THE ROLE OF PRIORS IN VISUAL PERCEPTION AND THEIR APPLICATIONS IN COMPUTER VISION

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Vijai Thottathil Jayadevan

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Edward J. Delp,Chair

Department of Electrical and Computer Engineering

Dr. Zygmunt Pizlo

Department of Psychological Sciences

Dr. Charles A. Bouman

Department of Electrical and Computer Engineering

Dr. Mireille Boutin

Department of Electrical and Computer Engineering

Approved by:

Dr. Dimitrios Peroulis

Head of the School Graduate Program

To my parents, brother and wife.

ACKNOWLEDGMENTS

I would like to thank my advisers Dr. Edward Delp and Dr. Zygmunt Pizlo for their support and guidance throughout the doctoral program. Having only worked on computer vision problems prior to arriving at Purdue, I had no exposure to the field of visual perception. When Dr.Delp introduced me to Dr.Pizlo's work on visual perception, I found it to be both interesting and inspiring. I'm grateful to Dr.Pizlo for helping me understand the world of visual perception through the numerous productive discussions we had over the years. I have benefitted immensely from these discussions and also from his vast knowledge of the historical developments in the field. I would also like to thank Dr. Mireille Boutin and Dr. Charles Bouman for serving on my committee and for reviewing my work.

I would like to thank Dr. Tadamasa Sawada for his help and guidance in designing and conducting the psychophysical experiments that are part of this dissertation. I would also like to thank my lab mate Aaron Michaux for his help and support. I have always enjoyed the academic and non-academic discussions we had.

Last but not the least, I would like to thank my parents, Savithri and Jayadevan, my brother, Sajai and my wife, Ranjini. Their unconditional love and support has kept me going through the toughest of times, and without them none of this would have been possible.

This research was supported by the National Eye Institute of the National Institutes of Health under award number 1R01EY024666-01. The contents of this dissertation is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

ABSTRACT

Thottathil Jayadevan, Vijai PhD, Purdue University, May 2020. The Role of Priors in Visual Perception and their Applications in Computer Vision . Major Professors: Edward J. Delp and Zygmunt Pizlo.

Three-dimensional (3D) vision is an ill-posed inverse problem. The formation of the 2D image of a 3D shape/scene is the forward problem, and inferring the 3D shape/scene from the image is the inverse problem. The ill-posedness is related to the fact that any given 2D image is consistent with infinitely many 3D interpretations. In order to produce a unique and ideally correct interpretation, one has to impose constraints (aka priors) on the family of possible interpretations. Symmetry, compactness, minimal surface area, planarity etc., are some priors used by the visual system to deal with the ill-posedness of the problem. In the first part of this dissertation, a psychophysical experiment conducted to better understand how these priors operate in the visual system is discussed. In the second part, a method that uses some of these priors to recover 3D shapes from a single image is described. And in the last part, a translational symmetry based algorithm to extract curve skeletons from 3D point clouds by decomposing the point clouds into its parts is presented.

Prior studies have found that, the perception of symmetric abstract polyhedral shapes, can be well modeled using the above mentioned priors and binocular depth order information [1]. In this study, it is shown that these priors can be used to model asymmetrical shapes obtained from affine distortions of symmetric shapes. The experiment shows that the perception of symmetrical shapes is closer to veridical in comparison to asymmetrical shapes. Metrics to measure asymmetry of abstract polyhedral shapes and to measure shape dissimilarity between two polyhedral shapes are introduced. A control experiment which proves the goodness of the model is also

presented. A website was developed with all the shapes used in the experiment, along with the user reconstructed shapes and the model reconstructed shapes.

To recover 3D shapes from a single view, symmetry and planarity constraints are used. Long smooth curves are extracted from the edge map of an image by solving the shortest (least-cost) path problem, where the cost function penalizes large interpolations and large turning angles. Optimal curve matches, that minimize the number of planes required to approximate the final 3D reconstruction, are then found. This optimization problem is framed as a binary integer program.

To extract curve skeletons from 3D point clouds, the cloud is decomposed into its parts. Generalized cylinders (GCs) are used to represent parts. Since, the axis of a GC is an integral part of its definition, the parts have natural skeletal representations. Cross-sections of parts are first detected and parts are then grown starting from this initial cross-sections. Translational symmetry, the fundamental property of GCs, is employed to grow the parts. A large number of such candidate parts are grown starting from different positions in the point cloud. Each part is assigned a score based on how well these parts can be represented as a GC. An optimization algorithm is then employed to select the best subset of parts, from within the candidate parts, to represent the decomposition of the object.

1. INTRODUCTION

We see things in the world in 3D and our visual perception is almost always veridical. To a common man, this statement would not be controversial. But, that is definitely not the case in the long history of research on visual perception. Even in recent times, there has been numerous studies that question the veridicality of visual perception [2, 3]. More radical theories about the nature of perception exist. Such theories, mostly based on evolutionary psychology, question the very existence of a visual representation of the world. One such example is the interface theory of Hoffman et al. [4]. The nature of visual representation, i.e., whether it is 2D or 3D, is also a topic of debate. Hermann von Helmholtz, owing to the 2D nature of the retinal image, suggested that the visual representation is 2D and not 3D [5]. The argument here is that, having seen objects in the world from different views many times, the visual system then draws on this memory to enable veridical perception. David Marr, in his influential work [6], proposed the concept of a 2.5D representation. According to this theory, the visual system first infers the visible surface orientations from depth cues. In the next step, this surface orientation information is used in combination with the 3D models stored in memory to obtain the full 3D representation. A related question here, is whether visual perception is innate or is it developed as a result of learning (the empiricism vs. nativism debate).

Research on human 3D shape perception started with two seminal papers published the same year in the same journal: Hochberg & McAlister's [7] and Wallach & O'Connell [8]. The authors of these papers addressed the fundamental question in vision: how is the 3D percept of an object produced from a single 2D retinal image. Hans Wallach, who received his training with one of the founding fathers of the Gestalt Psychology, started his paper with a note of disappointment that despite numerous attempts, no Gestalt Psychologist, nor anyone else, was able to show how Prägnanz,

or simplicity principle can produce veridical 3D percepts of shapes. So, he turned his attention to the competing tradition of empiricism, and set out to demonstrate that it is learning based on motion cues, rather than any innate simplicity predilection, that teaches human observers about the three-dimensionality of objects. Wallach and O'Connell showed that when the observer looks at a stationary shadow of an unstructured 3D polygonal line object, he never perceives a 3D shape. But, when the 3D object rotates, the changing shadow of the rotating object leads to a 3D percept (they called this "kinetic-depth-effect"). This way, Wallach & O'Connell provided evidence that motion, in the absence of any other cue, can produce a 3D percept. However, they did not explain how the 3D percept is actually produced by 2D motion cue. It was Hay [9] and later Ullman [10] and Longuet-Higgins [11], who formulated a mathematical and computational theory of kinetic depth effect. The computational theory is called, after Ullman [10], structure from motion theorem (SFM). What was obvious to the authors of SFM and was not to Wallach & O'Connell, is that 3D shape cannot be computed from motion cue without a rigidity predilection, or constraint. Indeed, all explanations of SFM rely on the assumption that the individual views are images of the same object and the object is at least approximately rigid. So, in a sense, Wallach who was looking for an empiristic theory of 3D vision, in which the role of simplicity constraints is minimal or absent, altogether, ended up providing experimental evidence for the operation of such a constraint.

