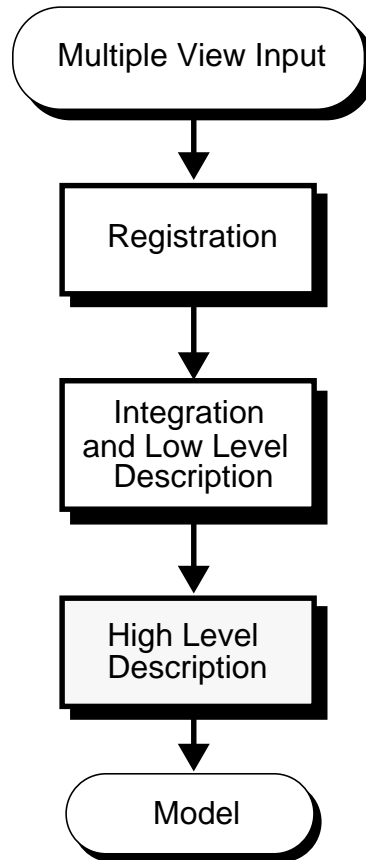
Ahuja and Veenstra [1] studied the problem of constructing octree models for 3-D objects using object silhouettes from orthogonal views with orthographic projections. Their method constructs an octree by computing the intersection of the projection cylinders of the object silhouettes. Potmesil [55] presents an improved algorithm which allows arbitrary viewpoint and assumes perspective projection. His algorithm is based on recursively projecting the octree nodes into the image plane to determine the value of the node. The inter-frame transformations are found by tracking the images of special markings on the objects. Because these methods use object silhouettes, they can not handle objects with concavities in general. Range images can be used in this case to help reduce the ambiguities associated with intensity images [19]. Liedtke *et al.* [44] also constructed a voxel representation from registered multiple silhouette images. The voxel



- We have developed a system for constructing a triangulated surface description for complex, non-star-shaped objects from multiple registered range images. The system is based on a dynamic mesh surface model which starts as a tiny sphere, and evolves over time under an inflation force and finally reaches the object surface, which accomplishes the mapping of a unit sphere to the object surface. The system addresses the description and the data integration at the same time, and has many advantages over current dynamic systems or deformable models based on global minimizations.
- We have developed a new scheme to evaluate triangle fitting errors in the context of data integration for merging multiple range images.

### **1.3 Dissertation Organization**

The dissertation is organized in two parts. The first part deals with range image registration (Chapter 3) and the second part deals with constructing triangulated surface models for simple (Chapter 4) and complex objects (Chapter 5). A survey of related research in object modeling is presented following this chapter and before the main parts. Most of the material presented in this dissertation has been published before (see [15], [16], [17]and [18]).



*Figure 1.4 Proposed object modeling system flow chart, the shaded part are not covered in this dissertation.*

## 1.2 Contributions

Here is a summary of contributions of this dissertation:

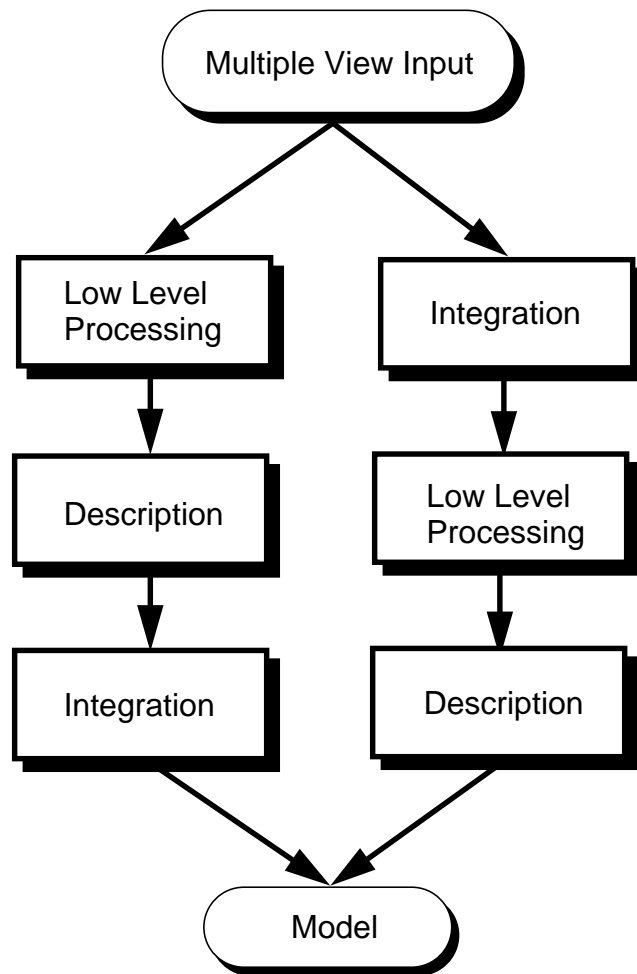
- We have developed a system for range image registration. The system is based on minimizing an intrinsic distance measure between the surface segments represented by the two range image views. The formalization of the least-squares results in faster convergence than other methods based on point correspondence reported in the literature. The algorithm does not rely on feature extraction and is suitable for any object with piecewise smooth surfaces.

