



Single View Reconstruction of Smooth Shapes From Contour

Paria Mehrani and James H. Elder

Technical Report EECS-2017-01

December 20 2017

Department of Electrical Engineering and Computer Science
4700 Keele Street, Toronto, Ontario M3J 1P3 Canada

Single View Reconstruction of Smooth Shapes From Contour

Paria Mehrani and James H. Elder

Department of Computer Science and Engineering
York University

Based on the Qualifying Examination Report of
December 18, 2013

Abstract

Inferring the three dimensional layout of a scene and the shape of objects from a single image is effortless to humans. This has been a long-lasting problem in computer vision. Reconstructing a three dimensional model from a single view has diverse applications, for example, creating novel views from a photograph. This problem is naturally ill-posed and various assumptions have been made to limit the set of solutions. This paper reviews single view reconstruction methods focusing on the problem of shape-from-contour.

Contents

List of Figures	3
List of Tables	5
1 Introduction	6
2 Piecewise Planar Models	11
2.1 Single view metrology [11]	12
2.2 Automatic photo pop-up [15]	16
2.3 Learning depth from single monocular images [18]	20
2.4 Notes on piecewise planar models	23
3 Preliminary Concepts	29
3.1 Differential geometry concepts	29
3.2 Shape from contour terminology	39
4 Shape Theories	44
4.1 Problem definition	44
4.2 Theory of shape-from-contour	46
4.3 Note on shape theories	48
5 Skeleton-based Methods	49
5.1 Playing with Puffball [2]	50
5.2 Teddy [1]	51
5.3 Notes on skeleton-based Methods	56
6 Optimization-based Methods	59
6.1 Visible surface reconstruction	60
6.1.1 Reconstruction of free-form scenes [3]	60
6.1.2 Shape, albedo and illumination recovery [4]	62
6.1.3 Shapecollage [5]	66
6.1.4 Notes on visible surface reconstruction methods	71
6.2 Solid object reconstruction	71
6.2.1 Deformable models	72
6.2.2 Volume occupancy models	77

6.2.2.1	Non-parametric object reconstruction [7]	77
6.2.2.2	Object reconstruction via Cheeger sets [8]	80
6.2.3	Coordinate mapping model [9]	84
6.2.4	Triangulated model [10]	87
6.2.5	Notes on solid object reconstruction methods	93
6.3	Notes on optimization-based methods	95
7	Conclusion	101
A	Total Variation	105
B	Skeleton and Medial Axis Transform	109
C	Bezier Curves	112
C.1	Bernstein Polynomials	112
C.2	Bezier Curves	113
	Bibliography	116

List of Figures

1.1	Various objects can give rise to the same image.	7
1.2	A classification of methods for reconstructing a smooth 3D model from a 2D contour.	10
2.1	Computing the distance of arbitrary points in the world.	13
2.2	Measuring height.	14
2.3	An example of a reconstructed 3D model by Criminisi et al..	15
2.4	An example of results by Hoiem et al..	16
2.5	Different steps of the photo pop-up algorithm.	17
2.6	Reconstruction of the 3D models from label map by Hoiem et al..	20
2.7	Extraction of feature vectors by Saxena et al..	21
2.8	Depth maps computed by Saxena et al..	24
2.9	Example of results by Saxena et al..	25
2.10	A failure example of the method by Hoiem et al..	27
3.1	A surface is a mapping from a subset of \mathbb{R}^2 to \mathbb{R}^3	30
3.2	Changes of the surface normal when moving on the surface.	34
3.3	Normal and geodesic curvatures of a curve on a surface patch.	35
3.4	The principal directions on a cylinder and a saddle shape surface patch.	37
3.5	Rim, outline, silhouette of a surface.	41
3.6	The contour of a self-occluding object.	42
3.7	Cusps of a contour.	43
4.1	Projection of a sphere onto the image plane.	45
4.2	Surface shape at points of positive/negative normal curvature.	47
4.3	Surface with negative Gaussian curvature.	48
5.1	Medial Axis Transform.	50
5.2	Puffball 3D reconstruction of a hand.	51
5.3	Puffball and self-occlusion.	52
5.4	Operations available through the Teddy interface.	53
5.5	Finding the spine of the closed curved in Teddy.	54
5.6	Steps of inflating the object in Teddy.	55
5.7	Reconstruction of 3D models using the Teddy interface.	55
5.8	Comparison of reconstruction between puffball and Teddy.	57

6.1	Constraints employed by Zhang et al. [3]	63
6.2	Reconstruction results by Zhang et al. [3]	63
6.3	The results of the algorithm proposed by Barron et al. [4].	66
6.4	Steps of the shapecollage algorithm.	67
6.5	An example of maps for each patch	68
6.6	Shapecollage depth labeling.	69
6.7	Examples of the reconstructed normal maps in shapecollage.	70
6.8	Deformations of the model using the method by Terzopoulos et al. [6].	72
6.9	Parameterization of the spine and tube suggested by Terzopoulos et al.	74
6.10	The effect of the function h in Equation 6.15 on the shape of the object.	79
6.11	Reconstruction process employed by Oswald et al. [7].	80
6.12	Examples of reconstructed surface using Oswald's method [7].	81
6.13	Reconstructed results using Toppe et al.'s [8] algorithm.	82
6.14	Changing the volume of the surface in [8].	83
6.15	Allowing non-smoothness on the surface by user interactions.	83
6.16	Comparison of the methods in [7] and [8].	84
6.17	The contour generator in the parameter space.	85
6.18	User-specified contour segments in the parameter space.	86
6.19	Sensitivity of the model by Prasad et al. [9] to parameter.	88
6.20	An example of a contour with the T-junctions and visible/hidden cusps labeled.	88
6.21	An example of a hidden cusp joining a T-junction and a visible cusp.	90
6.22	Clustering of vertices in smoothsketch.	92
6.23	Results of the method introduced by Karpenko et al.	93
6.24	An example of a labeled image by Karsch et al. [38]	96
6.25	Average cue ratings.	97
A.1	Recovering the clean image using the total variation norm.	107
A.2	Three similar functions according to TV norm.	108
B.1	The medial axis and MAT of a "B" shape.	110
B.2	The medial axis and grassfire height functions.	111

List of Tables

2.1 Comparison of piecewise planar methods. 28

6.1 Comparison of optimization-based methods. 100

Chapter 1

Introduction

Single view reconstruction is a long-standing problem in the computer vision community.

One of the goals of computer vision is to reconstruct a three dimensional model of scenes or objects from a single 2D image. The task is effortless to humans; we can easily answer questions about the geometry of the scene, determine if an object in the image has self-occlusions, describe what the shape of an object is, etc.

Reconstructing a three dimensional model from a single view has diverse applications, for example, creating novel views from a photograph (e.g. architectural applications), surveillance, and 3D reconstruction of organs for medical purposes. In addition, with the availability of stereo displays and the popularity of 3D movies, the film industry has become interested in converting 2D movies into 3D. This process is labor-intensive, and it would be of interest if computers could automatically perform the conversion.

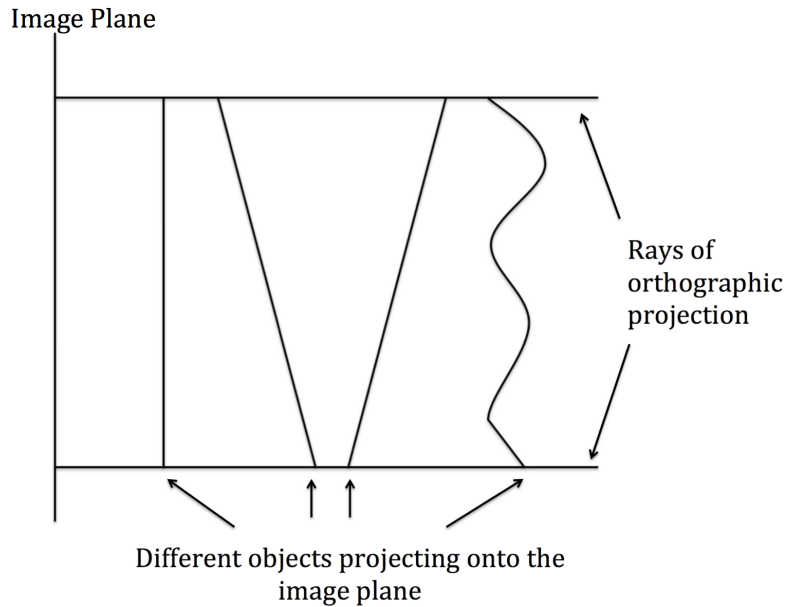


Figure 1.1: Various objects can give rise to the same image.

In general, the 3D reconstruction problem is ill-posed: infinitely many objects can give rise to the same image. Figure 1.1 shows how various objects can result in the same picture. The figure depicts a 1D image with 2D objects.

One approach to deal with the ill-posedness of the problem is to impose constraints on it. Some methods assume that objects present in a scene are piecewise planar. Another possibility, however, is to assume that the objects are smooth. The smoothness assumption is not sufficient for shape recovery; hence, cues extracted from the image and also the knowledge of image formation are employed to reconstruct the 3D shape. These cues may be one or a combination of shading, texture, and/or contour. In the case of using a single cue, the problem is called shape from X, where X is one of the shading, texture, or

contour.

In shape from shading, the goal is to recover the shape from variations of shading on a surface. In this case, various assumptions are made on the reflectance model and the light source. The problem, then, would be to recover the surface shape from the gray level of each pixel in the image.

Shape from texture depends on regular repetition of patterns on the surface. These patterns are assumed to be small enough to be approximately considered planar. The orientation of the surface at each point is recovered from the distortions of the texture.

The problem of shape from contour, described in detail in Chapter 4, is to recover the shape from the occluding contour. The contour is a source of local shape information and limits the set of plausible shapes that gave rise to an observation.

Although these problems have been researched separately and advances have been remarkable, each has shortcomings. For example, assumptions such as presence of a single light source or lack of interreflectance among objects in shape from shading, and presence of textured surfaces in shape from texture, are very limiting. To alleviate these shortcomings, some methods take advantage of a combination of these cues and solve for shape. In this work, however, the focus is on methods for recovery of shape from contour. Once the shape is recovered from contour, additional cues can be employed for further refinement.

Methods for reconstructing smooth objects from contour can be divided into two groups: the ones formulating the problem in an optimization framework, and the ones inflating the silhouette around its skeleton to get a 3D model.

Figure 1.2 presents a classification of methods for reconstructing a smooth 3D shape from an observed contour covered in this paper.

Although these methods impose constraints on the shape of objects, these constraints do not seem to be adequate. As a result, usually additional constraints are imposed through user interactions. In reconstruction of shape from a single image, however, the ultimate goal is to perform the task automatically.

In Chapter 2, I begin by discussing methods assuming piecewise planarity of the objects. Then, in Chapter 3, some relevant concepts from differential geometry and the shape-from-contour literature for smooth shapes are introduced. Chapter 4 is dedicated to discussing theories on reconstruction of a smooth shape from contour. Chapter 5 reviews the skeleton-based approaches. In Chapter 6, optimization-based algorithms are discussed. Conclusions and future directions are discussed in Chapter 7.

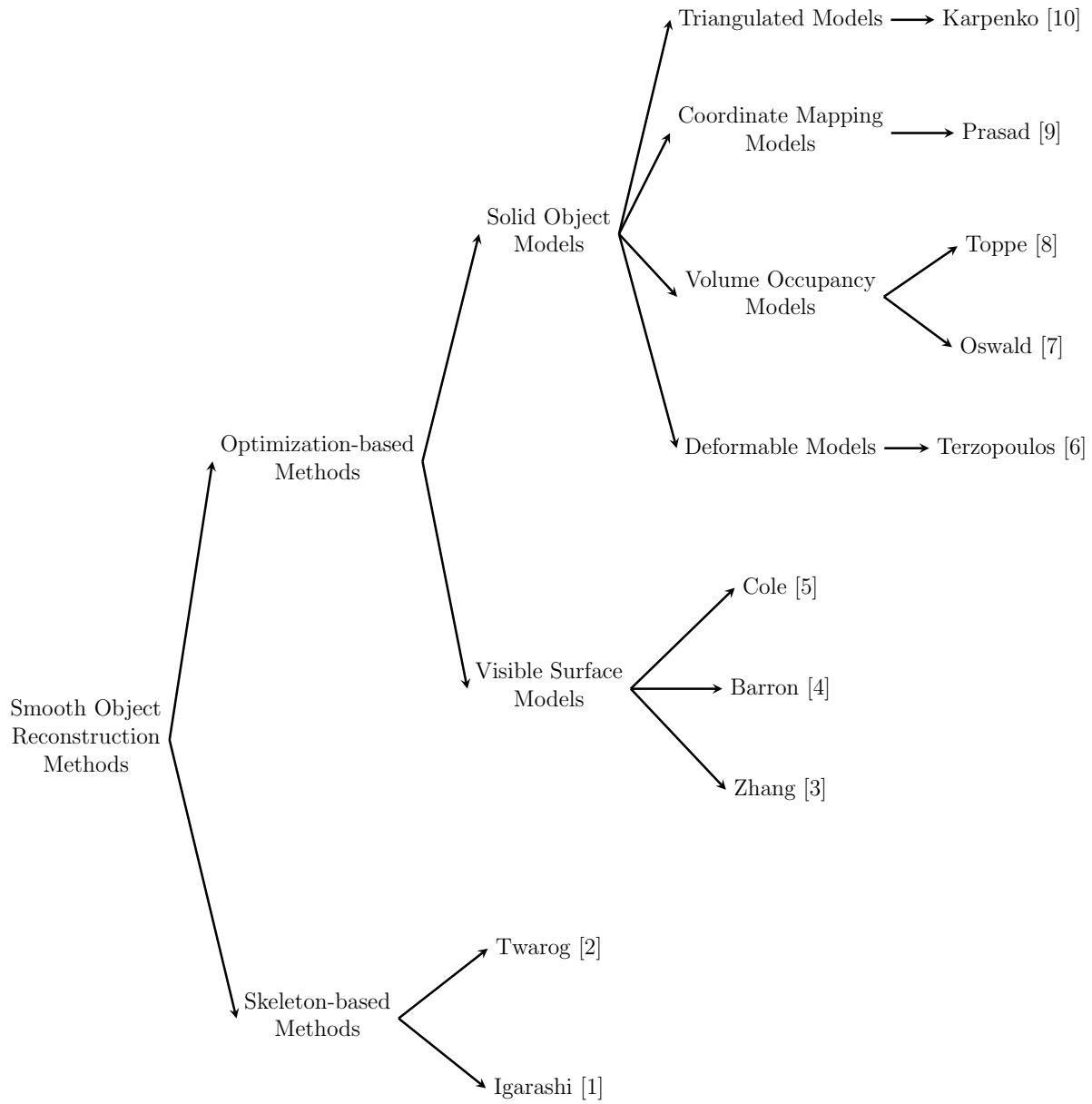


Figure 1.2: A classification of methods for reconstructing a smooth 3D model from a 2D contour.

Chapter 2

Piecewise Planar Models

When the goal is to reconstruct an entire indoor or outdoor scene from a single image, usually piecewise planarity is assumed. The assumption is effective particularly for reconstruction of buildings in an outdoor image or the walls in a room. However, a poor 3D model will be reconstructed in the presence of smooth objects or objects that are not piecewise planar such as leaves of trees in the image.

This chapter describes some of the most influential piecewise planar methods for outdoor scene reconstruction.

2.1 Single view metrology [11]

Criminisi et al. [11] introduced a novel method for extracting the geometric information of a scene from a single image. Although they achieve an accurate reconstruction, their method is highly interactive. It computes an affine structure of points, and with the true measurement of some references, the metric structure can be reconstructed. The single view metrology algorithm reconstructs a 3D model by computing two types of measurements between the image plane and the planar surfaces in the scene:

- Distance between points on a planar surface;
- Distance of points from a planar surface.

Distance between points on a planar surface

Given four corresponding points between the image and a planar surface in the world, one can compute the distance between any arbitrary points on the surface [12]: expressing the points in homogeneous coordinates, one must first find the homography, \mathbf{H} , mapping the points, \mathbf{x} , on the image plane to the points, \mathbf{X} , on the world plane. That is,

$$\mathbf{X} = \mathbf{H}\mathbf{x}. \tag{2.1}$$

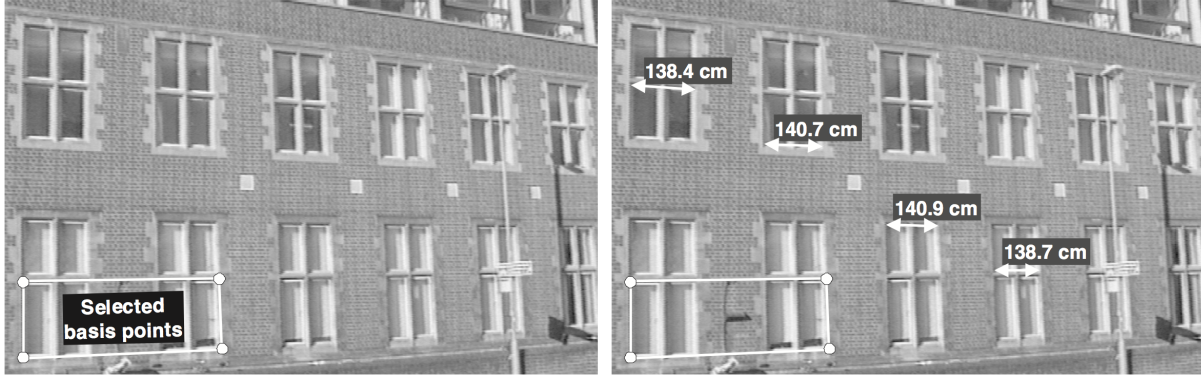


Figure 2.1: The distance of arbitrary points on a reference plane are computed given the correspondence between four points in the world and four points in the image plane.

The 3×3 matrix \mathbf{H} has 8 degrees of freedom. This homography can be computed using the Direct Linear Transformation (DLT) algorithm [12] given four points of correspondence between the image plane and the planar surface. Having computed the homography \mathbf{H} , one can map any two points in the image to the scene plane and compute the 3D distance between the two points. Figure 2.1 shows how the width of the windows can be measured. First, the homography mapping the image points to the corresponding points on the wall is computed. Then, two points on either edge of each window are mapped to their corresponding points in the world. Computing the Euclidean distance of the world points yields the 3D width of the windows.

Distance of points from planar surfaces

The typical goal here is to compute the height of objects, i.e., distance from the ground plane. Two examples of this process are shown in Figure 2.2. Criminisi et al. [13, 11]

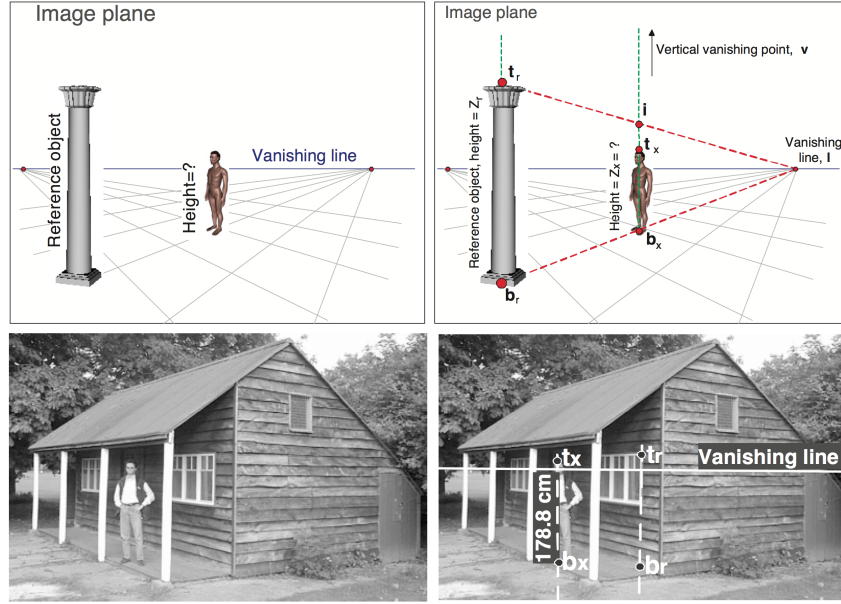


Figure 2.2: Measuring height. Given the height of a reference object, the height of another object can be computed. In the bottom row, the height of the man is computed given the distance of the window top to the ground (adapted from [11]).

showed that if the vanishing point of the vertical direction, \mathbf{v} , the vanishing line of the ground, \mathbf{l} , the top and bottom points of a reference object, \mathbf{t}_r and \mathbf{b}_r , and top and bottom points of the object to be measured, \mathbf{t}_x and \mathbf{b}_x are given, the following equality holds.

$$\alpha Z_i = -\frac{\|\mathbf{b}_i \times \mathbf{t}_i\|}{(\mathbf{l} \cdot \mathbf{b}_i)\|\mathbf{v} \times \mathbf{t}_i\|}, \quad i = r, x \quad (2.2)$$

where Z_r and Z_x are the heights of the reference object and the object to be measured respectively, and α is a scaling factor. Therefore, given the 3D height of a reference object, the value of α can be computed. Then, substituting α into the equation for the object to be measured, the height of the object will be found.

Given the two types of measurement described above, walls and objects in the scene

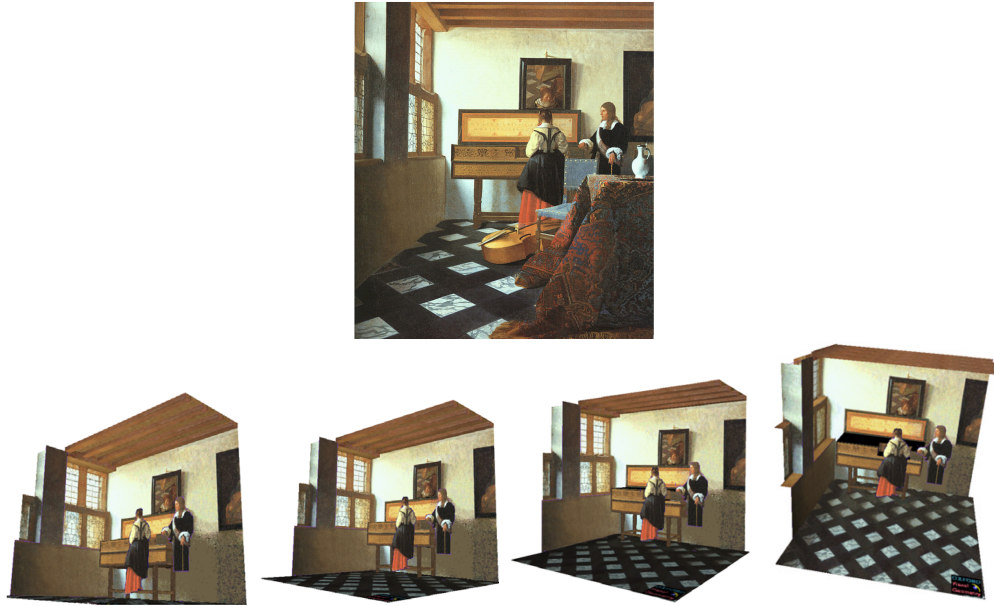


Figure 2.3: An example of a reconstructed 3D model. The first row shows the input image, and the second row shows the reconstructed model. The smooth objects are removed from the model and the ground plane is filled in using texture synthesis algorithms. All the objects in the image, even the human objects, are projected onto planes (adapted from [11]).

can be manually segmented and projected to the appropriate position with the computed height. Usually, smooth objects are removed from the scene. In such an instance, the area the object had occluded is filled in using techniques such as texture synthesis [14].