Simplicity constraint was an explicit motivation behind the second seminal contribution to our understanding of 3D shape perception that was mentioned in the previous paragraph [7]. Hochberg & McAlister used Kopfermann's [12] observation on the role of viewing direction in perception of a 3D Necker (transparent) cube from a single 2D line drawing. They observed that a 3D cube, which is characterized by a high degree of simplicity or redundancy, is always perceived as a 3D shape, except for very special 2D views, which themselves are very simple. They, like Wallach & O'Connell, did not provide a computational theory of how the 3D percept is produced from a single 2D image, but their paper inspired others to continue research

on the role of constraints in 3D shape perception (e.g., [13–18]). We now know that constraints are absolutely essential in 3D shape recovery because this recovery is an ill-posed inverse problem [19].

Pizlo et al., over the past couple of decades, built on this idea of visual perception as an ill-posed inverse problem. The story of how these ideas lead to a comprehensive computational model of visual perception is captured in the book titled "Making a machine that sees like us" [20]. In this computational model, symmetry plays a very important role as a fundamental constraint employed by the human visual system. An appreciation for the fundamental role of symmetry in visual perception is essential in understanding the motivation behind all the work done as part of this dissertation. Equally important is understanding the idea of an ill-posed inverse problem and the key role of priors in solving such problems. The following sections (sections 1.1 and 1.2) provide a brief overview of these concepts. These sections are followed by section 1.3, which illustrates the effectiveness of symmetry prior in solving the inverse problem of 3D shape reconstruction. And finally, section 1.4, provides an overview of the work proposed in this dissertation.

1.1 Symmetry and Shape

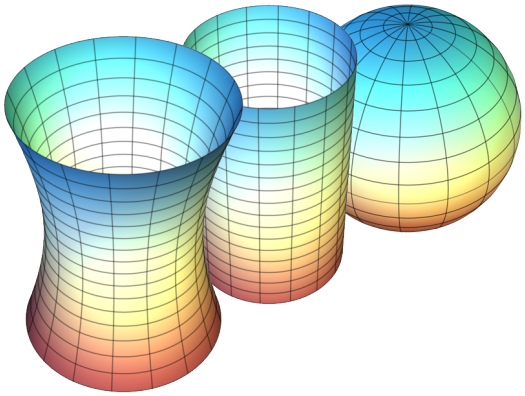
Defining shape, an intuitively obvious concept to one and all, had been a difficult task throughout history. Conventionally, shape was understood to be some abstract property which is invariant under transformation. More specifically, two shapes are said to be the same if one shape can be transformed into the other by *rigid motion, size-scaling* and/or *reflection* (i.e., by a similarity transformation). The opposing conclusions reached by Rock & DiVita [21] and Biederman & Gerhardstein [22], in their respective studies on shape constancy (the ability to identify the same shape when viewed from different directions), highlight a problem that such a definition can cause. While the former concluded that shape constancy does not exist, the latter found that shape constancy does exist. In an attempt to resolve this ambiguity, Pizlo

et al. noticed that, one of the problems with such a definition, is the assumption that all objects have shape. Or in other words, the idea that shape can exist without shape constancy is the problem. Does a random pattern of dots, a crumbled piece of paper or a naturally occurring rock have shape? Does it make sense to talk about shape without shape constancy?

(a) Reflection symmetry

(b) Spiral Symmetry

(c) Rotational symmetry



(d) Translational Symmetry

Fig. 1.1. Examples of different types of symmetries

Common sense tells us that shape constancy very well exists in our day to day life. This debate about the existence or non-existence of shape constancy can be settled if we understand how Pizlo et al. defines shape. According to this definition, *the shape of an object refers to all of its spatially global symmetries (its self-similarities) as measured by the group of rigid motions, reflections, and size-scaling of the parts within*

the object itself. According to this definition of shape, the amount of symmetries within an object would decide the amount of shape it has. Therefore, shape of an object can be measured on a scale from 0 to 1. All biological forms exhibit some form of symmetry and therefore have shape. For instance, animal and human bodies are mirror-symmetric. Apart from this, human and animal body parts, like limbs and fingers, display translational symmetry. Similarly tree trunks and stems of plants exhibit translational symmetry and seashells are spirally symmetric. See Figure 1.1 for some examples. Animals are mirror symmetrical because of the way they move. Plants are symmetrical because of the way they grow [23]. Man-made objects are symmetrical because of the functions they serve. A completely asymmetrical object would be dysfunctional.

With these ideas in mind, we notice that, bent wires, the stimuli used by Rock & DiVita [21], had very little shape. While Biederman & Gerhardstein [22] used abstract objects composed of generalized cones, which had lots of shape, as stimuli. This simply means that the latter was studying shape while the former was not. This also explains the different conclusions reached by these studies. The experiment conducted by Pizlo & Stevenson [24] and the one by Li & Pizlo [25], provide further support for the above mentioned definition of shape.

Fig. 1.2. Symmetry in man-made objects

1.2 Visual Perception as an Ill-posed Inverse Problem

Three-dimensional (3D) vision is an ill-posed inverse problem. Forming a 2D retinal or camera image of a 3D object is a forward problem and it is described in the rules of geometrical optics. Forward problems are usually easy: they are well-posed and well-conditioned. By well-posed we mean that a solution exists, it is unique and depends continuously on the data [26]. By well-conditioned we mean that the solution is computationally stable. More generally, in the theory of inverse problems, forward problem refers to producing data from a model, where a model could be a physical object whose image is taken by a camera, patient's chest whose health status is represented in the chest X-ray, center of an earthquake which produced mechanical vibrations detected by sensors on the surface of the Earth, or a natural law, such as Newton's second law of motion, tested by a college student by measuring times and distances of a free-falling object [27]. An inverse problem refers to going from data to a model. So, an inverse problem consists in making inferences about the true state of affairs "out there" that led to the data at hand. Most inverse problems in science and engineering are ill-posed and/or ill-conditioned. In 3D vision, the ill-posedness is related to the fact that any given 2D image is consistent with infinitely many 3D interpretations. In order to produce a unique and, ideally, a correct interpretation, one has to impose constraints (aka priors) on the family of possible interpretations [19,20,28–31]. Ames's chair is a classical example illustrating these observations (go to: `http://shapebook.psych.purdue.edu/1.3/`). When a 2D image is consistent with a chair interpretation, we see a chair despite the fact that this image could have been produced by a set of disconnected parts.