object surface, and we also need to reason about the ambiguities and the possible conflicts resulting from the incomplete, and possibly wrong, segmentation results from the lower levels when integrating them at high level. The accuracy of the integration is another problem for this approach, because some information is lost during the processing when we go to a high level to perform the integration, as is done by Parvin and Medioni [51].

The second method has the advantage of being able to use all the surface information for description, therefore it is much easier for us to take a more geometrical approach than a “vision” approach. Another advantage of the second method is that since the integration is done at low level, we have a better chance to reduce the effect of sensor noise and quantization errors during the data merging process. The difficulty of this approach lies in choosing an appropriate representation scheme for the integration and in achieving it. As will be shown in the coming this dissertation, it is relatively easy to perform data integration at low level for simple objects, e.g. in a spherical coordinate system. However, it is in general very difficult to find an appropriate parameterization to represent the entire surface of an arbitrary object (we only consider objects with smooth surfaces). The common approach in previous researches for shape description is to formulate the problem in terms of a minimization process of fitting certain shape model to the input data, which is not guaranteed to result in a global optimum and hence often requires a good initial approximation.

Based on the above analysis, our approach is to combine the integration process and the description process into one, as illustrated in Figure 1.4. We use a triangulated approximation as shape representation to perform data integration. The reasons for using an intermediate representation based on triangulation is, first, we think a triangular surface approximation presents a good trade-off between high and low level representations and is also a powerful representation in describing any complex shapes and theoretically to any degree of accuracy. This allows us not to address the segmentation issue at this moment. Secondly, we believe that, just like in 2-D case [60], an analytical surface representation will be very helpful in the process of segmentation and description both in terms of surface and volume, and a triangulated surface approximation is a first step towards building a smooth analytical surface representation [24].