Figure 2.3 shows an example of a reconstructed image. Note that the smooth objects on the ground are removed and the ground is filled in by texture synthesis algorithms. In addition, although it might not be clearly visible from the images, the two human objects are simply segmented and projected onto vertical planes.



Figure 2.4: Hoiem et al. [15] reconstruct a scene similar to children’s popup books. From left, the original image, and the reconstructed model from two novel views. The green horizontal plane corresponds to the ground and the building is modelled by planes perpendicular to the ground plane (adapted from [15]).

2.2 Automatic photo pop-up [15]

Hoiem et al. [15] introduced an *automatic* method for single view reconstruction of outdoor scenes. However, unlike [11], instead of recovering the precise geometry of the scene, they infer *geometric classes* such as ground, sky, and vertical. A 3D model is reconstructed from the labeled regions of the image. This model is similar to children’s pop-up books: a horizontal plane represents the ground, while perpendicular planes correspond to the planar surfaces of objects in the scene. Figure 2.4 depicts a reconstructed example.

The steps of the geometric class labelling proposed by Hoiem et al. are shown in Figure 2.5. In the first step, the image is decomposed into small homogeneous regions called superpixels [16]. Figure 2.5b shows superpixels of the input image.

The superpixels provide geometric support and have more information than a single pixel. For the purpose of geometric labelling, features such as color, texture, location and shape are extracted from the superpixels. Color helps to distinguish between green grass and blue sky, while location tends to label pixels on top of the image as the sky

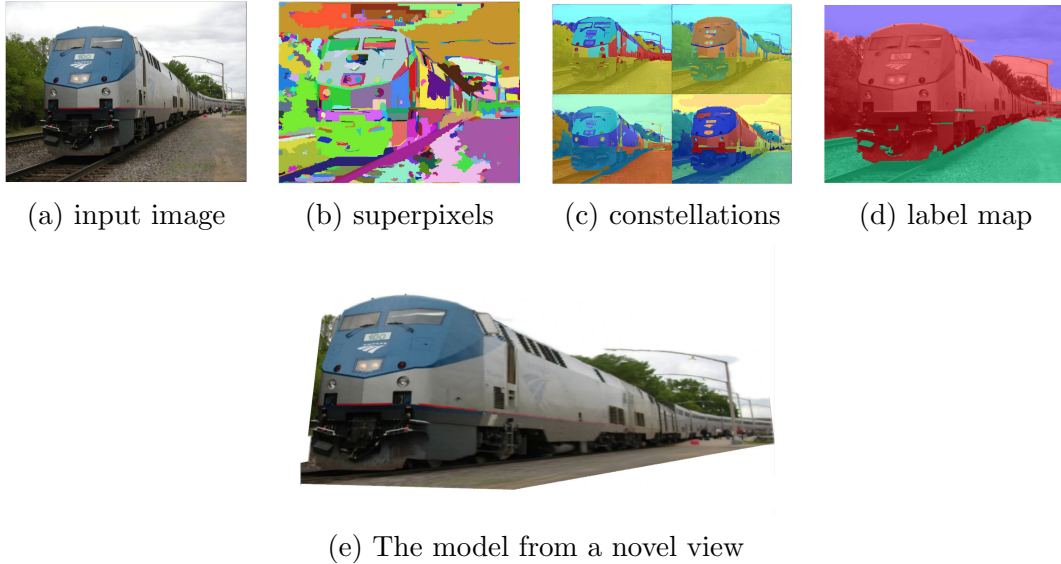


Figure 2.5: Different steps of the photo pop-up algorithm (adapted from [15]).

and the ones in the lower regions as the ground. Texture is useful to distinguish between textured grass and textureless sky. To compute more complex features such as the features capturing the geometry of the region, the superpixels are merged to form larger regions called constellations (see Figure 2.5c).

The constellations are obtained by merging superpixels that are likely to share a common geometric label. Since the constellations are larger regions, a homogeneous labelling is not guaranteed. Therefore, multiple constellations are formed for each superpixel. In particular, the process of assigning superpixels to constellations is performed multiple times to yield multiple constellation configurations for the image.

The process of assigning superpixels to constellations is as follows:

Let us assume N_c is the number of constellations. In [15], N_c varies between 3 and

25. First, N_c randomly chosen superpixels are assigned to these constellations. Then, each remaining superpixel is assigned to the constellation most likely to share its label. In other words, for all configurations of the superpixels assigned to the constellations, Hoiem et al. choose the one which maximizes the average pairwise log-likelihoods¹:

$$S(\mathbf{C}) = \sum_{k=1}^{N_c} \frac{1}{n_k(1-n_k)} \sum_{\substack{i,j \in C_k \\ i \neq j}} \log P(y_i = y_j | |\mathbf{z}_i - \mathbf{z}_j|), \quad (2.3)$$

where n_k is the number of superpixels in constellation C_k , and y_i and \mathbf{z}_i are the i^{th} superpixel label and feature vectors respectively. $P(y_i = y_j | |\mathbf{z}_i - \mathbf{z}_j|)$, the probability of two superpixels having the same label, is learned from the training data using the logistic regression version of Adaboost [17]. Note that, in forming the constellations, while a large value of N_c gives poor spatial support, a small value increases the risk of mixed labels in a constellation.

In the next step, the confidence in each geometric label for each constellation, $P(y_k = l | \mathbf{x}_k, C_k)$, and the homogeneity likelihood of the constellation, $P(C_k | \mathbf{x}_k)$, where \mathbf{x}_k is the feature vector of the constellation C_k , are estimated. These likelihoods are learned by employing Adaboost [17]. The training data in this case contains constellations labeled as ground, vertical, sky, or mixed. The mixed label is assigned when all the superpixels of the constellation are not labeled the same. Then, the label of each superpixel is estimated

¹Although $P(y_i = y_j | |\mathbf{z}_i - \mathbf{z}_j|)$ is the posterior, here, the language of [15] has been followed.

by marginalizing over the constellation likelihoods:

$$P(y_i = t|\mathbf{x}) = \sum_{k:s_i \in C_k} P(y_k = t|\mathbf{x}_k, C_k)P(C_k|\mathbf{x}_k), \quad (2.4)$$

where s_i is the i^{th} superpixel.

Having all the superpixels labelled, each image is transformed into a label map (see Figure 2.5d). The model is created by:

1. finding the boundary polyline between the ground and vertical regions using the hough transform (see Figure 2.6a),
2. discarding the sky region,
3. setting the camera parameters.

Most of the camera parameters are set manually, but remain unchanged for all the images, except for the horizon which is estimated automatically for each image (the yellow dotted line in Figure 2.6b). This line is estimated by intersecting nearly parallel lines and fitting a line minimizing the $L_{\frac{1}{2}}$ -distances. The $L_{\frac{1}{2}}$ has been used due to its robustness to outliers. Finally, the 3D model is reconstructed by folding the vertical region across the ground-vertical boundary so that it stands vertical to the ground plane. In addition, the vertical region is cut along its border with the sky region. An example of the cut and fold step is shown in Figure 2.6b where the solid lines represent the folds and the dashed curve is

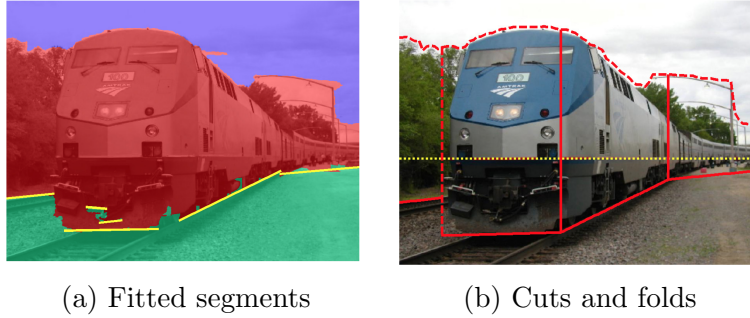


Figure 2.6: Reconstruction of the 3D model from the label map (adapted from [15]).

where the model is cut. The 3D model of the train image can be found in Figure 2.5e.

2.3 Learning depth from single monocular images [18]

Saxena et al. [18], have also developed an automatic method for single-view 3D scene reconstruction. However, instead of reconstructing a piecewise planar model, a depth map is computed and piecewise and vertical smoothness is assumed.

The image is divided into rectangular patches and two types of feature vectors are formed for each patch: features for absolute depth, and features for relative depth. These feature vectors are computed from the monocular features extracted from each patch. The monocular features include texture variation, texture gradient, haze, and color giving a total of 17 features for each patch. The absolute depth features are supposed to capture the global properties of the image. This feature vector contains monocular features extracted at three scales in addition to the monocular features of its four immediate neighbors. Vertical smoothness is enforced by including an information summary of the column the

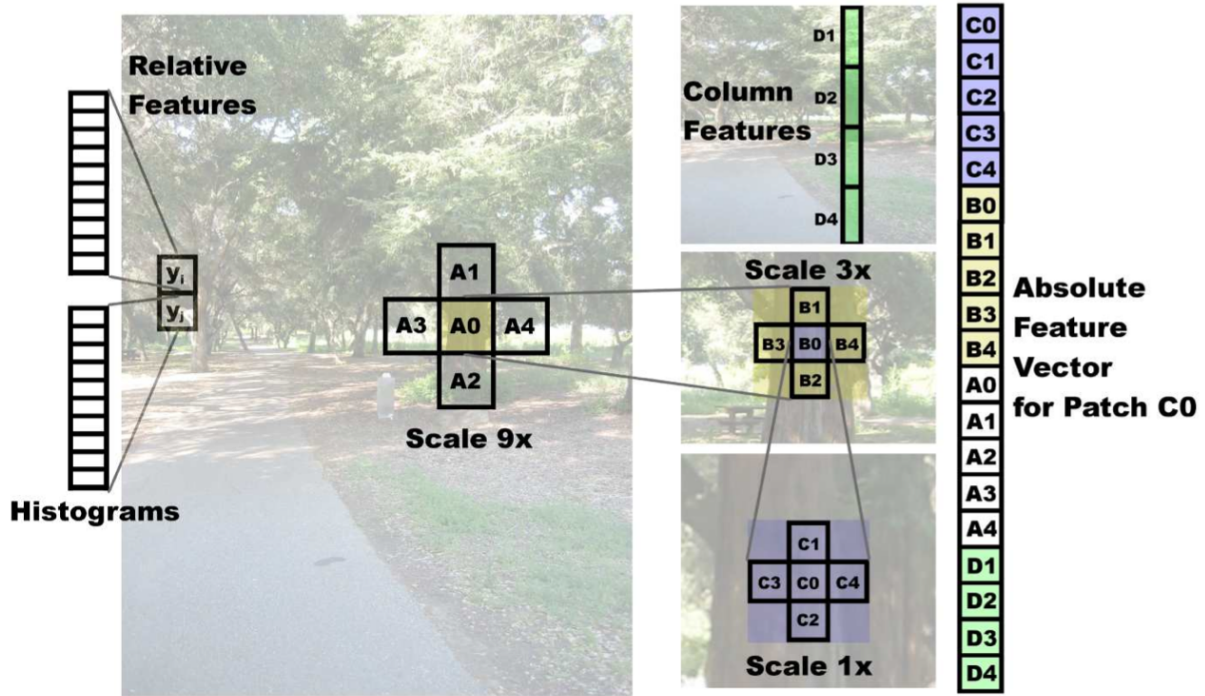


Figure 2.7: The absolute and relative feature vectors extracted at different scales (adapted from [18]).

patch lives in.

In contrast to the absolute depth features, the relative depth features represent the local properties of the image, that is, how two neighboring patches are related. For this purpose, a 10-bin histogram of each of the 17 monocular features for the patch are computed and the difference of this histogram with that of the neighboring patches forms the relative depth features. Figure 2.7 illustrates how the absolute and relative depth features are formed.

The posterior of the depth is then modeled as:

$$P(d|X; \theta, \sigma) = \frac{1}{Z} \exp \left(- \sum_{i=1}^{M_1} \frac{(d_i(1) - x_i^T \theta_r)^2}{2\sigma_{1r}^2} - \sum_{s=1}^3 \sum_{i=1}^{M_s} \sum_{j \in N_s(i)} \frac{(d_i(s) - d_j(s))^2}{2\sigma_{2rs}^2} \right) \quad (2.5)$$

where $d_i(s)$ is the depth of patch i at scale s , $N_s(i)$ is the set of four neighbors of patch i at scale s , M_s is the total number of patches at scale s , x_i is the absolute depth feature vector for patch i , and θ and σ are the model parameters. While the first term in Equation 2.5 models the depth as a function of the absolute depth features at different scales, the second term assures the neighboring patches have similar depths. The relative depth features are employed in estimating the variance σ_{2rs} in the second term. Saxena et al. reported that computing the maximum a posteriori estimate for a given test image takes about 2-3 seconds including feature extraction.

Saxena et al. reported that depth computation is more robust to outliers if a Laplace distribution is employed instead of a Gaussian in Equation 2.5. That is,

$$P(d|X; \theta, \lambda) = \frac{1}{Z} \exp \left(- \sum_{i=1}^{M_1} \frac{|d_i(1) - x_i^T \theta_r|}{\lambda_{1r}} - \sum_{s=1}^3 \sum_{i=1}^{M_s} \sum_{j \in N_s(i)} \frac{|d_i(s) - d_j(s)|}{\lambda_{2rs}} \right). \quad (2.6)$$

Using the Laplace distribution not only has the benefit of robustness to outliers due to the heavier tail of the distribution, but also allows for sharp transitions between regions. Quantitatively, the reported errors for the two models are similar. However, the depth

maps computed by employing the Laplace distribution look more promising. Figure 2.8 shows computed depth maps.

2.4 Notes on piecewise planar models

There are other 3D scene reconstruction methods based on piecewise planar models [19, 20]. However, the three algorithms described here are among the influential approaches and representative of a spectrum from reconstructing the precise geometry of the scene to merely finding the depth map. The spectrum also varies from purely geometrical methods to those relying on photometric appearance cues.

Criminisi et al. [11] and Hoiem et al. [15] explicitly assume piecewise planarity. In contrast, Saxena et al. [18] assumed vertical smoothness by including the features of the column that a patch resides the its feature vector. One can think of this approach as relaxing the piecewise planarity assumption. Note that the shape of smooth objects could potentially be recovered in such a setting. For example, even though the reconstruction is not accurate, Saxena recovered the structure of trees and bushes as multi layer objects. Trees, however, are mapped to a single vertical plane by Hoiem.

Figure 2.9 compares the recovered structure of trees by the two aforementioned methods. In this figure, note that the Laplace model introduced by Saxena considers the highly unstructured leaves as outliers. Therefore, the model mapped the leaves to separate pla-

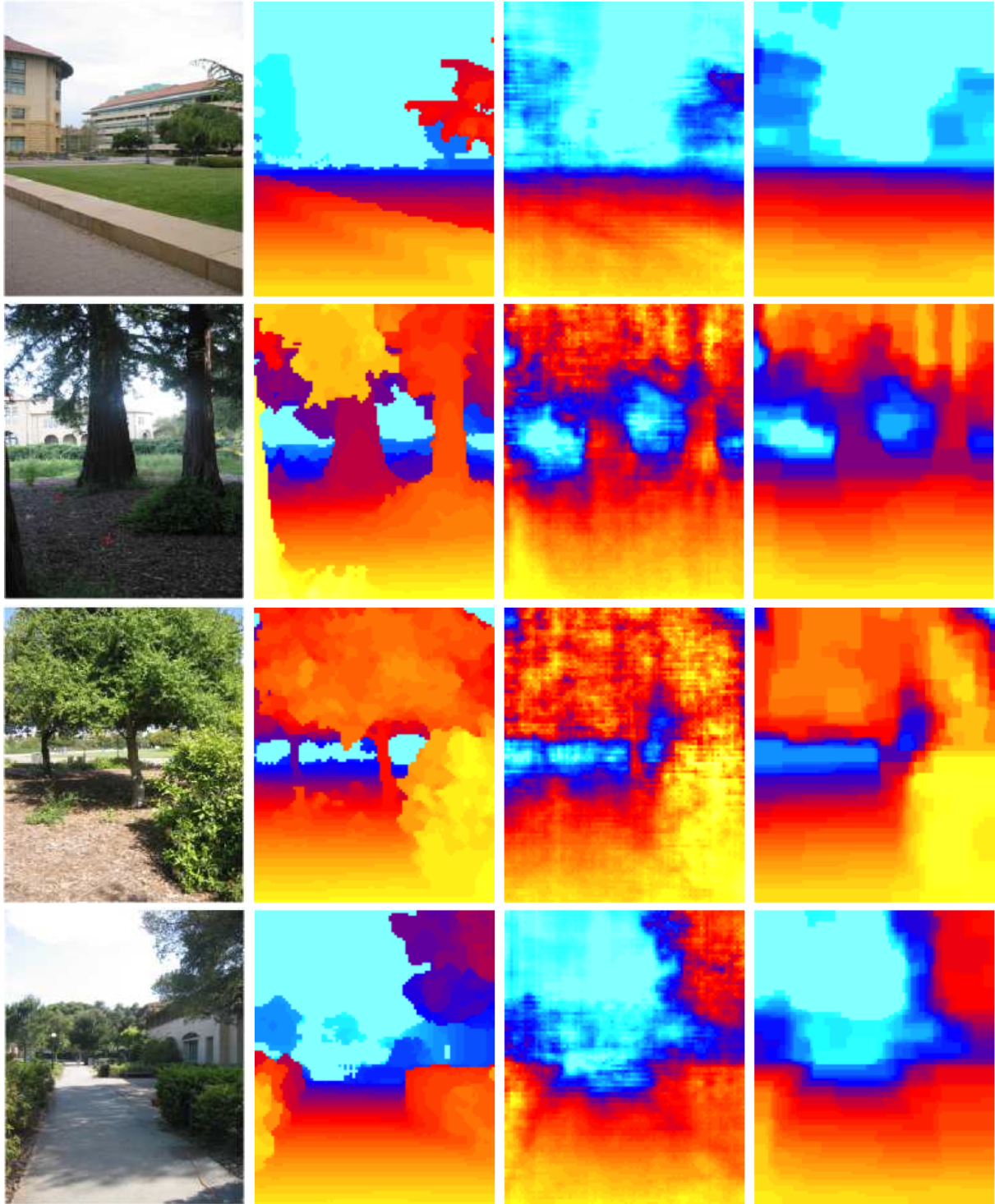
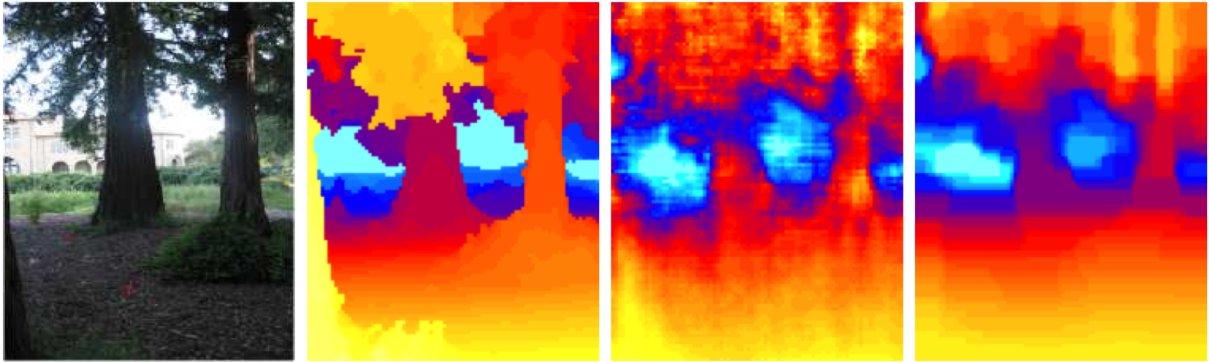


Figure 2.8: Depth maps computed by the method introduced in [18]. From left: the input image, ground truth, the depth map computed by the Gaussian model, the depth map computed by the Laplace model. This figure shows the robustness of the Laplace model to outliers and error in the training data. Depth map pixels vary in color from light red (yellow) for near objects to light blue for far objects (adapted from [18]).



(a) Output of [18]. From left: input image, ground truth, depth map predicted by the Gaussian model, depth map retrieved by the Laplace model.



(b) Output of [15]. From left: input image, reconstructed model from a novel view.

Figure 2.9: Output examples of [18] and [15] for images including trees. Although in [18], the Laplace model has better performance compared to the Gaussian model, the latter results in more realistic depth maps when highly unstructured objects such as trees and bushes are in the image (adapted from [18] and [15]).

nar regions. However, the Gaussian model assigned the leaves to different depth values which resulted in a more realistic reconstruction of the trees.

Unlike Hoiem’s approach which extracts features from superpixels, Saxena’s method extracts features from rectangular patches. Obviously, rectangular patches ignore region boundaries. Consequently, features might be extracted from inhomogeneous patches. In other words, the features extracted from rectangular patches might be noise-corrupted compared to the ones extracted from superpixels and might result in inaccurate localization of depth edges.

Contrary to the other two methods, Hoiem’s algorithm is based on a single ground plane assumption. This restricted model does not work, for instance, in the presence of stairs in the image. Additionally, unlike Saxena et al., who label neighboring patches similarly unless evidence of dissimilarity exists, Hoiem and colleagues do not take the relative relation of each superpixel with its neighbors into account. Strictly speaking, each superpixel is labeled independently from its neighbors, and thus the label map might contain isolated superpixels. To deal with this, Hoiem et al. update such isolated superpixels to match their neighbors. Unfortunately, this heuristic yields undesirable reconstructions when the area the isolated superpixel occupies is relatively large. An example of this effect is captured in Figure 2.10.

Comparing [15, 18] with Criminisi’s work [11], one notices the dependence of the



Figure 2.10: A failure example of the method by Hoiem et al. (adapted from [15]).

latter approach on the input from the user. Other than measuring the distances and marking correspondences, the user is supposed to segment the significant objects as well as walls in the image. This user segmentation could be eliminated by using semi-automatic segmentation techniques [21] for objects, and automatic methods to infer where walls are. Also, note that the success of this method relies upon the accuracy of the automatically extracted vanishing line and points.

In short, we discussed and compared methods for reconstructing 3D models of scenes. Table 2.1 completes this comparison. In spite of vast differences in the algorithms, they all shared a common assumption: piecewise planarity. The assumption works for man-made scenes without smooth objects in the image. Put differently, relatively poor results are obtained in the presence of smooth objects, or when an image of a natural scene is to be reconstructed.

Table 2.1: Comparison of piecewise planar methods. Note that one can have a 2.5D² reconstruction from Saxena’s depth maps with some pre-processing.