So, what are some of the priors that can be used to deal with the ill-posedness of 3D vision? Given its central role in the definition of shape, it should not be surprising that symmetry is an important prior. The fact that we see a 3D object called a "chair" is not because we have seen many chairs in our life, but rather that this kind of an object is the only 3D symmetrical interpretation of this 2D image. Moreover, symmetry is

ubiquitous in nature and in man-made structures (Figures 1.1 and 1.2). In fact, we are very likely to see a symmetric object whenever such an interpretation is possible (see the demo at: `http://shapebook.psych.purdue.edu/2.1/Symmetry/`). Planarity is another prior used by the visual system. The famous Ames room illusion and the demo at `http://shapebook.psych.purdue.edu/2.1/Planarity/` provide evidence for this fact. The idea here again is that we are very likely to perceive planarity whenever such an interpretation is possible. Compactness is yet another example of a prior that the visual system employs (see the demo at: `http://shapebook.psych.purdue.edu/2.1/Compactness/`). Compactness of a shape is the ratio V^2/S^3, where V is the volume and S is the surface area. Li et al. [32] showed that, for symmetric objects viewed monocularly, maximum compactness and minimum surface priors could be used to recover the perceived 3D shape from the 2D image. Specifically, the 3D shape that maximizes the ratio, V/S^3 will model the perceived 3D shape very well.

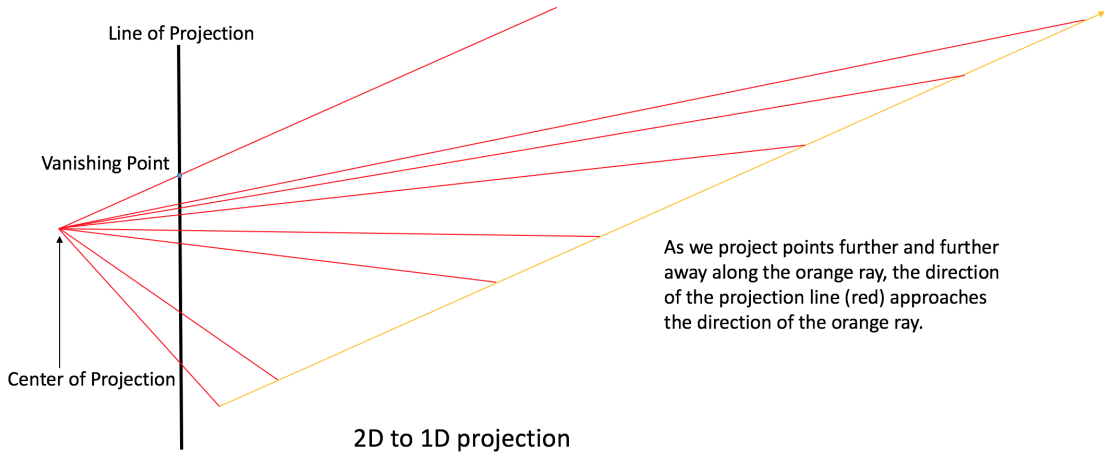
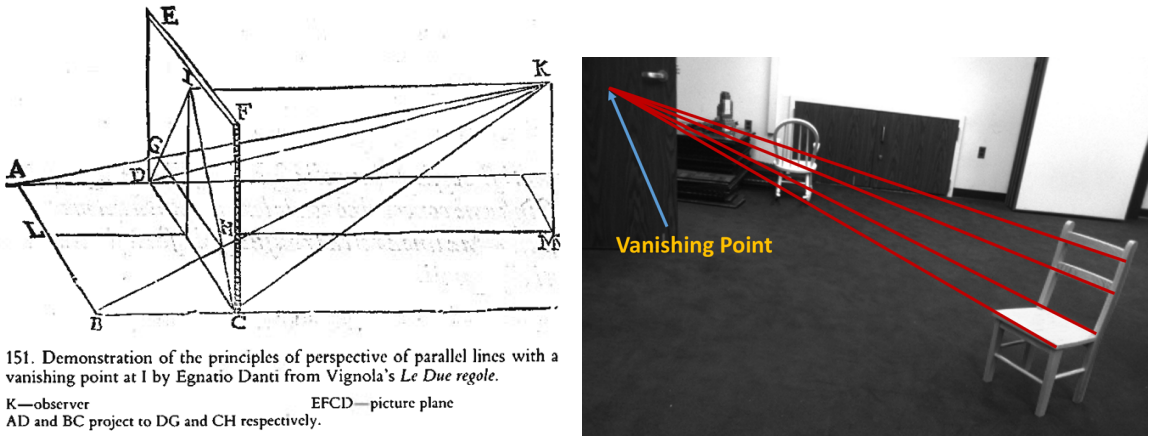


Fig. 1.3. Illustrating the concept of vanishing point using a 2D to 1D projection

1.3 The Symmetry Correspondence Problem

If three things are known, a bilaterally symmetric 3D shape can be reconstructed from its perspective image. These three things are: 1) the intrinsic parameters of the camera, 2) the symmetry plane and 3) the symmetry correspondence between various parts of the shape in the image. The intrinsic parameters of the camera can be obtained by calibrating the camera. The vanishing points in the image can tell us about the symmetry plane(s) (a shape could have multiple planes of symmetry). A vanishing point is a point on the image plane where the perspective projection of 3D parallel lines intersect (see Figure 1.4). Therefore, a vanishing point represents a particular direction in the 3D world. To understand this better, look at Figure 1.3. It represents a 2D to 1D projection. As we project points further and further away along the orange ray, the direction of the projection line (red) approaches the direction of the orange ray. This is why the vanishing point is said to be the projection of a point at infinity. Let a symmetry line be defined as a line connecting two 3D points that are symmetric with respect to a symmetry plane. Figure 1.4 (b), shows a symmetric chair. The red lines are the projections of symmetry lines. By definition, all the symmetry lines for the object shown in Figure 1.4 (b) are perpendicular to the plane of symmetry, and hence represent the same 3D direction. Therefore, there is a vanishing point in the image corresponding to that direction, let us call it V_P. If we imagine a 3D line with the same direction as the symmetry lines ,and passing through the center of perspective projection, this line will intersect the image plane at V_P. Therefore, if we know the vanishing point, we also know the direction of the normal of the symmetry plane. Two pixels, p and q, are said to symmetrically correspond if they are the projections of two 3D symmetric points. If the symmetry correspondence of all the pixels representing the object is known, and if the normal of the symmetry plane is known, then the object can be accurately reconstructed in three dimensions up to a scale factor. If the shape is of particular interest, this scale factor does not really matter, as the shape is scale invariant. Therefore, once

we know the normal of the symmetry plane, a convenient distance to the symmetry plane can be assumed to set the symmetry plane exactly. An expression for the 3D coordinates of two symmetrically corresponding points, given the symmetry plane and the intrinsic camera parameters can be derived as follows.