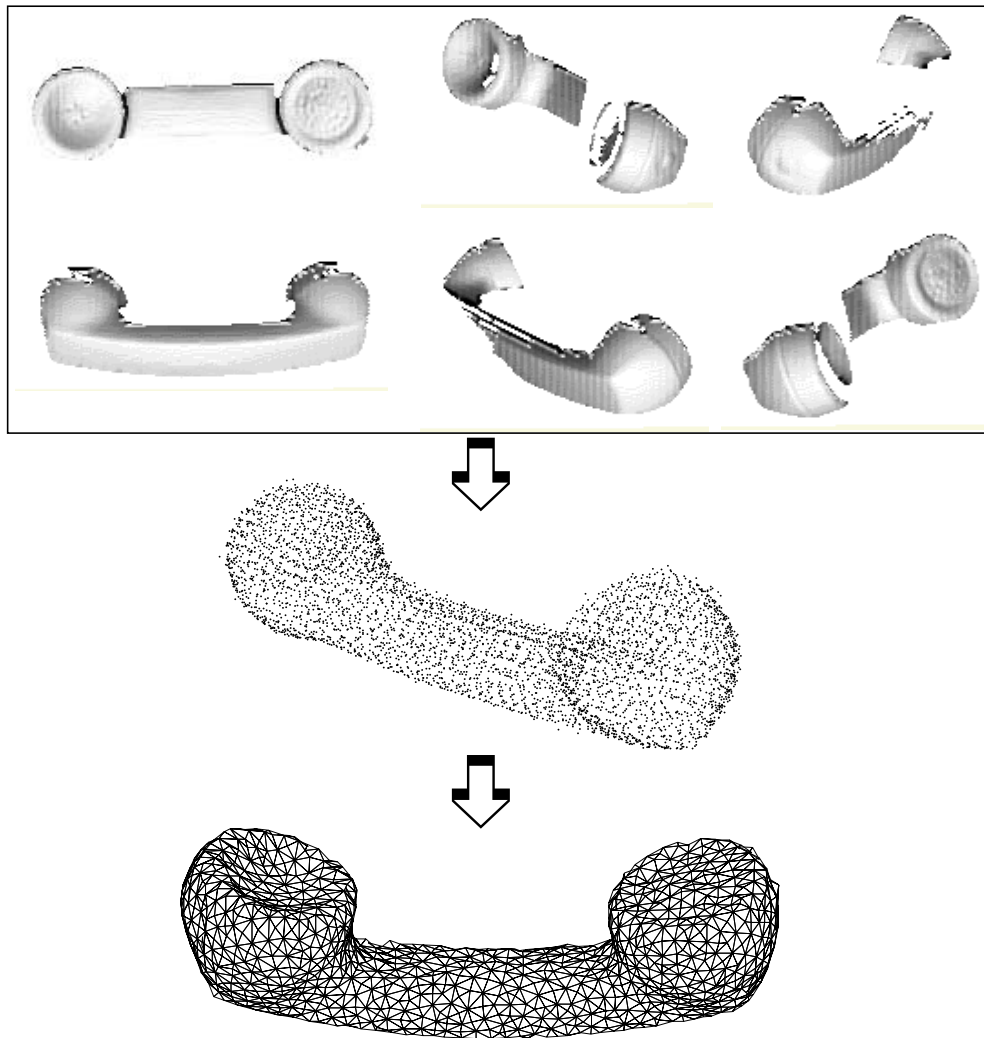
It must be point out that in some situations (e.g. [32]), integration of the registered range images is carried out by simply gathering all the 3-D points from the range image views. In doing so, valuable information in the range images, e.g. the neighborhood information of each range image pixel, is lost, which can be used in the description process. For this reason, we always maintain the range images in their original format. The assumption that the range images are densely sampled allows us to perform local surface approximation with good results except at depth discontinuities.



*Figure 1.3 Two different ways of performing integration and description.*

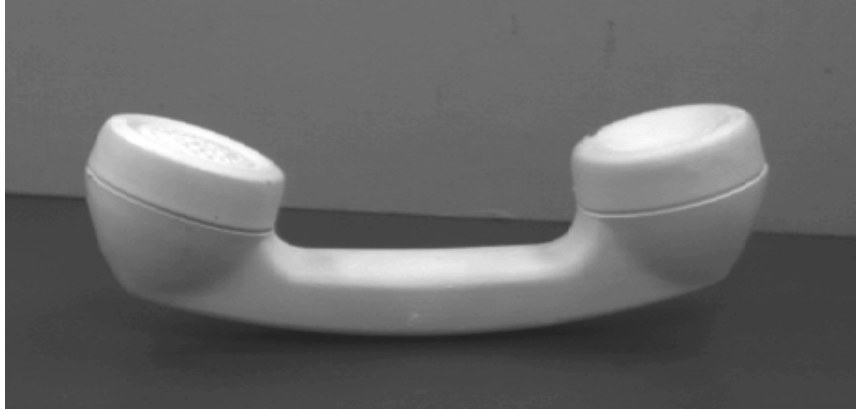
### 1.1.2 Data Integration and Description

As discussed above, due to self-occlusion, multiple range images of the object from a set of vantage points are needed in order to describe the complete surface of an object. There are two possible approaches to solving the shape description problem using multi-view input, as shown in Figure 1.3. One is to describe each view and then integrate the results to obtain a complete description. The other is to integrate the data from the views and then construct the description. The first method has the advantage of being able to use many existing systems design for shape description from single view input. The drawbacks of this method are that single views are inherently ambiguous, therefore we must predict the occluded shape from the visible portion of the



*Figure 1.2 Steps in object modeling: data acquisition, registration and description. Top: the range images taken of the telephone handset shown as shaded images, middle: sample data from the range images after registration, bottom: the constructed surface model shown as a wireframe*

transformation that minimizes a distance measure between a pair of range images. This distance measure is based on the distance from points on the first range image to the first-order approximation of the second surface and hence is an intrinsic measure (invariant to sensor orientation). The system is not designed to solve the general optimization problem and it assumes an approximate initial guess, which in turn may be provided by some pose estimation systems discussed above.