Method	Input	Output	Approach	Novel Views ³	Target Structure	Interactive
Criminisi et al. [11]	Single image and user input	Precise geometry	Pure geometric	Yes	man-made	Yes
Hoiem et al. [15]	Single image and learned model parameters	Geometric classes and scaled 3D model	Probabilistic	Yes	man-made and natural scenes	No
Saxena et al. [18]	Single image and learned model parameters	Depth map	Probabilistic	No	man-made and natural scenes	No

²2.5D is a viewer-centered 3D model of the scene from a 2D image. In this model, the surfaces are put in depth. However, there are gaps in the model where no information is available.

³Novel views refers to the views of the scene observed from a different point.

Chapter 3

Preliminary Concepts

Definition of some concepts referred to in this paper are provided in this chapter. These concepts are of two types: differential geometry concepts, and the concepts used by researchers in the area of shape from contour.

3.1 Differential geometry concepts

The ultimate goal of shape from contour is to reconstruct the 3D shape of an object from its 2D bounding contour. Therefore, knowledge of 3D surfaces and their properties is necessary. This section provides the essential definitions. More details can be found in [22].

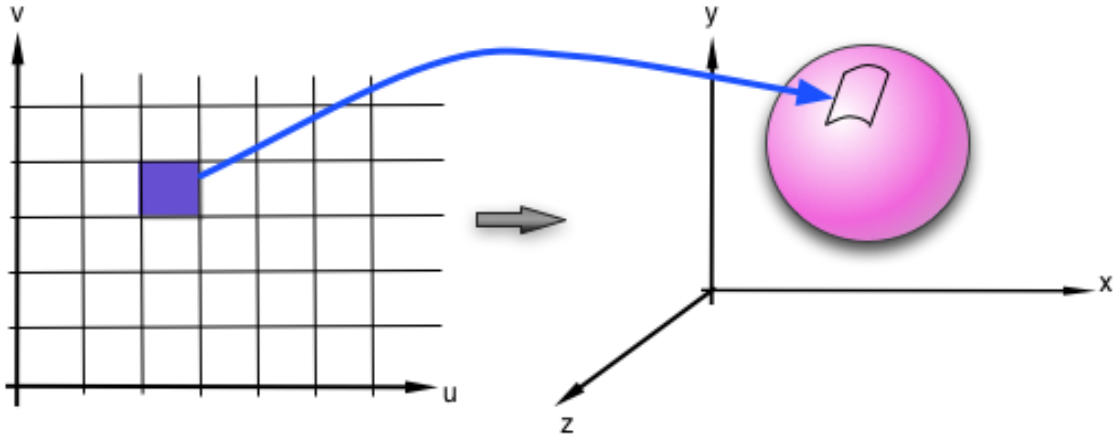


Figure 3.1: A surface is a mapping from a subset of \mathbb{R}^2 to \mathbb{R}^3 .

Smooth Surfaces

A 3D surface, represented parametrically, is a mapping from a 2D space to \mathbb{R}^3 . That is, $\mathbf{f} : U \rightarrow \mathbb{R}^3$, where \mathbf{f} is the mapping and U is the 2D parameter space. Figure 3.1 shows the mapping from $U \subset \mathbb{R}^2$ to \mathbb{R}^3 . A 3D surface, then, can be represented as

$$\mathbf{f}(u, v) = (x(u, v), y(u, v), z(u, v)). \quad (3.1)$$

The mapping \mathbf{f} is called smooth if the partial derivatives of its three components are continuous for all orders. The first and second order partial derivatives of \mathbf{f} are

$$\mathbf{f}_u = \frac{\partial \mathbf{f}}{\partial u} = \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u} \right), \quad \mathbf{f}_v = \frac{\partial \mathbf{f}}{\partial v} = \left(\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v} \right), \quad (3.2)$$

$$\frac{\partial^2 \mathbf{f}}{\partial u^2} = \mathbf{f}_{uu}, \quad \frac{\partial^2 \mathbf{f}}{\partial u \partial v} = \mathbf{f}_{uv}, \quad \frac{\partial^2 \mathbf{f}}{\partial v^2} = \mathbf{f}_{vv}. \quad (3.3)$$

Surface tangents and normals

Let us assume $\gamma(t)$ is a smooth curve on the surface patch σ on the surface S , i.e.,

$$\gamma(t) = \sigma(u(t), v(t)). \quad (3.4)$$

Clearly, the tangent to the curve at a point P is also tangent to the surface patch, and

$$\frac{d\gamma}{dt} = \sigma_u \frac{du}{dt} + \sigma_v \frac{dv}{dt}. \quad (3.5)$$

The two derivatives σ_u and σ_v are special tangents to the surface: the tangent plane of the surface patch σ at point P is the span of these two special vectors. Consequently, the unit normal to the tangent plane at P , called the unit standard normal to the patch σ at P , can be computed from

$$N_\sigma = \frac{\sigma_u \times \sigma_v}{\|\sigma_u \times \sigma_v\|}. \quad (3.6)$$

The first fundamental form

When dealing with surfaces, it is important to measure lengths and areas. The useful tool to this end is the first fundamental form. Koenderink [23] describes the first fundamental form as a machine with two slots. It takes two tangent vectors and returns the dot product of them in the tangent space. The first fundamental form is often represented by a symmetric matrix

$$\mathbf{I} = \begin{bmatrix} E & F \\ F & G \end{bmatrix}, \quad (3.7)$$

where,

$$E = \sigma_u \cdot \sigma_u, \quad F = \sigma_u \cdot \sigma_v, \quad G = \sigma_v \cdot \sigma_v. \quad (3.8)$$

As an example on how the first fundamental form shows up in computation of lengths, consider computing the arc-length of a curve $\gamma(t) = \sigma(u(t), v(t))$ starting from a point $t = a$ to b . Then, the arc-length is:

$$s = \int_a^b \|\gamma'(m)\| dm. \quad (3.9)$$

It can easily be verified that,

$$\|\gamma'\|^2 = Edu^2 + 2Fdudv + Gdv^2, \quad ^1 \quad (3.10)$$

¹Here, and also in Equation 3.11, $du \equiv \frac{du}{dt}$ and $dv \equiv \frac{dv}{dt}$.

and substituting into Equation 3.9 yields

$$s = \int_a^b (Edu^2 + 2Fdudv + Gdv^2)^{\frac{1}{2}} dm. \quad (3.11)$$

Therefore, Equation 3.11 shows how the first fundamental form can be used to compute lengths. The first fundamental form is also called the metric tensor for it gives measurements of length. See [22] for more information on how the first fundamental form helps to compute surface areas.

The second fundamental form

Where surface shape is concerned, the second fundamental form is the helpful tool. By the shape we mean how the tangent plane turns, or equivalently, how the normal to the surface turns, when we move from one point to another on the surface.

The second fundamental form, also known as the shape operator, can be represented as a symmetric matrix

$$\mathbf{II} = \begin{bmatrix} L & M \\ M & N \end{bmatrix}, \quad (3.12)$$

where,

$$L = \sigma_{uu} \cdot \mathbf{N}, \quad M = \sigma_{uv} \cdot \mathbf{N}, \quad N = \sigma_{vv} \cdot \mathbf{N}. \quad (3.13)$$

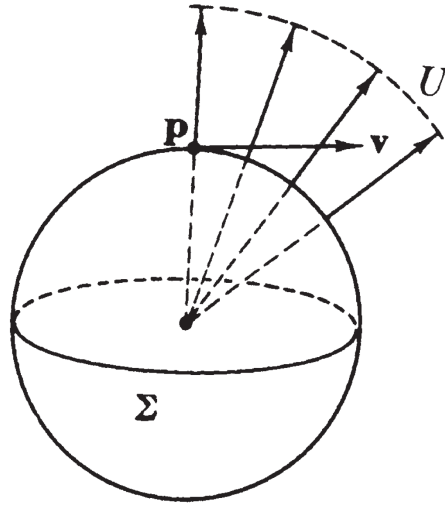


Figure 3.2: The figure shows how the normal to the sphere changes when moving from P in the \mathbf{v} direction. Note that in the text the normal vector is represented by \mathbf{N} . In this figure, adapted from [24], the normal is represented by \mathbf{U} .

The second fundamental form returns a vector representing the direction and the rate of change of the normal to the surface if one moves in the direction of a tangent vector \mathbf{v} at point P . Figure 3.2 depicts the change in the normal to a sphere when moving from P in the direction of \mathbf{v} . Clearly, the inner product of this vector with another vector, say \mathbf{w} , on the tangent plane of P shows the rate of change of the surface normal in the direction \mathbf{w} when moving from P in the \mathbf{v} direction. In other words, one can think of \mathbf{II} as a two slot machine which takes two tangent vectors at a point on the surface. This operator returns the inner product of one of the input vectors and the vector representing the normal change when moving in the direction of the other input vector.

The symmetric matrix of the second fundamental form introduces new geometrical entities such as Gaussian and mean curvatures. In what follows, we explore these important

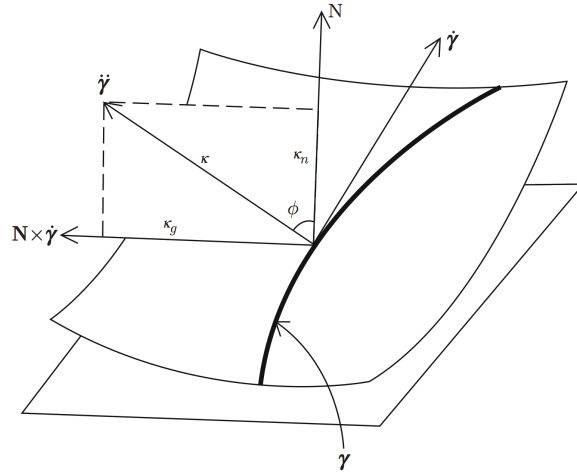


Figure 3.3: Normal and geodesic curvatures of a curve on a surface patch. Note that $\dot{\gamma}$ and $\ddot{\gamma}$ correspond to γ' and γ'' in the text, and denote the first and second derivatives of the curve γ respectively (adapted from [22]).

geometrical entities.

Normal and principal curvatures

The curvature of curves living on a surface is influenced by its shape. Suppose $\gamma(t) = \sigma(u(t), v(t))$ is a unit-speed curve on the surface patch σ . Let γ' be the tangent to the curve, \mathbf{N} the surface normal, and $\mathbf{N} \times \gamma'$ a vector perpendicular to both \mathbf{N} and γ' . As shown in Figure 3.3, the vector γ'' is perpendicular to γ' and hence can be written as a linear combination of the two vectors \mathbf{N} and $\mathbf{N} \times \gamma'$,

$$\gamma'' = \kappa_n \mathbf{N} + \kappa_g \mathbf{N} \times \gamma', \quad (3.14)$$

where κ_n and κ_g are called the normal and geodesic curvatures respectively. The curvature of the curve is given by

$$\kappa^2 = \kappa_n^2 + \kappa_g^2. \quad (3.15)$$

In addition, one can show that the normal curvature in the \mathbf{v} direction is

$$\kappa_n(\mathbf{v}) = \mathbf{II}(\mathbf{v}) \cdot \mathbf{v}. \quad (3.16)$$

Geometrically, the normal curvature represents the acceleration of the curve along the surface normal.

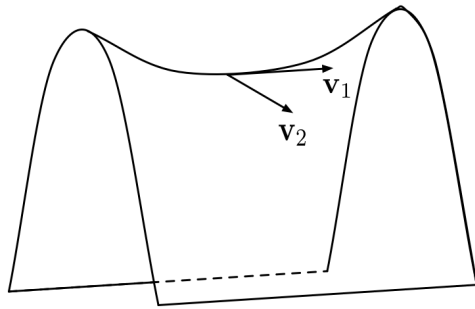
The minimum and maximum values of the normal curvature of a surface at point P are called the principal curvatures, usually denoted as κ_1 and κ_2 . The two unit vectors pointing in the directions of the minimum and maximum normal curvatures are called the principal vectors.

The principal curvatures are important entities in recovering the shape of the surface.

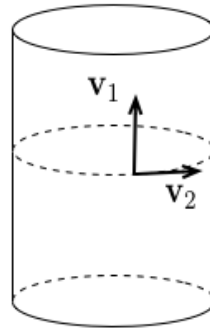
In general, one of the four cases below can happen:

1. κ_1 and κ_2 have the same sign (both positive or both negative). Then, the shape of the surface is convex or concave. An example is a sphere which is convex everywhere.
2. κ_1 and κ_2 have opposite signs. Then, the surface is hyperbolic or saddle-shaped.

An example is shown in Figure 3.4a.



(a) The principal curvatures corresponding to the principal directions \mathbf{v}_1 and \mathbf{v}_2 have opposite signs.



(b) The principal curvature along \mathbf{v}_1 is zero while along \mathbf{v}_2 it is nonzero.

Figure 3.4: The principal directions on a cylinder and a saddle shape surface patch.

3. Only one of the two principal curvatures is zero. Then, the surface is parabolic.

A simple example is a cylinder with zero curvature along the direction parallel to its axis, and nonzero curvature in the perpendicular direction. Figure 3.4b depicts these directions on a cylinder.

4. Both κ_1 and κ_2 are zero. In this case, the point p is called a planar point on the surface. All the points on a plane have this property.

The Euler formula and its dual

The Euler formula and its dual are important in expressing the relation between surface curvature and the curvature of an image contour projected from the surface. This formula expresses the relation between the normal curvature in a certain direction and the principal curvatures. Suppose \mathbf{e}_1 and \mathbf{e}_2 are the principal directions at a point on the surface, and \mathbf{d} is a viewing direction having angle ϕ with \mathbf{e}_1 , i.e., $\mathbf{d} = \mathbf{e}_1 \cos \phi + \mathbf{e}_2 \sin \phi$. Then, according

to the Euler formula, the normal curvature of the surface along \mathbf{d} is

$$\kappa_n(\phi) = \kappa_1 \cos^2 \phi + \kappa_2 \sin^2 \phi. \quad (3.17)$$

Now, consider the curve obtained from intersecting a plane perpendicular to \mathbf{d} and the surface. The dual of Euler's formula [25] expresses the relation between the curvature of this curve, call it κ_c , and the principal curvatures as

$$\kappa_c^{-1}(\phi) = \kappa_1^{-1} \sin^2 \phi + \kappa_2^{-1} \cos^2 \phi. \quad (3.18)$$

The curvature κ_c is called the transverse curvature [25].

Gaussian and mean curvatures

Assuming κ_1 and κ_2 are the two principal curvatures of a surface patch, by definition the Gaussian curvature of the surface patch is

$$K = \kappa_1 \kappa_2, \quad (3.19)$$

and the mean curvature is,

$$H = \frac{1}{2}(\kappa_1 + \kappa_2). \quad (3.20)$$

Gaussian and mean curvatures, also called the intrinsic and extrinsic curvatures, are of importance in the field of differential geometry. Of interest are surfaces with vanishing mean curvature called minimal surfaces. These surfaces have a minimum area subject to some constraints. One can obtain such a surface by passing a wire frame through a soap solution. Later, we will see how the concept of minimal surfaces is employed for shape recovery from a single image.

3.2 Shape from contour terminology

This section includes the definition of concepts² introduced over the years by researchers working on the shape-from-contour problem. Note the variety of terms used by researchers for naming some of the concepts. For the purpose of completeness, all the terms are included here to the best of this author's knowledge.

Rim

Consider viewing a smooth opaque object. Parts of the object are visible to the viewer, while other parts are occluded by the visible portion of the object. The curve on the surface separating the visible and occluded parts of the object is called the rim by Koenderink [23]. It is also called the terminator, the occlusion boundary, extremal boundary,

²Most of the definitions in this section are taken from [23].

or the contour generator [26].

Now, consider the set of points on the surface the visual direction is tangent to. This set is called the pre-image in [23], and the rim in [25]. The rim, in this case, is a set of smooth closed curves (loops) on the surface. The important fact about the rim is that it is a collection of smooth *space* curves—not necessarily a planar curve. Also note that not all of the rim need be visible.

Contour, silhouette, and outline

Koenderink [23] defines the contour as “the image of the rim in the natural projection” and “where the tangent planes project onto lines rather than planes”. In the literature, the contour is also called the apparent contour [26]. Although the rim of smooth objects is a closed smooth curve, the contour is not necessarily smooth.

Koenderink [23] defines a silhouette as “part of the visual field occupied by the projection of the object”. Another useful notion is the outline: the part of the contour that bounds the silhouette.

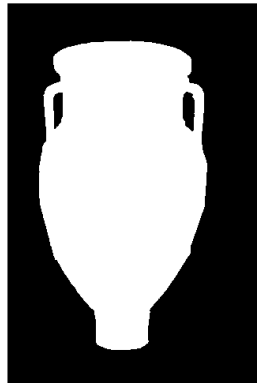
Figure 3.5a shows a 2D image of a vase. The rim, the visible part of which is indicated in red, is not a planar curve. In particular, the top and bottom of the vase are clearly not in the same plane as the rest of the rim. The silhouette and outline of the vase are shown in figures 3.5c and 3.5d respectively.



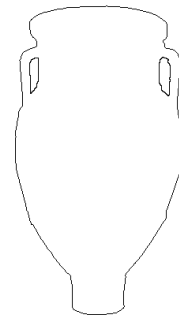
(a)



(b)



(c)



(d)

Figure 3.5: (3.5a) the original image, (3.5b) projection of the visible part of the rim indicated in red, (3.5c) the silhouette, (3.5d) the outline. Image taken from the Berkeley Segmentation Dataset [27].

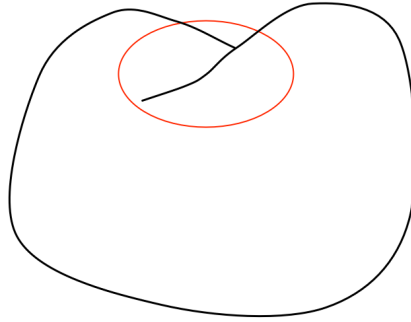


Figure 3.6: The contour of a self-occluding object. The self-occlusion is enclosed by a red curve.

Self-occlusion and cusp

A self-occlusion occurs when a view vector grazes and then penetrates the object. An example of the projection of a self-occlusion is shown in Figure 3.6.

Another interesting concept that we will come across in the following chapters is a cusp. Cusps are singularities of curves. Cipolla et al. [28] illustrate and describe a cusp very clearly. In their words, cusps appear “when a ray is tangent not only to the surface but also to the contour generator”. They also add that, “this will occur when viewing a hyperbolic surface patch along an asymptotic direction.” Figure 3.7, adapted from [28], depicts the cases when a cusp occurs. If the surface is translucent, one can easily see both branches of the cusp. For an opaque surface, only one branch of the cusp is visible and the other is occluded. In this instance, the contour ends [29].

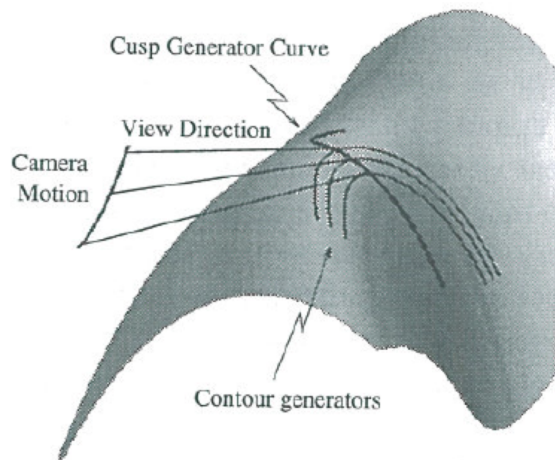


Figure 3.7: Cusps of the contour occur when the viewing direction is not only tangent to the surface, but also tangent to the rim. Note that the authors in [28] study the deformations of the apparent contour as the viewer moves. Hence, the figure depicts the contour generator from three different views. Figure adapted from [28].

Chapter 4

Shape Theories

The rest of this paper focuses on the reconstruction of smooth objects. The input is the contour, sometimes the silhouette/outline. Often, other information such as intensity gradient is used for reconstruction. In this chapter, we define the problem of shape from contour, and discuss theories on how to solve this problem.

4.1 Problem definition

Consider a smooth object projected onto an image plane. Figure 4.1 shows a simple example. The contour of the object may be extracted manually, semi-automatically [21], or automatically (although reliable, automatic object segmentation remains an open problem). Although the semi-automatic and automatic segmentation methods might yield

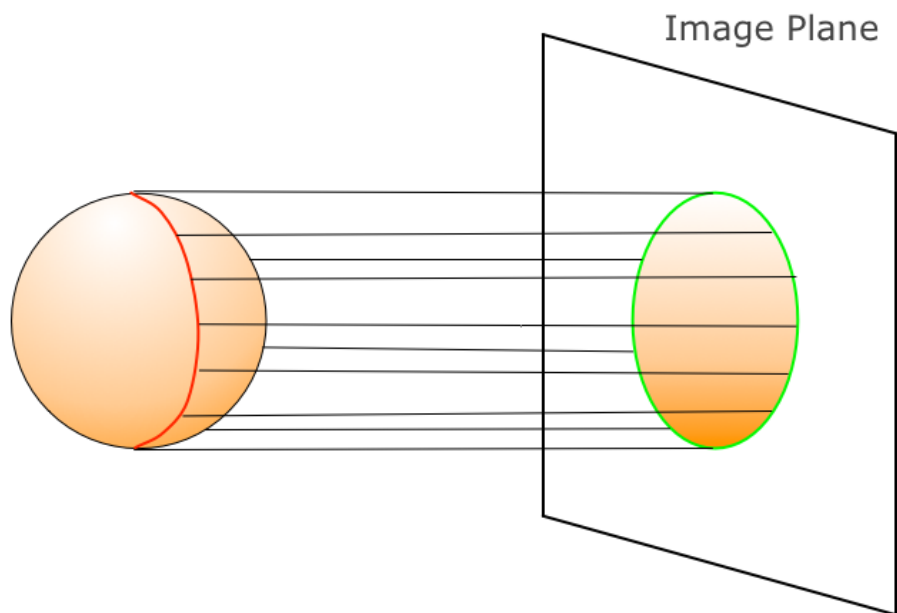


Figure 4.1: Projection of a sphere onto the image plane. The visible part of the rim is shown as a red curve on the sphere while the contour is represented by a green curve. The parallel lines match points on the sphere to points on the contour. Here, orthographic projection is assumed.

noisy contours, here, we assume a noise-free contour is given. Shape-from-contour is then the problem of reconstructing the 3D shape of the smooth object given its 2D contour or outline/silhouette.

Inferring the shape of objects from 2D contours is an effortless task for humans. For example, we can easily estimate the 3D shape of the vase by just looking at the contour in Figure 3.5d. There are mathematically sound theories relating the contour and the shape of the object in the vicinity of the rim [25]. However, there are still infinitely many objects satisfying the conditions at the rim. In short, the shape-from-contour problem is ill-posed, and it calls for constraints and assumptions to limit the solution set.