(a) (b)

Fig. 1.4. a) Vanishing point illustration by Egnatio Danti and b) Vanishing point illustrated on a real image

Without loss of generality, let the principal point on the image be $(0, 0)$. Note, principal point is the point on the image plane at which the centre of perspective projection is projected. I.e., it is the point of intersection of the optical axis and the image plane. Let the equation of the symmetry plane be $n_1x + n_2y + n_3z - d = 0$, where the unit vector $\hat{\mathbf{n}} = (n_1, n_2, n_3)$ represents the normal of the symmetry plane. Let image points $\mathbf{p}$ with coordinates (x_p, y_p) and $\mathbf{q}$ with coordinates (x_q, y_q) correspond symmetrically (See Figure 1.5). Let $\vec{\boldsymbol{d_p}} = (x_p, y_p, -f)$, where f is the focal length, represent the 3D vector from the origin (center of projection) towards $\mathbf{p}$ on the image plane. Let $\hat{\mathbf{d}}_\mathbf{p}$ represent the unit vector in the direction $\vec{\boldsymbol{d_p}}$, i.e., $\hat{\mathbf{d}}_\mathbf{p} = \vec{\boldsymbol{d_p}}/|\vec{\boldsymbol{d_p}}|$, where $|\vec{\boldsymbol{d_p}}|$ represents the magnitude of the vector $\vec{\boldsymbol{d_p}}$. Similarly, let $\hat{\mathbf{d}}_\mathbf{q}$ represent the unit vector pointing towards the image point $\mathbf{q}$. Let the 3D point $\mathbf{P}$, of which $\mathbf{p}$ is the projection, be given by $k_p\hat{\mathbf{d}}_\mathbf{p}$, where k_p is a positive scalar value which determines how much to travel along the direction $\hat{\mathbf{d}}_\mathbf{p}$ to get to the 3D point $\mathbf{P}$. Then, the 3D

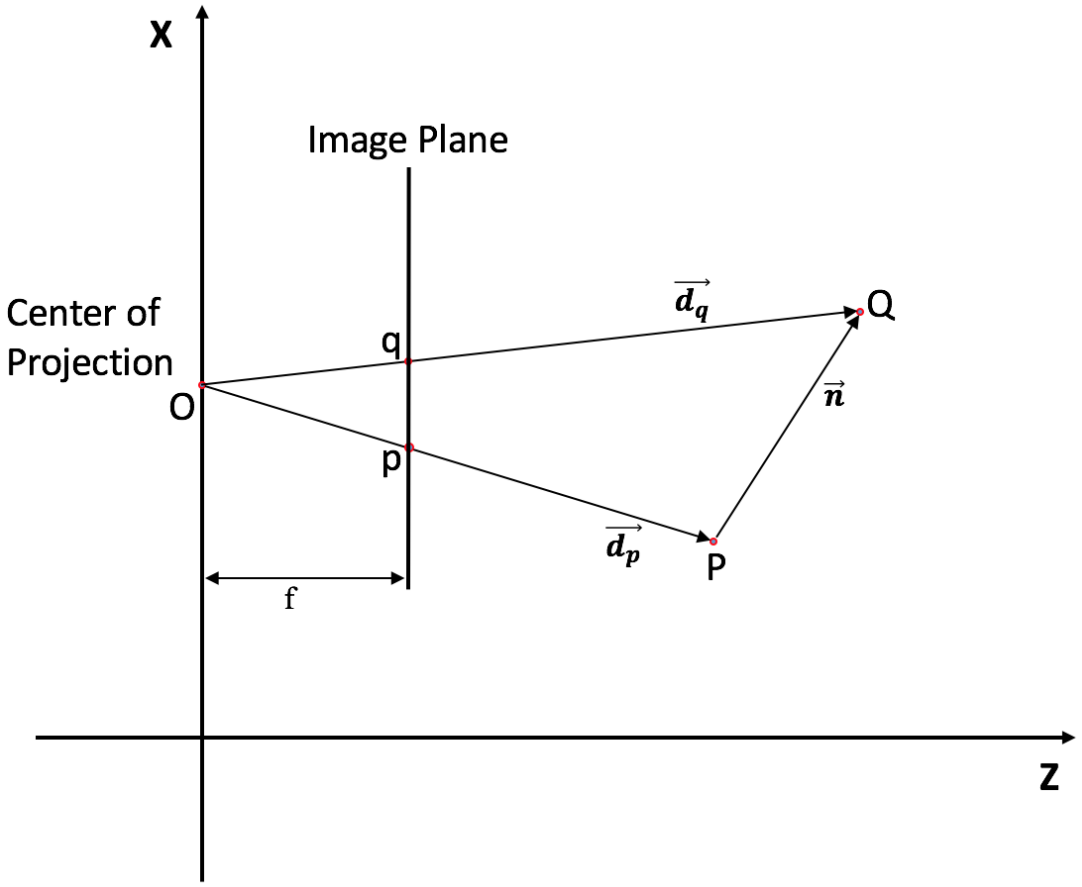


Fig. 1.5. 3D coordinates from symmetry correspondence

point $\mathbf{Q}$, of which $\mathbf{q}$ is the projection, is given by $k_p\hat{\mathbf{d}}_\mathbf{p} + k_n\hat{\mathbf{n}}$, where k_n is a scalar value which determines how much to travel along the direction $\hat{\mathbf{n}}$ from $\mathbf{P}$ to get to the point $\mathbf{Q}$.

The midpoint of the 3D line segment joining $\mathbf{P}$ and $\mathbf{Q}$ will lie on the symmetry plane. Therefore,

$$\begin{aligned} \hat{\mathbf{n}} \cdot \frac{(2k_p\hat{\mathbf{d}}_\mathbf{p} + k_n\hat{\mathbf{n}})}{2} &= d \\ \implies 2k_p(\hat{\mathbf{n}} \cdot \hat{\mathbf{d}}_\mathbf{p}) + k_n - 2d &= 0 \end{aligned} \tag{1.1}$$

The vector $\vec{\boldsymbol{d}}_\mathbf{3} = (-y_q, x_q, 0)$ is perpendicular to $\hat{\mathbf{d}}_\mathbf{q}$. Let $\hat{\mathbf{d}}_\mathbf{3} = \vec{\boldsymbol{d}}_\mathbf{3}/|\vec{\boldsymbol{d}}_\mathbf{3}|$. Since $\vec{\boldsymbol{d}}_\mathbf{3}$ is perpendicular to the vector from the origin to $\mathbf{Q}$, we have:

$$\begin{aligned} \hat{\mathbf{d_3}} \cdot (k_p \hat{\mathbf{d_p}} + k_n \hat{\mathbf{n}}) &= 0 \\ \implies k_p(\hat{\mathbf{d_3}} \cdot \hat{\mathbf{d_p}}) + k_n(\hat{\mathbf{d_3}} \cdot \hat{\mathbf{n}}) &= 0 \end{aligned} \tag{1.2}$$

Equations1.1 and 1.2 represent a system of two linear equations with two (k_p and k_n) unknowns, which can be easily solved.