*Figure 1.1 A telephone handset.*

that each range image can only cover the visible portion of the object surface, more than one range images from different viewpoints must be taken in order to describe the complete surface of the object. Once range images are taken, we need to represent the acquired data in a common reference frame before we can start to build the description. In order to do that, we must find the relative positions of the range images by using a process called registration. Once the registration is done, the next step is to construct a description from the set of aligned, or registered, range images. This process is shown in Figure 1.2.

### **1.1.1 Registration**

To register a pair of range images is to find the rigid 3-D transformation between them so that the overlapping areas of different range images covering the same segment of the object surface are aligned with each other. This is the same as finding the motion between the range images or estimating the pose of the object in one range image related to the other [61], except that in feature-based motion or pose estimation systems the accuracy of the registration is not adequate for shape description purposes because of the error introduced in feature detection process. In addition, correspondence between features extracted from the two range images must be established. Thus a registration system that does not rely on feature extraction and correspondence, and can produce accurate result, is desirable. The difficulties lie in the formalization of such a problem as a minimization problem and the design of a working system that can deliver accurate registration results for a usually non-convex minimization problem, which is the topic in the first part of this dissertation.

Our solution to this problem is an iterative registration algorithm that works on raw range images without feature extraction. We formalize the registration problem as finding the rigid 3-D

# 1

## Introduction

We address the problem of building a complete surface model automatically for an existing three dimensional (3-D) object by “looking” at it from various viewpoints. A model of an object is the mathematical description of the entire shape of the object. Such a model is needed in vision systems for tasks such as recognition. It can also be used in robotics systems to carry out planning and manipulation tasks. Constructing computer models from existing objects is a very important technique for applications such as reverse engineering (reverse CAD) and computer graphics (see [4], [66] and [49]).

By “looking” at (as opposed to touching) the object, depending on the kind of sensor we are using, we can get two types of information, visual (intensity) and range (3-D coordinates). There is a whole line of research in vision in developing methods for shape description using intensity images, which includes stereo-, motion- and contour-based techniques. However, these techniques usually need first to extract the corresponding 3-D information from the intensity image before they can actually start to construct any description. Range images, which capture the geometry of 3-D object surfaces as (densely) sampled arrays of 3-D points, are more directly suitable for shape description purpose because they allow us to focus our attention on the actual description of the shape, rather than on extracting the 3-D data from a complex function of 2-D images. Many difficult vision tasks are thus bypassed. The availability of the sensors (e.g. [62] and [63]) and the quality of the data compared to those obtained from intensity images also make range image a good choice.

In the following sections, we first introduce the major steps in constructing complete models of objects using range images and then we give a summary of the contributions of dissertation.

### 1.1 Issues in Object Modeling Using Range Images

Figure 1.1 shows a telephone handset, a typical object with piecewise smooth surface that we would like our system to be able to model. Since range images are just like intensity images in



