Creating Generative Models from Range Images

Ravi Ramamoorthi

Computer Science Department
California Institute of Technology

Caltech-CS-TR-98-05

CS-TR-98-05

Creating Generative Models from Range Images

Thesis by
Ravi Ramamoorthi

In Partial Fulfillment of the Requirements
for the Degree of
Master of Science

California Institute of Technology
Pasadena, California

1998
(Submitted March 2, 1998)

Contents

Abstract

We describe a new approach for creating concise high-level generative models from one or more approximate range images. Using simple acquisition techniques and a user-defined class of models, our method produces a simple and intuitive object description that is relatively insensitive to noise and is easy to manipulate and edit. The algorithm has two inter-related phases—*recognition*, which chooses an appropriate model within a given hierarchy, and *parameter estimation*, which adjusts the model to fit the data. We give a simple method for automatically making tradeoffs between simplicity and accuracy to determine the best model. We also describe general techniques to optimize a specific generative model. In particular, we address the problem of creating a suitable objective function that is sufficiently continuous for use with finite-difference based optimization techniques. Our technique for model recovery and subsequent manipulation and editing is demonstrated on real objects—a spoon, bowl, ladle, and cup—using a simple tree of possible generative models.

We believe that higher-level model representations are extremely important, and their recovery for actual objects is a fertile area of research towards which this thesis is a step. However, our work is preliminary and there are currently several limitations. The user is required to create a model hierarchy (and supply methods to provide an initial guess for model parameters within this hierarchy); the use of a large pre-defined class of models can help alleviate this problem. Further, we have demonstrated our technique on only a simple tree of generative models. While our approach is fairly general, a real system would require a tree that is significantly larger. Our methods work only where the entire object can be accurately represented as a single generative model; future work could use constructive solid geometry operations on simple generative models to represent more complicated shapes. We believe that many of the above limitations can be addressed in future work, allowing us to easily acquire and process three-dimensional shape in a simple, intuitive and efficient manner.

Acknowledgements

Firstly, I would like to thank my advisor, Jim Arvo, for being a constant source of ideas and inspiration, and for creating many of the illustrations in the text. My former advisor, Al Barr, deserves special mention for guiding me during my first steps in research, and for many preliminary discussions on the subject matter contained in these pages. Jean-Yves Bouguet provided many helpful hints for data acquisition, and made many insightful comments that have significantly improved the quality of exposition.

This thesis represents only one part of the wonderful experience I've had these past 4 years as an undergraduate at Caltech, and I would like to take this opportunity to thank other faculty members who have mentored me during the course of research projects—Brad Werner, and K. Mani Chandy. Tom Tombrello deserves special thanks for the Ph11 program, a model for undergraduate research.

I would like to thank all the students of the Computer Science Department for support over the years, and especially the members of the Graphics Group. Most of my years as an undergraduate here have been spent in Blacker Hovse, and I thank each of its inhabitants for making it such a great place to live in.

I thank my sister, Roopa, for her advice and unflagging encouragement; and my parents for their unwavering support.

This work was supported in part by the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-8920219), an Army Research Office Young Investigator award (DAAH04-96-100077), the Alfred P. Sloan Foundation, and an equipment donation from Hewlett-Packard.

Chapter 1

Introduction

1.1 Motivation

It has now become possible to easily acquire true 3-dimensional data of actual objects. In this way, 3D objects may be *scanned* in much the same way as a 2D photograph. Reasonably accurate *point-clouds* or *range data*—a collection of 3-dimensional points on the surface of the object to be scanned in—from 3D objects [6, 31] can be obtained. There has also been much interest recently in simple methods for range data acquisition using structured light. In this thesis, we use two of these techniques, due to Bouguet and Perona[4] and Trobina[30]. In the first approach, 3-dimensional shape is inferred from the deformation of the shadow of a hand-held rod rolled over a plane on which the object is placed. The second method[30] involves the projection of alternating black-and-white stripes onto the object to be scanned. Both approaches use off-the-shelf hardware and are portable and easy to set up.

For use in graphical applications, these point-clouds are usually transformed into polygonal meshes [6, 15], or spline patches [9, 19]. However, these approaches often provide an unintuitive representation of the object and are difficult to manipulate. In addition, a huge amount of data is required since the meshes usually contain many thousands of triangles.

For modeling many man-made objects, *generative models* proposed by Snyder[27, 28] provide an attractive alternative. The modeled object is represented by a hierarchical tree of operators that provides a logical description of the object's structure. Designers can intuitively specify, examine, and modify the model. In this paper, we describe methods to invert this modeling process. Given a user-

defined hierarchy of models such as that shown in the lower left of figure 1.2, the system recovers an appropriate generative model within this hierarchy from range data of an actual object.

The hierarchy from which the examples in this thesis are derived is shown in the lower left of figure 1.2 and reproduced in figure 5.1. A number of objects such as bowls, ladles, and spoons as well as cylinders and surfaces of revolution can be adequately modeled. Assume we have a collection of objects such as cylinders, bowls, and spoons that we wish to represent, and which are effectively modeled by our chosen hierarchy. By using the range data acquisition techniques in [4] or [30], and the algorithms described in this thesis, we may automatically obtain accurate parametric models of the input objects. If we wish to represent a different class of models, the user must define the appropriate generative model hierarchy or extend our hierarchy to model the desired objects.

1.2 Benefits of the Proposed Method

Simplicity: To acquire the range data, we use simple techniques based on structured light due to Trobina[30] and Bouguet and Perona[4]. The latter approach requires only a desk-lamp, pencil and checkerboard apart from the camera. Using input from these approaches, we output a simple and intuitive object description in the form of a compact generative model.

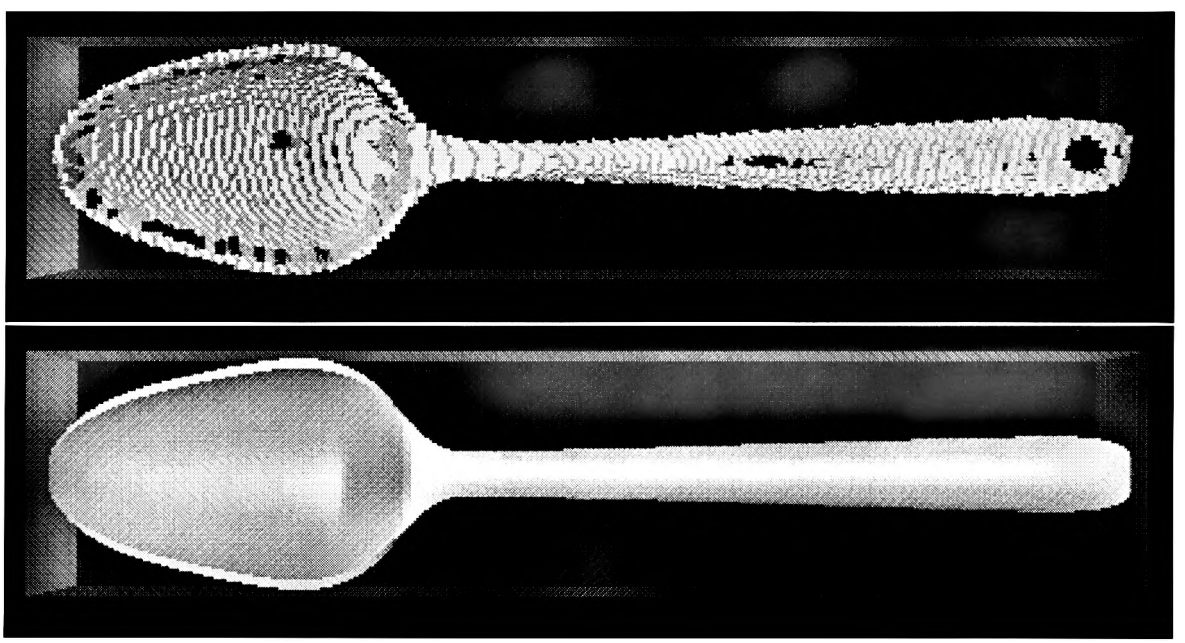

Figure 1.1: **Top:** *Original range data (a single range image artificially colored) embedded in a cuboidal bounding box.* **Bottom:** *Recovered generative model with the same bounding box. The model is a smooth and compact representation.* □

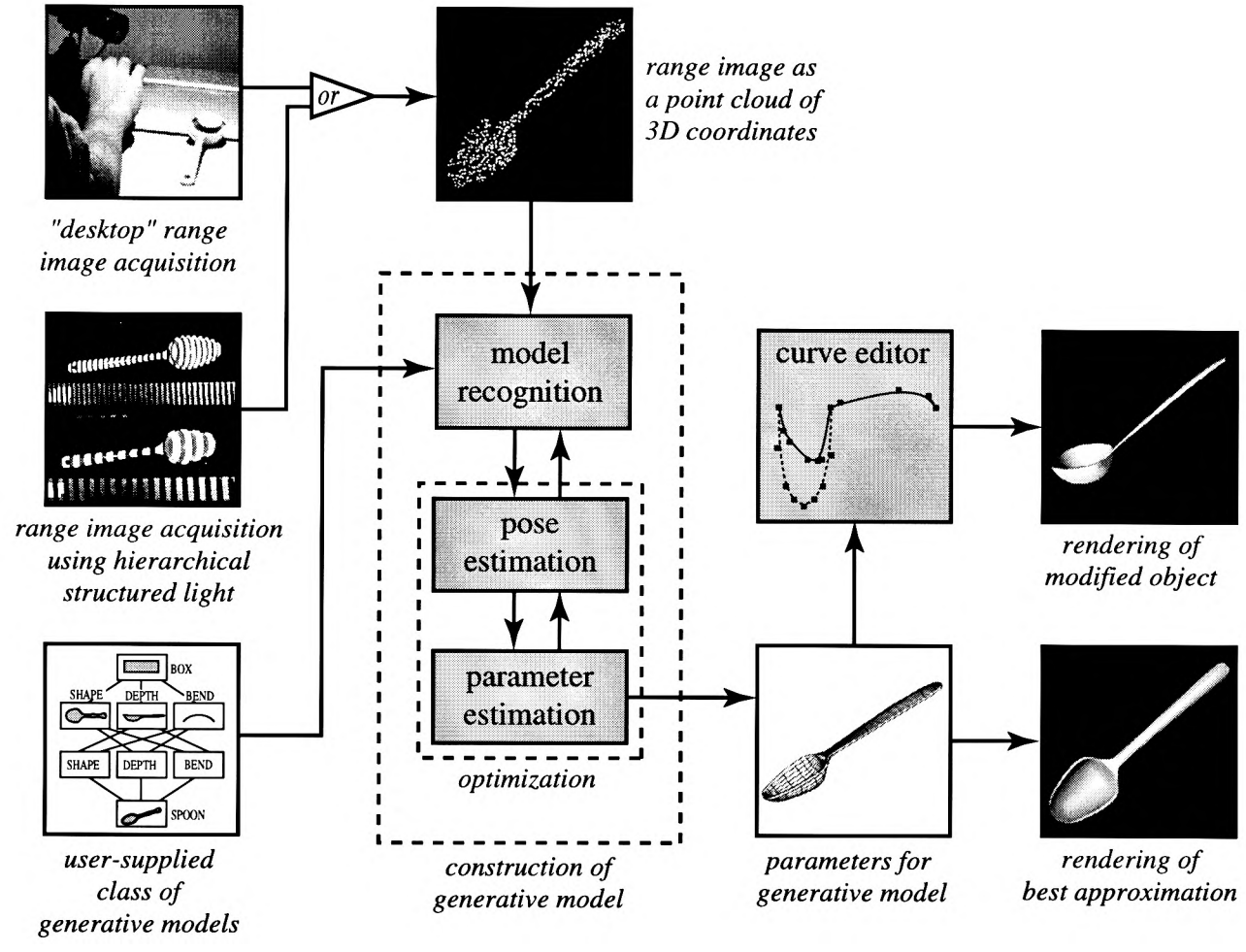

Figure 1.2: *Pictorial representation of generative model creation. The algorithm takes range data (in the form of a point-cloud) and a generative model hierarchy as input. An appropriate generative model is then chosen, and parameters are optimized to output an accurate and concise description that can be edited.* □

Robustness: Since we can make use of known symmetries in the given hierarchy, an accurate model can often be recovered from incomplete or noisy range data, making this approach more robust to noise than mesh creation. Most of our examples use only one noisy and incomplete range image, while creation of a polygonal mesh would require many accurate and properly aligned range images. We also demonstrate that even where the hierarchy cannot adequately model the range data, the approach yields a simple model that mimics the original object to the extent possible.

Compactness: Generative models provide a very concise representation—we need only store the operations at each level in the tree that describes the model and the control points representing the

parametric curves that constitute the leaves of this tree. This can provide a saving of several orders of magnitude over the number of parameters a triangular mesh would require.

Intuitiveness: Since the generative model is expressed in terms of parametric curves that correspond to logical features of the object, it is much easier to understand and manipulate than the corresponding polygonal mesh. This has obvious applications in recognition and editing. The generative model can be adapted to the object being represented. A sphere, a spoon and a cylinder can each be represented explicitly as high-level objects whereas a mesh or spline-patch represents these fundamentally different shapes with the same primitive.

1.3 Related Work

Recovery Methods for Specific Shapes: Much research in the computer vision community has focused on methods for recovering object shape for specific primitives such as superquadrics, generalized cylinders and hyperquadrics. Representative references are [5, 14, 20]. While these methods are useful within our framework for estimating model parameters, our major contribution is not in parametric recovery for specific models, but in selecting an appropriate model and presenting general techniques for optimizing the curves constituting this representation. Further, most of the representations used in previous work have a small number of parameters instead of curves, and research has largely been restricted to recovery of a specific type of model such as quadrics[10] or generalized cones[26] instead of a model within a more general hierarchy. In a similar spirit as our work, Debevec et al.[7] propose a way to recover polyhedral models from photographs of architectural scenes. While this research is similar to ours, we employ more complex shapes, and automate the recovery process further—the user need specify neither the specific model nor the particular edges of interest. However, we require range data instead of photographs and use a pre-defined model hierarchy.

Grammars: Lin and Fu[22] discuss the use of plex grammars[11] to describe composition of objects, which is analogous to our use of a hierarchical tree structure as a model representation. An object is represented as a composition of primitive objects. These primitives can be joined together via translations, rotations and connection at curves or surfaces. This is a general framework for representing

complex objects. By contrast, in this thesis, we are concerned with composing curves with a tree of operators to derive a single generative model. Future work could involve composition of generative models to represent more complicated objects. While this could be done with standard computer graphics operations, grammars might provide a more general and powerful framework.