4.2 Theory of shape-from-contour

Koenderink [25] showed how some properties of the surface can be inferred from the apparent contour. In particular, he observed that the Gaussian curvature, K , of the surface is given by

$$K = \kappa_c \kappa_n, \tag{4.1}$$

where κ_c and κ_n are the transverse and normal curvatures associated with the viewing direction, respectively. This equation can be confirmed by replacing κ_n and κ_c from equations 3.17 and 3.18. Assuming perspective projection, one can verify that the apparent curvature is

$$\kappa_{\text{app}} = d\kappa_c, \tag{4.2}$$

where $d > 0$ is the distance of the vantage point to P . Therefore,

$$\kappa_{\text{app}} = \frac{dK}{\kappa_n}. \tag{4.3}$$

Clearly, from the definition of Gaussian curvature, $K > 0$ corresponds to convexities or concavities, and $K < 0$ represent saddle shape regions. Noting that only positive normal curvature leads to a visual contour (see Figure 4.2), Equation 4.3 constrains the surface shape given the apparent contour. In other words, there are two cases:

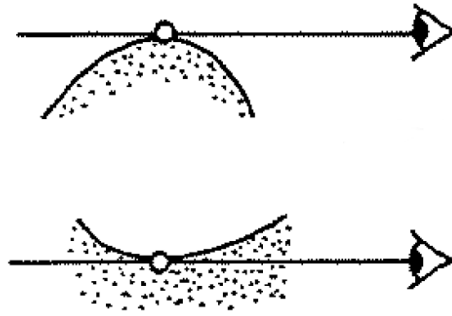


Figure 4.2: The top figure shows a surface with positive normal curvature in the viewing direction. The bottom one shows that negative normal curvature does not lead to a visible contour.

- $\kappa_{\text{app}} > 0$ means the surface is convex at P ,
- $\kappa_{\text{app}} < 0$ means the surface is saddle-shaped at P .

In short, Equation 4.3 states that the sign of the apparent curvature and the Gaussian curvature are the same.

An interesting case is when $\kappa_{\text{app}} < 0$ and the normal curvature vanishes (Figure 4.3).

In this instance, the apparent curvature becomes infinite and the contour has a cusp.

Moreover, the apparent contour ends at this point where the surface is saddle-shaped

(Koenderink et al. [29]).

In summary, Koenderink and colleagues [25, 29] showed how the curvature of the apparent contour *locally* constrains the shape of the surface. However, this constraint is only qualitative, specifying the sign of curvature but not the magnitude. Further, at all other points within the object silhouette, the surface is formally unconstrained.

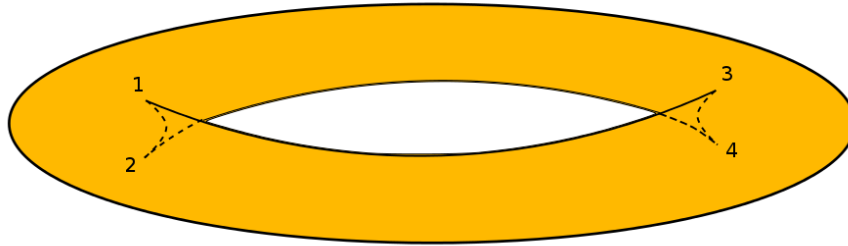


Figure 4.3: The figure shows a bagel with negative Gaussian curvature where the contour cusps. There are a total of four cusps for the bagel. The branches of the cusps which are invisible, are shown with dashed lines, and the visible branches are shown with solid lines. Cusps 1 and 3 have one visible and one hidden branch, while for cusps 2 and 4 both branches are hidden. At the cusps, the normal curvature vanishes, and the apparent contour becomes infinite.

4.3 Note on shape theories

The theories on shape from contour convey how the shape should look from its contour or outline. In other words, these theories mainly play the role of constraining the possible 3D shapes for the observed 2D outline. However, they do not introduce constraints for obtaining a quantitative 3D shape for the object. In addition, in some cases, these theories might give rise to multiple, sometimes mutually exclusive, interpretations for the 3D shape of the object. Although these theories do not suggest how to favor one interpretation over another, in practice, employing other assumptions, for example, that the object lies on a ground plane and is viewed from above, might help. In addition, defining the problem in a probabilistic framework and reconstructing the most probable shape is another reasonable direction.

Chapter 5

Skeleton-based Methods

This chapter is dedicated to discussing methods that inflate the silhouette directly according to a 2D skeleton. Using these methods, the reconstructed shape depends on both the skeletonization and inflation algorithms. Although the various definitions for skeleton might result in the same skeleton in the continuous space, the discrete implementations usually yield different skeletons, and hence different 3D shapes are obtained. The inflation also determines the shape. This step in skeleton-based methods is usually performed heuristically.

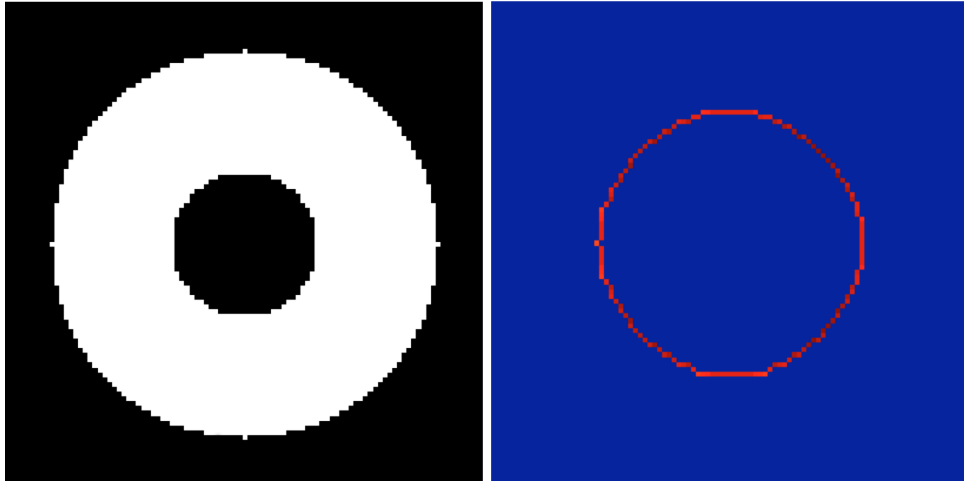


Figure 5.1: Medial Axis transform of a bagel. The silhouette of a bagel (left), and its MAT (right).

5.1 Playing with Puffball [2]

Puffball [2] is a simple inflation method to obtain a 3D model from a given 2D silhouette.

The method is based on the principle “anywhere you can place a circle, place a sphere”.

The reconstructed surface will be smooth if the outline is smooth. If the outline has sharp corners, the surface will have creases.

The algorithm starts by using the Medial Axis Transform (MAT)¹ to compute the skeleton of the silhouette. The skeleton consists of the union of the centers of the maximally inscribing circles of the outline. Augmenting this description with the radii of the circles rendered the representation complete.

Figure 5.1 depicts the silhouette of a bagel and its MAT. In puffball, each circle is replaced by a sphere and the union of all these spheres yields the 3D surface. The puffball

¹See appendix B for more information on MAT

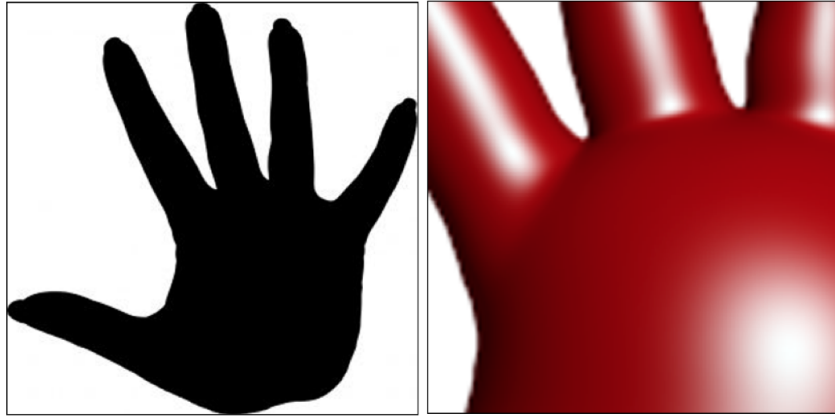


Figure 5.2: Puffball 3D reconstruction of a hand. The silhouette of a hand and 3D model of part of it reconstructed using the puffball algorithm. The palm is very puffy due to having points far from the outline (adapted from [2]).

method is simple, fast, automatic and accepts objects of any genus type. Nonetheless, there are disadvantages of the puffball algorithm:

- The reconstruction can be too puffy (Figure 5.2).
- The rim is constrained to be planar and frontoparallel.
- Not recognizing self-occlusions might result in reconstructions of genus types other than that of the object. For example, the string in Figure 5.3 is of genus type 0, while self-occlusion results in a surface of genus type 1 using puffball.

5.2 Teddy [1]

Similar to puffball, Teddy, the method introduced by Igarashi et al. [1], reconstructs a 3D model by computing and lifting a skeleton for the outline. In contrast to puffball,



Figure 5.3: Puffball and self-occlusion. The string in this figure is of genus type 0. Not recognizing self occlusions by puffball results in a surface of genus type 1 for this image.

Teddy provides interactive tools to incrementally make the object more complex. For example, it permits certain user operations such as introducing an extrusion, cutting, and smoothing, etc. Figure 5.4 shows screenshots of the sketch-based interface and the different operations supported by this method. Here, we skip the details of the operations provided by the interface and rather focus on the algorithm introduced for finding and lifting the skeleton.

In order to find the skeleton, the continuous curve drawn by the user is converted into a closed polygon with unit length sides. An example is shown in Figure 5.5(a). This polygon is triangulated using the constrained Delaunay triangulation algorithm. The triangles are labeled as terminal, junction or sleeve triangles shown by various colors in

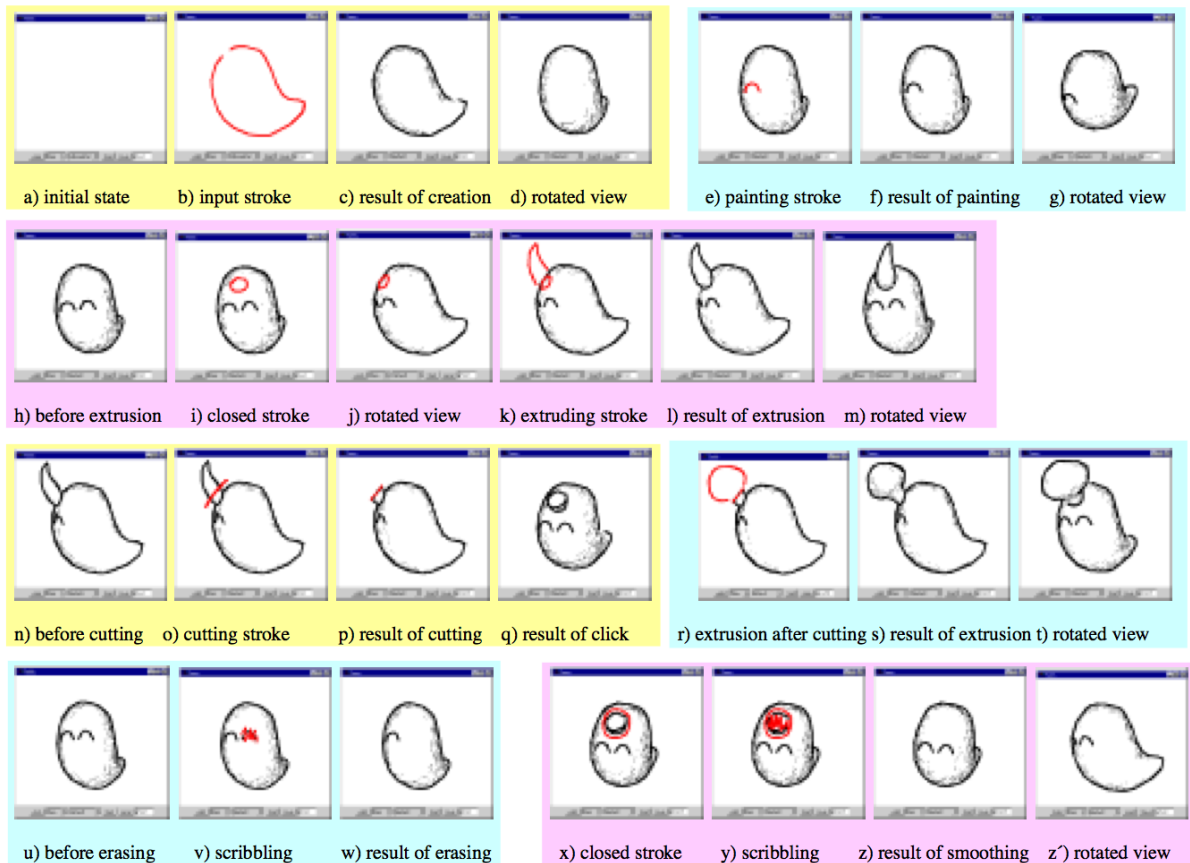


Figure 5.4: Operations available through the Teddy interface. The sketch-based interface designed by Igarashi et al. [1]. (a) - (d) depict the creation of the object, (e) - (g) painting on the 3D model, (h) - (m) extrusion, (n) - (q) cutting, (r) - (t) extrusion after cutting, (u) - (w) erasing a painted curve, (x) - (z') smoothing the model (adapted from [1]).

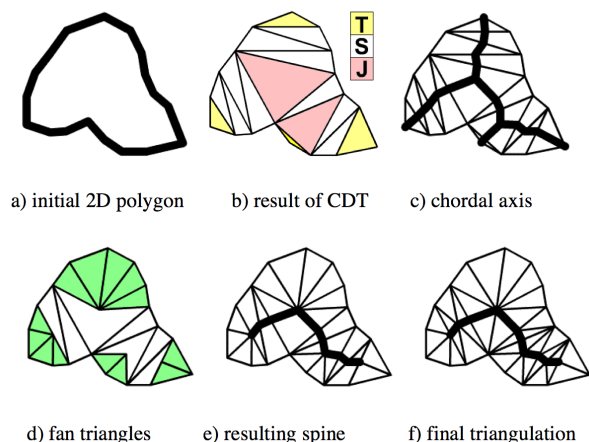


Figure 5.5: Finding the spine of the closed curved in Teddy to create a 3D model for the stroke. The letters T, S, and J represent terminal, sleeve, and junction triangles. (adapted from [1]).

Figure 5.5(b). The terminal triangles have two external edges, while the junction ones have no external edges. The sleeve triangles have one external edge. A primary spine is found by connecting the midpoint of the internal edges and also the midpoint of the junction triangles. This spine is pruned to eliminate insignificant branches by merging the terminal triangles with their neighbors and introducing triangulations which resemble the shape of a fan. These are called the fan triangles (depicted in Figure 5.5(d)). The final spine results by connecting the central point of fan triangles and the midpoint of the remaining internal edges from the primary triangulation. This spine, which is a polyline, is the base of the inflation.

The inflation starts by first elevating each line segment of the spine proportional to the distance of its endpoints to the outline. Then the internal edges of the triangulation are elevated into quarter ellipses where one axis of the ellipse is defined by the length of

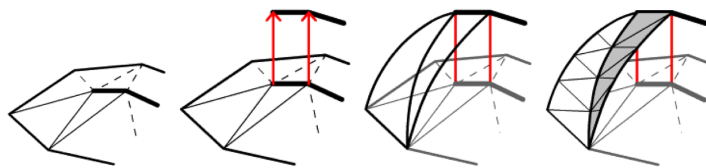


Figure 5.6: Steps of inflating the object in Teddy. From left to right: the triangulated silhouette with the pruned spine, the elevated spine, elevating the internal edges into quarter ovals, triangulated inflated mesh (adapted from [1]).

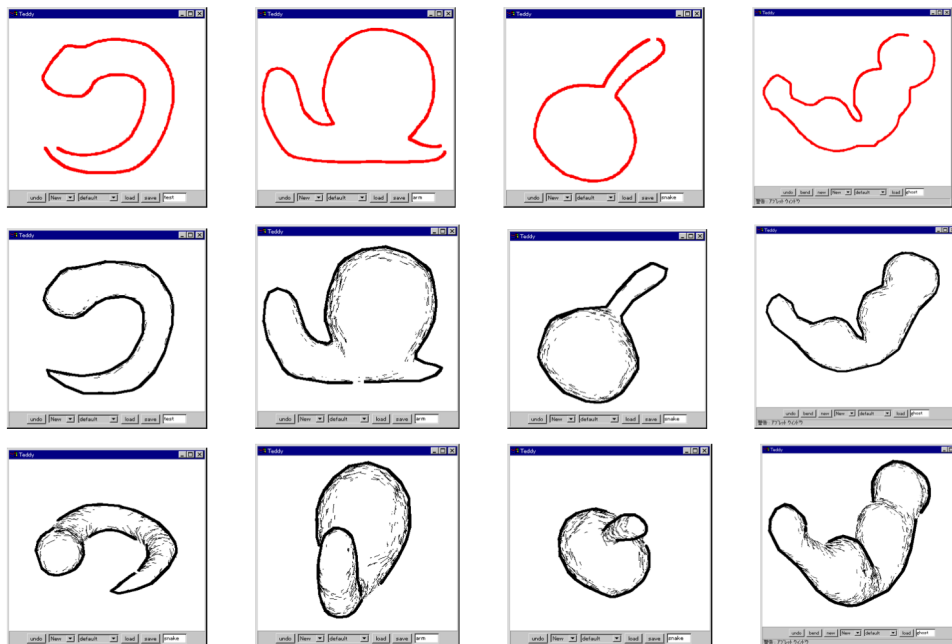


Figure 5.7: Reconstruction of 3D models using the Teddy interface. The first row is the stroke drawn by the user. The second row shows the reconstructed surface. The last row depicts the surface from a novel view (adapted from [1]).

the internal edge, and the other by the amount the corresponding spinal line segment is lifted. The neighboring quarter ellipses are considered as being attached to each other and the surface area between them is triangulated to obtain a triangulated 3D model. The back of the object is reconstructed similarly based on a symmetry assumption. Figure 5.6 shows the steps of inflating the model. Some objects reconstructed by this method are shown in Figure 5.7.

Teddy is a simple method and works in real time. In addition, it yields a triangulated mesh which is useful for post-editing. As it is based on inflation, it works best for puffy objects. It also allows objects of genus 0 only and requires many user interactions to create complex objects. Moreover, the method does not work for objects with self-occlusion. As in the puffball method, the rim is assumed to be planar in the first step. A non-planar rim can be obtained through user interaction.

5.3 Notes on skeleton-based Methods

As discussed earlier in this chapter, the skeleton-based methods consist of two steps: finding the skeleton, and inflation to obtain a 3D model. Therefore, the shape of the 3D model depends on the algorithms for finding the skeleton and inflation.

The two skeleton-based methods discussed in this chapter are both simple, fast and automatic. Although the reconstructed shapes might look similar qualitatively (see Figure 5.8 as an example), these methods are different in detail and yield quantitatively different results. These variations are due to the differences in skeletonization and inflation steps. Unlike the skeleton found by puffball, the skeleton found in Teddy is a polyline, and also its vertices are not necessarily the center of the maximally inscribing circles. Moreover, the skeleton found by Teddy requires an additional pruning step. In other words, the skeletons found by these methods might be different for the same



Figure 5.8: Comparison of reconstruction between puffball and Teddy. From left to right: Input outline, reconstruction of the surface by Igarashi's [1] approach from two views, reconstruction using puffball (adapted from [2]).

silhouette which in turn might result in quantitatively different 3D models.

In terms of inflation, the radius of the maximally inscribing circle determines the amount of inflation in puffball, while in Teddy, the inflation is based on the average distance of the spine vertex to the external vertices. As a result, the amount of inflation might be different for same skeleton points.

Although simple and fast, these approaches have the following disadvantages:

- They are based on the assumption that the rim is planar and frontoparallel which limits the set of objects these methods can reconstruct accurately.
- They work well for puffy objects only.
- Other cues such as shading, texture, etc. can not be easily incorporated into these algorithms.
- While puffball reconstructs objects of any genus types, Teddy is limited to objects of genus type 0.

- Neither method has a mechanism for dealing with self-occlusions automatically.

In general, the skeleton-based methods are difficult to generalize to the reconstruction of more general objects.

Chapter 6

Optimization-based Methods

The approaches discussed in this chapter reconstruct a 3D model of an object from its contour/silhouette by optimizing an objective function subject to constraints. Smoothness of the surface is a commonly used constraint for 3D reconstruction. These optimization-based methods are grouped into two categories: methods reconstructing the visible surface only, and methods reconstructing a solid object. Here, we start with a discussion of methods reconstructing the visible surface given a single still image.

Note that for the methods in this chapter, the first step is to segment the object of interest from the input image. State-of-the-art algorithms for segmentation can be employed for this purpose [30, 21], although in general this remains an unsolved problem. Here, the assumption is that a noise-free segmentation is available.

6.1 Visible surface reconstruction

The focus in this section is on methods for reconstructing the visible surface rather than reconstructing a solid object. The reconstructed surface is smooth except across a limited number of curves. Although the output of these methods is a depth map, it can be transformed into a 2.5D model¹. Gaps present at depth discontinuities might be visible if the 2.5D model is viewed from a novel viewpoint.

6.1.1 Reconstruction of free-form scenes [3]

The method introduced and implemented by Zhang et al. [3] takes a single image, and reconstructs a free-form scene. The reconstruction consists of the visible surface of the objects in the image.

In order to reconstruct free-form scenes, Zhang et al. [3] use the weighted thin-plate energy to guarantee smoothness, and reconstruction is achieved interactively. In particular, if $f(x, y)$ represents the depth map of the scene, and $g_{i,j} = f(id, jd)$ is the depth map on a discrete grid where i and j correspond to the coordinate of the samples and d is the

¹Definition of 2.5D models is provided in Chapter 2

distance of the points on the grid, the thin-plate energy is

$$\begin{aligned}
\mathbf{Q}_0(\mathbf{g}) = & \frac{1}{2d^2} \sum_{i,j} [\alpha_{i,j}(g_{i+1,j} - 2g_{i,j} + g_{i-1,j})^2 \\
& + 2\beta_{i,j}^2(g_{i+1,j+1} - g_{i,j+1} - g_{i+1,j} + g_{i,j})^2 \\
& + \gamma_{i,j}(g_{i,j+1} - 2g_{i,j} + g_{i,j-1})^2]
\end{aligned} \tag{6.1}$$

where \mathbf{g} is the vector with components $g_{i,j}$, and $\alpha_{i,j}$, $\beta_{i,j}$, and $\gamma_{i,j}$ are weights that take either 0 or 1 values. The value of these weights is determined according to input from the user.

Zhang et al. expect the input from the user to be in the following constraining forms:

- Point constraints: the user specifies the depth or normal of some points on the surface
- Depth discontinuities and creases: these two features are handled by adjusting the weights of the terms in the thin-plate energy.
- Planar region: the user indicates the regions with no curvature
- Fairing curve constraints: this option allows the user to draw a curve across which the surface must be smooth. This is achieved by minimizing the variations of the surface gradient across the fairing curve. These constraints are added to the objective function to be minimized.

Figure 6.1 shows a simple surface satisfying the constraints imposed by the user, and texture-mapped examples are illustrated in Figure 6.2. Note that this method does not return a solid object. Due to the discontinuities in depth and modeling objects as surfaces rather than solid objects, the reconstructed model viewed from other directions might not seem appealing. This effect is particularly evident in the last row of Figure 6.2. In addition, it is evident from Figure 6.2 that a great many constraints were imposed by the user. In other words, the reconstruction is mainly performed by the user through extensive interactions. One obvious drawback of asking for so many constraints from the user is the excessively large amount of time required to specify these constraints. Although minimizing the objective function is very fast, as reported by Zhang et al., specifying the constraints is very time consuming. For example, for the Van Gogh image in the third row of Figure 6.2, the reconstruction time included 45 seconds for the optimization part and about 1.5 *hours* for specifying the constraints. The optimization is very fast due to employing a method introduced by Szeliski [31].