## ABSTRACT

Building computer models of existing 3-D objects from sensor data is an interesting yet very difficult problem. This research has applications in computer vision research as well as in industry for such purposes as inspection and reverse-CAD. The acquired models can also be used for robotic planning and manipulation. We identify three key steps in the model construction process, which are, 1) data acquisition, 2) data integration and 3) description. We use range images of an object from multiple viewpoints as input to our system since range images provide direct 3-D measurements of the object surface. To integrate the range image views, we develop a new algorithm for range image registration, which can find an accurate rigid 3-D transformation between a pair of range images given coarse transformation estimates. Our formalization of the registration problem is less constrained than the commonly used schemes based on point-correspondence, resulting in a much faster convergence. A global registration scheme is also presented to integrate all the range image views for the description task. Our description scheme can achieve both integration and description at the same time. We use a dynamic triangulated mesh with spring attachment between the vertices as the shape model. We introduce an inflation force inside the balloon-like model initially placed inside the object with a spherical shape. The inflation force expands and deforms the mesh until it reaches the object surface. A dynamic mesh subdivision and a local adjustment scheme are also incorporated so that the mesh eventually fits the entire surface of the object, accomplishing the mapping from a sphere to any complex object surface. The novel point of the approach is that only inflation force is introduced during the mesh deformation process. The problem with other deformable model and dynamic mesh methods, which is the reliance on a good initial guess for the global optimum, is thus no longer an issue. Experimental results for modeling both simple and complex objects from real range images are given.

## LIST OF TABLES

<b>Table 3.1</b>	Registration convergence test: rotation in X axis .....	35
<b>Table 3.2</b>	Registration convergence test: rotation in Y axis .....	35
<b>Table 3.3</b>	Registration convergence test: rotation in Z axis .....	36
<b>Table 3.4</b>	Registration convergence test: translation in X axis .....	36
<b>Table 3.5</b>	Registration convergence test: translation in Y axis .....	37
<b>Table 3.6</b>	Registration convergence test: translation in Z axis .....	37
<b>Table 5.1</b>	Statistics of the test results .....	75

Figure 4.6	Iterative triangle subdivision. . . . .	58
Figure 4.7	Two rendered views of the constructed triangulation of the object “Wood”. .59	
Figure 4.8	Three rendered views of the constructed triangulation of the object “Tooth”. . . 60	
Figure 5.1	Dynamic surface model illustrated. . . . .	62
Figure 5.2	The inflating balloon model as illustrated in a 2-D case (from left to right and from top to bottom): the initial state, intermediate states and the final state. .64	
Figure 5.3	Line-surface intersection: searching for a corresponding point in the normal direction. . . . .	65
Figure 5.4	Subdivision of triangle mesh: (a) before A is subdivided, (b) after A is subdivided and subdivision is propagated to both B and C. . . . .	69
Figure 5.5	Rearranging the local configuration to eliminate long and thin triangles. . . .70	
Figure 5.6	Sample range images used in constructing the Phone model and Renault model in Figures 5.9 and 5.10, shown here as shaded intensity images. . . . .	74
Figure 5.7	Stages of an inflating balloon inside the Phone, showing the movement of the right front only. The wireframes are superimposed with sample points from the input range images. . . . .	75
Figure 5.8	Examples of the balloon model for Wood and Tooth: (a) the wireframes of the obtained balloon models and (b) the rendered shaded images of the models. .76	
Figure 5.9	The final balloon model for the Phone: (a) a wireframe, (b) and (c) smoothly shaded images. . . . .	77
Figure 5.10	A wireframe and a rendered image of the reconstructed model for the Renault automobile part. The small picture in the middle is the intensity image of the actual object. Note that the wireframe is not produced by a true hidden-line removal algorithm. . . . .	78
Figure B.1	The components of the LCRF system. . . . .	94

Figure 3.8 Relationship between Cartesian, cylindrical and spherical coordinate systems..	
39	
Figure 3.9 The wood blob and four of its range images used for model construction. .	.41
Figure 3.10 Constructed model for the wood blob in spherical coordinate image for the wood blob in Figure 3.9, where the horizontal and vertical axes correspond to the q and f angles respectively, while the image value encodes the radius r.	.42
Figure 3.11 The rendered images (a) and the wireframe drawings (b) of the constructed model of the wood blob. ....	.43
Figure 3.12 The original model plaster tooth (a) and the constructed model (b) in spherical image form. ....	.44
Figure 3.13 The rendered images (top and middle rows) and the wireframe drawings (bottom row) of the constructed model for the tooth. ....	.45
Figure 3.14 Sample range images from the teapot. ....	.47
Figure 3.15 Range image slices of a teapot before (a) and after (b) registration with the global registration scheme. The initial positions of the roughly registered slices (a) are computed from the system calibration information.. ....	.48
Figure 3.16 Registration error distribution for the teapot range images before (a) and after (b) registration with the global registration scheme. ....	.49
Figure 4.1 Initializing the ellipsoidal shell from an icosahedron with its faces subdivided.	52
Figure 4.2 Result of projecting the ellipsoidal shell onto the surface of an object. Shown with the wire frames are sample data points from the range image views of the object.. ....	.53
Figure 4.3 A 3-D triangle projected into range image parameter space. ....	.54
Figure 4.4 The problem in defining data points in evaluating triangle approximation error. The data points in the shaded area are not appropriate because of the oblique viewing angle of the sensor relative to the surface patch. The hashed triangle is preferred but cannot be obtained.. ....	.55
Figure 4.5 Triangles are subdivided in groups with the same type of approximation errors. The dashed lines define the new triangles.. ....	.57