Mesh simplification and editing: Hoppe et al.[16] describe techniques to optimize meshes. In this approach, a simplified mesh is compared to the original pointwise. However, the simplified mesh will still contain the same primitive—the triangle—and will still not represent many models accurately. For instance, a sphere can obviously be described by one parameter—the radius—but a triangular tessellation requires significantly more data. By contrast, generative models have the power to switch to a different representation as appropriate, e.g. from a sphere to a cylinder. In addition, our chapter on optimizing generative models discusses some improvements over the objective functions used in Hoppe et al.[16]. A major application of our research for graphics is the intuitiveness of the representation that makes models easy to interpret and edit.

1.4 Limitations of our Approach

There are a number of important limitations in our current approach that deserve mention. While the number of different shapes that can be expressed logically with generative models is an advantage, this also means that an appropriate model hierarchy must be created by the user to model a desired set of input objects. We currently rely upon user-defined methods to estimate particular curves of the generative model (although these estimates may be crude). While we touch on how these estimates can be automatically derived from curves for simpler models in chapters 3 and 4, we have not developed this approach completely. Note that we derive an approximation to the range data; the full complexity of the data is not retained. Further, we deal only with the problem of obtaining a single generative model. The composition of simple generative models to represent more complex objects is not dealt with in this thesis. Finally, note that we have shown our results only for a relatively simple subtree of generative models. While we believe this to represent a proof of concept, a complete system would probably use a much larger class of input models.

Our approach depends upon existing methods of range data acquisition. The input point-cloud is

assumed to be a *good* representation. In practice, this means that we manually clean the data to remove stray points or points picked up from the environment and not relevant to the object being modeled. Further, we do not touch on alignment of range images which is of prime importance in obtaining a seamless 3-dimensional representation. Where alignment of multiple views was required, we used a mechanical device (a rotary platform) to get accurate information on the transformation connecting the views.

Organization

The rest of this thesis is organized as follows: First, chapter 2 gives some background information on range data acquisition, generative modeling, and fitting of parametric models on which the material in this thesis builds. In chapter 3, we describe our basic framework for recovering the appropriate generative model based on the complexity of the range data given as input. In chapter 4, we discuss our methods for fitting a selected generative model to range data, and discuss some general optimizations in this context. In chapter 5, we give details of the specific generative hierarchy we use to generate the results given in chapter 6. We present our conclusions and directions for future work in chapter 7.

Chapter 2

Background

This chapter is intended to give a brief introduction to some of the concepts on which this research builds—range data acquisition, generative modeling, and optimization to fit parametric models.

2.1 Range Data Acquisition

A *range image* is equivalent to a photograph with additional depth information—in effect, a 3-dimensional scan. Instead of an intensity at each pixel, as in a conventional digitized photograph, we associate a distance from the camera. This is the depth information. The ordinary intensity image can be useful for modeling the reflectance properties of the object to be scanned, but this thesis does not address issues of texture and reflectance modeling.

The range image can be acquired in a variety of ways—by use of laser scanner[21], triangulation[25], and with structured light. These are all *active* methods in that energy is projected onto the object, and range data is obtained from observing its reflection. *Passive* methods may also be used such as shape from properties such as shading and silhouettes. However, these algorithms are generally not as reliable as *active* range data acquisition. Humans acquire a sense of depth by stereoscopic vision as well as higher level cognitive processing. While there has been some research on such methods (for instance, [29]), use of stereo requires solving a correspondence problem, for which sufficiently robust algorithms have not yet been developed. Jarvis gives a good introduction and survey of range data acquisition[18]. Besl and Jain[2] survey three-dimensional object recognition and include a discussion

of range data acquisition. Parthasarathy et al.[24] give a classification of methods to acquire range data.

To acquire a seamless 3-dimensional representation of the object being modeled, we must *align* multiple range images obtained from different angles. This is analogous to composing photographs taken from different angles to get a better idea of shape. Alignment of range images is an interesting topic. A volumetric method of merging scans is detailed in [6] along with an arrangement that allows for full motion around the object. In case the precise transformation between the different range images is not known, computing a correct alignment can be extremely difficult. This is because methods to align multiple range images must establish correspondences between the same physical point on multiple data sets, which is not always easy to do. To avoid this problem, we used a rotary platform where necessary to obtain an accurate transformation between views from different angles (in the same plane). It should be noted that one of the advantages of our technique is that we can often get good results from only one range image—in these cases, no alignment is required—or a cylindrical view of the object—in which case a simple rotary platform sufficed. One of the goals of this thesis is to provide methods to obtain fairly accurate models with minimal hardware, so we have used a simple rotary stage rather than a more complicated motion apparatus.

While methods for range data acquisition have been known for decades, there has recently been a lot of interest in simple techniques for range imaging which make the process simple, cheap, and portable. Indeed, one of the primary goals of this thesis is to show that for a certain class of objects, we can use simple data acquisition techniques together with the algorithms described in this thesis to produce a simple and intuitive object description. It is hoped that the development of such methods will allow ordinary users to become comfortable with representing 3-dimensional objects in much the same way that photography has become commonplace. Below, we discuss two techniques for range data acquisition that require minimal hardware and can easily be constructed from off-the-shelf components. These were used to provide the range data input for our system.

2.1.1 Hierarchical Structured Light

In this method[30], we use a projector and a camera. First, the projector and camera are calibrated (a checkerboard pattern is used for this purpose), and the relative transformation between them is noted. The setup is shown in figure 2.1.

Figure 2.1: *The apparatus for range data acquisition using structured light. We include the object to be scanned in, the projector, the camera, the checkerboard pattern for calibration, the monitor and the computer displaying point-clouds of the object. Photograph courtesy of Jean-Yves Bouguet.* □

A sequence of alternating black and white stripes are projected as a gray code onto the object from the projector. The gray code is used to minimize error (as proposed by Inokuchi et al.[17]) owing to misclassification of a point as light or dark. By considering the pixel location on the camera and the projector code derived from the striped pattern, we may derive the 3D location of that point using our calibration data. Since this can be done for all visible points, we are able to acquire a range image.

Figure 2.2 shows an example of the stripe images that result. Objects from the environment such as the table are removed by using a contrast mask before processing or by removing stray points manually from the range data. An example of the resulting point cloud (which has been made sparser to show the individual points clearly) is found in figure 2.3.

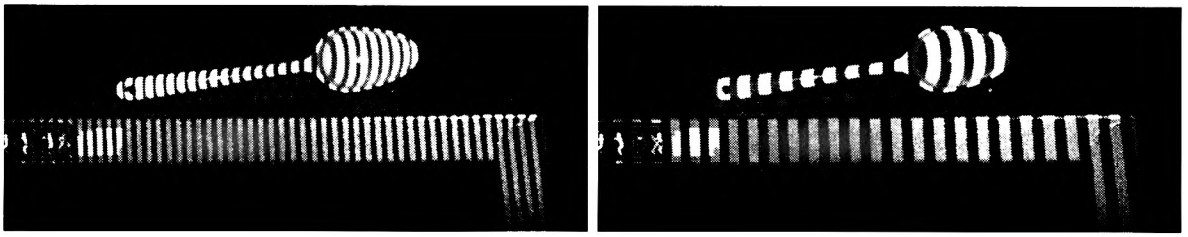

Figure 2.2: *Two patterns of alternating light and dark stripes projected onto the object to be scanned. The stripes are at different resolutions and project a gray code onto the object.* □

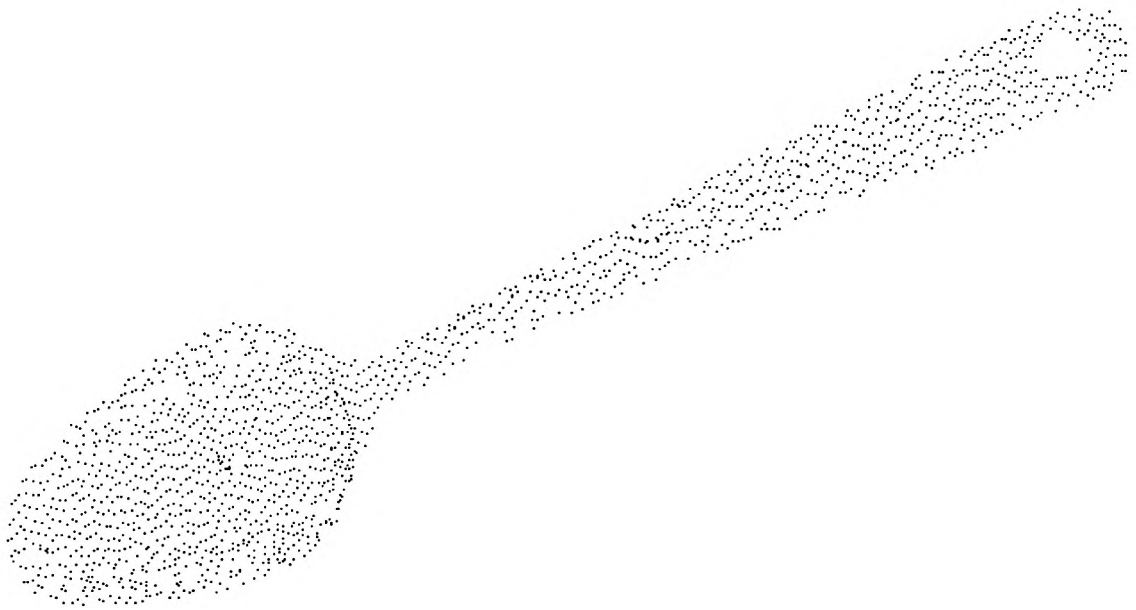

Figure 2.3: *The point cloud got from range imaging of a spoon using structured light. The point cloud has been made sparser in the figure to show the individual points clearly.* □

2.1.2 Desktop Range Imaging

Recently, Bouguet and Perona [4] proposed an approach designed to use minimal hardware. They observe the deformation of the shadow cast by a hand-moved rod over a plane on which the object is cast after calibration of the camera and point light-source. This approach is very easy to set up. An image of the setup is shown in figure 2.4 and illustrates the simplicity of the technique.

Note that the camera image places each visible point along a line. The depth is unknown. However, in the setup of figure 2.4, we know that all points in the shadow of the rod lie on the plane formed by the light-source and the line segment containing the undeformed shadow (where the shadow does lies on the plane). The intersection of this plane with the line from the camera allows us to infer depth. By moving the rod, we calculate depth for all visible points.

Figure 2.4: *Showing the method of Bouguet and Perona. The light-source isn't shown in the figure and the only other devices needed are a checkerboard and pencil for calibration only. The rod is moved manually and depth is inferred from deformation of the shadow.* □

Limitations: A few of the limitations of our range data acquisition techniques are discussed below. Highly specular objects or dark objects can often not be imaged. This is because there is little contrast between views obtained by projecting dark and light stripes, or between views obtained when the object is in shadow versus being lighted. Also, there is a size limitation—our system is currently set up to image objects whose characteristic length is about 10 cm. While it may be possible to extend the approach to work on objects significantly outside this size range, this has not yet been tested. We deal only with static scenes; dynamic scene descriptions are not modeled. The object to be scanned in must be rigid to allow it to be easily rotated for viewing from multiple viewpoints. Our current system makes it hard to scan in flexible objects.

2.2 Generative Modeling

Snyder [27] gives a comprehensive description of generative modeling. Here, we briefly mention the important features that we use in our system. A generative model can be represented:

$$S(u,v) = f(u,v,c_1,c_2,c_3,...) \tag{2.1}$$

where S is the surface, f is a vector function represented as a tree of primitive operators acting on curves $c_1, c_2, c_3, ...$ The curves are maps from the real line to 2-dimensional space; i.e. $c_i : \Re \rightarrow \Re^2$. u and v are parameters to the curves (and hence, the surface). By convention, we usually use $u, v \in [-1, 1]$ It can be seen that the three-dimensional shape is built from two-dimensional curves—that may for instance, control cross-sections or scaling relationships. Thus, the shape is specified in a modular way, being built from lower-dimensional components that are easier to manipulate and visualize.

An example of a generative model that is simple to specify, and yet reasonably expressive is the *banana* model [27, p. 68,69] developed by Snyder. Here, the surface S is given by an affine transform:

$$S(u,v) = M(v)\gamma(u) + T(v) \tag{2.2}$$

where $M(v)$ is a linear transformation in 3D, and $T(v)$ is a translation. M, γ and T may then be represented with parameterized curves. The affine transform is composed of a rotation, translation and scaling. Each component could be represented with a different curve, each parameterized by v. γ represents the cross-section parameterized by u. It is also possible to easily represent simpler models such as profile products [1]:

$$S(u,v) = \begin{pmatrix} \gamma_1(u)\delta_1(v) \\ \gamma_2(u)\delta_1(v) \\ \delta_2(v) \end{pmatrix} \tag{2.3}$$

where $\gamma(u) = [\gamma_1(u), \gamma_2(u)]$ is the cross-section curve and the profile curve is $\delta(v) = [\delta_1(v), \delta_2(v)]$.

Generative models are intuitive to use, and fairly general. Unlike simple shapes such as superquadrics, hyperquadrics, spheres, or cylinders which are parameterized by a number of (not always intuitive) parameters, generative models differ in having curves, not isolated parameters, as inputs to or at the leaves of the operator tree f in equation 2.1. Further, these curves usually correspond directly to a logical feature of the model such as a cross-section, a transformation, or a scaling factor. Because of their intuitive nature and expressive power, generative models are convenient for modeling many

man-made objects. To create a generative model, the user needs to know the surface description—in other words, the user must specify the precise symbolic description of the model. The constituent curves must then be created by adding control-points in a curve-editor. Note that the methods of this thesis automate the above procedure—the curves are automatically determined, and the appropriate model (symbolic representation) is chosen within a user-specified hierarchy.

The principal benefits of generative models are that they are compact, very general, and allow intuitive model specification. Further, for a set of generative models, it is usually easy to create a hierarchy of models with links from *parent* to *child* or a shallower to a deeper level corresponding to making the model more complex, usually by adding a single curve as explained in Chapter 3.

Our work allows generative models to be more widely used by presenting algorithms for their recovery from range data. We note that a principal hurdle in generative modeling is that the user must specify the function f in equation 2.1, often using complicated mathematical operations. However, once a generative model hierarchy has been created, a user can recover an appropriate model and the parametric curves constituting it without needing to fully understand the mathematics. By interactively modifying the curves, he may edit the model in an intuitive way without knowing the precise mathematical representation of the object being modeled.

2.3 Fitting of Parametric Models using Optimization

There is a wealth of literature on fitting parametric models to data. See the survey by Bolle and Vemuri [3] for instance. In some cases, the parametric model may be represented by an implicit function that depends on some parameters:

$$f(x, y, z) = 0$$

where x, y and z are the co-ordinates. In this case, an application of f to each data-point gives an error-norm, and the model parameters that determine f can be varied to minimize this norm. The simplest example is if f represents a sphere centered about the origin. Then $f(x, y, z) = x^2+y^2+z^2-r^2$ where the sole model parameter is the radius r. Here, r can be varied to best fit the data. The error norm defined by f will in general be algebraic, and not based on purely geometric disparity. This can lead to problems, and various error norms can be proposed to correct some of these problems.