From 1.2,

$$k_p = \frac{-\hat{\mathbf{d_3}} \cdot \hat{\mathbf{n}}}{\hat{\mathbf{d_3}} \cdot \hat{\mathbf{d_p}}} k_n$$

$$\text{Let } \gamma = \frac{\hat{\mathbf{d_3}} \cdot \hat{\mathbf{n}}}{\hat{\mathbf{d_3}} \cdot \hat{\mathbf{d_p}}} \implies k_p = -\gamma k_n \tag{1.3}$$

Substituting equation 1.3 in 1.1,

$$k_n = \frac{2d}{1 - 2\gamma(\hat{\mathbf{d_p}} \cdot \hat{\mathbf{n}})} \tag{1.4}$$

Once we have k_p and k_n, $\mathbf{P}$ and $\mathbf{Q}$ can be obtained. Note that $\hat{\mathbf{d_3}}$ can be chosen in multiple ways as there are infinitely many directions perpendicular to $\hat{\mathbf{d_q}}$. Choosing a vector lying on a plane parallel to image plane (i.e., with z-coordinate zero, as we did here) is a stable choice.

1.4 Current Work

Given the central role that symmetry plays in visual perception, the central theme of the work described in this dissertation is symmetry. There are three main parts to this dissertation. In the first part a psychophysical experiment is described which aims at studying the priors employed by the visual system in the perception of near-symmetrical shapes. The second part focusses on 3D reconstruction of a shape from a single real image. The broad idea is to employ symmetry and planarity constraints to reconstruct shapes. The third and final part employs translational symmetry to decompose a 3D point cloud into parts. A skeletal representation of 3D shapes is

obtained from this decomposition. A very brief discussion of these parts is presented below, with details to follow in the subsequent chapters.

1.4.1 Modeling near-symmetrical shapes

Li et al. tested the subjects' ability to recover 3D symmetrical shapes from one or two 2D images (monocular and binocular viewing) [1]. Li et al. also formulated a computational model that emulated subjects' performance. In this prior work, the subject and the model adjusted one parameter representing the 3D aspect ratio of a symmetrical interpretation. It is known that the aspect ratio is the only free parameter when a 3D shape is recovered from a single 2D orthographic image by using 3D symmetry constraint. All other characteristics of the 3D shape are uniquely determined by the 3D symmetry constraint. Recall that symmetry constraint can uniquely recover a 3D shape from a perspective image. Li assumed a weaker case, namely the case of an orthographic image, because when the range of the object in depth is small compared to the viewing distance, perspective projection reduces to an orthographic projection. Li et al. showed that the human visual system chooses a 3D symmetrical shape which maximizes a modified compactness expressed as V/S^3, where V and S are the volume and the surface area of the object. The subjects never reported seeing a 3D asymmetrical object when presented with a 3D symmetrical one. So, not allowing the subject to recover asymmetrical objects seemed justified.

But in everyday life many objects are not perfectly symmetrical and they are perceived as not perfectly symmetrical [33]. This is true when we look at a chair with a broken leg or when we look at an animal body which is not mirror symmetrical due to articulation of limbs. This kind of cases can often be handled by recovering a perfectly symmetrical shape and then modifying the recovered object by removing a part or changing its 3D position [20]. But what about objects, whose global symmetry has been distorted? In order to shed light on such cases, we had the subjects look at 3D shapes produced by 3D affine distortion of symmetrical shapes. Will these

shapes be perceived as symmetrical? If not, what is the percept? In order to allow the subject produce what they see, we gave them 3 sliders to adjust: these three sliders represented all possible 3D affine transformations of the 3D shape in front of them that leave a 2D orthographic image unchanged. Experiments were run in both monocular and binocular condition. We show that simple models with a symmetry, compactness and minimal surface priors can explain the monocular percept. In the binocular condition, the only additional term required is the binocular depth order. We ran a control experiment to prove the goodness of the models. The results of this control experiment and some interesting observations obtained by analyzing this data are reported. A couple of metrics to measure asymmetry of a polyhedral shape and to measure the shape dissimilarity between two polyhedral shapes are introduced. A website with all the shapes (including the real shapes, user reconstructed shapes and model shapes) was developed. This website would serve as a useful resource, for interested readers, to directly see all the results from the experiment and to evaluate for themselves (qualitatively) the results obtained.

1.4.2 3D reconstruction of shapes from a real image

The problem of 3D reconstruction of a shape from a single image, is framed as a curve matching problem. Once we assume that the object to be reconstructed is symmetric, all that is left to do is: (i) estimate the plane of symmetry, and (ii) establish the symmetry correspondence between the various parts of the object. The edge map of the image of an object serves as a representation of its 2D shape; establishing symmetry correspondence means identifying pairs of symmetric curves in the edge map. In this work, we assume that the vanishing point, which establishes the symmetry plane up to a scale factor, is known. In addition, we also assume that the focal length and the direction of gravity are known. We extract long smooth curves from the edge map by solving the shortest (least-cost) path problem, where the cost function penalizes large interpolations and large turning angles. We then find the optimal

curve matches that minimize the number of planes required to approximate the final 3D reconstruction. This optimization problem is framed as a binary integer program and solved to obtain the curve matches. Qualitative results of reconstructing images are presented. While some of the images were taken in the lab using camera with known intrinsic parameters, results for some images downloaded from the internet is also provided.

1.4.3 Skeleton extraction from 3D point clouds by decomposing it into parts

Decomposing a point cloud into its components and extracting curve skeletons from point clouds are two related problems. Decomposition of a shape into its components is often obtained as a byproduct of skeleton extraction. In this work, we propose to extract curve skeletons, from unorganized point clouds, by decomposing the object into its parts, identifying part skeletons and then linking these part skeletons together to obtain the complete skeleton. We believe it is the most natural way to extract skeletons in the sense that this would be the way a human would approach the problem. Our parts are generalized cylinders (GCs). Since, the axis of a GC is an integral part of its definition, the parts have natural skeletal representations. We use translational symmetry, the fundamental property of GCs, to extract parts from point clouds. This method can handle a large variety of shapes because of the generality of GCs. We also show how this method can be used to extract skeletons from and identify parts of noisy point clouds. A part based approach also provides a natural and intuitive interface for user interaction. We demonstrate the ease with which mistakes, if any, can be fixed with minimal user interaction with the help of a graphical user interface.

1.5 Contributions

- Through psychophysical experiments we show that the monocular perception of near-symmetrical polyhedral shapes can be well modeled by symmetry, compactness, minimal surface area priors. The same priors in combination with the binocular depth order information can be used to model the binocular perception of such shapes.

- We introduce a metric to measure the shape difference between two polyhedral shapes and a similar metric to measure the amount of asymmetry in such shapes.

- We show that the perception of symmetric shapes are closer to veridical in comparison to near-symmetric shapes. We also show that, for symmetric shapes, there is no systematic depth distortion when viewed binocularly.