6.1.2 Shape, albedo and illumination recovery [4]

In recent work, Barron and colleagues [4] suggested exploiting intensity for shape (a depth map) and albedo recovery. Other than the constraint from intensity, they also employ constraints from the occluding contour.

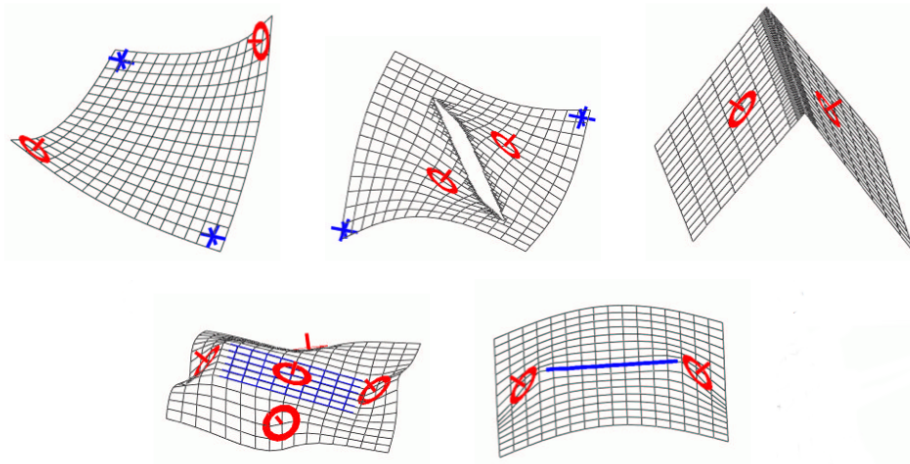


Figure 6.1: This figure depicts the reconstruction results given certain constraints. From left to right, top to bottom: point and normal constraint, depth discontinuity, crease constraint, planar region, a fairing curve minimizing curvature (adapted from [3]).


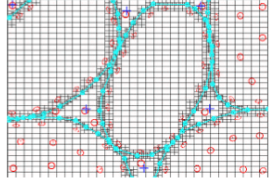


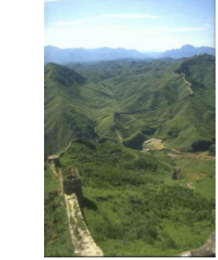
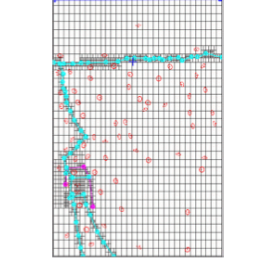
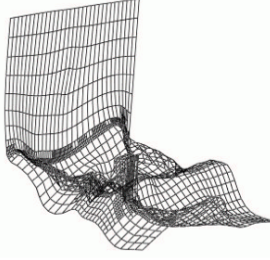
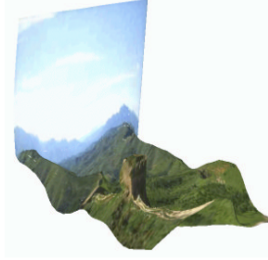

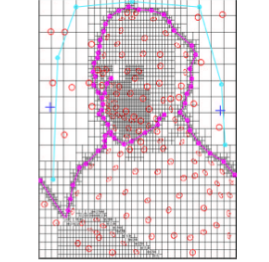

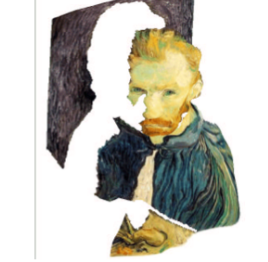
original image	constraints	3D wireframe	novel view
			
			
			

Figure 6.2: Examples of single view modelling by Zhang et al.'s method (adapted from [3]).

They assumed Lambertian reflectance and orthographic projection. With these two assumptions, the log-intensity image can be written as

$$I = A + S(Z, L). \quad (6.2)$$

The log-shading, $S(Z, L)$, is a function of the depth Z and illumination L , and A is the log-albedo map. Given an intensity map, Equation 6.2 leads to the following optimization problem:

$$\begin{aligned} & \underset{Z, A}{\text{minimize}} && g(A) + f(Z) && (6.3) \\ & \text{subject to} && I = A + S(Z, L), \end{aligned}$$

where $g(A)$ and $f(Z)$ are priors over albedo and shape respectively.

The albedo prior, $g(A)$ is a weighted combination of two priors: smoothness and minimal entropy. The smoothness prior assumes the albedo map to be piecewise smooth, and the minimal entropy forces this map to have a small number of albedos.

The prior on shape, $f(Z)$, on the other hand, has three components: flatness, occluding contour, and smoothness priors. The flatness assumption forces the surface to be facing the viewer. The occluding contour prior, relying on the orthographic projection assumption, requires the z -component of the normal to the surface at the occluding contour

to be zero. Then, the occluding contour prior is enforced by minimizing

$$f_c(Z) = \sum_{i \in C} \sqrt{(N_i^x(Z) - n_i^x)^2 + (N_i^y(Z) - n_i^y)^2}, \quad (6.4)$$

where N_i^x and N_i^y are the x and y components of the surface normal at the point i on the occluding contour C , and n_i^x and n_i^y are the normals to the corresponding points on the outline.

The smoothness prior defined by Barron et al. reduces the variation of the mean curvature, $\|\nabla H\|$, over the surface. Under this prior, the ideal surface has constant mean curvature.

Finally, all these priors are incorporated into Equation 6.3. This equation is rearranged to

$$\underset{Z}{\text{minimize}} \quad g(I - S(Z, L)) + f(Z). \quad (6.5)$$

The illumination is assumed to be known in this equation. In case L is to be recovered, the problem is reformulated as a MAP estimation, and L is introduced as a latent variable which is marginalized. A uniform prior over L is assumed.

Barron et al.'s method can potentially reconstruct surfaces with non-planar rims. Some examples can be seen in Figure 6.3. However, creases and self-occlusions are not handled and the depth map is undesirably smooth wherever there are creases or depth

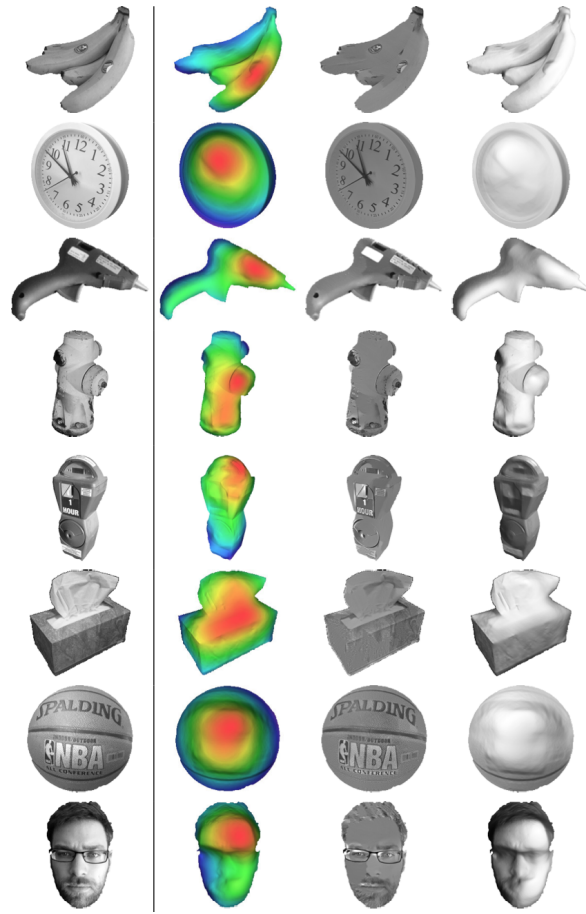


Figure 6.3: The results of the algorithm proposed by Barron et al. [4]. From left to right: the input grayscale image, the depth map, albedo map, and shading image. The red and blue colors in the depth map mean near and far respectively (adapted from [4]).

discontinuities.

6.1.3 Shapecollage [5]

Cole et al. [5] employed a dataset of 3D shape patches to reconstruct the shape of the object. A similar approach was introduced by Hassner and Basri [32]. However, their approach learns the shape of a single class of objects.

The algorithm proposed by Cole and colleagues [5], called shapecollage, takes the

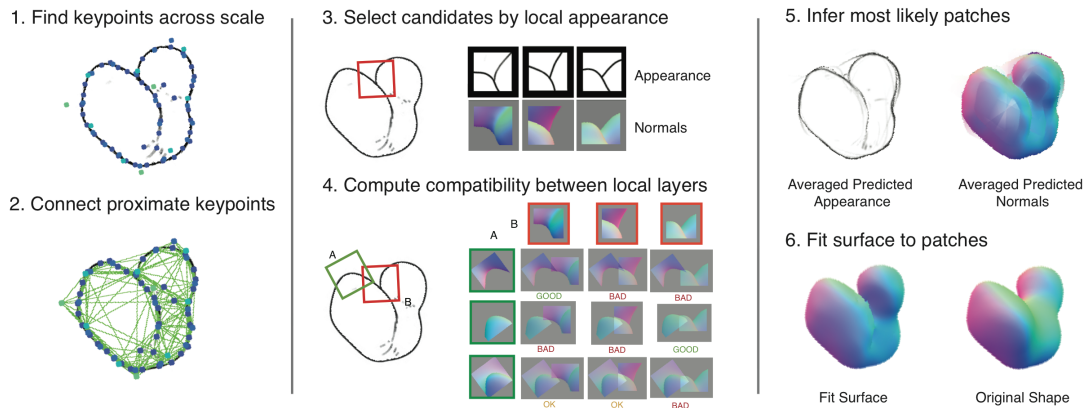


Figure 6.4: Steps of the shapecollage algorithm. After constructing the graph of keypoints (step 2), potential shape patches are selected from the shape database (step 3). These candidates are then scored based on compatibility (step 4), and employing loopy belief propagation results in the shapecollage (step 5). The final smooth surface is obtained by fitting a thin-plate spline surface to the shapecollage (step 6) (adapted from [5]).

apparent contour and the rendered appearance of synthesized objects as input². In the first step, they extract keypoints of the input at different scales. A graph connecting the keypoints is constructed. Then, a set of potential 3D patches are found from the database of 3D shape patches. The min-sum belief propagation algorithm [33, 34] is then deployed to estimate the best set of compatible shape descriptors for the keypoints. These shape descriptors form a sparse shape representation called a shapecollage. A smooth surface is then reconstructed by fitting a thin-plate spline surface to the shapecollage. Figure 6.4 shows the steps of the shapecollage algorithm.

The database of shape patches contains five maps for each patch: an appearance map, a normal map, a depth map, a map of occluding contours and an ownership map shown

²The appearance of the object depends on the rendering style. In this work, the rendering styles employed include texture only, diffuse and glossy shading with texture.

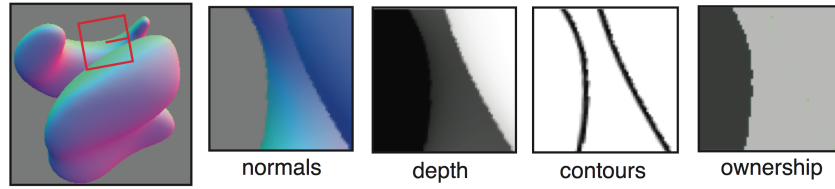


Figure 6.5: An example of maps for each patch. The appearance map is shown by a red box on the left.

in Figure 6.5. The ownership map is in fact part of the object silhouette overlapping with the patch. In other words, this map specifies, within each patch, which pixels are in the silhouette. Each of these maps are useful for reconstructing a new shape.

The keypoints in the test image are extracted using gradient information. Then, candidate patches are chosen according to similarity in the appearance. Each candidate is scored according to two factors: first, the match in appearance with the test image, i.e. the average difference in the pixel value between the blurred versions of the test and candidate patches, and second, the compatibility with the candidates of the neighboring keypoints. The latter factor is essential to avoid conflict scenarios, for example, when two overlapping neighboring patches claim opposite occlusion assignment for the same contour. An example of such a case is shown in Figure 6.6, case (b).

To resolve such issues, Cole et al. introduced a graph corresponding to the overlap area for each candidate. If two candidate patches overlap, two identical graphs are considered—one for each candidate patch. This graph is called a region graph. The configuration of the region graph is determined by the contour of the test image. This contour divides the

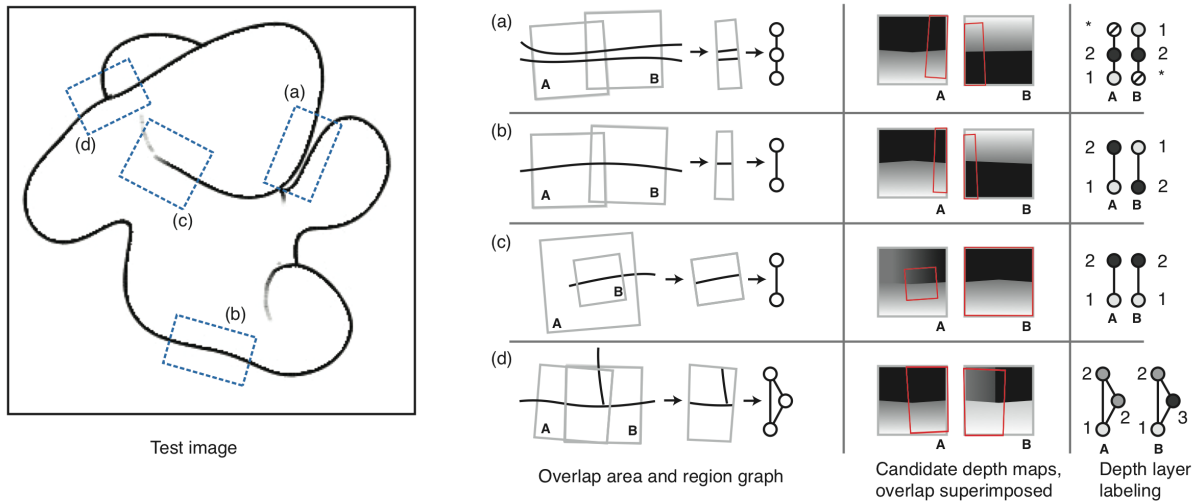


Figure 6.6: From left to right: the input image with challenging areas marked, overlapping regions with the corresponding graph, candidate depth maps, ordinal depth labelling.

overlap area of the two patches into disconnected components (See the column illustrating “the overlap area and region graph” in Figure 6.6). Each of these components is called a region. For each region defined by the contour, a node is designated. For example, the overlap region of patch (b) in Figure 6.6 has been divided into two regions by the contour, and hence, the region graph has two nodes. Then, the average depth of the pixels in each region is computed. These averages are sorted and an ordinal depth is assigned to the nodes (last column of Figure 6.6). As an example, in Figure 6.6, the average depth according to each candidate depth map for patch (b) is computed in each region of the overlap. This average depth is greater for the region on top of the overlap according to candidate map “A”. Therefore, the ordinal depth assigned to the region graph node corresponding to the top region is greater than that of the node corresponding to the lower overlap region. At this step, if the region graph of the two candidates match, they

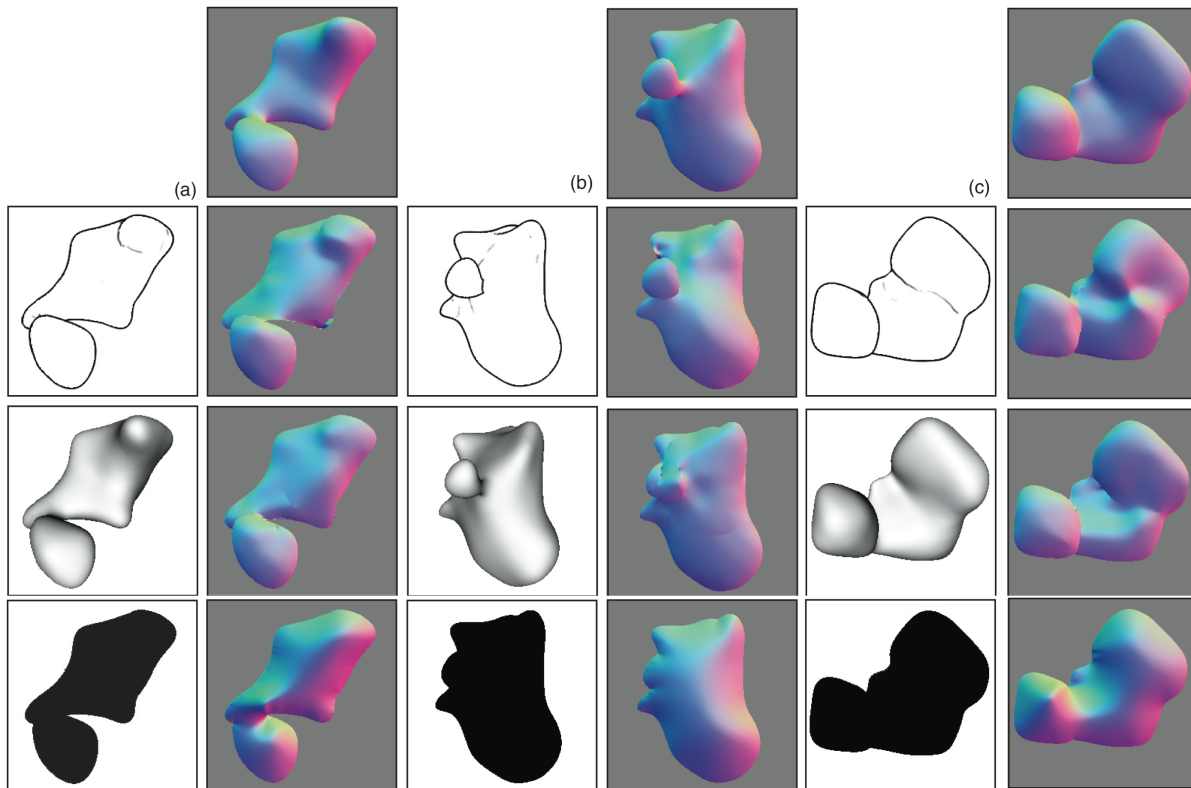


Figure 6.7: Examples of the reconstructed normal maps. The first row is the ground truth. The second and third rows show the apparent contour and appearance map with the reconstructed normal maps. The last row depicts the normal map obtained by interpolating the normals at the outline (adapted from [3]).

are considered as compatible.

Examples of reconstructed synthesized surfaces are depicted in Figure 6.7. The last row in this figure shows the normal map computed by simply interpolating the outline normals. In particular, the normals at the outline are employed as soft constraints in an L_1 minimization of the thin-plate energy. Clearly, the contour can improve reconstruction of the 3D shape.

Shapecollage scores the similarity of two patches based on the appearance. This approach works for their synthetic database, however, it is unclear whether it would work

for natural images.

6.1.4 Notes on visible surface reconstruction methods

The methods discussed in this section all reconstruct the shape of a smooth visible surface given the occluding contour and, in some cases, appearance cues (shading, junctions, etc.).

Smoothness can be relaxed if depth/orientation discontinuities are evident.

The output of these methods is a depth map (Table 6.1). Although the depth map can give a relatively good sense of the shape, it is not a solid object and viewing the output from various viewpoints might introduce gaps in the model. Therefore, as one of the purposes of 3D reconstruction is viewing the object from a novel view, these models are not optimal.

The “visible surface” methods reviewed here can potentially reconstruct objects with non-planar rims accurately.

6.2 Solid object reconstruction

The approaches discussed in the section focus on reconstructing solid objects. These methods are divided into deformable models, volume occupancy models, coordinate mapping models, and triangulated models. The following sections review these models in order.

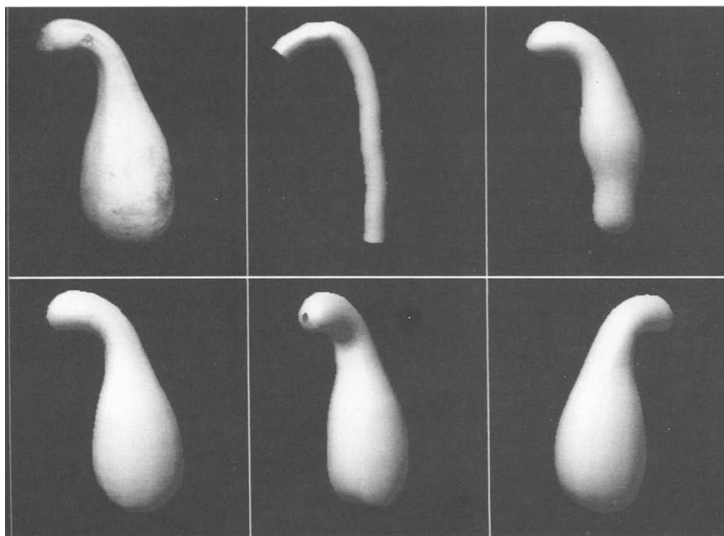


Figure 6.8: Deformations of the model reconstructing a squash using the method by Terzopoulos et al. [6]. Top row, from left to right: the original image, initialized model, the symmetric model at an intermediate step. Bottom row: the reconstructed model from different views (adapted from [6]).

6.2.1 Deformable models

Deformable models, introduced by Terzopoulos et al. [35] have been extensively used in computer vision. These models, represented as curves or surfaces, deform by responding to internal and external forces.

Terzopoulos et al. [6] introduced a cylindrical model which deforms iteratively and yields a 3D model of the object in the image. The deformations are determined by external forces pushing the model toward significant image intensity gradients. At the same time, the internal forces keep the model smooth. Deformations of the model reconstructing a squash is depicted in Figure 6.8.

The deformable model is comprised of a tube coupled with a spine (see Figure 6.9).

An energy functional is associated with the tube-spine model. This energy is comprised of two terms:

$$E = E_{\text{int}} + E_{\text{ext}}, \quad (6.6)$$

where E_{int} and E_{ext} denote the internal and external energies. The external energy term guides the model toward the outline. In addition, it keeps the model symmetric around the spine. On the other hand, the internal energy ensures the model remains smooth. The energy functional is defined for each of the spine and the tube separately. The spine is a space curve $\mathbf{v}(s) = [X(s), Y(s), Z(s)]$ where $s \in [0, 1]$. Its internal energy is defined as

$$E_{\text{int}}^S(\mathbf{v}) = \frac{1}{2} \int_0^1 w_1(s) |\mathbf{v}_s|^2 + w_2(s) |\mathbf{v}_{ss}|^2 ds \quad (6.7)$$

where \mathbf{v}_s and \mathbf{v}_{ss} are first and second order derivatives of \mathbf{v} with respect to s . Using the Euler-Lagrange equation, the minimum value of the function can be obtained from

$$\frac{\delta E_{\text{int}}^S}{\delta \mathbf{v}} = \frac{\partial^2}{\partial s^2} \left(w_2 \frac{\partial^2 \mathbf{v}}{\partial s^2} \right) - \frac{\partial}{\partial s} \left(w_1 \frac{\partial \mathbf{v}}{\partial s} \right). \quad (6.8)$$

The weighting function $w_1(s)$ controls the “tension” along the spine as $w_2(s)$ controls its shape. Intuitively, one can think of the spine as a rubber band which has limits for expansion and bending. Figure 6.9 illustrates the parameterization of the spine and the tube.