Another possibility is for x , y and z to be known explicitly in terms of parameters u and v as is often the case for generative models. Refer to equation 2.1. The challenge in this case is to associate a data point with given u and v for comparison to the model. Chapter 4 on fitting a specific generative model gives a clear discussion of our methods, and why they improve in some cases on previous work. The quantities involved in computing correspondence are often challenging to compute analytically, and our approach is general enough to allow the model to be represented in tessellated form. This is particularly challenging because most optimizers require objective functions that are twice-differentiable.

Optimization

In fitting of a parametric model, we will need to use an unconstrained nonlinear minimizer. We used the Sequential Quadratic Programming routine E04UCF in the NAG libraries [23]. Here, we briefly describe how this optimizer works.

Let $\mathbf{x}$ denote the current best guess to minimize the objective function ϕ. We seek to update:

$$\overline{\mathbf{x}} = \mathbf{x} + \alpha \mathbf{d} \tag{2.4}$$

where $\overline{\mathbf{x}}$ is the new best guess, α is the step-size and $\mathbf{d}$ is a direction determined by solving the quadratic subproblem obtained from a Taylor expansion about the current best guess to quadratic order:

$$\min \ \mathbf{g}^\top \mathbf{d} + \frac{1}{2}\mathbf{d}^\top \mathbf{H}\mathbf{d} \tag{2.5}$$

where $\mathbf{g}$ is the gradient of the objective function and $\mathbf{H}$ is the corresponding Hessian. A straightforward gradient descent can then be performed. The Hessian is updated using Broyden-Fletcher-Garb-Shapiro quasi-Newton update (see [8] for a reference). A good general reference is Gill et al.[12].

Differentiability: In order to compute the gradients and the Hessian, most numerical routines require the objective function to be continuously twice differentiable. The gradients may either be computed analytically, or by using a finite difference approximation. It is easier to use a finite difference approximation, since it can be computed automatically, and this is the approach used in this thesis. However, analytic gradients are more accurate and can lead to better results provided they can be computed efficiently.

Chapter 3

Framework for Model Recognition

Here, we discuss our general framework for choosing the appropriate generative model from within a user-defined class to best fit the range data. The model recognition process is based on a compromise between model accuracy and simplicity. Although the approach detailed here is general, we will often refer for illustrative purposes to the specific tree used to create the results shown. This hierarchy which is pictured on the lower left of figure 1.2 (a larger version is reproduced in figure 5.1) and in figure 6.1 is inspired by the spoon model created by Snyder [27, p. 83]. Our model hierarchy can represent spoons, bowls and ladles as well as a number of common shapes as discussed in chapter 5.

3.1 Important Concepts

3.1.1 Model Hierarchy

The model hierarchy consists of a number of levels corresponding to the complexity of the model as measured by the number of curves constituting it. For our class of models, level 0 consists of a "box" of constant thickness with only 3 parameters for the length, width and depth. The box model essentially defines a bounding box for the data. Deeper levels consist of *refining* one or more of these parameters by representing them as curves instead of constant values. For example, the edge from the box to the "depth" node in figures 1.2 and 6.1 corresponds to *refining* the depth by representing it as a curve. The hierarchy can also be thought of as a tree; going from parent to child generally corresponds to adding a single curve. For instance, the box is the *parent*, while the model with refined depth is the *child*.

While the parameters for curves can vary continuously, there may also be parameters which take

on one of a set of discrete values. In this case, we may use one of these values in the recognition phase, but choose the value which leads to the lowest error before final model selection and optimization. For example, in the tree shown in figure 1.2 and discussed in greater detail in chapter 5, each node can be split corresponding to a straight or circular cross-section. Thus, there is a link in the hierarchy corresponding to changing the cross-section used. In practice, we have found it convenient to use the simple rectangular (straight) cross section for the recognition process. After recognition, we choose the cross-section—straight or circular—leading to lower error. The branch for the appropriate cross-section is the lowest part of the tree in figure 1.2. For completeness, the results show an example (the bowl) where we actually split each node.

3.1.2 Estimation of Co-ordinate Axes

During the fitting process, we align the range data with the axes as labeled by the specific generative hierarchy being used. This is done simultaneously with estimation of the first or *root* model. We use the principal axes of the data to infer the co-ordinate directions. The co-ordinate representation of the generative model imposes an asymmetry between the axes, making it impossible to label them without knowledge of the specific hierarchy used. However, we can consider all six possible labelings and pick the one which leads to the best results after an appropriate model is recognized and then optimized. Model-dependent axis-labeling techniques can be used where available, and details for our hierarchy are found in chapter 5.

3.1.3 Parameter Estimation

For generative models, we need a way to estimate parametric curves for model recognition and optimization. A starting point exists in the form of the parent model at the previous stage in the hierarchy. For instance, when we change from the box to a generalized cylinder, we are refining the shape curve from the starting point of a simple rectangular bounding box. Optimization over the additional degrees of freedom can be used to estimate the newly-added curve as shown in figure 4.3.

In case reasonably accurate model-dependent ways to estimate parameters are known such as those for our tree of models discussed in chapter 5, we do not need to optimize in the recognition phase, but

still do so after an appropriate model is chosen. We emphasize that the model-dependent estimates may be quite crude, since they merely provide an initial guess for the global optimization phase.

Figure 3.1 presents a summary of our entire algorithm. Below, we discuss each step in detail. The reader may wish to refer to figure 6.1 for examples of applying the algorithm.

Basic Algorithm

1. Acquire Range Data
2. Set current estimate for the model to the root node of hierarchy and calculate error of fit for root node after estimating parameters.
3. For each child of current model, estimate parameters and calculate error of fit. Add complexity to calculate total cost.
4. If cost for any child is less than that for current model estimate, reset current estimate to child with minimal cost and goto step 3.
5. Optimize.
6. Check for possible simplifications.

Figure 3.1: *An overview of the entire algorithm.* □

3.2 Algorithm Description

Step 1. Acquire Range Data: We use simple and portable techniques based on structured lighting to acquire range data. Trobina [30] describes a method using a projector where alternating patterns of dark and light are projected onto an object. Bouguet and Perona[4] infer shape from the shadow of a rod moved over a plane on which the object is placed. Both techniques are simple, portable, and require minimal hardware.

Step 2. Fit to root model: Let ρ denote our current model estimate which we initially set to the root node of the hierarchy—for our hierarchy, the box. An error-of-fit function ϕ is computed based on the average spatial deviation between the range data and the model as defined in equation 4.10 —details are given in the next chapter. The total cost C is set equal to the error of fit.

Step 3. Fit children to Data: For each child denoted by $\gamma_i(\rho)$ of the best guess ρ, we estimate parameters and calculate the error of fit $\phi(\gamma_i)$. For instance, if $\rho = box$, the children are $\gamma_1 = shape, \gamma_2 = depth, \gamma_3 = bend$. The parameter estimation is **greedy**. When a constant is refined into a curve, only that curve is estimated; the curves already constituting ρ are not changed. This is not necessarily the best strategy—addition of more degrees of freedom may change the best values for the existing parameters—but it is simple, efficient and scalable. We then calculate a cost function for each child:

$$C(\gamma_i) = \phi(\gamma_i) + \triangle(\gamma_i) \tag{3.1}$$

where $\triangle$ is a penalty for complexity. We use $\triangle(N) = NQ$ where N is the level in the hierarchy of a specific model and Q is a constant that controls the tradeoff between simplicity of the output model and accuracy.

Step 4. Reset ρ: The previous stage calculated the cost function for ρ and all its children if any. Of these, if the child with lowest cost $C(\gamma_i(\rho))$ has a lower cost than ρ, we make it the new best fit ($\rho = \gamma_i$) and go back to step 3. Otherwise, exit to step 5. For instance, if $\rho = box$ and *shape* has the lowest cost out the children of *box*, and $C(shape) < C(box)$, we now set $\rho = shape$ and loop back to step 3. On the other hand, if $C(shape) > C(box)$, we exit to step 5.

As with parameter estimation, we are using a **greedy** algorithm to choose the best model. We consider only the descendants of the current best guess ρ to choose the next best guess. Conversely, a node is considered only if it is linked to the best guess ρ at some point.

Step 5. Optimize: Once we have a model ρ of the right complexity, we refine the model parameters or curves by using a global optimization process—discussed in the next chapter—that is capable of fine-tuning all the parameters simultaneously, and giving us the best generative model of a particular type.

Step 6. Simplify: Once the optimized curves constituting the generative model are obtained, we can check for simplifications. We currently check for the curves being close to semi-circular, one or more curves being symmetric (in which case we use the side for which we have more complete data),

and identicality of two or more curves. In these cases, a simple representation of the curve can be obtained. This in turn can lead to simpler object types. The same effect can be obtained by including all simpler object types in the hierarchy, but may lead to an unnecessarily large number of primitives at each level.

The simplification step occurs after, and not before optimization, because the initial guess for the optimization might intentionally be simple or crude, and we must be sure that the optimal curve can be simplified.

Chapter 4

Fitting a specific generative model

In this chapter, we give some general techniques to fit a specified generative model to range data. The specific generative model is assumed chosen using the methods of the previous chapter. The primary contribution of this chapter is to develop our general error-of-fit functions (used for model recognition in the previous chapter), and show how they can be minimized by optimization—step 5 in our general framework just described. We also discuss our methods for fitting and parameterizing curves from extracted silhouettes, and show how curve-fitting and optimization are combined to yield an accurate generative model.

4.1 Optimization

We require a general objective function to compare range data and a generative model. For greatest generality, we require our methods to be *independent* of the specific generative model used and work whether or not the quantities to be computed have analytic forms; for instance, the model may exist only in tessellated form. Since the model can be an arbitrary function of the parametric curves constituting it, we use a general unconstrained nonlinear minimizer. For example, we have used the Sequential Quadratic Programming routine in NAG[23] with numerical computation of the gradients. Because the optimization process is more robust when the objective function varies smoothly with the parameters, we use objective functions that are guaranteed to be C^p continuous where p can be made arbitrarily large.

We discuss two objective functions corresponding to 2D and 3D error-of-fits, either of which may

be used to obtain an optimal model. The 2D error-of-fit is well suited for optimizing particular curves by projecting onto the appropriate plane as shown in figure 4.3, while the 3D error-of-fit gives a clear measure of the average spatial deviation between range data and the model.

(x_i^m, y_i^m)	Model point in 2D plane
(x_i^m, y_i^m, z_i^m)	Model point in 3D
ϕ	Objective function
$\triangle_r$	Area of projected range data
$\triangle_m$	Area of projected model
$\triangle_c$	Common area
$\max(a, b)$	Maximum of a and b
$\min(a, b)$	Minimum of a and b
$\Uparrow(\cdot,\cdot), \Downarrow(\cdot,\cdot)$	C^p continuous max/min
$Y^m_{\min}(x), Y^m_{\max}(x)$	Model silhouettes
t	Support of kernel
$K(\cdot, t)$	Kernel of support t
$B(\cdot, t)$	Blend function of support t
(α, β, γ)	Fractional Parameterization
$x^m_{\Downarrow}, x^m_{\Uparrow}$	C^p min and max values in x of model
$y^m_{\Downarrow}(x^m), z^m_{\Downarrow}(x^m, y^m)$	C^p min values for y and z
$y^m_{\Uparrow}(x^m), z^m_{\Uparrow}(x^m, y^m)$	C^p max values for y and z
$\mathbf{p}_i^m, \mathbf{p}^r$	Vectors on model and range data
$\mathbf{P}^m(\mathbf{p}^r)$	Corresponding model point for $\mathbf{p}^r$
μ	Penalty for a hole

Table 4.1: *Some important symbols.* □

4.1.1 2D Error of Fit

Here, we discuss the comparison of two "images"—a series of points projected onto a given plane which for notational purposes we will assume to be the X-Y plane but which can actually be any plane or combination of planes (such as the sum of the errors for projection onto the X-Y and X-Z planes). We shall denote the i^{th} point from the acquired range data by (x_i^r, y_i^r) and the i^{th} point from tessellating the model by (x_i^m, y_i^m). (x_i^m, y_i^m) is assumed to be arbitrarily smooth in the parameters of the model. Our goal is to design a general optimization technique which given as inputs (x_i^r, y_i^r), and a starting guess for model parameters—corresponding to a guess for (x_i^m, y_i^m)—produces a "best-fit". We will also discuss the simplifications that result when analytic formulae for model contours can be found.

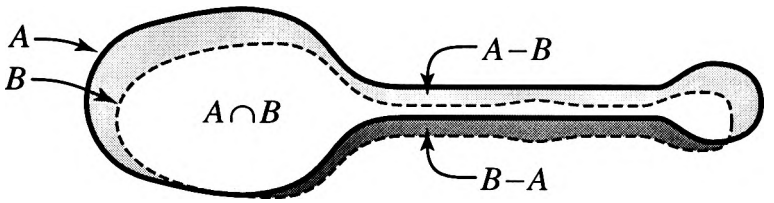

Figure 4.1: *Motivation for the 2D objective function which is the symmetric difference between the range data and the model:* $(A - B) \cup (B - A)$. □

2D Objective Function: We will use a simple objective function illustrated in figure 4.1—the area not common to the two projections. Clearly, when this area is 0, the images coincide. We start with the definition:

$$\phi = \triangle_r + \triangle_m - 2\triangle_c \tag{4.1}$$

where ϕ is the objective function, $\triangle_r$ is the area of range image, $\triangle_m$ is the area of the image from the model, and $\triangle_c$ is the common area. The terms on the right are defined by:

$$\begin{aligned} \triangle_r &= \int_{-\infty}^{\infty} (Y^r_{\max}(x) - Y^r_{\min}(x))\, dx \\ \triangle_m &= \int_{-\infty}^{\infty} (Y^m_{\max}(x) - Y^m_{\min}(x))\, dx \\ \triangle_c &= \int_{-\infty}^{\infty} \max(0, l_c(x))\, dx \end{aligned} \tag{4.2}$$

where $Y^m_{\max}(x)$ refers to the maximum value of the model as a function of x. $Y^m_{\min}(x)$, $Y^r_{\max}(x)$ and $Y^r_{\min}(x)$ are similarly defined with the values for the range data being computed in the same way as those for a tessellated model. In cases of holes, there may be more than one maximum or minimum. We will not discuss these cases here, but our argument can be extended by considering each segment separately. $l_c(x)$ stands for the common "length" as a function of x and is given by:

$$l_c(x) = \min(Y^m_{\max}(x), Y^r_{\max}(x)) - \max(Y^m_{\min}(x), Y^r_{\min}(x)) \tag{4.3}$$

Range of Integration: Where $Y^r_{\max}(x)$ and $Y^r_{\min}(x)$, or $Y^m_{\max}(x)$ and $Y^m_{\min}(x)$ are undefined—that is, if the points from range data or model do not extend into a given region—the integrand(s) is(are) defined as 0. This means the effective range of integration is the extent in x.