- We introduce a method which uses planarity and bilateral symmetry to perform 3D shape reconstruction from a single image. We present a method (shortest path with turning angle penalty) to extract long smooth curves from edge maps. We also introduce a shape metric that can be used to evaluate if two 2D curves are the projections of (approximately planar) symmetric 3D curves.

- A method to decompose 3D point cloud into parts is presented. The parts are represented by GCs and to the best of our knowledge we present the first method which explicitly uses translational symmetry, the most fundamental property of the GCs, to extract them. Since, the parts are GCs the algorithm can be used to extract skeletal representation of 3D shapes represented as point clouds.

- We present an improved version of an existing 3D point cloud registration algorithm by adapting it to the task of GC extraction. We demonstrate the advantages of a part based approach to skeleton extraction and also present a GUI to demonstrate the ease with which users can interact with a part based skeleton extraction algorithm.

2. MODELING PERCEPTION OF NEAR-SYMMETRICAL SHAPES

2.1 3D Shapes, 2D Orthographic Projections and 3D Recovery

Consider a 3D mirror-symmetrical shape represented by N feature points. This shape can be represented by a $3\times N$ matrix P. Assume that the XY plane is the image plane. Recall that in an orthographic projection, translation of an object along the Z coordinate does not change the image, and translation of an object in X and Y directions results in corresponding translations of the image. Assume that the Y coordinates of the pairs of mirror-symmetrical points are identical. This means that the tilt of the symmetry plane is zero (the symmetry plane contains the Y axis). This situation is illustrated in the following animation: `http://shapebook.psych.purdue.edu/2.2/`. If this assumption is not satisfied, the coordinate system can always be rotated around the Z-axis to make the tilt zero. An orthographic image p of the 3D point P can be computed as follows:

$$p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} P \tag{2.1}$$

Equation (2.1) means that an orthographic image is computed by simply omitting the Z coordinate. When an orthographic image of a mirror-symmetrical 3D shape is given, the 3D shape can be recovered up to one unknown parameter. We assume that the symmetry correspondence problem has been solved in the orthographic image. This means that all image points are grouped into pairs of corresponding points. Consider one such pair p_1 and p_2 that are images of 3D mirror-symmetrical points P_1 and P_2. When the tilt of the symmetry plane is zero, the Y coordinates of p_1, p_2, P_1 and P_2 are all the same. So, the Y coordinate is given—it does not have to

be recovered. What needs to be recovered are the depths of the 3D points: Z_1 and Z_2 [34]. This is done as follows:

$$\begin{aligned} Z_1 &= \frac{X_2 - X_1 cos(2\sigma)}{sin(2\sigma)} \\ Z_2 &= \frac{X_1 - X_2 cos(2\sigma)}{sin(2\sigma)} \end{aligned} \tag{2.2}$$

The slant σ of the symmetry plane is the free parameter determining the 3D aspect ratio and 3D orientation of the recovered shape (see `http://shapebook.psych.purdue.edu/2.2/`). All pairs of symmetrical points of the 3D shape are recovered this way. Note that when slant is 45°, the equations in (2.2) become very simple. As pointed out above, the human visual system chooses the unknown slant, σ, which maximizes V/S^3 (see `http://shapebook.psych.purdue.edu/2.3/`, where the shape on the lower right maximizes this ratio). It turns out that good recoveries are also produced when V and S refer to the 3D convex hull of the recovered 3D shape. Convex hull is needed when 3D contours are recovered from a line drawing of a shape (see `http://shapebook.psych.purdue.edu/1.2/` for several examples).

Consider now a pair of 3D symmetrical shapes from the one-parameter family characterized by Equations (2.2). We can write an equation mapping one such 3D shape into the other. This mapping applies to individual points. We no longer have to keep track of symmetry correspondence, because this correspondence is already in the 3D shapes. So, we only have to show how each 3D point of one shape is transformed into a 3D point in the other shape. Let the (X,Y,$Z(1)$) be a point in 3D shape 1, and (X,Y,$Z(2)$) be the transformed point in 3D shape 2. Note that the X and Y coordinates of these two points must be the same by the virtue of the fact that these two 3D shapes produce the same 2D orthographic image. So, only the Z coordinate changes, as follows:

$$\begin{bmatrix} X \\ Y \\ Z(2) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{cos(2\sigma(1)) - cos(2\sigma(2))}{sin(2\sigma(2))} & 0 & \frac{sin(2\sigma(1))}{sin(2\sigma(2))} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z(1) \end{bmatrix} \tag{2.3}$$

where $\sigma(1)$ and $\sigma(2)$ are the slants of the symmetry plane of these two shapes. In our previous shape recovery experiments [1], the subject was adjusting the slant of the symmetry plane of the recovered shape so that the recovered shape matched the reference shape. The error of the match was evaluated by $log_2(|\frac{tan(\sigma(1))}{tan(\sigma(2))}|)$. This error essentially measures the ratio of aspect ratios of the two shapes, and so it represents how different the two shapes are.

In the experiment reported in this work, we made several changes compared to Li et al. [1]. First, the experiment consisted of a number of trials, some of which contained 3D symmetrical reference shapes while others contained 3D asymmetrical reference shapes. Asymmetrical shapes were produced by applying a 3D affine transformation to a symmetrical shape. Let H be a 3D symmetrical shape oriented such that the normal of the symmetry plane is aligned with the x-axis (tilt is zero). An asymmetrical reference shape H' was produced as follows:

$$H' = TH \tag{2.4}$$

where

$$T = \begin{bmatrix} t_{11} & 0 & 0 \\ 0 & 1 & 0 \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$$

Note that t_{11}, t_{31}, t_{32} and t_{33} are positive real numbers. In a trial with a symmetrical reference shape, $H' = H$ (the 3D affine transformation, T, was an identity transformation). The subject was shown a static perspective image of H' with a random orientation R and a rotating 3D shape H'', where:

$$H'' = ARH' \tag{2.5}$$

and

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

The subject was asked to deform H'' by adjusting a_{31}, a_{32} and a_{33} of A so that H'' matched the perception of H'. The form of matrix A implies that X and Y coordinates of RH' and H'' are identical, which means that their 2D orthographic images are identical. More specifically, the family of 3D shapes H'' is the entire 3D affine family corresponding to the 2D orthographic image of RH'. The 3D percept will be called veridical when H'' recovered by the subject is identical with H'.

Similar to Li et al. [1], the subject was shown a 2D perspective image and was adjusting the 3D shape in such a way that a 2D orthographic, rather than perspective image, stayed the same. This approximation was considered reasonable, considering the actual sizes of shapes used, relative to the viewing distance. This experiment could be run without using this approximation. That is, one can form a three-parameter projective family of 3D shapes that all produce the same 2D perspective image. The fact that our models produced good fits suggests that the orthographic approximations used by the model were acceptable. Otherwise, the models would have to have an additional term measuring the difference between the orthographic and perspective image. The fact that such a term was not needed in fitting the model to the subject's data suggests that the orthographic approximation we used was acceptable.