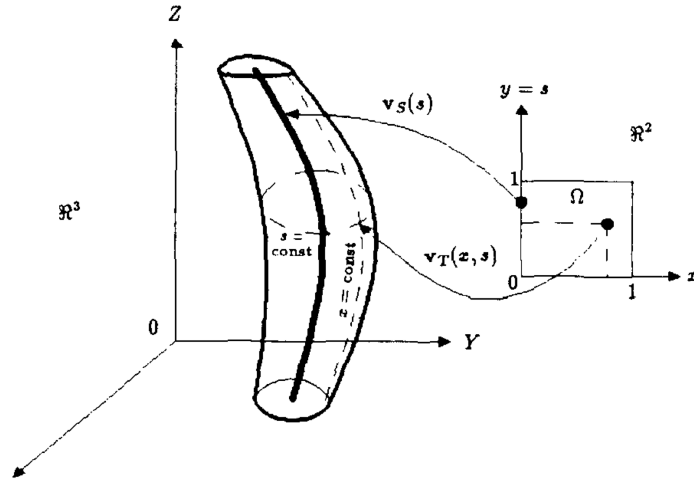


Figure 6.9: Parameterization of the spine and tube representing the 3D model suggested by Terzopoulos et al. In this figure, a tube coupled with a spine (the line in the middle) are shown. Each of the tube and the spine are represented parametrically. The mapping $\mathbf{v}_T(x, s)$ maps the points in the parameter space onto the points on the tube. The mapping $\mathbf{v}_S(s)$ maps the points on the y -axis in the parameter space onto the points on the spine (adapted from [6]).

Similarly, the tube, represented parametrically, is denoted as $\mathbf{v}(x, y) = [X(x, y), Y(x, y), Z(x, y)]$. The tube can be assumed to be a rubber sheet. Then, the internal energy of the tube is given by the thin-plate energy

$$E_{\text{int}}^T(\mathbf{v}) = \frac{1}{2} \int_0^1 \int_0^1 w_{1,0} |\mathbf{v}_x|^2 + w_{0,1} |\mathbf{v}_y|^2 + w_{2,0} |\mathbf{v}_{xx}|^2 + 2w_{1,1} |\mathbf{v}_{xy}|^2 + w_{0,2} |\mathbf{v}_{yy}|^2 dx dy. \quad (6.9)$$

The minimum can be found from the Euler-Lagrange equation

$$\begin{aligned} \frac{\delta E_{\text{int}}^T}{\delta \mathbf{v}} = & \frac{\delta^2}{\delta x^2} \left(w_{2,0} \frac{\delta^2 \mathbf{v}}{\delta x^2} \right) + 2 \frac{\delta^2}{\delta x \delta y} \left(w_{1,1} \frac{\delta^2 \mathbf{v}}{\delta x \delta y} \right) + \\ & \frac{\delta^2}{\delta y^2} \left(w_{0,2} \frac{\delta^2 \mathbf{v}}{\delta y^2} \right) \\ & - \frac{\delta}{\delta x} \left(w_{1,0} \frac{\delta \mathbf{v}}{\delta x} \right) - \frac{\delta}{\delta x} \left(w_{0,1} \frac{\delta \mathbf{v}}{\delta x} \right). \end{aligned}$$

In this equation, the weight functions $w_{2,0}$, $w_{1,1}$, and $w_{0,2}$ control the curviness of the sheet while $w_{1,0}$ and $w_{0,1}$ control the length of the sheet along the x and y directions respectively.

The external energies of the model are twofold: the energies capturing interactions between the tube and the spine, and the energy dictated from the image data. The interaction energies are introduced to assure

1. The spine is in axial position in the tube.
2. The tube is radially symmetric around the spine.
3. All the points (x, s) on the tube, for each constant value of s , have planar radial deformations: expansion or contraction.

The energy dictated by the image data is incorporated to guarantee that after the final iteration the projection of the model matches the 2D silhouette of the object. At each

iteration, this energy guides the boundary of the tube to regions with significant intensity changes in the image. This effect is achieved by defining the energy as

$$E_{\text{img}}[\mathbf{v}^T(x, s)] = \psi(x, s) |\nabla G_\sigma * I(\Pi(\mathbf{v}^T(x, s)))|, \quad (6.10)$$

where G_σ is a Gaussian filter with the standard deviation σ , and $\Pi[\mathbf{v}^T(x, s)]$ denotes the projection of the tube to the image, I . The weighting function $\psi(x, s)$ is zero everywhere, and close to 1 for the points near the rim of the tube.

Note that satisfactory reconstruction depends on the initialization: the user is required to initialize the model so that the spine is not far from the medial axis of the silhouette³. Additionally, the tube-spine model is of genus type 0, and thereby only objects of genus type 0 can be reconstructed by this model. Also, this approach can easily get stuck in local minima. Finally, consider the example of an object with an extrusion and suppose the extrusion has caused shadowed regions on the object surface. Because this method assumes significant intensity changes are at the boundaries, it might see the border of the shadowed region as the boundary. In this case, the model gets attracted to the shadow of the extrusion rather than the actual outline.

³See appendix B for more details on medial axis.

6.2.2 Volume occupancy models

6.2.2.1 Non-parametric object reconstruction [7]

Oswald et al. [7] formulate the single view reconstruction problem in an energy minimization framework. That is, they minimize the energy functional

$$E(u) = E_{\text{data}}(u) + \nu E_{\text{smooth}}(u), \quad (6.11)$$

where $u : \mathbb{R}^3 \rightarrow \{0, 1\}$ is an indicator function representing the surface Σ implicitly. Each point in \mathbb{R}^3 is labeled as 1 if it is inside the surface, and 0 otherwise. The smoothness term in Equation 6.11 is expressed in terms of the weighted total variation norm⁴:

$$E_{\text{smooth}} = \int_V g(x) |\nabla u| d^3x, \quad (6.12)$$

where the weighting function, $g(x)$, controls the smoothness of the surface. The value of this function at each point is set according to the input received from user interactions.

The data term consists of the silhouette and volume constraints. That is,

$$E_{\text{data}} = \int_V u(x) \phi_{\text{sil}}(x) d^3x + \int_V u(x) \phi_{\text{vol}}(x) d^3x, \quad (6.13)$$

⁴see appendix A for more details on total variation norm.

where $\phi_{\text{sil}}(x)$ assigns large positive values to the points whose projection onto the image plane falls outside the silhouette. This term ensures the projection of the reconstructed object matches the silhouette. On the other hand, the function $\phi_{\text{vol}}(x)$ ensures that a smooth *non-flat* surface is obtained. This function assigns $+1$ or -1 depending on the distance of the point x to the image plane. That is,

$$\phi_{\text{vol}}(x) = \begin{cases} -1 & \text{if } \text{dist}(x, \Omega) \leq h(\pi(x)) \\ +1 & \text{otherwise,} \end{cases} \quad (6.14)$$

where $\text{dist}(x, \Omega)$ is the distance of the point x to the image plane Ω , $\pi(x)$ is the projection of x to the image, and $h(\pi(x))$ is a function of the distance of $\pi(x)$ to the outline. More specifically, it is

$$h(p) = \min\{\lambda_{\text{cutoff}}, \lambda_{\text{offset}} + \lambda_{\text{factor}} \times \text{dist}(p, \partial S)^k\} \quad (6.15)$$

where the parameters $k, \lambda_{\text{cutoff}}, \lambda_{\text{offset}}, \lambda_{\text{factor}}$ control the shape of the object as shown in Figure 6.10. Therefore, the function $\phi_{\text{vol}}(x)$ not only specifies a maximum depth for the points on the surface through $h(\pi(x))$, but also relates the depth of the points to the distance of their projection to the outline. This constraint inflates the surface to a certain volume while the point projecting to the innermost point on the silhouette has

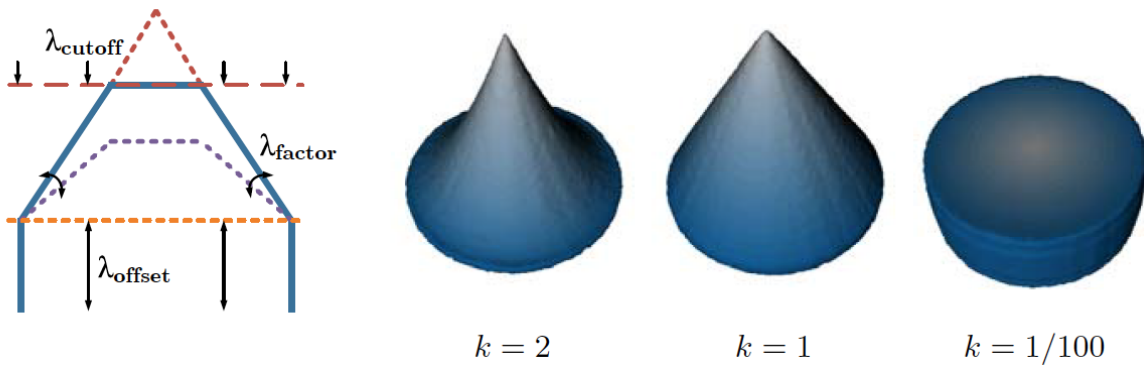


Figure 6.10: The effect of the parameters of the function h in Equation 6.15 on the shape of the object. Note how the function for $k = 1$ looks like a distance transform.

the maximum depth.

The minimizer of the energy functional in Equation 6.11 is therefore a smooth non-flat minimal surface whose projection is the silhouette. To minimize this energy functional, the binary assumption on u is relaxed. In other words, u is considered to be continuous, and then the energy is minimized. Once a continuous solution is achieved, the surface is reconstructed by thresholding u .

Post-editing by modifying the model parameters allows the user to make changes to the surface (Figure 6.11). Adjusting the parameters of the data term deforms the shape of the object. On the other hand, the user can create creases on the surface by manually setting the weight function in the smoothness term. Examples of creases on the reconstructed surfaces are shown in Figure 6.12.



Figure 6.11: Reconstruction process of [7]. From left to right: the original image with blue and red scribbles drawn by a user specifying the object and background respectively for segmentation of the object, the silhouette (extracted by user), the reconstructed surface, the surface after post-editing by a user (adapted from [7]).

6.2.2.2 Object reconstruction via Cheeger sets [8]

Following the approach introduced by Oswald et al. [7], Toppe et al. [8] suggested imposing a *fixed* volume as a hard constraint. In this case, the volume must be known in advance, or the user must change it interactively to obtain the desired result. The new volume constraint eliminates the parameters controlling the model through $h(\pi(x))$ and the thickness of the object does not have to be proportional to the distance from the outline. This effect could be seen on the reconstructed models using the method of [7]. For example, the third image from the left in Figure 6.11 depicts the pot which is thicker in the middle since the corresponding silhouette points are farther away from the outline.

The energy functional includes a smoothness term similar to Equation 6.11. However, the minimization is performed over the set of surfaces satisfying the silhouette and volume constraints. Similar to [7], the binary assumption over u is relaxed and the convex energy functional is minimized. The 3D surface is then reconstructed by sorting the voxels by decreasing value of $u(x)$, and setting the first V voxels are set to 1 and the rest to 0, where

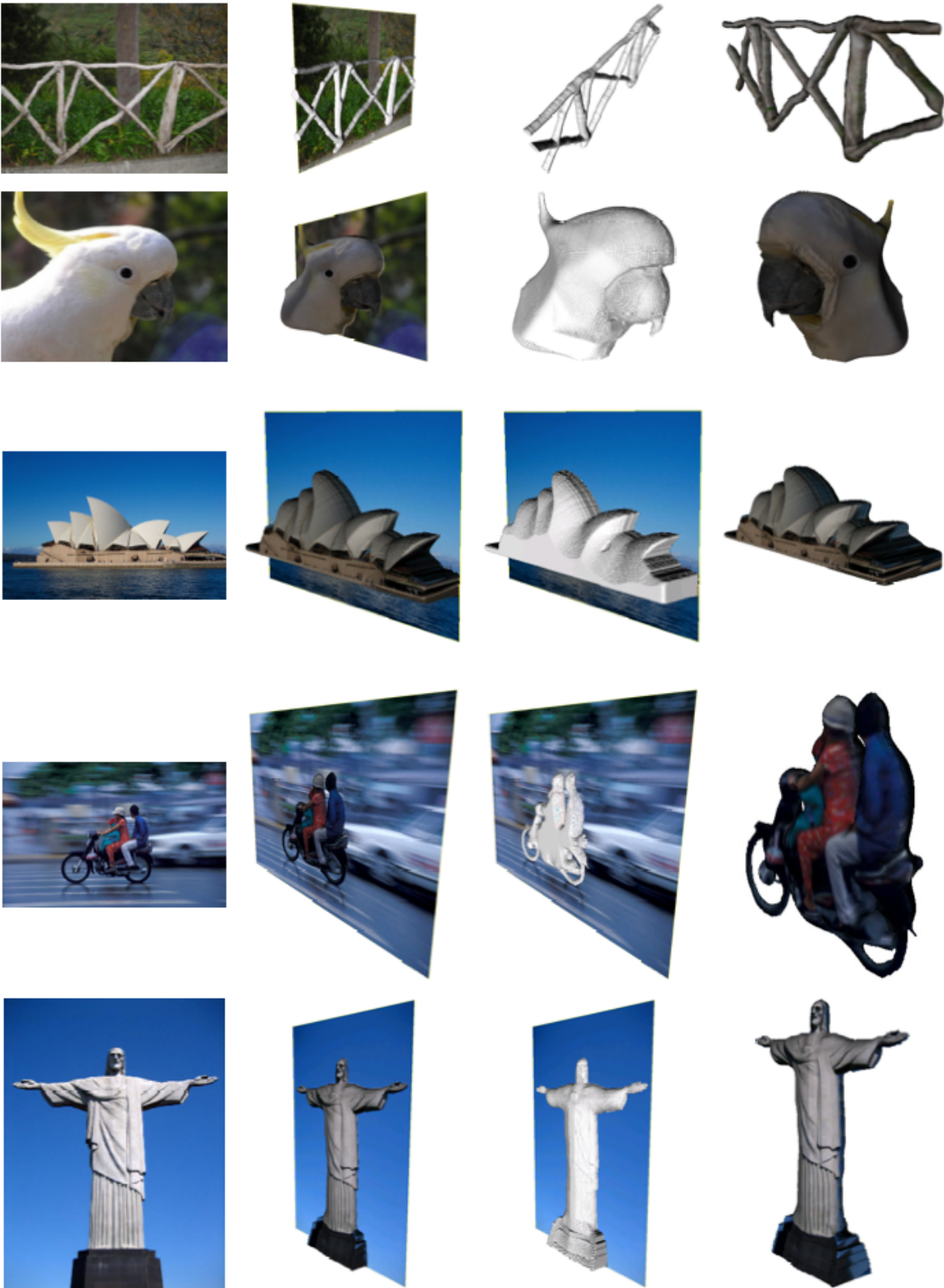


Figure 6.12: Examples of reconstructed surface in [7] (adapted from [7]).



Figure 6.13: From left to right: the original image, the reconstructed surface using Toppe et al.’s [8] algorithm, the textured surface (adapted from [8].)

V denotes the volume of the object. Toppe et al. have shown that the method generally results in a compact solid object. Figure 6.13 depicts an example of a reconstructed balloon.

In the post-editing phase, the user can vary the volume to the desirable amount after the first reconstruction. Results of varying the volume after the first reconstruction are shown in Figure 6.14. Accommodating local non-smoothness such as creases and sharp edges, similar to [7], is performed by changing the weight function, $g(x)$ in Equation 6.11. Novel views of a reconstructed object with user-specified creases is illustrated in Figure 6.15.

Notes on volume-occupancy models

The algorithms proposed in [7] and [8] both require the model to be smooth, though the user might specify creases on the surface. While in the former the volume is a soft



Figure 6.14: From left to right: the original image, the initial reconstruction, the result after varying the volume to +30%, and +40% volume (adapted from [8]).

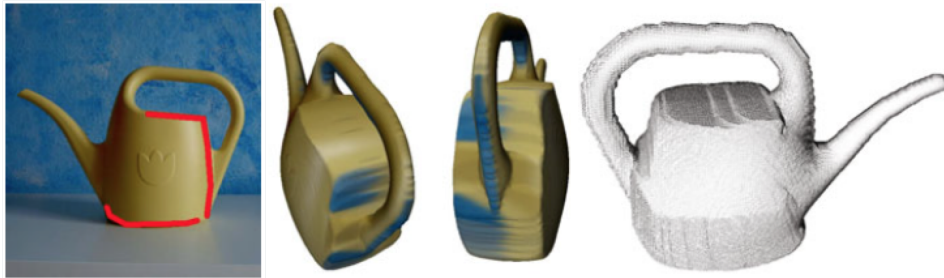


Figure 6.15: From left to right: Input image with the user-specified lines for local non-smoothness, two views of the textured reconstructed object, the reconstructed geometry (adapted from [8]).

constraint, the latter imposes it as a hard constraint. Therefore, the latter method requires the user to know a good approximation of the volume in advance. Otherwise, additional user interaction is necessary after the first reconstruction. In contrast, the distance-transform-like volume constraint in the former method results in unsatisfying surfaces especially when the object is round. See Figure 6.16, middle, as an example.

Both methods accept objects of any genus type thanks to the implicit representation of the surface. However, the implicit representation does not give much control over the shape of the surface unless the weight function $g(x)$ is employed in order to locally change the smoothness of the surface. This operation needs user interaction in the post-editing



Figure 6.16: From left to right: the original image, the reconstructed geometry by [7], the reconstructed geometry by [8] (adapted from [8]).

phase. Additional editing options, such as specifying points in the space to be included in the volume or changing the local shape of the surface, have not been provided to the user.

Neither algorithm recognizes self-occlusions and both assume the rim to be planar. This assumption limits the set of target objects that can be reconstructed accurately.

6.2.3 Coordinate mapping model [9]

Prasad et al. [9] formulated the reconstruction of objects as a quadratic optimization problem with linear constraints. The reconstructed surface is obtained by minimizing the thin-plate energy. The objective function is minimized subject to constraints from the outline, user-specified depth for some surface points, self-occlusion and creases.

As before, the thin-plate energy guarantees the surface smoothness. It is in the form

$$E(\mathbf{r}) = \int_0^1 \int_0^1 \|\mathbf{r}_{uu}\|^2 + 2\|\mathbf{r}_{uv}\|^2 + \|\mathbf{r}_{vv}\|^2 dudv \quad (6.16)$$

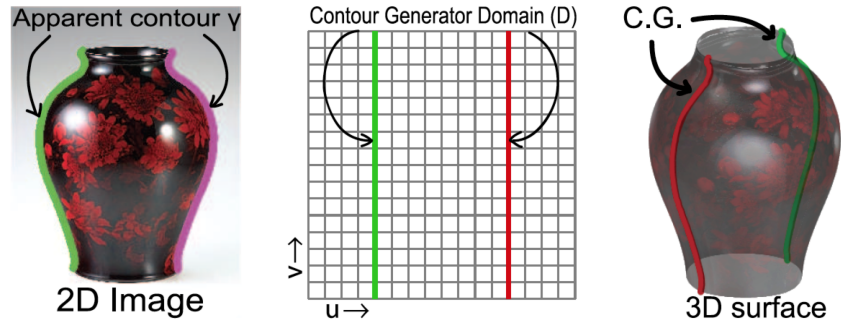


Figure 6.17: From left to right: The 2D image with the outline shown as pink and green curves, the contour generator in the parameter space, and the contour generator on the 3D surface (adapted from [9]).

where $\mathbf{r} : [0, 1]^2 \rightarrow \mathbb{R}^3$ represents the surface parametrically in the uv -space. Figure 6.17 depicts the contour generator in the uv -space.

As expected, the trivial solution to the thin-plate energy is a flat surface. This solution is avoided by adding an inflation constraint to the problem. Two types of inflation constraints are defined: interpolation and approximation constraints. The interpolation constraint forces the surface to pass through certain points in \mathbb{R}^3 , whereas, the approximation constraint requires the surface to pass near some designated points in \mathbb{R}^3 . These constraints are set by the user.

In addition to inflation, a constraint ensuring the contour generator projects onto the apparent contour is imposed. Another constraint sets the z -component of the surface normal at the rim to be zero. Although the normal constraints should be sufficient for avoiding flat reconstructions, it is not clear why Prasad et al. [9] claimed the inflation constraint is necessary for avoiding flat surfaces.

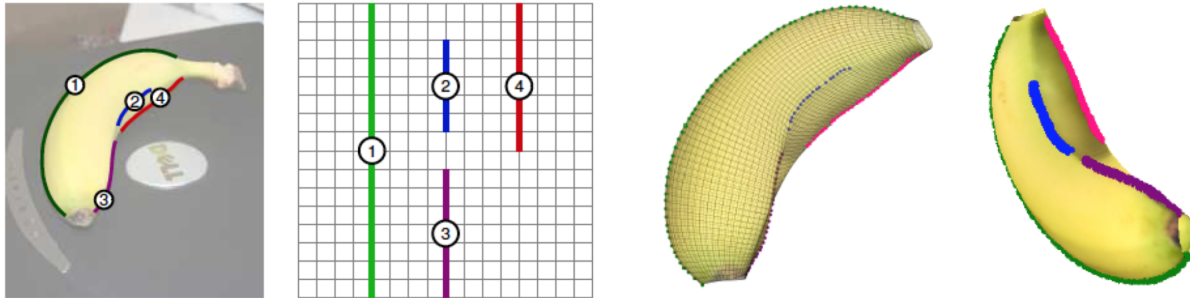


Figure 6.18: From left to right: The original image with the user-specified contour segments, the contour segments in the parameter space, the reconstructed surface from two different views (adapted from [9]).

Minimizing the thin-plate energy with the constraints mentioned above results in smooth surface reconstruction. The model is further generalized by allowing self-occlusion. In this case, the contour generator is split into segments with different u values. Figure 6.18 depicts an example where splitting the rim into segments gives rise to a non-planar rim. The user determines the u values for each segment. Moreover, the user might select an additional segment representing crease discontinuities. Such a segment is shown in blue, segment number 2, in Figure 6.18.

Another set of constraints empower the model with the capability of reconstructing objects of genus types 0, 1 and 2. For example, to recover a cylindrical topology, the constraint is

$$\mathbf{r}(0, v) = \mathbf{r}(1, v) \quad \forall v, \quad (6.17)$$

in addition to the other constraints on the first and second derivatives for surface smoothness at $u = 0$ and $u = 1$.

Also, surface creases are permitted on the surface by replacing the thin-plate energy with a membrane energy

$$\|\mathbf{r}_u\|^2 + \|\mathbf{r}_v\|^2. \quad (6.18)$$

Although the method of Prasad et al. [9] accommodates self-occlusion, creases and objects of genus 0 - 2, this comes at the price of increased user-interaction: none of these additional features are automatic in the algorithm. Moreover, the parametric representation of the surface requires the user interactions happen in the parameter space (uv -space): additional interactions might be necessary to compensate for an untrained user in order to obtain a satisfying reconstruction. The model is also sensitive to parameter assignments. This effect is illustrated in Figure 6.19.