Continuity: The function $\triangle_c$ as defined above may be C^1 discontinuous because of the mins and maxes. Removing the outer max will help us by providing a correct gradient even in regions where the two datasets don't overlap. C^p continuity can be maintained through the use of blending functions:

$$\triangle_c = \int_{-\infty}^{\infty} [\Downarrow\!(Y^m_{\max}(x), Y^r_{\max}(x)) - \Uparrow\!(Y^m_{\min}(x), Y^r_{\min}(x))]\, dx \tag{4.4}$$

Here, $\Uparrow\!(\cdot,\cdot)$ and $\Downarrow\!(\cdot,\cdot)$ denote C^p continuous versions of max and min respectively which reduce to the standard max and min functions when the two points in question are far enough apart. We will begin by defining:

$$\begin{aligned} c(a,b) &= \frac{a+b}{2} \\ r(a,b) &= \frac{|\,a-b\,|}{2} \\ \Uparrow\!(a,b) &= c + f(r) \\ \Downarrow\!(a,b) &= c - f(r) \\ r \geq t &\Rightarrow f(r) = r \end{aligned} \tag{4.5}$$

The standard max and min functions result when $f(r) = r$ everywhere, and the modulus sign in the definition of r results in a derivative discontinuity when $r = 0$. For C^p continuity, f must be C^p continuous and satisfy $f(u) = u$ when $u \geq t$ where t is the "support" of f—the region where it differs from the traditional value. Also, the first p derivatives of f should vanish at 0. The top part of figure 4.2 shows this construction. If we desire the property that $\Uparrow\!(a,b)$ lie between a and b, then $\Uparrow\!(a,a) = a$ and $f(0) = 0$. In this case, we can define f as follows (see the bottom part of figure 4.2 for an illustration):

$$f(r) = r(1 - B(r,t)) \tag{4.6}$$

Here, $B(u,t)$ is a C^p continuous blending function of finite support t ($B(u,t) = 0$ when $u \geq t$ and $B(u,t) = 1$ when $u \leq 0$) with $B(0,t) = 1$ such that the first p derivatives vanish at $u = 0$ and $B(t-u,t) = 1 - B(u,t)$. An inspection of the bottom middle part of figure 4.2 will show that this construction requires there to be regions where f will be less than that for the traditional maximum

or minimum, and consequently there will be regions where the continuous maximum is less than the discrete version while the continuous minimum is greater than the discrete version.

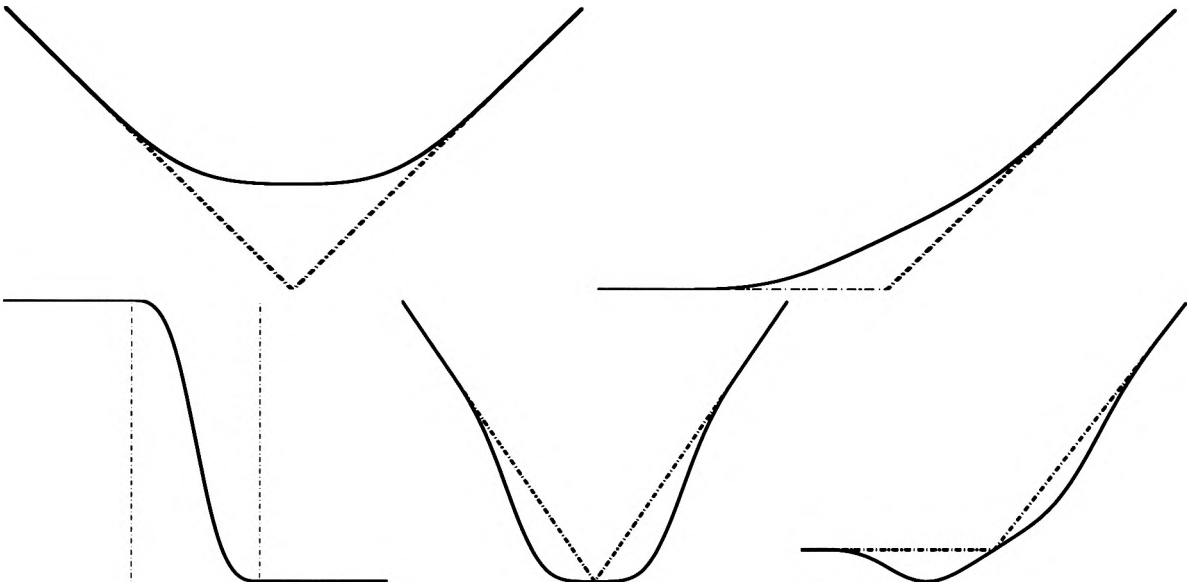

Figure 4.2: C^p *continuous maximum:* $\Uparrow(a,b)$. **Top Left:** *Solid line shows* f *as a function of* $b-a$*, while dotted line shows the traditional relation* $f=r$ *which is seen to have a derivative discontinuity at 0.* **Top Right:** *Solid line shows* $\Uparrow(a,b)$ *as a function of* b *- a while dotted line shows the traditional maximum and is seen to have a derivative discontinuity where* b *becomes larger than* a*.* **Bottom Left:** *Blend function with vertical lines showing extent of blend from* 0 *to* t*.* **Bottom Middle:** f *as defined in equation 7 as a function of* $b-a$ *with the solid line showing the continuous version and the dotted line, the traditional maximum.* **Bottom Right:** *Solid line shows* $\Uparrow(a,b)$ *while dotted line shows the traditional version.* □

The integrals remains smooth in the face of isolated discontinuities in $Y^m_{\max}(x)$ and $Y^m_{\min}(x)$ as functions of (x_i^m, y_i^m), which allows us to use a "box-kernel" as discussed next.

Evaluating the Integral: If $Y^m_{\max}(x)$ and $Y^m_{\min}(x)$ can be expressed in closed form, the integrals may be evaluated analytically. Otherwise, the model must be tessellated, and $Y^m_{\max}(x)$ and $Y^m_{\min}(x)$ found numerically. In this case, we use a simple "box-kernel" of support t to evaluate $Y^m_{\max}(x)$ and $Y^m_{\min}(x)$:

$$\begin{aligned} Y^m_{\min}(x) &= \min(y_i^m : \mid x_i^m - x \mid \leq t) \\ Y^m_{\max}(x) &= \max(y_i^m : \mid x_i^m - x \mid \leq t) \end{aligned} \tag{4.7}$$

where t is a suitable width of the same order as the point spacing in x. The evaluation of $Y^r_{\max}(x)$ and $Y^r_{\min}(x)$ is similar. As already discussed under the paragraph on the range of integration, in regions where no suitable y_i^m exists, $Y^m_{\min}(x)$ and $Y^m_{\max}(x)$ are undefined and the appropriate integrals are taken as 0 since there is no contribution to area from a region in which there is no data.

More complicated methods may produce better results, but also make evaluation of the integral for ϕ more difficult. Also note that evaluation by quadrature does not preserve continuity in the face of isolated discontinuities. The "silhouette" step functions are represented as a list of intervals with each interval corresponding to a particular (x_i^m, y_i^m). The list can be evaluated with one pass through the collection of points making this technique quite efficient.

Figure 4.6 shows the construction of the "silhouette" functions. The right of the figure shows a way to make them continuous as required for our 3D error of fit, but not necessary here.

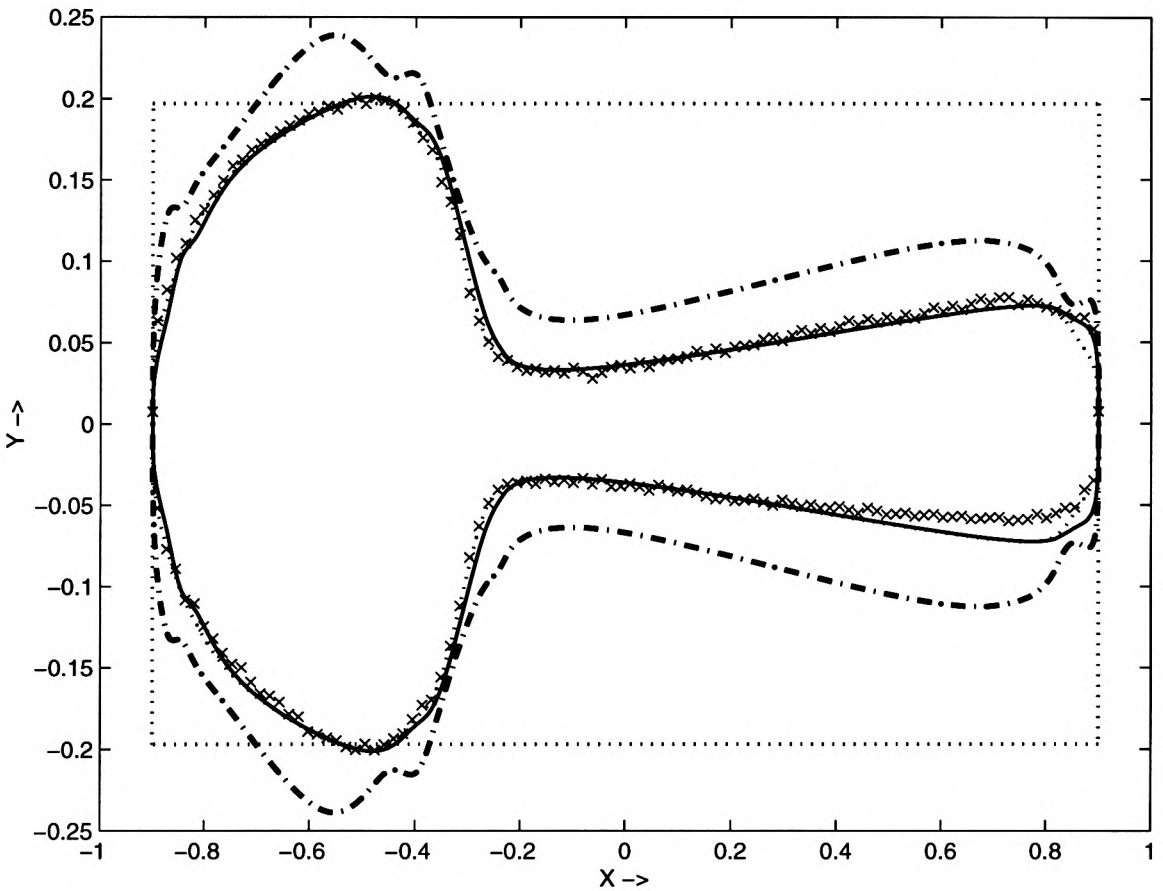

Figure 4.3: *Showing the results of optimizing using 2D error of fit. The spoon model is projected onto the X-Y plane and the shape curve is optimized. The dashdot(-.) line shows the noisy initial silhouette for the model, the solid line the optimized silhouette and x marks the silhouette from range data. The dotted rectangle indicates the rectangular shape derived from the box model. Optimization of this produces the dotted(..) curve which is virtually indistinguishable in most regions from the solid line.* □

Results: Figure 4.3 shows an example of our technique. The (tessellated) spoon model was projected onto the X-Y plane and the shape curve was optimized. The x marks show the silhouette for the range

data. The dashdot(-.) line shows the initial conditions produced by adding random noise to parameters estimated using the curve-fitting techniques to be discussed later. The noise was added to validate the usefulness of the method for far off initial guesses. The solid curve is what we get after optimization and is seen to be quite close to the data. Note that while our specific model is symmetric about the X axis, the range data is not similarly constrained, which accounts for most of the final error. Analytic evaluation of the objective function gave similar results.

The dotted rectangle shows the guess for the box model. The dotted curve(..) shows the results of optimizing this trivial initial guess and is seen to be virtually indistinguishable from the solid line in most regions, demonstrating the robustness of the optimization procedure. As discussed in section 3, this allows us to derive good estimates even when no model-dependent estimation procedure is available. In order to set up this optimization, we merely moved the Y components of all control points (except at the ends) to the values they would have for the box model. A complete implementation must also locate the positions of the control points, perhaps based on the curvature of the final result. We leave the full development of this approach to future work. The regularizing term discussed in section 4.3 has not been added, so there are small kinks in the optimized result.

4.1.2 3D Error of Fit

Rather than projecting onto a given plane, we can directly compare two 3D data sets from acquired range data (x_i^r, y_i^r, z_i^r) and from the model (x_i^m, y_i^m, z_i^m). For each point on the range data, we will associate a corresponding point on the model—for other applications like texture mapping, we may wish to change the direction of correspondence. The squared distance between these two points averaged over all points of interest will give us our objective function. We will do this by first projecting a given point in the range data (x_i^r, y_i^r, z_i^r) on to a unit cube parameterization (α, β, γ). We then map (α, β, γ) to a point on the model (x_i^m, y_i^m, z_i^m) and compare (x_i^m, y_i^m, z_i^m) with (x_i^r, y_i^r, z_i^r).

Nearest Neighbor: The simplest way to make correspondences is by considering the nearest neighbor in the model to a given point on the range data. Hoppe et al.[16] use a similar idea. However, figure 4.4 illustrates the ill-effects of nearest neighbor based correspondence. The left image shows the way the texture map would appear if the range data were noiseless and exactly matched the model.

On the middle and right, the model has been translated and random noise added as in figure 4.3. The middle figure shows the effect of a nearest-neighbor type correspondence. On the left of this image, for a large region, almost all points map onto the same region of the range data, and a similar effect is observed in cases where the model is badly scaled. In the rightmost figure, we use our fractional mappings to determine the texture map from the range data, and we see that a reasonable result is obtained.

Also note that our comparisons are based on geometric properties of the generative model and range data and are general. By contrast, much of the literature on fitting algebraic surfaces uses a specific algebraic objective function which while having the benefit of being easy to compute and preserving continuity, has the disadvantage that it does not always correspond well to geometric proximity and may depend on the particular parameterization used. We believe that our approach can be beneficial in any context where two datasets need to be compared and matched.