Equation (2.5) is deceptively simple. There are two important characteristics of this equation. First, when H' is symmetrical, the family of shapes represented by H'' contains both symmetrical and asymmetrical shapes. A one-parameter family of 3D symmetrical shapes represented by Equation (2.3) is obviously included, because the transformation in Equation (2.3) is a special case of the transformation in Equation (2.5). However, the family represented by Equation (2.5) contains many asymmetrical shapes as well. This fact is not too surprising. The converse relation is, however, surprising. When H' is an asymmetrical shape produced by an affine transformation, Equation (2.4), of a symmetrical shape H, the family of shapes represented by H'' in Equation (2.5) also contains both symmetrical and asymmetrical shapes (see Appendix A for a proof of this statement). In particular, this family contains a one-

parameter family of 3D symmetrical shapes. So, the image of a 3D symmetrical shape is consistent with both symmetrical and asymmetrical shapes. Similarly, the image of a 3D asymmetrical shape is consistent with both symmetrical and asymmetrical shapes. If the subject has a unique 3D percept, the percept is either symmetrical or asymmetrical, and in either case, the percept must use priors, such as symmetry and compactness. (Some previous authors did discuss the affine family in the context of 3D shape perception [35], but they focused on the ambiguity of 3D shape perception, claiming, incorrectly, that the observers never perceive a unique metric structure. Our subjects did perceive a unique metric structure of 3D shapes and the 3D percept was either veridical or close to veridical. We show in the present work that the unique and veridical percept is produced by the application of a priori constraints to 2D retinal images.) If the percept is sometimes asymmetrical, then it is obvious that a symmetry prior competes with other priors and with binocular data, when the session involves binocular viewing. Which priors are needed in the cost function to account for the subject's 3D percept? What are the relative weights of the priors and the sensory data? Can the same cost function account for monocular and binocular percepts? How veridical is the subject's percept? These questions are at least partially answered in the experiments reported below.

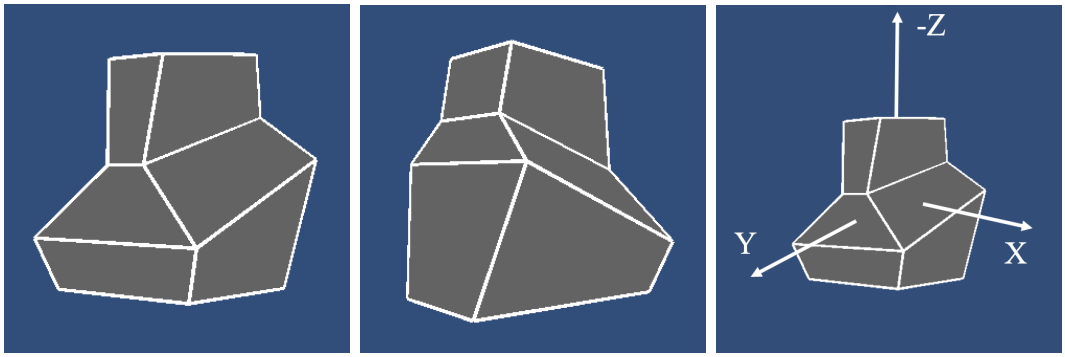
2.2 Psychophysical Experiment on 3D Shape Recovery

2.2.1 Stimuli

The stimuli were 3D symmetrical polyhedra and 3D asymmetrical polyhedra. The asymmetrical stimuli were created by distorting symmetrical ones. Our symmetrical polyhedra were similar to those used by Li et al. [1]. Two views of one symmetrical shape are shown in Figure 2.1.

We used abstract shapes to avoid biases, if any, due to familiarity [24,36]. All our shapes had 16 vertices. They can be thought of as consisting of three hexahedrons, with one of the faces of each hexahedron being coplanar with one face of the other

two hexahedrons. The symmetry plane of the symmetrical shape was perpendicular to this planar surface of the shape, and the x-axis was the normal of the symmetry plane.



(a) (b) (c)

Fig. 2.1. An example of a symmetrical polyhedron. The "top" of the shape is shown in (a), (b) shows the flat (planar) "bottom" of the shape and (c) shows the shape's coordinate system.

As mentioned earlier, three-dimensional symmetry is a prior used by the human visual system. Li et al. [1] also showed the significance of compactness prior in 3D shape perception of symmetric shapes. In this study, we show how these two priors operate when the 3D shape is not symmetrical. We generated 3D shapes with varying degrees of asymmetry and compactness. To measure asymmetry of a shape, we use a normalized average of absolute difference of corresponding angles. Specifically, consider $V_1, V_2, \cdots, V_8$ to be the eight vertices of a symmetrical polyhedron on one side of the symmetry plane. Each of these vertices has a symmetrical counterpart. Let these symmetrical counterparts be represented by $V_1', V_2', \cdots, V_8'$. Each subset of 3 vertices from all 16 define three angles depending on their order. For instance, the three vertices $V1$, $V2'$ and $V3$, define angles (V_1, V_2', V_3), (V_3, V_1, V_2') and (V_2', V_3, V_1). For each of these angles, we have its symmetric counterpart. For the three angles mentioned above, the symmetric counterparts are (V_1', V_2, V_3'), (V_3', V_1', V_2) and (V_2, V_3', V_1'), respectively. The number of unique angles is therefore given by $\binom{16}{3} \times 3$. Let these

unique angles (in radians) be represented by the ordered set $A = \{\alpha_1, \alpha_2, \cdots, \alpha_N\}$. We will use the differences between an angle and its symmetric counterpart as a measure of asymmetry. Let the ordered set $A' = \{\beta_1, \beta_2, \cdots, \beta_N\}$ represent the symmetric counterparts of the angles in set A. The measure of asymmetry can now be defined as:

$$asym = \frac{1}{N\pi} \sum_{i=1}^{N} |\alpha_i - \beta_i| \tag{2.6}$$

Note that there are only $N/2$ unique angular differences that can be considered, as we do not need to count each angle twice. Since we are using the absolute value, the above calculation would give us just that. Note that among all unique angles possible, there will be angles that do not represent angles formed by edges of the polyhedron. The division by $N\pi$ is to normalize the metric to a number between zero and one. For a perfectly symmetrical shape, the measure will be zero. Our asymmetrical shapes are affine transformations of symmetrical ones. So, even though the asymmetrical shapes do not have a plane of symmetry because they are not symmetrical, we can use a unique "symmetry" correspondence which is the same as the correspondence in the symmetrical shape that was used to produce the asymmetrical one. Perceptually, our asymmetrical shapes do look like distorted symmetrical ones. Therefore, one can refer to our asymmetrical shapes as "near-symmetrical".