6.2.4 Triangulated model [10]

Smoothsketch, introduced by Karpenko et al. [10], was mainly proposed as a component to a sketch-based interface such as Teddy [1]. The main focus in designing this method has been on reconstructing objects with non-planar rim and self-occlusion. This method consists of three steps: contour completion, lifting, and smoothing.

The contour completion phase takes as input only the visible part of the contour, and identifies cusps, hidden cusps and T-junctions. Figure 6.20 depicts an example of a contour with the T-junctions, visible and hidden cusps marked. These special parts of

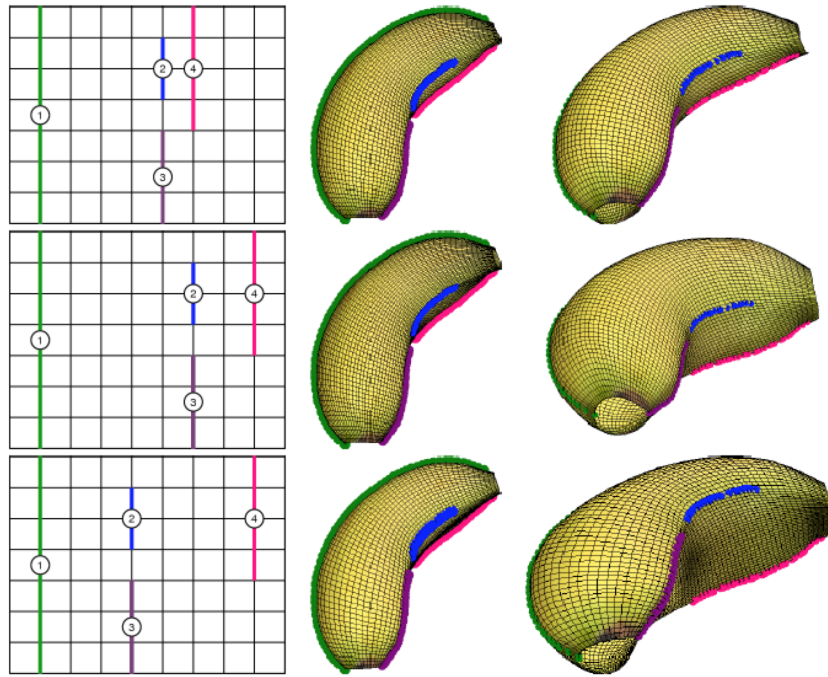


Figure 6.19: This figure shows the sensitivity of the model to parameter assignment. Each row shows a setting in the parameter space with the reconstructed model from two views (adapted from [9]).

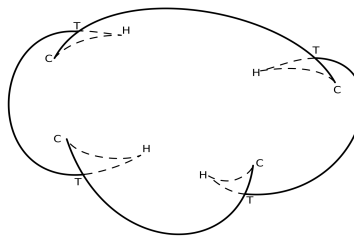


Figure 6.20: An example of a contour with the T-junctions and visible/hidden cusps labeled (adapted from [10]). The letter T represents T-junctions, C is for visible cusps, and H shows the hidden cusps.

the contour, called the endpoints, are joined during the completion process. For example, in Figure 6.20, all of the T-junctions are joined to a visible and hidden cusp. Similarly, each visible cusp is connected to a T-junction and a hidden cusp. More specifically, each connection joins two endpoints, and the algorithm computes the likelihood of all pairs. Then, the most likely configuration is chosen using a greedy algorithm.

Generally, three cases of joining endpoints might happen: joining two T-junctions, two cusps, or a T-junction and a cusp. For the first two cases, they compute the energy of the pair as

$$E_{\text{pair}} = E_{\text{curve}} + E_{\text{endpoints}}. \quad (6.19)$$

E_{curve} is the energy of the connecting Bezier curve⁵, controlled by the curve length and its curviness. E_{endpoint} measures the similarity of the two endpoints. This measure is based on the tangent directions of the points.

In the case where the two endpoints are a cusp and a T-junction (Figure 6.21), the location and direction of a hidden cusp is determined from a pre-built table. The probability of the hidden cusp is defined as the product of the probabilities of the two curves connecting the hidden cusp to the visible cusp and the T-junction. In order to build the table, they put one of the two points, either the visible cusp or the T-junction, at the origin with its tangent in the x direction, and the other on a unit circle. Then, many C^1

⁵More details on Bezier curves can be found in Appendix C.



Figure 6.21: An example of a hidden cusp joining a T-junction and a visible cusp (adapted from [10]).

random walks are generated from the point at the origin, and the location and direction of the point the walk intersects the unit circle is recorded. A similar approach is performed for the second point situated on the unit circle. Among all the generated pairs of points and directions, the one which was most likely to be the endpoint of both sets of random walks is chosen. The entries of the table, then, specify the location and direction of the hidden cusp for given location and direction of the second point. The table of locations and directions of the hidden cusp is created one time and used every time the contour completion algorithm is performed. The energy of the two endpoints is then the sum of the energy of the Bezier curves connecting the hidden cusp to the two endpoints. Pairing of a T-junction and a cusp is shown in Figure 6.21.

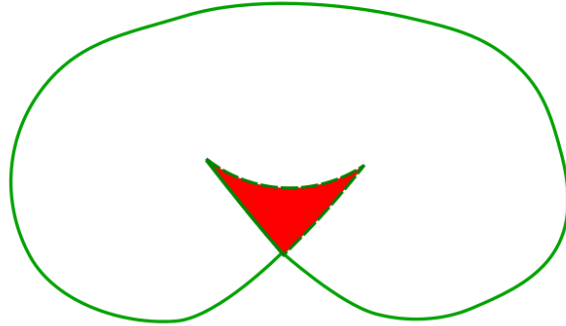
The contour completion algorithm allows the contour to be labeled according to Huffman rules [36]. The silhouette is then decomposed into a disjoint union of regions based on the labeled contour. In Figure 6.20, as an example, the silhouette is divided into five regions, four of which are confined by the curves connecting T-junctions, visible and hidden cusps, and one region inside the outline not enclosed by these special endpoints. Each region is triangulated and ordinal depth of the vertices of each region are determined

based on the labeled contour.

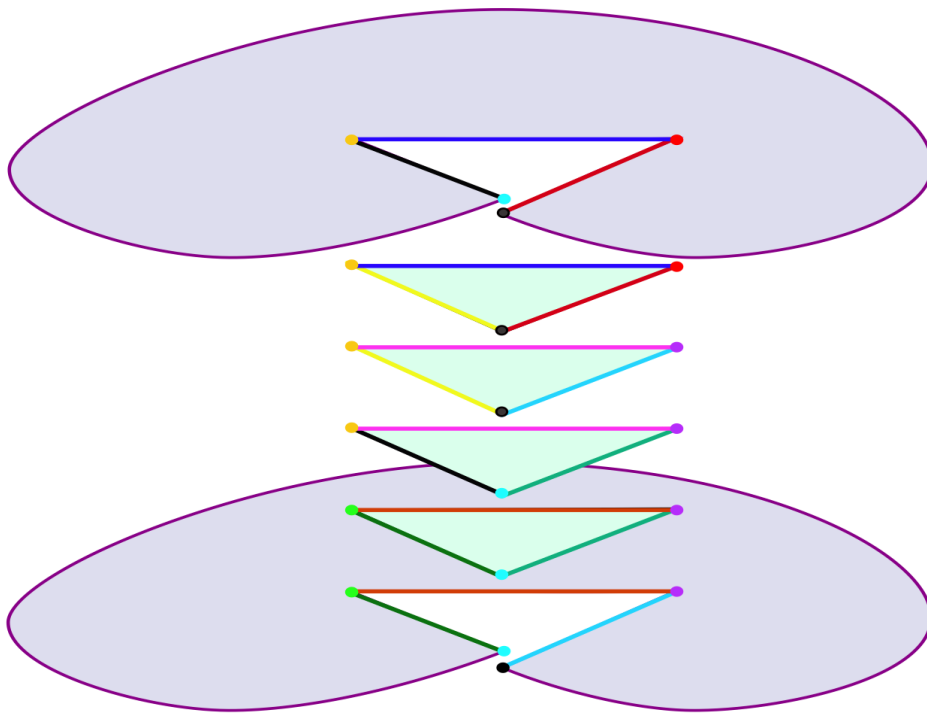
To reconstruct a solid object, the algorithm makes copies of each region, and these copies are called panels. For example, in Figure 6.22, there are two panels for the largest region shown in purple: one for the front of the solid object, and one for the back. Also, the algorithm clusters the edges and vertices of the panels according to criteria suggested in [37]. In Figure 6.22, the edges/vertices belonging to the same cluster are shown with the same color. The vertices and the edges in the same cluster from different panels are stitched together. The result will have each panel in the correct ordinal depth.

Assigning depth to the points on the surface is performed by interpolating the depth of the points on each edge according to the depth of the two vertices of the edge. The depth of the faces of the model are also determined similarly. To make the model more smooth, remeshing is performed at this step to make the triangles equal in size. The authors have reported a smoother shape is obtained after remeshing while the 3D model is still sharp at the edges corresponding to the rim.

To have an even smoother reconstruction, the model is inflated by defining a mass-spring system. The vertices contain masses while two types of springs are introduced. The length springs force the length of edges of the triangles to get as close to zero as possible while the “pressure-force” springs push each triangle outward in its normal direction. This system is iteratively optimized to reach its equilibrium. Examples of reconstructed



(a) A bean with the visible cusp and the T-junction joined. Here, the contour has two disjoint regions.



(b) The panels corresponding to the contour of the bean shown in 6.22a. Also, the vertices/edges in the same cluster are shown with same color. The four intermediate triangles correspond to the four surfaces that project to the image region shown in red in Figure 6.22a.

Figure 6.22: Clustering of vertices in Smoothsketch (adapted from [10]).

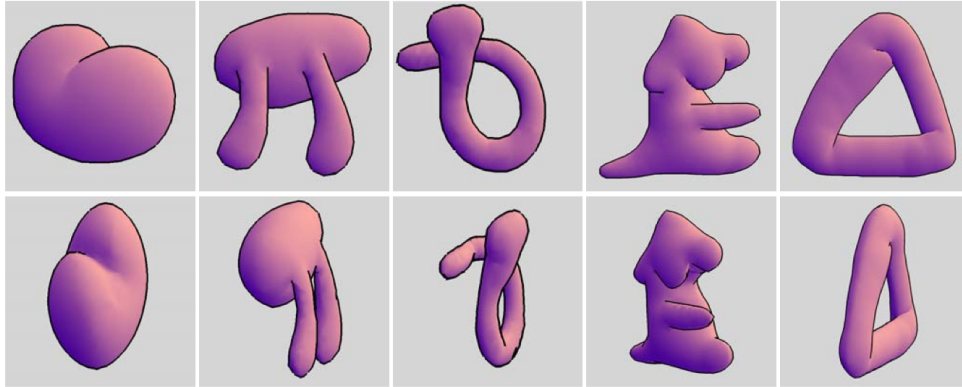


Figure 6.23: Some of surfaces reconstructed using the method introduced by Karpenko et al. from two different views (adapted from [10]).

surfaces are shown in Figure 6.23.

In short, the smoothsketch algorithm achieved the goal of allowing self-occlusions and depth discontinuities in three heuristic steps. For example, the mass-spring system, is an iterative algorithm that seems to have many parameters to tune while the authors have reported it to be sensitive to the size of the triangles.

6.2.5 Notes on solid object reconstruction methods

Viewing the reconstructed objects obtained from these methods from any viewpoint will be equally pleasant, as opposed to the surfaces obtained from the methods reconstructing the visible surface only. As a result, these objects have the advantage of usability. That is, once reconstructed, they can be located in any 3D scene. However, the solid object methods must have special mechanism to deal with self-occlusions. In addition, one must remember that reconstructing a solid object means reconstructing part of the original

object which has not been visible, while this problem does not arise for the methods reconstructing the visible surface. In [7, 8, 10], the problem has been tackled by assuming that the object is symmetric in depth. Terzopoulos et al. [6] solves this issue by starting with a generalized cylinder, i.e. a solid object itself, and Prasad et al. [9] take advantage of user interactions for specifying the depth of the points on the back of the object.

Among the five approaches reviewed here, Oswald et al. and Toppe et al. reconstruct objects with a planar rim. The other three methods reconstruct objects with non-planar rims accurately. The non-planarity of the reconstructed objects by Terzopoulos' method is due to allowing the spine to be a space curve rather than a planar curve. In the methods by Prasad et al. and Karpenko et al., recognizing self-occlusions results in objects with non-planar rims.

Finally, it is worth highlighting that all the solid object methods employ user interactions except the method by Karpenko et al. However, the automatic reconstruction property has added to the complexity of this method. In addition, the reconstructed object might not be as smooth as desired. However, one should remember that the obtained reconstruction is a triangulated mesh and can be easily further polished in a post-editing phase using numerous software resources.

6.3 Notes on optimization-based methods

In general, all of these optimization methods maximize the smoothness of the reconstructed surface. Since the problem is ill-posed, and maximizing the smoothness of the desired surface results in a flat surface, some constraints are imposed. These constraints were either imposed through user interactions or by choosing candidate shapes from a dataset of 3D shapes.

Satisfying the smoothness and non-flatness requirements results in satisfying reconstructions for a limited set of objects. Additional contour constraints empower models to handle more general objects with self-occlusions. Similarly, other cues from the image, such as shading and intensity changes, can be employed to further constrain the surface.

In recent work, Karsch and colleagues [38] studied the effectiveness of various shading and contour cues in shape recovery. These cues include: silhouette, self-occlusion, folds⁶, and shading. A weighted combination of the constraints imposed by these cues is minimized to get the 3D reconstruction of the object. The input to the algorithm is a set of user-specified curves shown by various colors in Figure 6.24. Red and cyan represent parts of the outline across which the surface is smooth and non-smooth respectively. Self-occlusion is shown by green, and folds are orange.

The imposed constraints are very simply formulated. For example, the silhouette

⁶Creases are called folds in [38].

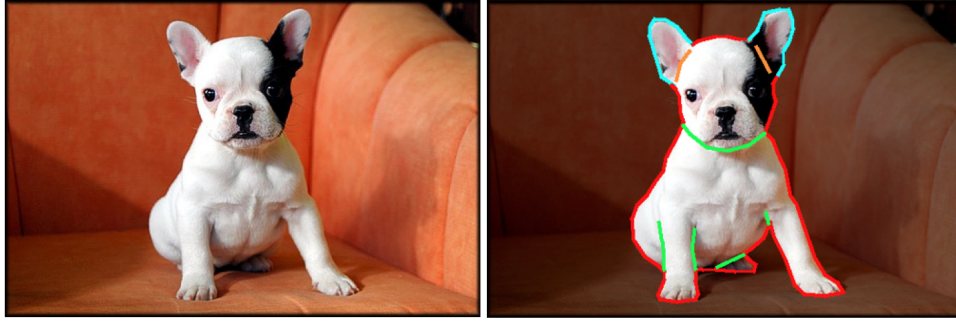


Figure 6.24: The original image on the left, and the curves labelling different cues superimposed on the image (adapted from [38]).

constraint is the same as the contour constraint used by Barron et al. [4]. This constraint, formulated in Equation 6.4, is restricted to the smooth parts of the outline. The same formulation is employed for the self-occlusion curves. Note that the user also marks which part occludes the other. Therefore, the direction of the normal to this curve can be easily determined. The fold constraint imposed by the location of the fold curve and its convexity/concavity ensures the normals along the fold are consistently oriented. The shading cues incorporated into the formulation are that of Barron et al.’s [4]. Two regularization terms, i.e. two shape priors from Barron et al. [4], are added to the objective function to guarantee smoothness of the surface.

In order to evaluate the effect of these cues in a fair setting, the weight of cues are equal in the objective function. The silhouette constraint is always imposed in all the experiments; other cues are evaluated separately and in combination, both qualitatively and quantitatively. In the qualitative setting, performed on 17 of 20 object categories of the VOC12 dataset [39], users are shown the reconstructed surface, asked to rate the

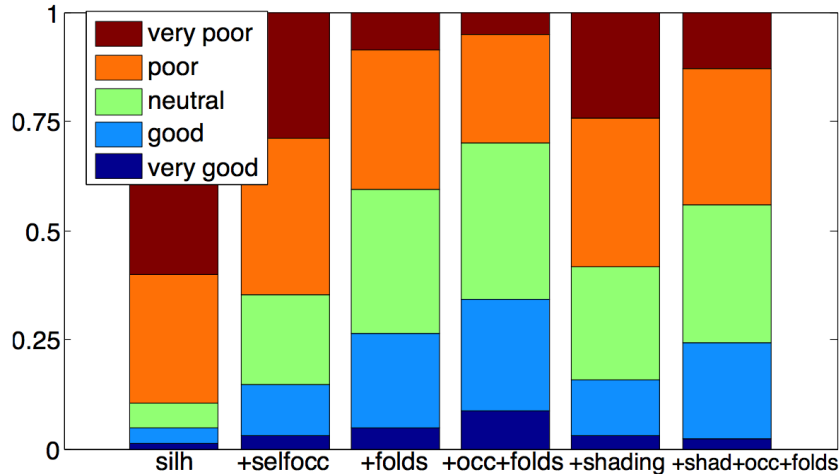


Figure 6.25: Average rating of the reconstructions for the objects in the VOC dataset. All the reconstructions employ silhouette as a constraint. This has been shown by a “+” before the name of the other constraints used for the reconstruction. Users have clearly selected better ratings for the combination of silhouette, self-occlusion and folds. Note that this combination has also higher amount of neutral ratings. Another interesting observation is the confidence of users when rating the reconstructions with the silhouette constraint only. The average neutral rating for these reconstructions is less than the other settings (adapted from [38]).

reconstruction quality, and later are required to recognize the object category. Figure 6.25 plots the result of the ratings by human subjects. Interestingly, subjects gave the highest rating to the combination of silhouette, self-occlusion and folds constraints; adding shading lowered ratings.

Karsch et al. also evaluated the effect of features quantitatively over the MIT Intrinsic Image dataset [40]. They computed the error in normal and depth of the surface points. Similar to the qualitative evaluations, the silhouette + self-occlusion + folds combination out-performed the other combinations. The reconstructed surfaces were also given to two object recognition algorithms [41, 42]. The RGB-D kernel method [41] was given the

RGB values along with the geometric and shading cues. For these methods, similar to the qualitative results, the silhouette + self-occlusion + folds combination yielded better recognition results. In contrast, VLFeat [42] showed better recognition performance when the silhouette and self-occlusion were the cues.

The study performed by Karsch and colleagues [38] confirms the importance of self-occlusion and folds in reconstruction. An open question remains: how does the specific reconstruction model affect results? For example, does replacing the smoothness constraint with the thin-plate energy still confirm the effectiveness of self-occlusion and folds over other cues?

Among the optimization-based methods studied in this chapter, Zhang et al. [3] and Prasad et al. [9] are the only two employing both folds (creases) and self-occlusion cues. Therefore, according to the study by Karsch et al., these two are expected to yield better results compared to the rest. However, one must keep in mind that the accuracy of the reconstructed objects for both of these methods depends strongly on user interactions. This limiting requirement, however, leads to considering the methods by Cole et al. [5] and Karpenko et al. [10] for automatic shape reconstruction. These two approaches recognize self-occlusions, but do not employ fold cues. Nonetheless, they both yield reasonably satisfying 3D shapes. Moreover, the reconstructed shapes can be further modified by users if needed. Also, one possibility is to add fold cues to these two methods.

Table 6.1 compares the optimization-based methods reviewed in this section.

Table 6.1: Comparison of optimization-based methods.

Methods	Smoothness term	Rim	Creases	Self-occlusion	Genus type ⁷	Surface representation	Reconstructed surface type	User interaction
Zhang [3]	Thin-plate energy	Non-planar	Yes	Yes	Any	Height map	Depth map	Yes (input + post-editing)
Barron [4]	Gradient of mean curvature	Non-planar	No	No	Any	Height map	Depth map	No
Cole [5]	Thin-plate energy	Non-planar	No	Yes	Any	Shape patches at keypoints	Depth + normal maps	No
Karsch [38]	Gradient of mean curvature	Non-planar	Yes	Yes	Any	Height map	Depth map	Yes (input)
Terzopoulos [6]	Thin-plate energy	Non-planar	No	No	0, 1	Parametric	Solid object	Yes (initialization of the model + external forces)
Oswald [7]	Weighted TV	Planar	Yes	No	0	Implicit	Solid object	Yes (post-editing)
Toppe [8]	Weighted TV	Planar	Yes	No	0	Implicit	Solid object	Yes (post-editing)
Prasad [9]	Thin-plate energy	Non-planar	Yes	Yes	0, 1, 2	Parametric	Solid object	Yes (input + post-editing)
Karpenko [10]	mass-spring energy	Non-planar	No	Yes	Any	Triangulated mesh	Solid object	No

⁷In this column, only the genus type a method can reconstruct accurately has been reported.

Chapter 7

Conclusion

In this paper, we discussed the problem of reconstructing the 3D shape of objects from a single image, particularly from the occluding contour. This problem is ill-posed and therefore various assumptions have been made for shape recovery. I considered two cases: piecewise planar objects, and smooth objects.

Piecewise planar methods vary from purely geometrical to probabilistic (learning-based) models. These methods are often suitable for reconstructing architectural scenes and buildings.

Methods reviewed in this paper for smooth objects take the contour (silhouette) of the object as input, sometimes with additional information from the user, to recover the 3D shape. There are studies and theories [29, 25] in the literature showing the relationship between the contour and the shape of the object. These studies, though very important,

do not suggest a practical algorithm for obtaining a 3D model for the object.

Algorithms for recovering the shape of smooth objects from the 2D contour are divided into two category: skeleton-based and optimization-based. From this review, I believe the most important limitations of shape from contour methods to be:

- **Lack of a standard dataset:** The experimental results reported for methods discussed in this paper were restricted to a limited set of objects. Although there are 3D shape datasets available [43], there is lack of a standard 3D dataset specifically designed for the field of shape from contour.
- **Lack of quantitative comparison:** Aside from the dataset problem, these methods, except for [38], provide no quantitative comparison against others. In [38], the quantitative comparison is performed by using the recovered shapes in various object recognition methods. The recognition rates were then reported as a quantitative measure of shape recovery success. Various measures for 3D shape similarity have been introduced for 3D shape retrieval [44, 45, 46] that could be employed for comparison of the recovered shapes against ground truth.
- **No unified shape model:** Shape is represented in one of the following forms: implicitly, parametrically, or as a triangulated mesh. Each of these representations has its own advantages and disadvantages, as discussed in Chapter 6.

- **User interactions:** With the exception of [2, 5, 10], success depends on the input received from the users. In some cases the input is given only once at the beginning, while in others the user makes changes to the 3D model by providing more constraints on the problem. Sometimes, a huge amount of input from the user is needed. In addition, for some of the methods, only a trained user can get a satisfying model with few interactions. That is, untrained users need more interactions to get a visually satisfying surface. Puffball [2], although automatic, is a very simple method with major shortcomings (Chapter 5). Smoothsketch [10] has many parameters to tune. Shapecollage [5], appears to work reasonably well, but has only been evaluated on a limited, synthetic dataset.
- **Rim planarity:** Existing methods either assume the rim is planar, or use additional information other than the bounding contour to infer a non-planar rim. For example, in smoothsketch [10], analysis of self-occlusions in addition to the bounding contour, allows inference of a non-planar rim.
- **Orthographic projection:** All the methods reviewed assume orthographic projection.