## LIST OF FIGURES

Figure 1.1 A telephone handset. . . . .	2
Figure 1.2 Steps in object modeling: data acquisition, registration and description. Top: the range images taken of the telephone handset shown as shaded images, middle: sample data from the range images after registration, bottom: the constructed surface model shown as a wireframe. . . . .	3
Figure 1.3 Two different ways of performing integration and description. . . . .	4
Figure 1.4 Proposed object modeling system flow chart, the shaded part are not covered in this dissertation. . . . .	6
Figure 3.1 Two range image views of the Mozart bust shown as shaded intensity images. The image sizes are 217¥422 and 221¥420 respectively. . . . .	18
Figure 3.2 Defining the distance measure between 3-D surfaces P and Q illustrated in a 2-D case. . . . .	24
Figure 3.3 Intersecting a line with a digital surface illustrated in a 2-D case. . . . .	27
Figure 3.4 The selected control points for Figure 3.1(a). . . . .	30
Figure 3.5 Registration results for the Mozart range images.(In the histogram figure, “u” stands for “mean”, “s” for “standard deviation” and “min” and “max” for minimal and maximal values respectively). . . . .	31
Figure 3.6 Registration results for the model tooth. (Continued on next page.). . . . .	32
Figure 3.6 (continued) Registration results for the model tooth. . . . .	33
Figure 3.7 Coordinate system for the range images used in this dissertation. . . . .	34

<b>APPENDIX A. The Least-Squares Solution for Registration . . . . .</b>	<b>.91</b>
<b>APPENDIX B. Experimental Range Finder System . . . . .</b>	<b>.94</b>

3.3.6	<i>Experiments</i>	29
3.3.7	<i>Stability with Respect to Initial Conditions</i>	33
3.3.8	<i>Comparisons and Analyses.</i>	34
3.4	Integration of Multiple Range Images for Compact Objects	38
3.4.1	<i>Object-Centered Representation.</i>	38
3.4.2	<i>Global Registration</i>	39
3.4.3	<i>Results and Discussion.</i>	40
3.5	Handling Complex Objects	40
3.6	Summary	47
<b>Chapter 4. Surface Level Integration: Simple Objects.</b>		<b>51</b>
4.1	Introduction	51
4.2	Mapping a Triangulated Shell by Projection	51
4.2.1	<i>Initializing and Projecting the Triangulated Shell</i>	52
4.3	Approximation via Triangle Subdivision	53
4.3.1	<i>Triangular Patch Approximation Error Evaluation</i>	54
4.3.2	<i>Triangle Subdivision and Reprojection</i>	57
4.4	Experimental Results	58
4.5	Summary	59
<b>Chapter 5. Modeling Complex Objects</b>		<b>61</b>
5.1	Introduction	61
5.2	Related Work	63
5.3	The Inflating Balloon Model	63
5.3.1	<i>The Correspondence Problem</i>	64
5.3.2	<i>A Simplified Dynamic Model</i>	65
5.3.3	<i>Spring Force and Inflation Force</i>	66
5.4	Subdivision and Adaptation of the Triangular Mesh	67
5.4.1	<i>Adaptive Triangle Mesh Subdivision</i>	68
5.4.2	<i>Local Mesh Adjustment</i>	70
5.5	Description of the Algorithm	70
5.6	Setting Up the Parameters	71
5.7	Adaptive Local Fitting, Holes and Noise	72
5.8	Experiment Results and Discussion	73
5.9	Summary	79
<b>Chapter 6. Conclusions and Future Research.</b>		<b>81</b>
<b>BIBLIOGRAPHY</b>		<b>83</b>