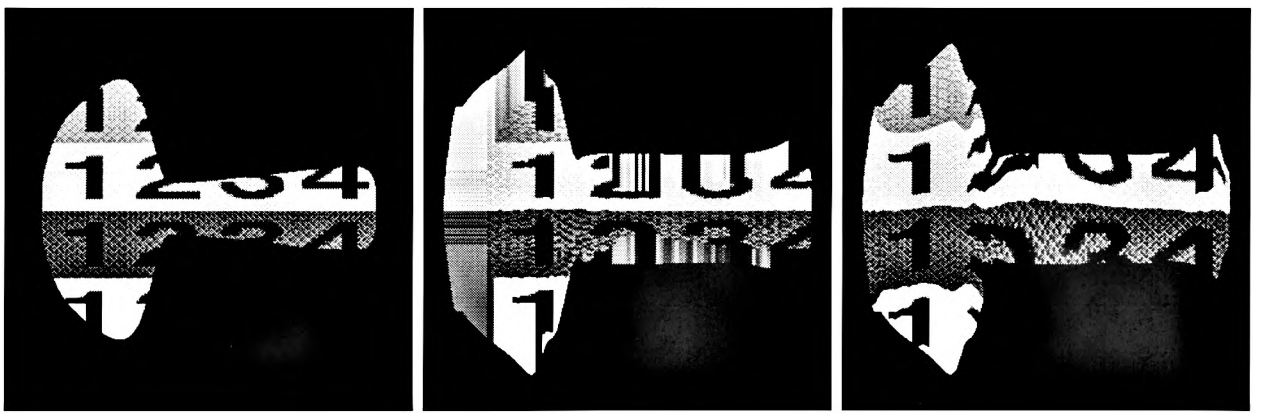

Figure 4.4: **Left:** *Texture map if range data were noiseless, and the model matched it exactly.* **Middle:** *Nearest-Neighbor when model is translated and has noise added.* **Right:** *Texture map using fractional mappings.* □

Corresponding Points: Using the *fractional distance* in x, y and z is simple and has some advantages over the nearest-neighbor method for finding good correspondences. The fractional distance is essentially the ratio of the co-ordinate of a point to the total extent of the data in that co-ordinate. Besides the different correspondence method used, our method differs from that in Hoppe et al.[16] in that we optimize directly over model parameters, not over individual points of the tessellated model. Thus, our objective function is generally nonlinear. Instead of iterating over minimization and repro-

jection, we introduce a general technique for making the correspondences vary C^p continuously with respect to the model parameters, which will help in the optimization phase.

Fractional Mapping: For simplicity, let us first assume that z^m is a single-valued function of (x^m, y^m). We first map a range data point onto the unit square $[0,1]^2$ (the inverse mapping):

$$\begin{aligned} \alpha(x^r) &= \frac{x^r - X^r_{\min}}{X^r_{\max} - X^r_{\min}} \\ \beta(x^r, y^r) &= \frac{y^r - Y^r_{\min}(x^r)}{Y^r_{\max}(x^r) - Y^r_{\min}(x^r)} \end{aligned} \tag{4.8}$$

We then map (α, β) onto the model (the forward mapping) where we assume that z^m is a known function of (x^m, y^m):

$$\begin{aligned} x^m(\alpha) &= (1-\alpha)x^m_{\Downarrow} + \alpha x^m_{\Uparrow} \\ y^m(x^m, \beta) &= (1-\beta)y^m_{\Downarrow}(x^m) + \beta y^m_{\Uparrow}(x^m) \\ z^m &= z^m(x^m, y^m) \end{aligned} \tag{4.9}$$

Here, $x^m_{\Downarrow}$ is a (C^p continuous) minimum value in x, $x^m_{\Uparrow}$ is the corresponding maximum; and $y^m_{\Downarrow}(x^m)$ and $y^m_{\Uparrow}(x^m)$ are similar for y (as a function of x). The paragraph on continuity will discuss how appropriate smoothness can be achieved in equation 4.9. Since the range data isn't being varied by the optimizer, its maxima and minima can be computed in a straightforward way. The objective function for a single range point is:

$$\phi(\mathbf{p}^r) = \| \mathbf{P}^m(\mathbf{p}^r) - \mathbf{p}^r \| \tag{4.10}$$

where $\mathbf{p}^r$ is a range data point, $\mathbf{P}^m(\mathbf{p}^r)$ is the correspondence generated by the fractional map, and $\| \mathbf{P}^m - \mathbf{p}^r \|$ is an appropriate norm—here, the square of the distance— between points $\mathbf{P}^m$ and $\mathbf{p}^r$. In the recognition phase, it is often more intuitive to use the simple distance norm rather than its square. We average over all range points to get the net objective function.

Generality: The above framework can be used even when z is a multi-valued function of x and y provided we can associate corresponding values of z for data and model. However, the number of values

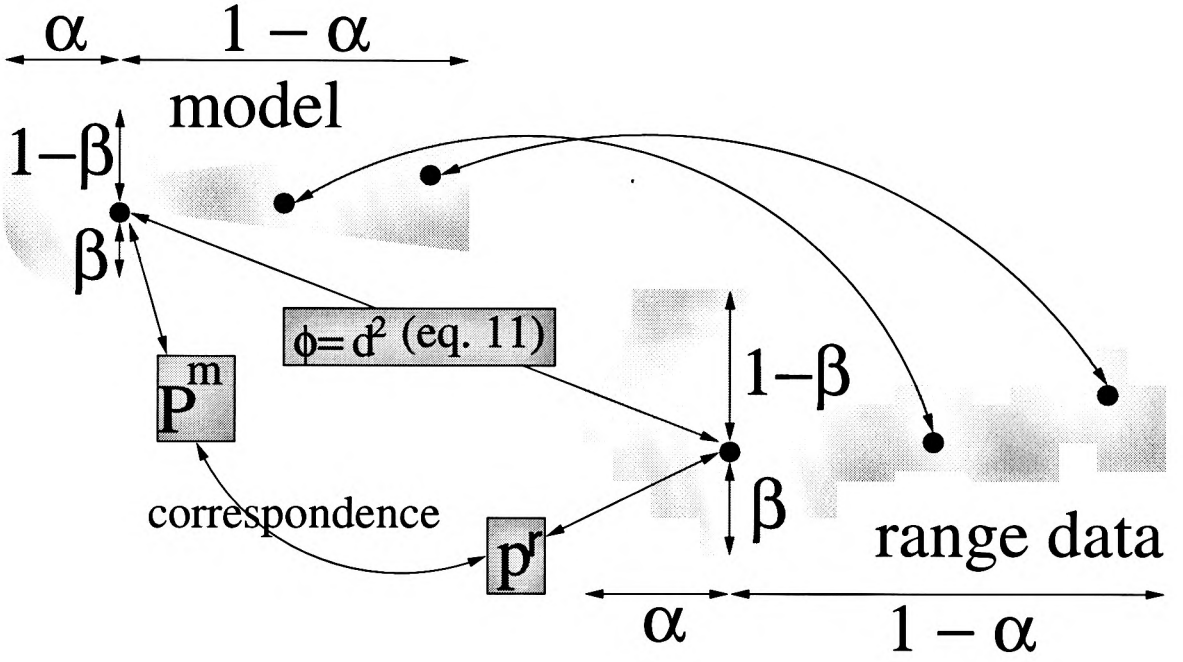

Figure 4.5: *Schematic of fractional mapping. We show the model and range data (as a step-function silhouette). For a point* $\mathbf{p}^r$ *in range data, we first find its fractional distance* α *in* x*. Then, considering the height as a function of* x*, we find the fractional distance* β *in* y*. We map these fractional values back to a point* $\mathbf{P}^m$ *on the model using equation 10. Note that* α *and* β *are ratios, not distances.* d *is the distance between* $\mathbf{P}^m$ *and* $\mathbf{p}^r$ *and its square gives the objective function. We show a few more correspondences. The correspondences act like springs, pulling the corresponding points together.* □

of z^m for given (x^m, y^m) could even change as model parameters are varied. For greatest generality, we introduce a fractional mapping in z too:

$$\begin{aligned} \gamma(x^r, y^r, z^r) &= \frac{z^r - Z^r_{\min}(x^r, y^r)}{Z^r_{\max}(x^r, y^r) - Z^r_{\min}(x^r, y^r)} \\ z^m(x^m, y^m, \gamma) &= (1-\gamma) z^m_{\Downarrow}(x^m, y^m) + \gamma z^m_{\Uparrow}(x^m, y^m) \end{aligned} \tag{4.11}$$

where $z^m_{\Downarrow}(x^m, y^m)$ and $z^m_{\Uparrow}(x^m, y^m)$ are minima and maxima for z and are functions of (x^m, y^m).

The model may be present only in tessellated form and the correspondence $\mathbf{P}^m(\mathbf{p}^r)$ need not map into an actual model point. Thus, we consider a small neighborhood around $\mathbf{P}^m(\mathbf{p}^r)$ to find the objective function, and modify equation 4.10 to read:

$$\phi(\mathbf{p}^r) = \frac{\sum_i [K(\mid \mathbf{p}^m_i - \mathbf{P}^m(\mathbf{p}^r) \mid, t) \parallel \mathbf{p}^m_i - \mathbf{p}^r \parallel]}{\sum_i K(\mid \mathbf{p}^m_i - \mathbf{P}^m(\mathbf{p}^r) \mid, t)} \tag{4.12}$$

where $\mathbf{p}^m_i$ is a point on the tessellated model and $\mid \mathbf{a} - \mathbf{b} \mid$ is the distance between points $\mathbf{a}$ and $\mathbf{b}$. $K(u,t)$ is a non-negative symmetric C^p continuous kernel of finite support t. t should be chosen to have a slightly larger magnitude than the spacing between discretized points of the tessellated model.

Symmetry of the kernel ensures continuity even though the Euclidean distance involves a square root and so has a C^1 discontinuity at 0. Because only a small number of nearby model points will make the kernel non-zero, we need only check a small area of the model.

Assumptions: We expect the forward and inverse mappings to exist. For there to be no $y^m_{\Downarrow}(x^m)$ or $y^m_{\Uparrow}(x^m)$ for a particular x, the model would have to split into disconnected parts, which we disallow. There is a possibility of holes, either when $z^m_{\Downarrow}(x^m, y^m)$ and $z^m_{\Uparrow}(x^m, y^m)$ do not exist or when the denominator in equation 4.12 vanishes. We will show how to smoothly blend holes into the objective function after we discuss simplifications and continuity considerations.

Simplifications: The framework above is general. In case more information is known about the model, it may be simplified somewhat. If analytic results for $x^m_{\Downarrow}$, $x^m_{\Uparrow}$, $y^m_{\Downarrow}(x^m)$, and $y^m_{\Uparrow}(x^m)$ are known, the forward mappings may be computed easily. Usually, z will either be a single-valued function of x and y or have a known number of values for given x and y. This corresponds to γ being a discrete integer variable instead of continuous. In this case, instead of doing the final scaling in z, we can merely associate corresponding discrete values of z, and eliminate the need for using the kernel by just using:

$$\phi(\mathbf{p}^r) = \mid \mathbf{P}^m(\mathbf{p}^r) - \mathbf{p}^r \mid^2$$

Continuity: We must verify that our objective function is smooth in the model parameters. It is sufficient to consider the correspondences as functions of (α, β, γ). We show how the maxes and mins in the forward map (equation 4.9 and the second part of equation 4.11) can be made arbitrarily smooth. Since the optimizer isn't varying the range data, which just defines a set of values in (α, β, γ), it is not necessary to observe smoothness in computation of maxima and minima for the range data.

Equations 4.5 and 4.6 already show how to compute $x^m_{\Downarrow}$ and $x^m_{\Uparrow}$ C^p continuously for two points. This can be symmetrically extended to more points by associativity and symmetrizing. For instance, for three points:

$$\Uparrow(a, b, c) = \frac{\Uparrow(a, \Uparrow(b, c)) + \Uparrow(b, \Uparrow(a, c)) + \Uparrow(c, \Uparrow(a, b))}{3}$$

By choosing the support to be small enough, we ensure that we always consider only a small number of points.

Now, we compute $y^m_{\Downarrow}(x^m)$ and $y^m_{\Uparrow}(x^m)$. Because of the integral below, $y^m_{\Downarrow}(x^m)$ and $y^m_{\Uparrow}(x^m)$ are C^p continuous in the model parameters provided the kernel is C^p continuous. Figure 4.6 depicts the process.

$$\begin{aligned} Y^m_{\min}(x^m) &= \min(y^m_i : \mid x^m_i - x^m \mid \leq t) \\ Y^m_{\max}(x^m) &= \max(y^m_i : \mid x^m_i - x^m \mid \leq t) \\ y^m_{\Downarrow}(x^m) &= \frac{\int_{x^m-t}^{x^m+t} K(x - x^m, t) Y^m_{\min}(x)\, dx}{\int_{-t}^{t} K(u,t)\, du} \\ y^m_{\Uparrow}(x^m) &= \frac{\int_{x^m-t}^{x^m+t} K(x - x^m, t) Y^m_{\max}(x)\, dx}{\int_{-t}^{t} K(u,t)\, du} \end{aligned} \tag{4.13}$$