Ways in which these shortcomings could be addressed include:

- Introducing a public 3D shape dataset for the problem of shape from contour.

- Employing a measure for comparison of similarity of 3D object shapes such as the ones investigated in [44, 45, 46].
- Quantitatively comparing the methods reviewed in the report according to the ground truth and the shape similarity measure.
- Introducing a method for identifying depth discontinuities and self-occlusion from the bounding contour.
- Generalizing to perspective projection.
- Employing a probabilistic approach that allows accurate fusion of multiple shape cues.

Appendix A

Total Variation

Total variation of functions have proved to be effective for removing noise from images. Rudin et al. [47] first suggested employing the total variation norm instead of the commonly used L_2 -norm for image denoising. They formulated image denoising as an optimization problem where the cost function is the total variation norm of the desired images.

The total variation norm of a function is in fact the L_1 -norm of its derivatives. That is, following Rudin's notation, the total variation of the function u on a bounded set $\Omega \in \mathbb{R}^2$ is

$$\text{TV}[u] = \int_{\Omega} |\nabla u| dx dy. \tag{A.1}$$

Consequently, Rudin et al. defined the image denoising problem as

$$\text{minimize} \quad \int_{\Omega} \sqrt{u_x^2 + u_y^2} \, dxdy, \quad (\text{A.2})$$

$$\text{s.t.} \quad \int_{\Omega} u \, dxdy = \int_{\Omega} u_0 \, dxdy,$$

$$\int_{\Omega} \frac{1}{2}(u - u_0)^2 \, dxdy = \sigma^2,$$

where u_0 and u are the observed and desired clean images respectively, and $\sigma > 0$ is the standard deviation of noise. The parameter σ is assumed to be known in advance.

Solving the optimization problem A.2 calls for solving the Euler-Lagrange equation

$$\frac{\partial}{\partial x} \left(\frac{u_x}{\sqrt{u_x^2 + u_y^2}} \right) + \frac{\partial}{\partial y} \left(\frac{u_y}{\sqrt{u_x^2 + u_y^2}} \right) - \lambda_1 - \lambda_2(u - u_0) = 0 \quad \text{in } \Omega, \quad (\text{A.3})$$

$$\frac{\partial u}{\partial n} = 0 \quad \text{on the boundary of the image, i.e. } \partial\Omega, \quad (\text{A.4})$$

where n is the noise. This equation is solved iteratively using the gradient descent method.

Results of denoising an image by solving the Euler-Lagrange equation above are shown in figure A.1.

According to the definition of total variation, i.e. Equation A.1, the total variation norm does not see any difference between the three functions plotted in figure A.2. As a result, unlike the L_2 -norm which make the sharp edges in an image smooth, the total

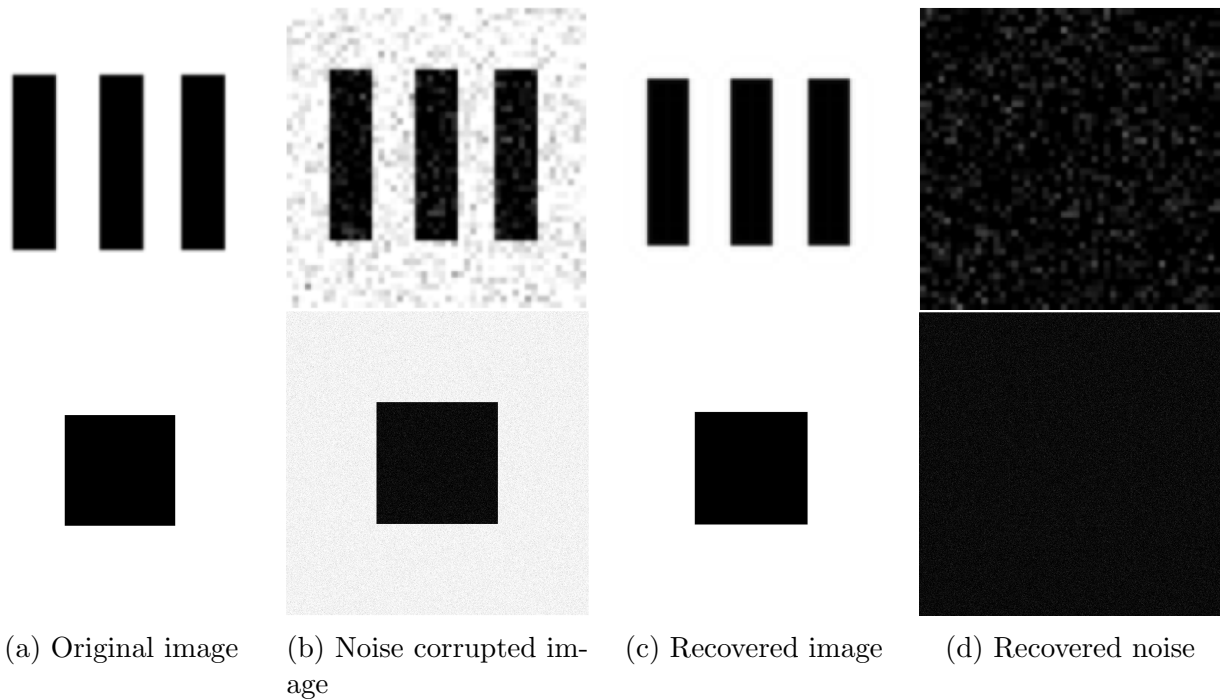


Figure A.1: Recovering the clean image using the total variation norm. The parameter σ was set to 30 for both images.

variation allows the sharp edges to remain untouched in the denoising process. In other words, the TV norm makes the signal smooth unless it observes strong evidence for discontinuity. This effect can be seen in the recovered image in figure A.1.

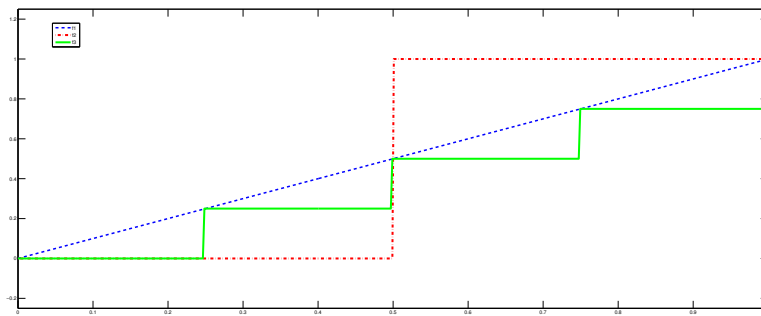


Figure A.2: The total variation norm does not see any difference between the three functions in this graph.

Appendix B

Skeleton and Medial Axis Transform

Finding the skeleton of a silhouette has been common practice for many applications in computer vision. Since the skeleton is one of the forms for shape representation, it has been used for shape recognition [48]. Also, as shown in Chapter 5, the skeleton can be employed for inflating a silhouette to reconstruct the 3D model of the object. There are different methods for finding the skeleton such as the medial axis transform [49] or using Voronoi techniques followed by a usually heuristic pruning step [1]. This appendix focuses on the medial axis transform.

Given the outline of a silhouette, its medial axis is the locus of points having more than one closest point on the outline. The medial axis is also called the “skeleton”. The medial axis with the distance of its points to the outline, i.e. the radius function, are called the *medial axis transform* (MAT). In technical texts, however, skeleton and MAT

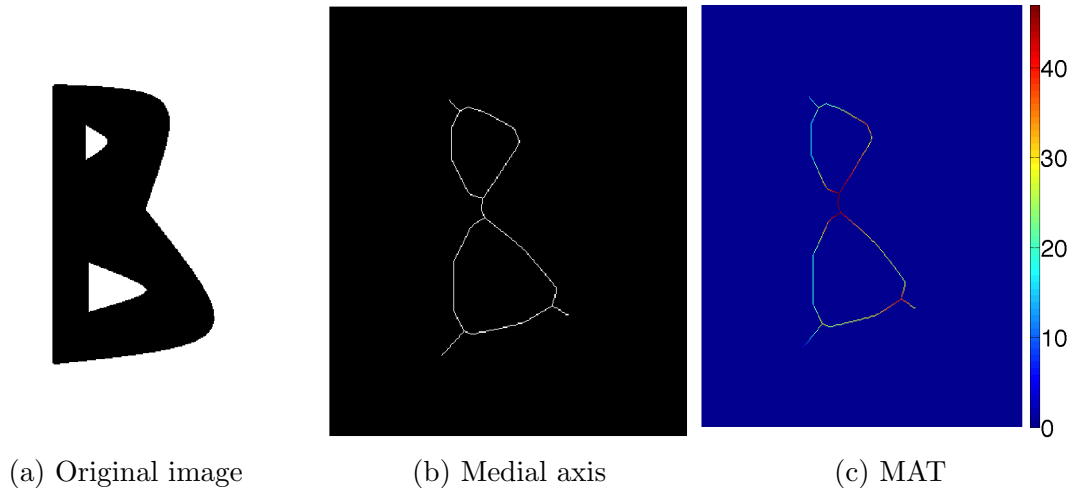


Figure B.1: The medial axis and MAT of a “B” shape.

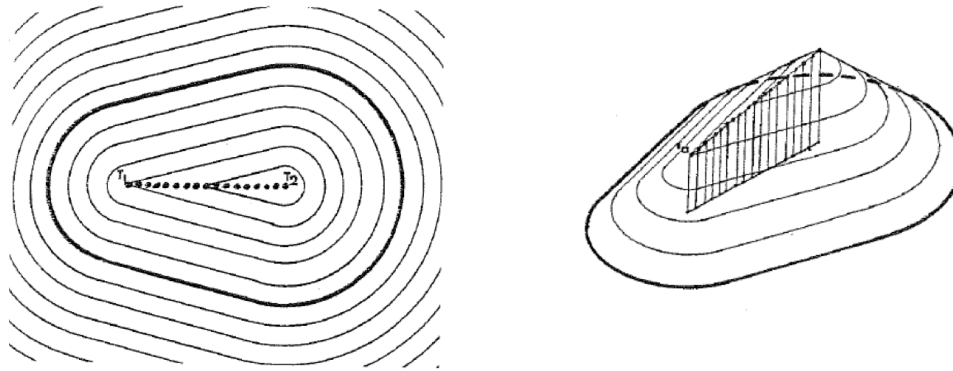
are used interchangeably. Figure B.1 shows the medial axis transform and the medial axis of a “B” shape.

As proposed by Blum [49], the grassfire height function can be used for finding the medial axis. Suppose the plane the outline of a silhouette lives in is made of extinguishable material and a fire is started from the points on the outline moving inward the silhouette.

Also, assume that the following three rules govern the propagation of the fire:

- Each point can be either burnt or not; nothing in the middle,
- Each point sets its neighbors on fire at a time proportional to its distance to the neighbor,
- Each point already burnt can not be set on fire again.

Drawing the border of the fire at each time gives the locus of points of equidistance to the outline. Therefore, to each point inside the silhouette, a number equal to its distance



(a) Contours of equidistance points to the outline. (b) The height function. The ridges of this function give rise to the medial axis.

Figure B.2: The medial axis and grassfire height functions (adapted from [49]).

to the outline can be assigned. This gives a function called the grassfire height function.

Figure B.2a depicts the contours of equidistance to the outline for the regions both inside and outside of the outline. Also, figure B.2b illustrates the grassfire height function.

The ridges of the grassfire height function give the medial axis. Then, the medial axis transform is simply the medial axis with the associated heights from the height function.

Appendix C

Bezier Curves

In this appendix, Bezier curves are introduced. The idea of Bezier curves is based upon Bernstein polynomials. These curves are extensively used in computer graphics for character animation [50, 51].

C.1 Bernstein Polynomials

A polynomial is in Bernstein form if it is a linear combination of Bernstein basis polynomials. Such polynomials can be used for approximating a function. The Bernstein basis polynomials of degree n are defined as:

$$b_{\nu,n}(x) = \binom{n}{\nu} x^{\nu}(1-x)^{n-\nu}, \quad \nu = 0, \dots, n. \quad (\text{C.1})$$

The Bernstein polynomial is then defined as:

$$B(x) = \sum_{\nu=0}^n \beta_{\nu} b_{\nu,n} \quad (\text{C.2})$$

where β_{ν} 's are called the Bernstein coefficients. One can see this linear combination as the weighted average of probabilities of success of a random variable with binomial distribution. Thereby, if all the coefficients are equal to 1, we have $B(x) = 1$ which is the probability of the sample space. Bezier curves are in fact Bernstein polynomials restricted to $x \in [0, 1]$.

C.2 Bezier Curves

Computer graphics benefit from Bezier curves for modelling smooth curves. These curves are usually specified by a number of control points. Bezier curves represent a curve parametrically. In practice, quadratic and cubic Bezier curves are often used. Below is an explanation of three special cases of Bezier curves:

- **Linear** : This case can also be seen as a convex combination of two control points p_0 and p_1 . Linear Bezier curve is

$$B(t) = (1 - t)p_0 + tp_1, \quad t \in [0, 1]. \quad (\text{C.3})$$

- **Quadratic:** Using three control points p_0 , p_1 , and p_2 , a quadratic Bezier curve is defined as:

$$B(t) = (1 - t)[(1 - t)p_0 + tp_1] + t[(1 - t)p_1 + tp_2], \quad t \in [0, 1]. \quad (\text{C.4})$$

Note that the quadratic Bezier can be rewritten recursively as:

$$B(t) = (1 - t)B_{p_0, p_1}(t) + tB_{p_1, p_2}(t), \quad t \in [0, 1], \quad (\text{C.5})$$

where $B_{x,y}(t)$ is the linear Bezier curve with x and y as the control points.

- **Cubic:** Similar to the quadratic Bezier curve, cubic Bezier curve with four control points $p_i, i = 0, \dots, 3$ can be defined recursively as:

$$B(t) = (1 - t)B_{p_0, p_1, p_2}(t) + tB_{p_1, p_2, p_3}(t), \quad t \in [0, 1], \quad (\text{C.6})$$

which yields,

$$B(t) = (1 - t)^3 p_0 + 3(1 - t)^2 t p_1 + 3(1 - t) t^2 p_2 + t^3 p_3, \quad t \in [0, 1]. \quad (\text{C.7})$$

A Bezier curve of degree n is, therefore, a linear combination of Bernstein basis poly-

nomials where the control points are the Bernstein coefficients.

Bibliography

- [1] T. Igarashi, S. Matsuoka, and H. Tanaka, “Teddy: a sketching interface for 3D freeform design,” in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH ’99, (New York, NY, USA), pp. 409–416, ACM Press/Addison-Wesley Publishing Co., 1999.
- [2] N. R. Twarog, M. F. Tappen, and E. H. Adelson, “Playing with puffball: simple scale-invariant inflation for use in vision and graphics,” in *Proceedings of the ACM Symposium on Applied Perception*, SAP ’12, (New York, NY, USA), pp. 47–54, ACM, 2012.
- [3] L. Zhang, G. Dugas-Phocion, J.-S. Samson, and S. Seitz, “Single view modeling of free-form scenes,” in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I–990–I–997 vol.1, 2001.
- [4] J. Barron and J. Malik, “Shape, albedo, and illumination from a single image of an unknown object,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 334–341, 2012.
- [5] F. Cole, P. Isola, W. Freeman, F. Durand, and E. Adelson, “Shapecollage: Occlusion-aware, example-based shape interpretation,” in *Computer Vision ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), vol. 7574 of *Lecture Notes in Computer Science*, pp. 665–678, Springer Berlin Heidelberg, 2012.
- [6] D. Terzopoulos, A. Witkin, and M. Kass, “Symmetry-seeking models and 3D object reconstruction,” *International Journal of Computer Vision*, vol. 1, pp. 211–221, 1987.
- [7] M. Oswald, E. Toppe, K. Kolev, and D. Cremers, “Non-parametric single view reconstruction of curved objects using convex optimization,” in *Pattern Recognition* (J. Denzler, G. Notni, and H. Süe, eds.), vol. 5748 of *Lecture Notes in Computer Science*, pp. 171–180, Springer Berlin Heidelberg, 2009.
- [8] E. Toppe, M. R. Oswald, D. Cremers, and C. Rother, “Image-based 3D Modeling via Cheeger Sets,” in *Asian Conference on Computer Vision*, nov 2010.
- [9] M. Prasad, A. Zisserman, and A. W. Fitzgibbon, “Single view reconstruction of curved surfaces,” in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, pp. 1345–1354, 2006.

- [10] O. A. Karpenko and J. F. Hughes, “SmoothSketch: 3D free-form shapes from complex sketches,” in *ACM SIGGRAPH 2006 Papers*, SIGGRAPH ’06, (New York, NY, USA), pp. 589–598, ACM, 2006.
- [11] A. Criminisi, I. Reid, and A. Zisserman, “Single view metrology,” *Int. J. Comput. Vision*, vol. 40, pp. 123–148, nov Nov. 2000.
- [12] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.
- [13] A. Criminisi, *Accurate Visual Metrology from Single and Multiple Uncalibrated Images*. PhD thesis, University of Oxford, Dept. Engineering Science, 1999. D.Phil. thesis.
- [14] A. A. Efros and W. T. Freeman, “Image quilting for texture synthesis and transfer,” in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH ’01, (New York, NY, USA), pp. 341–346, ACM, 2001.
- [15] D. Hoiem, A. A. Efros, and M. Hebert, “Automatic photo pop-up,” in *ACM SIGGRAPH 2005 Papers*, SIGGRAPH ’05, (New York, NY, USA), pp. 577–584, ACM, 2005.
- [16] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” *Int. J. Comput. Vision*, vol. 59, pp. 167–181, Sept. 2004.
- [17] M. Collins, R. E. Schapire, and Y. Singer, “Logistic Regression, AdaBoost and Bregman Distances,” in *MACHINE LEARNING*, pp. 158–169, 2000.
- [18] A. Saxena, S. H. Chung, and A. Y. Ng, “Learning depth from single monocular images,” in *NIPS 18*, MIT Press, 2005.
- [19] T. Xue, J. Liu, and X. Tang, “Symmetric piecewise planar object reconstruction from a single image,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 2577–2584, 2011.
- [20] P. Sturm, “A method for 3D reconstruction of piecewise planar objects from single panoramic images,” in *Omnidirectional Vision, 2000. Proceedings. IEEE Workshop on*, pp. 119–126, 2000.
- [21] C. Rother, V. Kolmogorov, and A. Blake, ““grabcut”: interactive foreground extraction using iterated graph cuts,” in *ACM SIGGRAPH 2004 Papers*, SIGGRAPH ’04, (New York, NY, USA), pp. 309–314, ACM, 2004.
- [22] A. N. Pressley, *Elementary Differential Geometry*. Springer, 2nd ed. 2010 ed., Mar. 2010.
- [23] J. J. Koenderink, *Solid shape*. MIT Press Cambridge, Mass, 1990.

- [24] B. O’Neill, *Elementary Differential Geometry*. Harcourt/Acedemic Press, second ed., 1997.
- [25] J. J. Koenderink, “What does the occluding contour tell us about solid shape?,” *Perception*, vol. 13, no. 3, pp. 321–330, 1984.
- [26] R. Cipolla and A. Blake, “Surface shape from the deformation of apparent contours,” *International Journal of Computer Vision*, vol. 9, no. 2, pp. 83–112, 1992.
- [27] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in *Proc. 8th Int’l Conf. Computer Vision*, vol. 2, pp. 416–423, July 2001.
- [28] R. Cipolla, G. Fletcher, and P. Giblin, “Following cusps,” *Int. J. Comput. Vision*, vol. 23, pp. 115–129, June 1997.
- [29] J. J. Koenderink and A. J. van Doorn, “The shape of smooth objects and the way contours end,” *Perception*, vol. 11, no. 2, pp. 129–137, 1982.
- [30] Y. Boykov and G. Funka-Lea, “Graph cuts and efficient n-d image segmentation,” *Int. J. Comput. Vision*, vol. 70, pp. 109–131, Nov. 2006.
- [31] R. Szeliski, “Fast surface interpolation using hierarchical basis functions,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 6, pp. 513–528, 1990.
- [32] T. Hassner and R. Basri, “Example Based 3D Reconstruction from Single 2D Images,” in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW ’06. Conference on*, pp. 15–15, 2006.
- [33] J. S. Yedidia, W. T. Freeman, and Y. Weiss, “Constructing free-energy approximations and generalized belief propagation algorithms,” *Information Theory, IEEE Transactions on*, vol. 51, pp. 2282–2312, July 2005.
- [34] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
- [35] D. Terzopoulos and K. Fleischer, “Deformable models,” *The Visual Computer*, vol. 4, no. 6, pp. 0178–2789, 1988.
- [36] D. A. Huffman, “Impossible Objects as Nonsense Sentences,” *Machine Intelligence*, vol. 6, pp. 295–323, 1971.
- [37] L. R. Williams, “Perceptual completion of occluded surfaces.” PhD thesis, 1994.
- [38] K. Karsch, Z. Liao, J. Rock, J. T. Barron, and D. Hoiem, “Boundary Cues for 3D Object Shape Recovery,” *Conference on Computer Vision and Pattern Recognition*, 2013.

- [39] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.” <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>.
- [40] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman, “Ground-truth dataset and baseline evaluations for intrinsic image algorithms,” in *International Conference on Computer Vision*, pp. 2335–2342, 2009.
- [41] L. Bo, X. Ren, and D. Fox, “Depth Kernel Descriptors for Object Recognition,” in *IROS*, September 2011.
- [42] A. Vedaldi and B. Fulkerson, “Vlfeat: an open and portable library of computer vision algorithms,” in *Proceedings of the international conference on Multimedia, MM '10*, (New York, NY, USA), pp. 1469–1472, ACM, 2010.
- [43] J. Zhang, R. Kaplow, R. Chen, and K. Siddiqi, “The mcgill shape benchmark,” 2005.
- [44] L. Zhang, M. J. da Fonseca, A. Ferreira, and Recuperação, “Survey on 3D Shape Descriptors,”
- [45] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, “Matching 3d models with shape distributions,” in *Proceedings of the International Conference on Shape Modeling & Applications, SMI '01*, (Washington, DC, USA), pp. 154–, IEEE Computer Society, 2001.
- [46] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, “On Visual Similarity Based 3D Model Retrieval,” *Computer Graphics Forum*, vol. 22, no. 3, pp. 223–232, 2003.
- [47] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Phys. D*, vol. 60, pp. 259–268, Nov. 1992.
- [48] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker, “Shock graphs and shape matching,” *International Journal of Computer Vision*, vol. 35, no. 1, pp. 13–32, 1999.
- [49] H. Blum, “A transformation for extracting new descriptors of shape,” *Models for the Perception of Speech and Visual Form*, pp. 362–380, 1967.
- [50] T. W. Sederberg and S. R. Parry, “Free-form deformation of solid geometric models,” *SIGGRAPH Comput. Graph.*, vol. 20, pp. 151–160, Aug. 1986.
- [51] P. Faloutsos, M. van de Panne, and D. Terzopoulos, “Dynamic free-form deformations for animation synthesis,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, pp. 201–214, 1997.