# CONTENTS

<b>ACKNOWLEDGEMENTS</b>	<b>iii</b>
<b>LIST OF FIGURES</b>	<b>vii</b>
<b>LIST OF TABLES</b>	<b>x</b>
<b>ABSTRACT</b>	<b>11</b>
<b>Chapter 1. Introduction</b>	<b>1</b>
1.1 Issues in Object Modeling Using Range Images	1
1.1.1 Registration	2
1.1.2 Data Integration and Description	4
1.2 Contributions	6
1.3 Dissertation Organization	7
<b>Chapter 2. Object Modeling: A Review</b>	<b>9</b>
2.1 Object Modeling Using Multiple Views: Data Integration	9
2.2 Triangulation	11
2.3 Deformable Models	12
2.4 Parametric Models	14
2.5 Other Approaches	15
2.6 Summary	16
<b>Chapter 3. Range Image Registration</b>	<b>17</b>
3.1 Introduction	17
3.2 Previous Work on Registration	19
3.3 Registration as Minimization	21
3.3.1 Problem Definition	21
3.3.2 Registration Evaluation Functions	22
3.3.3 Algorithm Description	25
3.3.4 Iterative Line-Surface Intersection Algorithm	26
3.3.5 Control Point Selection	29

## ACKNOWLEDGEMENTS

First of all, I thank my advisor and chairman of my dissertation committee, Dr. Gérard Medioni, for all his support, encouragement and patience during all these years of my study at USC. In the long way towards my dissertation, he encouraged creative thinking, provided necessary freedom and offered his advice and critical comments at every step. His systematic approach and scholarly insight helped me become a better researcher.

I thank Dr. Ramakant Nevatia, Director of IRIS, for offering his suggestions to my research, for his comments on my dissertation, and for serving on my dissertation committee. I thank him especially for sending me a letter when I was in China, which encouraged me to pursue my Ph.D. at USC, and for his guidance when I first joined the IRIS group.

I thank Dr. Douglas Ierardi for finding time in his busy schedule to serve on both my guidance and dissertation committees, and for his many suggestions during my qualifying exam. I also thank Dr. Aristides Requicha and Dr. Kenneth Goldberg for serving on my guidance committee.

I thank the faculty and staff members of the IRIS group and fellow students who created an excellent environment that I enjoyed working in, and made my stay at USC the most memorable period of time. Special thanks go to Dr. Keith Price for his help with my computing needs, to Andres Huertas for all his help and friendship, to Delsa Castelo and Dorothy Steele for helping me with my administrative needs. I thank my fellow students who gave me friendship and support both spiritually and academically during my long struggle in achieving my goal, among whom are Dr. Hillel Rom, Gideon Guy, Dr. Steven Cochran, Dr. Shou-Ling Peng, Dr. Bahram Parvin, Dr. Fridtjof Stein, Dongsung Kim and Chia-Wei Liao. I also thank Dr. Philippe Saint-Marc for introducing me to the range image world and for providing many of the range image processing elements.

My sincere thank goes also to Dr. Bala Kumar of CMU and K<sup>2</sup>T Inc. for providing the range finder software which is responsible for acquiring most of the range images used in this dissertation.

Finally, my deepest gratitude goes to my family members. I thank my parents for their support for my career choice. I thank my wife, Min Wang, for her support and understanding and for sharing my frustration and joy during my study. I thank her for giving birth to our son Bo Chen at the most difficult time when I was away, and for raising our son, who brought me new meanings of life.

*To my wife Min Wang and son Bo Chen*

DESCRIPTION OF COMPLEX OBJECTS  
USING  
MULTIPLE RANGE IMAGES

by  
Yang Chen

---

A Dissertation Presented to the  
FACULTY OF THE GRADUATE SCHOOL  
UNIVERSITY OF SOUTHERN CALIFORNIA

In Partial Fulfillment of the  
Requirements for the Degree  
DOCTOR OF PHILOSOPHY  
(Electrical Engineering)

August 1994

Copyright 1994     Yang Chen