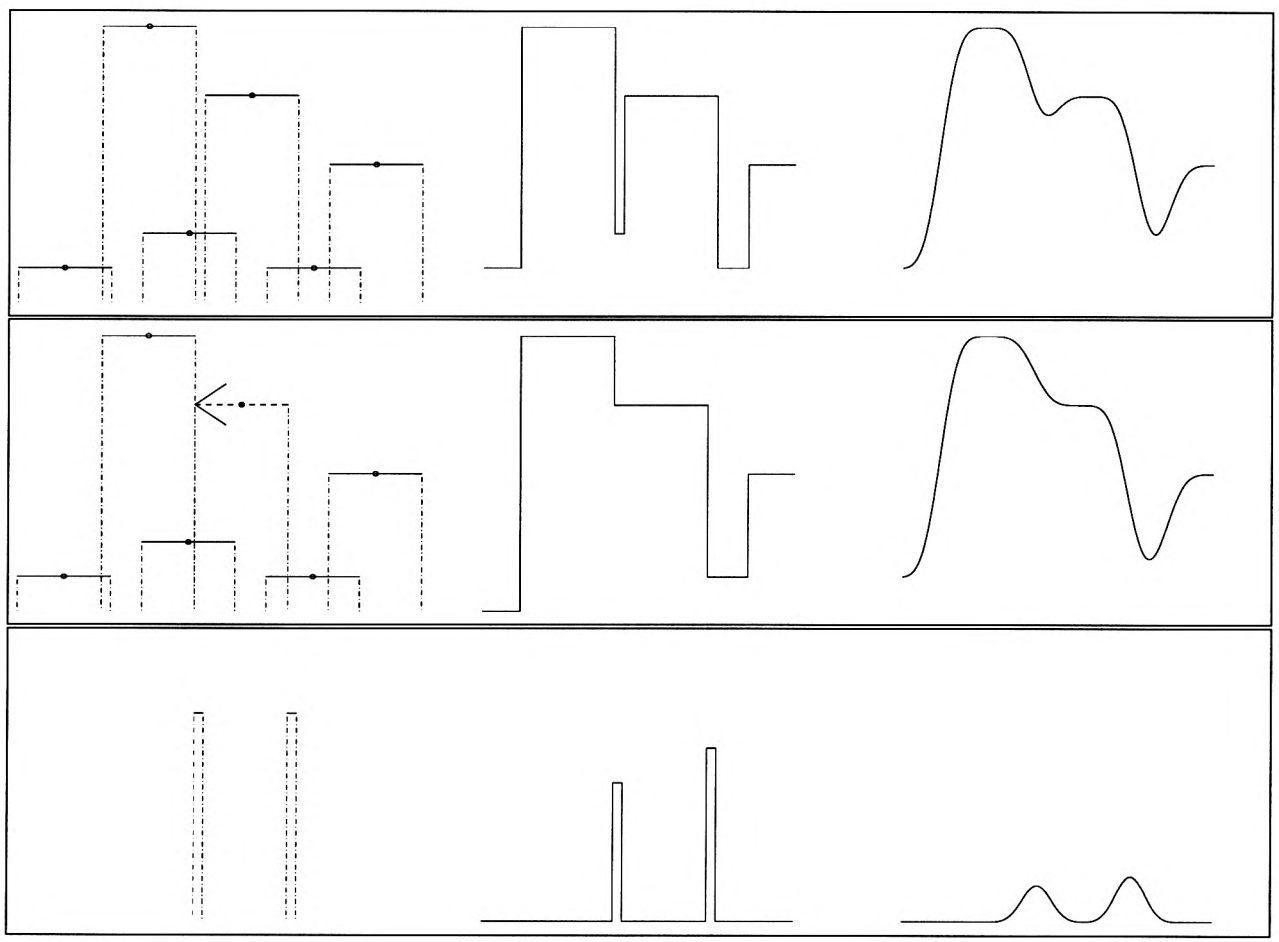

Figure 4.6: *The process of finding silhouettes. The left column of images shows the points as circles with horizontal lines indicating a width of t. The middle column shows the resultant silhouette and the right column the continuous maxima. In the middle row, one point (shown with a dotted line and arrow) has moved slightly to the left as compared to the top row. The bottom left shows the region affected, while the bottom middle and bottom right show the absolute value of the difference between the top and middle rows. The continuous maximum changes smoothly, and with small amplitude, compared to the discrete version.* □

Finally, we must compute the maxima and minima $z^m_{\Downarrow}(x^m, y^m)$ and $z^m_{\Uparrow}(x^m, y^m)$. Our requirements

are that the maxima and minima remain C^p continuous with respect to the model parameters and that they take on the expected values except in certain "boundary regions". We use the following method (we show the computation only for the minimum; the maximum is computed analogously):

$$\begin{aligned} Z^m_{\min}(x^m, y^m) &= \min(z^m_i : \mid (x^m_i, y^m_i) - (x^m, y^m) \mid \leq t) \\ \overline{Z^m_{\min}(x^m, y^m)} &= \overline{\min}(z^m_i : \mid (x^m_i, y^m_i) - (x^m, y^m) \mid \leq t) \\ z^m_{\Downarrow}(x^m, y^m) &= qZ^m_{\min}(x^m, y^m) + (1-q)\overline{Z^m_{\min}(x^m, y^m)} \end{aligned} \tag{4.14}$$

where the bar denotes the lowest value after removing the previous lowest, and the computation is repeated until q is 1. q is a blending function that ensures continuity in case the minimal value slips out of range. One possibility is:

$$q = B(\mid (x^m_i, y^m_i) - (x^m, y^m) \mid -(t-\delta), \delta) \tag{4.15}$$

where δ is a parameter—typically a small fraction of t—that controls the extent of the blend. If there is no model point within a radius t of (x^m, y^m), we apply the hole penalty discussed next.

Holes: In some cases, a range data point may map to a hole in the model. This usually indicates some fundamental incompatibility between the range data and the model (or a stray data point), and we deal with this by assigning a large penalty μ. This occurs either when the denominator in equation 4.12 vanishes or when $z^m_{\Downarrow}(x^m, y^m)$ and $z^m_{\Uparrow}(x^m, y^m)$ do not exist. In the latter case, we already have a blending function that smoothly blends between the hole penalty and the normal objective function (if what were previously a hole were to fall inside the model as a result of varying parameters of the model). In the former case, we can easily create a blending function:

$$\begin{aligned} q &= B\left(\left[\sum_i K(\mid \mathbf{p}^m_i - \mathbf{P}^m(\mathbf{p}^r) \mid, t)\right], \delta\right) \\ \overline{\phi(\mathbf{p}^r)} &= (1-q)\phi(\mathbf{p}^r) + q\mu \end{aligned} \tag{4.16}$$

where the bar on top stands for the modified objective function which will be equal to the original everywhere except in regions near holes and will make the transition with C^p continuity.

4.2 Curve-Fitting

The building blocks of generative models are curves, and most of our estimation algorithms discussed in chapter 5 will yield a set of (noisy) points from determining the silhouette of the data projected on to a given plane. The interpolant of this silhouette will be our desired curve. Below, we show how to represent this curve simply and efficiently.

Step 1. Input interpolated data: We take as input a set of points, that we seek to smooth and interpolate. This comes from the estimation schemes discussed in chapter 5. For notational purposes, we assume these points to lie on the X-Y plane.

Step 2. Filter: Since we expect the data to be noisy, we first filter to smooth it.

Step 3. Compute Curvature: A discrete measure of the curvature of the curve is then computed:

$$\begin{aligned} \triangle x_i &= x_{i+1} - x_i \\ \triangle y_i &= y_{i+1} - y_i \\ \theta &= \sum_i \left| \frac{(\triangle y_i / \triangle x_i) - (\triangle y_{i-1} / \triangle x_{i-1})}{(\triangle x_i + \triangle x_{i-1})/2} \right| \end{aligned} \tag{4.17}$$

where θ is the discrete curvature or importance measure. The curve is divided into a number of regions having equal net curvature, and cubic B-spline control points are added at the boundaries of each of these regions, and tripled at the end-points of the curve. The number of control points is proportional to the net curvature. The values of the control points are the corresponding data values.

Step 4. Optimize: The above steps suffice to choose an appropriate model in the recognition phase. When fitting to the chosen model, we will also optimize the parameters of the control points to minimize an energy function ϕ:

$$\phi = \sum_{i=1}^{N} (y(x_i) - y_i)^2 \tag{4.18}$$

where the right-hand side is a straightforward least-squares approximation to the N points being interpolated. At this time, any model-specific constraints discussed in chapter 5 must also be taken

into account—for instance, requiring that the shape curve be perpendicular to the X-axis at the endpoints.

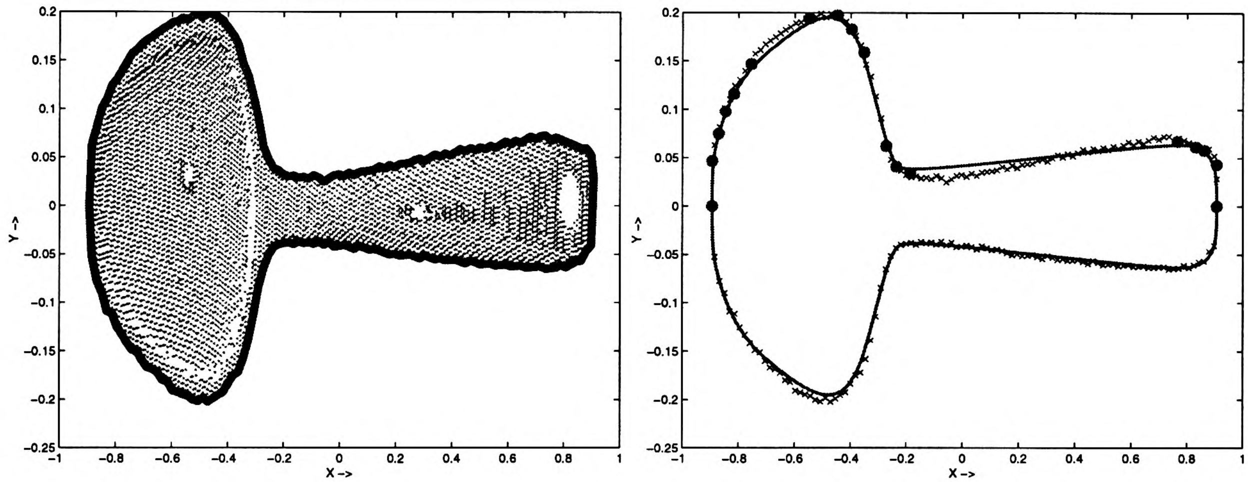

Figure 4.7: *The curve-fitting process. The left figure shows extraction of the silhouette while the right figure shows fitting of a smooth curve to the interpolated data.* □

Figure 4.7 shows the process of fitting. In the left figure, the range data is projected onto the X-Y plane and a (noisy) silhouette is extracted. In the right figure, we show the silhouette with x marks, and the estimated curve with a solid line with control points marked by circles. Since the model is symmetric about the X-axis, control points are shown for only one side, and the other side of the curve is shown only for illustrative purposes and is derived by symmetry. Note that there is a high density of control points in regions of large curvature with a corresponding low density in regions of low curvature.

4.3 Combining Curve-Fitting and Optimization

The simplest and most general method to combine curve-fitting and optimization is to estimate parameters using curve-fitting for model recognition, and then run a global optimization using the 2D or 3D error-of-fit function. A faster variant when the model-dependent parameter estimation is fairly accurate, is to use optimization only to refine the estimate of the co-ordinate axes (so the only parameters are those for translation and rotation and the procedure is relatively fast), and iterate between this and curve fitting (with the optimization in step 4). This corresponds to alternating between optimizing the *extrinsic* parameters and refining the *intrinsic* parameters of the model. It is always desirable to include a regularizing term in the objective function to enforce fairness of the results. Otherwise, noise

in the range data can cause the output curves to exhibit kinks and other undesirable behavior. A simple term based on a notion of curvature for each curve is:

$$\nu \int \left(\frac{d^2 y}{dx^2} \right)^2 dx$$

The constant ν weights the constraint to make its (initial) value of roughly the same magnitude as the other terms in the objective.

It should be emphasized that this is only a crude regularization term. Much further work is needed on determining what constitutes a perceptually suitable result, and adding suitable terms to the objective function.

Chapter 5

Discussion of Specific Model Hierarchy

In this chapter, we review the particular hierarchy we choose to model. A diagram of the tree we use is given in figure 5.1. Models at higher levels (deeper in the tree) are obtained by refining simpler models at lower levels. Following the edges leading from the root node to a particular interior node, one can derive the history of successive refinements.

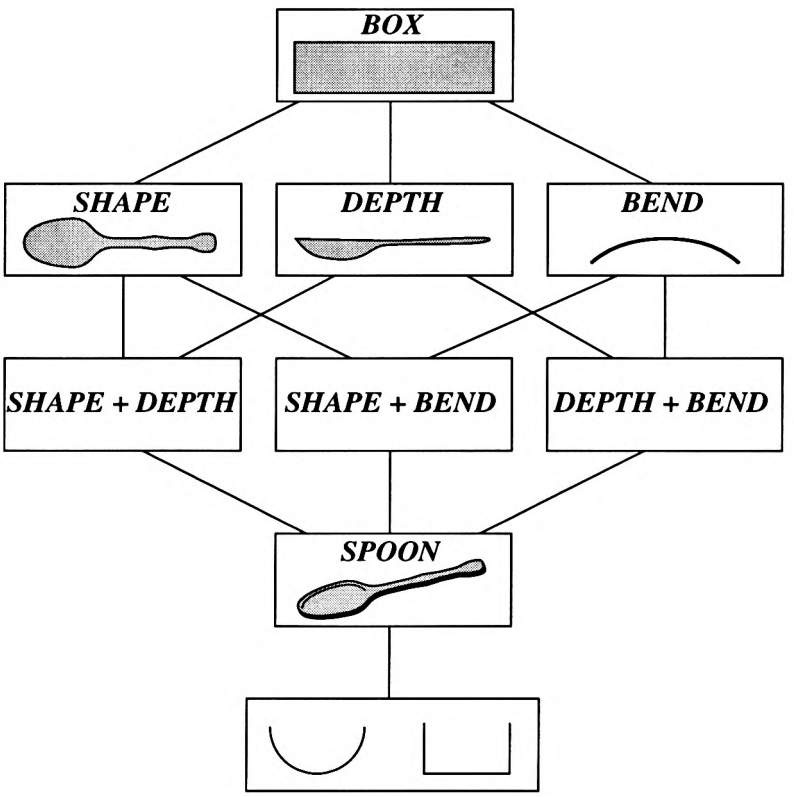

Figure 5.1: *The specific hierarchy we use.* □

5.1 General Concepts

Cross-Sections: We deal with two cross-sections in the Y-Z plane—circular and straight. Refer to the figures below. Cross-sections are defined by two parameters P and H. The circular cross-section is the circular arc connecting $(-P,0)$, $(P,0)$ and $(0,H)$ where $\mid H \mid \leq \mid P \mid$, while the straight cross-section is the similar rectilinear segment connecting the three points.

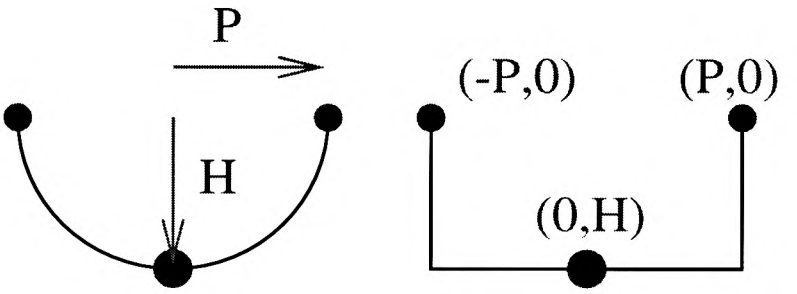

Figure 5.2: *Circular and straight cross-sections. Both pass through points* $(-P,0),(P,0),$ *and* $(0,H)$. □

The tree in figure 5.1 shows a final branch for the two possible cross-sections. As discussed in section 3, conceptually each node in the tree splits into two corresponding to the cross-section used. We simplify the tree somewhat by using the straight cross-section for the recognition phase. Within the framework of section 3, we can effectively treat this as a parent-child linkage except that both nodes have the same penalty for complexity.

Co-ordinate Axes: We find the co-ordinate axis directions by computing the principal axes of the data using the eigenvectors of the covariance matrix as done to compute bounding boxes in [13], or by considering the axes along which the range data has greatest extent—in this approach, we first find the longest axis using optimization, and then the longest axis orthogonal to the first one. In general, we may expect the use of the covariance matrix to be more consistent—for input of a uniformly sampled cube, the resulting axes are the axes of the cube. However, for elongated objects where a significant amount of data is missing, the use of the longest axis is more robust. Note that we expect the input range data to contain points only from the object of interest, not the environment—far off stray points can seriously affect determination of the longest axis. In practice, we enforce this by manual removing stray points in the range data collection phase before invoking the methods described in this thesis.

Axis Labeling: To label the axes, we consider the three co-ordinate planes. The one about which

the data has the greatest symmetry (computed by comparing a reflected version with the original using the error-of-fit techniques discussed in section 3) is the X-Z plane with the remaining axis forming the Y-axis. This follows from the symmetry of this hierarchy. To label the X and Z axes, we require that the silhouette of the range data in the X-Z plane lie parallel to the X axis (as will be the case for the "box" model which is what we are originally trying to fit). To avoid noise, we consider only the regions where the depth of the range data is greatest to compute this silhouette. It must be emphasized that this is only a crude model-dependent estimate, with a more accurate description given after the (hierarchy-independent) optimization step.

Thickness: For a complete solid representation, we must specify a thickness. Since the thickness is typically small, and there are weak clues in the range data regarding it, we do not optimize over it, but instead treat it as a constant value derived from looking at the projection in the Y-Z plane of the acquired data for a suitable X as illustrated in figure 5.3.

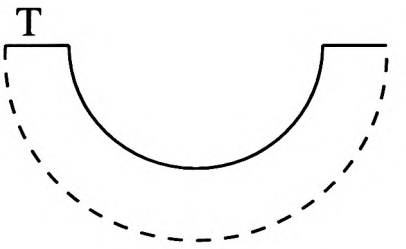

Figure 5.3: *Deriving the thickness.* □

5.2 Shapes Modeled

We now discuss the various shapes and curves inherent to our hierarchy. We will use the straight cross-section for most of these.

Box: There are 3 parameters for the length, width and depth corresponding to the extent of the range data. A mathematical representation for the box as a function of u and v is:

$$BOX = \begin{pmatrix} v\frac{l}{2} \\ ST(\frac{w}{2}, -d, u)[0] \\ ST(\frac{w}{2}, -d, u)[1] \end{pmatrix} \tag{5.1}$$

where $-1 \leq v \leq +1$, and u parameterizes the straight cross-section ($-1 \leq u \leq +1$). The other parameters to ST are P and H. Square brackets refer to the two components (X and Y) of ST. l, w, and d denote length, width and depth respectively.

Generalized Cylinder (Variable Shape): The projection of the box on the X-Y plane is a rectangle of dimensions l and w. Instead of treating w as a constant, it can be made a curve. This is the essence of descending one level down the tree of generative models —a curve is added for a better representation. This curve (which is assumed symmetric about the X-axis) is obtained by considering the silhouette of the projection of the range data onto the X-Y plane. The curve-fitting part of section 3 shows an example of this projection and the derived curve. A mathematical representation is then given by (where the shape is denoted by P):

$$GENCYL = \begin{pmatrix} P(v)[0] \\ ST(P(v)[1], -d, u)[0] \\ ST(P(v)[1], -d, u)[1] \end{pmatrix} \tag{5.2}$$

There are a few special restrictions on P. We assume that $P[1]$ goes to 0 at the end-points and is perpendicular to the X-axis there to ensure derivative continuity of the shape (which is implicitly reflected about the X-axis as a result of the symmetry in the cross-sections). We implement this by placing a triple knot at the end-points with $P[1] = 0$, and also ensuring that the next control point at both ends has $P[0]$ the same as the previous 3 to ensure the curve is perpendicular to the X-axis at the extremes.

Variable Depth: Instead of the depth being constant, it too can be made a curve like the shape. The depth is estimated by considering the silhouette in the X-Z plane. Calling the depth curve "H", we obtain:

$$VARDEPTH = \begin{pmatrix} H(v)[0] \\ ST(\frac{w}{2}, H(v)[1], u)[0] \\ ST(\frac{w}{2}, H(v)[1], u)[1] \end{pmatrix} \tag{5.3}$$

Similar to the generalized cylinder, we want $H[1]$ to go to 0 at the end-points. Also, as shown in figure 5.4, H may go from negative to positive. In case the derivative is discontinuous here, a doubled knot should be placed at the zero-crossing.

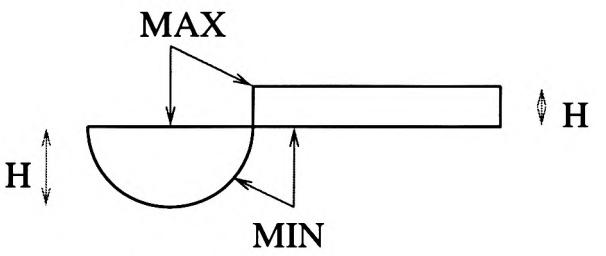

Figure 5.4: *A silhouette in the X-Z plane showing where H goes from being negative to positive.* □

Bend: Figure 5.4 shows the MAX and MIN curves in the X-Z projection. if these curves are not parallel to the X-axis, we must include a bend defined as

$$B(X) = CRV(X)$$

where $CRV = MAX$ when $H < 0$ and $CRV = MIN$ when $H > 0$.

Combinations of these curves can give a variety of models. We give below the representation for the complete spoon model (replacing ST with ARC and adding B for bend parameterized so $B[0] = P[0] = H[0]$).

$$SPOON = \begin{pmatrix} P(v)[0] \\ ARC(P(v)[1], H(v)[1], u)[0] \\ B(v)[1] + ARC(P(v)[1], H(v)[1], u)[1] \end{pmatrix} \tag{5.4}$$

5.3 Simplified Shapes

Chapter 3 discusses how in some cases the curve representations may be simplified. Here, we show how some simpler shapes may be derived through this process, demonstrating the variety of objects that can efficiently be modeled using this simple tree:

- *Right-Circular Cylinder:* Make P(v) a semi-circle in equation 5.2.
- *Hemisphere:* Using a circular cross-section, make P(v) and H(v) identical semi-circles.
- *Surfaces of Revolution:* Using a circular cross-section, set $P(v) = H(v)$. This gives half a surface of revolution. To get full spheres or surfaces, we would need to extend the ARC to draw out the whole circle.

Chapter 6

Results

We demonstrate results from fitting generative models to range data. For illustrative purposes, both the range images and the model were meshed to create the final images for display (colors are artificial). Internally, no meshing is done. Snyder [27] describes alternative methods to image generative models without mesh creation, but at the cost of loss of interactivity.

We use the 3D error-of-fit function ϕ defined by equation 4.12 using the simple distance norm after scaling the range data so the largest axis lies in the range -1 to $+1$. ϕ thus has a clear physical interpretation as the average fractional deviation between the model and the range data. In computing cost C, we use a complexity penalty per level Q of .02 after scaling. For estimation, we use the model-dependent estimation techniques of chapter 5, with parametric curves estimated using the curve-fitting approach of section 4.2. A kernel support of .04 was used in equation 4.12 with a 100-by-100 tessellation. A small blend parameter $\delta = 10^{-3}$ was used in equation 4.15, while a large hole penalty of $\mu = 10^{-1}$ was assigned in equation 4.16.

6.1 Recovered Objects

Spoon: The first model which uses the full generality of the framework is a spoon. A single range image (shown in figure 1.1) obtained using the method of Trobina[30] was used. In figures 6.1 and 6.2, we show the entire process of recovery with reference to the tree of models pictured in figure 1.2.

Bowl: Figure 6.3 shows the recovery of a bowl model from a single range image acquired using [30]. In regions where data is missing, the depth curve yields poor results, but the system automatically

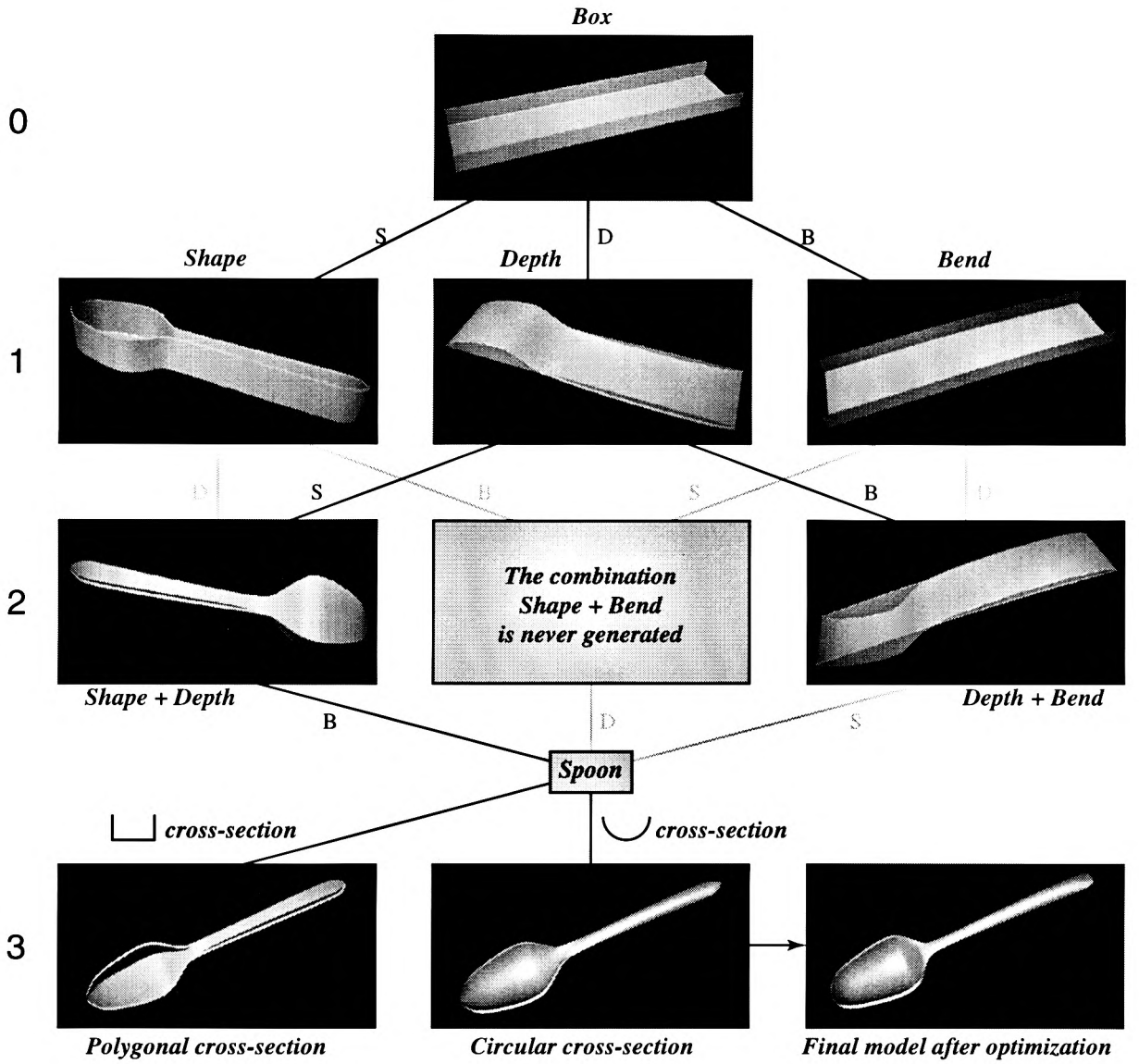

Figure 6.1: *Recovery of the spoon model. We show the tree over which we do recognition with inset images. Highlighted nodes were the best fit at some point, while nodes which are never reached (combination of shape and bend neither of which are ever best-guess) are not shown. The greedy algorithm requires that for a node to be reached, it be directly connected to a best-guess node, and these are the only links followed. Other links are shown in a lighter color. The final branch for straight and circular cross-sections is also shown where we pick the circular cross-section which has lower error, and optimize to get a best-fit model.* □

recognizes the symmetry of this curve about the vertical axis in the simplification phase (step 6 of the algorithm framework in section 3), and fills in the gaps. The system also automatically determines that the bowl shape is circular and that no significant benefit is gained from using a non-zero bend. We show error-of-fit data in figure 6.2 after splitting each node corresponding to straight and circular

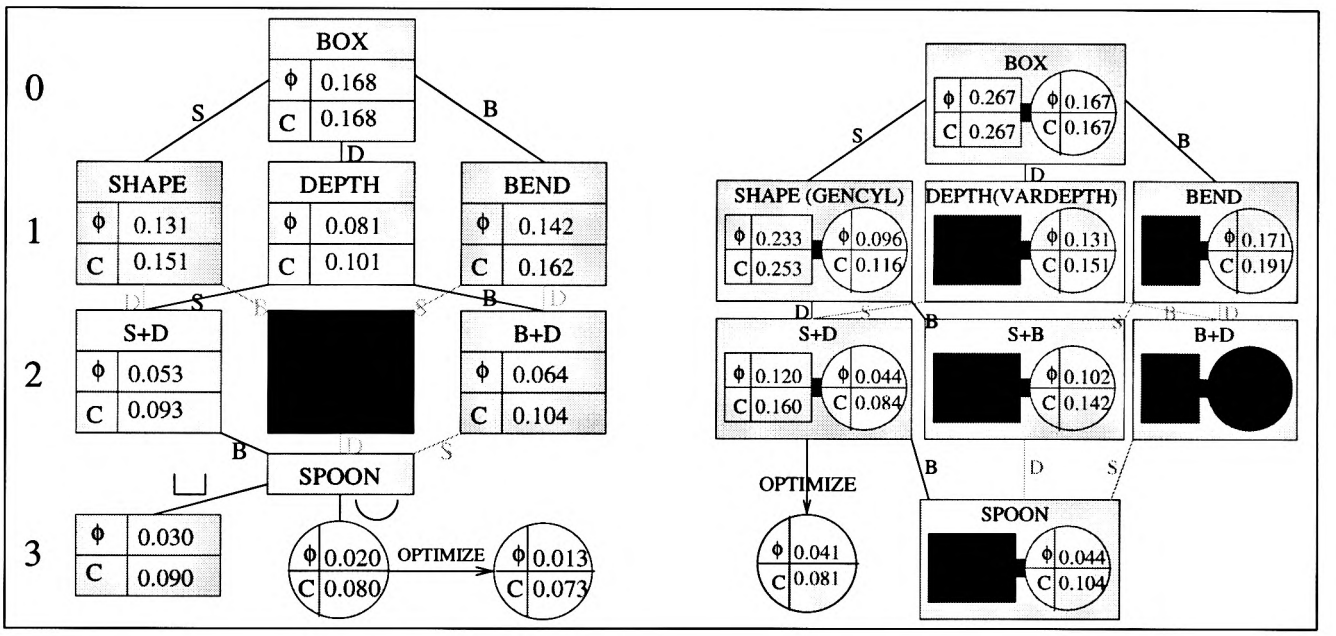

Figure 6.2: *Trees for recognition of the spoon (left) and bowl(right) with corresponding error-of-fits ϕ and costs C. The complexity penalty per level is .02. Best-guess nodes are shown in yellow; nodes that were never reached in dark red. The spoon tree corresponds to the top figure and the decision between straight and circular cross-sections is made only after recognition. In the bowl tree on the right, each node is split corresponding to straight and circular cross-sections and a choice is made at each level. Links exist corresponding to adding a curve or changing cross-section but not both simultaneously. In the bowl tree, adding bend increases the cost function so our final model is a level 2 combination of shape and depth.* □

cross-sections. Links exist between nodes either when the cross-section is changed *or* a single curve is refined using the same cross-section.

Ladle: In figure 6.4, we show the fitting of a generative model to a single range image of a ladle obtained using the method of Bouguet and Perona[4]. The algorithm does well even when the data is incomplete or noisy.

Cup: Figure 6.5 demonstrates the robustness of our approach even when the model does not match the range data well. We took 6 aligned range images of a cup (shown in different colors). The hierarchy we use is incapable of modeling the handle of the cup, but does a good job on the cylindrical portion while doing something reasonable for the handle. Segmenting the range data (equivalent to adding a union operator to our model hierarchy) and a minor augmentation of our tree of models (addition of full rectangular instead of just a semi-rectangular or straight cross-section) allows us to model the entire cup fairly accurately.

6.2 Compactness

The models discussed typically had fewer than a hundred parameters (the coefficients of the control points for the constituent curves). This is at least two orders of magnitude better than the results for meshing.

6.3 Editing

We demonstrate editing of the recovered spoon model shown in figure 6.1. Figure 6.6 demonstrates the ladle-like shape that we have created and shows how the new curves were obtained from the old ones by moving a few control points selectively in a few regions, making the editing process intuitive and simple.

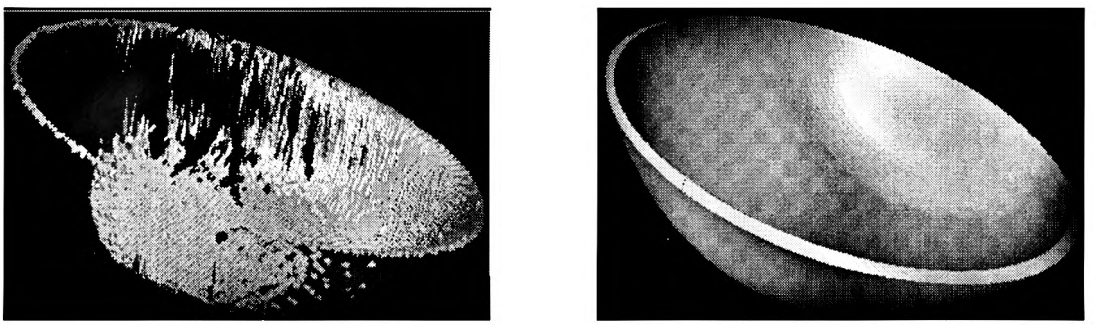

Figure 6.3: *Recovery of a simplified generative model for a bowl (right) from a single noisy incomplete range image (left). The system detects circularity and symmetry and that no bend need be added. As opposed to mesh-based reconstruction, we don't have bad spots where there is no information in the data.* □

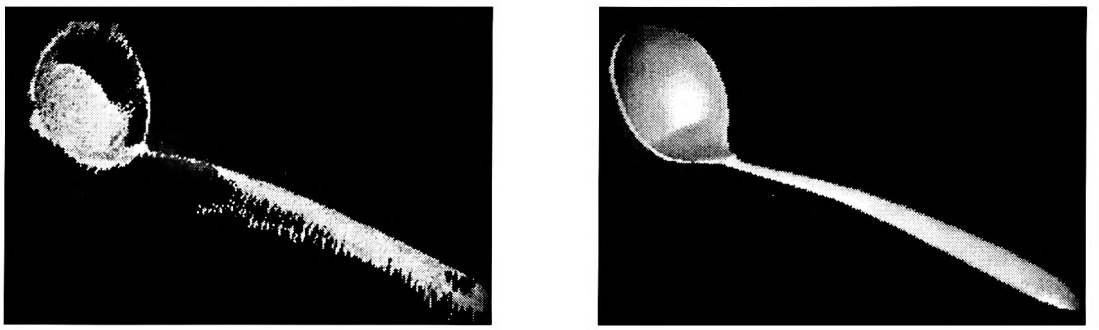

Figure 6.4: **Left:** *Range data for ladle acquired using the technique of Bouguet and Perona.* **Right:** *Reconstructed generative model which is seen to do quite well even where the original is incomplete.* □

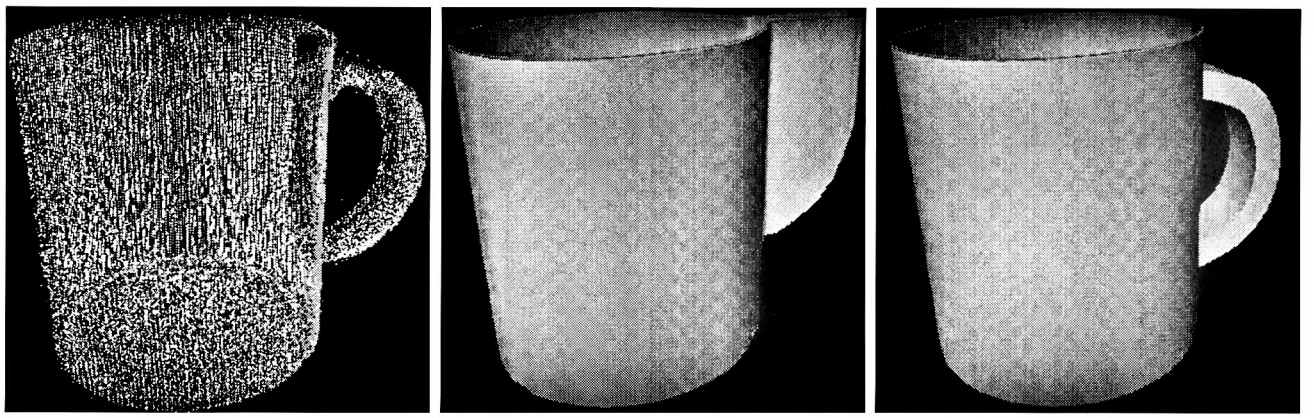

Figure 6.5: **Left:** *Range data for a cup got from aligning 6 views, each shown in a different color.* **Middle:** *Model's best estimate. The cup handle cannot be correctly represented, but the algorithm still does something reasonable, showing the robustness of the technique.* **Right:** *Fairly accurate model derived by segmentation and slightly augmenting our hierarchy.* □

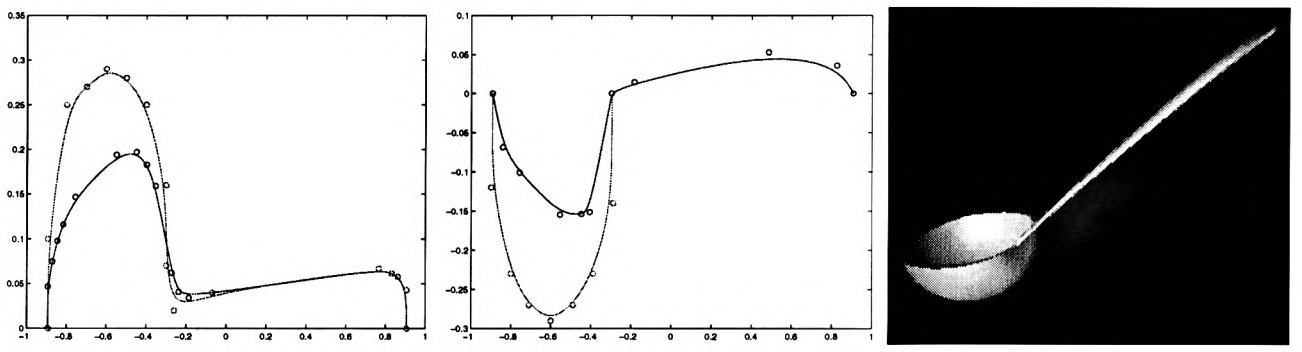

Figure 6.6: *Editing the spoon (recovery shown in figure 5.1) into a ladle-type shape. Only a few control points need to be moved to get a radically different shape.* **Left:** *Edited shape curve before(blue), and after(red) editing. Control points are shown as circles.* **Middle:** *A similar plot for the depth curve.* **Right:** *A view of the edited model.* □

Chapter 7

Conclusions and Future Work

We have presented a new method of going from range data to a model. Instead of deriving a polygonal mesh, we create a simple, intuitive and compact generative model. Demonstrating our ideas on a simple tree of models, we show how the ideas of model complexity and accuracy can be combined to select the appropriate model. Methods of curve-fitting are combined with special techniques for optimization to recover an appropriate optimized model. New models can be created very easily by editing shapes already recovered.

While we believe this is a first step towards an interesting area of future research, much further research is needed. Some limitations of our current system are detailed in section 1.4, and overcoming them will be a topic for future work. More work is also needed on perceptually-based objective functions and complexity-accuracy tradeoffs. Our objective functions based on average deviation between model and range data are currently very simple. As a first step, we could incorporate knowledge about the confidence with which each range data point is known into the objective function by considering the error in data acquisition. Further, a simple norm does not take into account perceptual qualities such as absence of discontinuities or near-discontinuities. Including such perceptual features in the objective function is important. Correct and efficient methods of curve representation especially in the presence of discontinuities is also a direction for future research. For instance, in the cup example before segmentation, a large number of control points are needed because the system does not recognize the discontinuities in the cup profile. Also, a future system would probably use displacement maps[19]

to keep the full complexity of the range data and have the ability to compose generative models to represent more complex objects.

In the future, we would also like the system to automate some tasks that currently require user interaction. In the method described in this thesis, the user must supply a generative hierarchy. It might be possible to automatically derive this hierarchy from a mathematical representation of the objects to be modeled, or to use a large pre-defined hierarchy of models. Further, the user currently supplies the initial guess for parameter optimization. To automate this procedure, we could use a *bootstrapping* technique where the simpler model at the previous stage in the hierarchy is used as a basis for determining the initial guess for model parameters. Chapter 3 discusses this approach briefly and favorable preliminary results are presented in figure 4.3, but we have not yet fully explored this approach.

Our vision encompasses a scenario where the user need merely select an arbitrary object, and the system will automatically supply an intuitive compact representation that is easy to visualize, manipulate and edit. This thesis represents one step toward that goal.

Bibliography

[1] A.H. Barr. Superquadrics and angle preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981.

[2] P.J. Besl and R.C. Jain. Three dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, 1985.

[3] Ruud M. Bolle and Baba C. Vemuri. On three-dimensional surface reconstruction methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1):1–13, Jan 1991.

[4] Jean-Yves Bouguet and Pietro Perona. 3D photography on your desk. In *International Conference on Computer Vision 98 proceedings*, pages 43–50, 1998.

[5] Liang-Hua Chen, Wei-Chung Lin, and Hong-Yuan Mark Liao. Recovery of superquadric primitive from stereo images. *Image and Vision Computing*, 12(5):285–296, June 1994.

[6] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH 96 proceedings*, pages 303–312, 1996.

[7] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH 96 proceedings*, pages 11–20, 1996.

[8] J.E. Dennis, Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice Hall, 1983.

[9] Matthias Eck and Hugues Hoppe. Automatic reconstruction of B-Spline surfaces of arbitrary topological type. In *SIGGRAPH 96 proceedings*, pages 325–334, 1996.

[10] O.D. Faugeras, M. Herbert, and E. Pauchon. Segmentation of planar and quadratic patches from range data. In *IEEE Conference on Pattern Recognition and Image Processing*, 1983.

[11] J. Feder. Plex languages. *Inform. Sci.*, 3:225–247, 1971.

[12] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization.* Academic Press, 1981.

[13] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. A hierarchical structure for rapid interference detection. In *SIGGRAPH 96 proceedings*, pages 171–180, 1996.

[14] Ari D. Gross and Terrance E. Boult. Recovery of SHGCs from a single intensity view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):161–180, Feb 1996.

[15] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH 92 proceedings*, pages 71–78, 1992.

[16] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *SIGGRAPH 93 proceedings*, pages 19–26, 1993.

[17] S. Inokuchi, K. Sato, and F. Matsuda. Range imaging system for 3d object recognition. In *Proceedings of the* 7^{th} *International Conference on Pattern Recognition*, pages 806–808, 1984.

[18] R. A. Jarvis. A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):122–139, 1983.

[19] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. In *SIGGRAPH 96 proceedings*, pages 313–324, 1996.

[20] Senthil Kumar, Han Song, Dmitry Golgof, and Kevin Bowyer. On recovering hyperquadrics from range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1079–1083, Nov 1995.

[21] R.A. Lewis and A.R. Johnston. A scanning laser rangefinder for a robotic vehicle. In *Proceedings of* 5^{th} *International Joint Conference on Artifical Intelligence IJCAI*, pages 762–768.

[22] W.C. Lin and K.S. Fu. A syntactic approach to 3-D object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):351–364, May 1984.

[23] Numerical Algorithms Group, Ltd. *NAG Fortran Library Document*, 1988.

[24] S. Parthasarathy, J. Burk, and J. Dessimoz. Laser rangefinder for robot control and inspection. In *Proceedings of the Society for Photo-Optical Instrumentation Engineers Conference on Robot Vision*, volume 336, pages 2–11, 1982.

[25] F.J. Pipitone and T.G. Marshall. A wide-field scanning triangulation rangefinder for machine vision. *International Journal of Robotic Research*, 2(1):39–49, 1983.

[26] S.A. Shafer and T. Kanade. The theory of straight homogeneous generalized cylinders and taxonomy of generalized cylinders. Technical Report CMU-CS-83-105, Jan 1983.

[27] John Snyder. *Generative Modeling for Computer Graphics and CAD.* Academic Press, 1992.

[28] John M. Snyder and James T. Kajiya. Generative modeling: A symbolic system for geometric modeling. In *SIGGRAPH 92 proceedings*, pages 369–378, 1992.

[29] D. Terzopoulos. Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing*, 24:52–96, 1983.

[30] Marjan Trobina. Error model of a coded-light range sensor. Technical Report BIWI-TR-164, ETH, Zurich, 1995.

[31] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *SIGGRAPH 94 proceedings*, pages 311–318, 1994.