Compactness is computed from the volume V and surface area S of the shape as V^2/S^3. After this ratio is normalized to the compactness of a sphere, one gets a number between zero and one, and this number is scale invariant. As mentioned before, we need to have shapes with varying degrees of symmetry and compactness as part of our stimuli. To accomplish this, each 3D shape (stimulus) is classified as belonging to one of the groups shown in Table 2.1. In the table, *asym* and *cmp* represent asymmetry and compactness measures of the 3D shape, respectively.

Table 2.1. Asymmetry and compactness characteristics of the 3D shapes

Symmetry / Compactness	$asym = 0$	$0.01 < asym < 0.11$	$asym > 0.11$
$cmp > 0.38$	Group 1	Group 2	Group 3
$0.2 < cmp < 0.38$	Group 4	Group 5	Group 6
$cmp < 0.2$	Group 7	Group 8	Group 9

We generated two shapes for each of the nine groups. The resulting 18 shapes formed a session in the experiment: 6 symmetrical shapes and 12 asymmetrical shapes. There were five such sessions, with a total of 90 shapes. To generate a stimulus, a random symmetrical shape was first generated and then modified to produce a shape belonging to one of the nine groups. Figure 2.1c depicts the orientation of the initial symmetrical shape with its symmetry plane orthogonal to the X-axis of the 3D Cartesian coordinate system. The symmetrical shape was aligned so that the planar bottom of the shape was parallel to the XZ-plane. To modify the compactness, the shape was either stretched or compressed along the X- or Z-axes. This process generated shapes belonging to groups 1, 4, and 7. To generate asymmetrical shapes, the original symmetrical shape was repeatedly sheared according to Equation 2.7 until the desired asymmetry level was obtained.

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.1 & 0.1 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.7}$$

By stretching/compressing the shape to modify compactness and by shearing the shape to modify asymmetry, stimuli belonging to all nine categories were generated.

Each session had two shapes from each of the nine groups listed in Table 2.1, one of which was shown to the subjects with the slant of the symmetry plane equal to 45° and the other with the slant of either 20° or 70°. Note that asymmetrical shapes

do not have any symmetry planes. In those shapes, the slant refers to the YZ-plane, which was the symmetry plane before shearing. The main motivation for using a 45° slant was to have a view that led, in Li et al.'s [1] experiment, to accurate recovery with both monocular and binocular vision. The motivation for using slants of 20° and 70°, which are close to degenerate views, was to produce large recovery errors. Once the slant of the shape's symmetry plane was decided, there was still one degree of freedom in specifying the exact view of the 3D shape, i.e., the shape can still rotate around the normal of the symmetry plane. A random angle of rotation was chosen, subject to two constraints. The first constraint was that at least 10 vertices of the shape needed to be visible, and the second constraint required at least 5 of the 8 vertices forming the "top" of the shape to be visible. These constraints were designed by Li et al. [1] to make sure that the 3D shape can be fully recovered by the human visual system and by the model.

The stimuli were generated using the game engine Unity (known commonly as Unity3D). To display the stimuli, we used Oculus Rift, a virtual reality (VR) head-mounted display. Oculus Rift comes with a head tracker and a joystick. Unity has built-in support for certain VR devices like Oculus Rift. The desired view of the shape can be obtained in Unity by setting the rotation, translation and scaling transformations of the shape appropriately. Unity also allows the use of both orthographic and perspective projections to display the shape. In our experiment, we used perspective projection. The interocular distance can be set in Unity, based on which Unity generates appropriate stereoscopic images for Oculus Rift. We used the head tracker of Oculus to make sure that the head movements of the subject did not change the simulated position of the eyes with respect to the shape being displayed. This means that when the subject moved his head, the shape moved the same way. This is like having the subject's head on a virtual chin rest. In the actual experiment, the subjects did not move their head a lot.

2.2.2 Procedure

In the experiment, a stationary reference 3D shape was shown in front of the subject and, on the right, a test adjustable shape was shown rotating. The adjustment involved the three parameters a_{31}, a_{32} and a_{33} from Equation (2.5). The values of these three parameters were chosen randomly at the beginning of each trial. However, these initial random values were confined to the range (–3, 3) for a_{31} and a_{32} and to the range (0.1, 3) for a_{33}. This was done to avoid highly distorted shapes. The subject's task was to look at the stationary reference shape and adjust the shape of the rotating test shape on the right side of the screen to match the perceived 3D shape of the reference shape. The continuous rotation of the test shape was around Y-axis. In order to make sure that the subjects compared 3D shapes and not 2D images, the test 3D shape was first rotated by 45° around the X-axis before applying the continuous rotation around the Y-axis. This ensured that none of the views of the rotating test shape matched the projection of the reference shape. The adjustment of the three parameters was done using the joystick. The rotating test shape matched, geometrically, the stationary reference 3D shape for the following values: $a_{31} = 0, a_{32} = 0$ and $a_{33} = 1$. The subject was not shown the values of the three parameters during the trial.

In the binocular viewing condition, the reference shape was rendered at a distance of one meter from the subject's cyclopean eye. In the "monocular" viewing condition, the subject still viewed with both eyes, but the shape was rendered at a distance of one kilometer. Note that the object's size was proportionally scaled up by a factor of 1000 so that the retinal size of the shape was similar in both viewing conditions. The average angular distance from the centroid of the shape to its vertex was 10.32°. The large distance in the "monocular" condition meant that the binocular disparity information available to the subject was negligible. Recall that binocular disparity is inversely proportional to the square of the viewing distance [37].

Each session consisted of 18 trials. Each subject ran a total of 5 monocular and 5 binocular sessions for a total of 90 trials (90 shapes) in monocular viewing and

90 trials in binocular viewing. The same 90 shapes were used in monocular and binocular viewing and all subjects were tested with the same shapes. However, the order in which individual subjects were tested with the five sessions and the order of monocular vs. binocular sessions were randomized. Three subjects were tested, including two of the authors (VJ and ZP). The third subject was naïve about the underlying theory. Each session of 18 trials took around 60–90 min to complete. The subjects were allowed to take breaks at any point of time. Before the actual experiment started, the subjects were given adequate practice sessions until they felt comfortable with the task. In the beginning of each trial, the subject was also asked whether he perceived the shape to be symmetrical or not. Then, the subject adjusted the rotating shape by using three controls of the joystick. Some shapes looked very close to perfectly symmetrical, and so, the subject had to decide whether he called the shape symmetrical or not. In other words, these binary responses might have confounded the percept with response bias. Note, however, that our main analyses that included building computational models ignored these binary responses. These responses were used, however, in some graphs that showed the relation between the shape recovered by the subject and the reference shape, or the shape recovered by the model.

2.3 Model

The data collected from the experiment were used to build models. The emphasis in this work is on the quality of the fit of the model to human results. This is why we present separate models for monocular and binocular conditions. Within each viewing condition (monocular or binocular), separate models are also considered for symmetric and asymmetric shapes. In the Discussion section, we comment on the possibility of simplifying this approach so that fewer models are used.

Li et al. (2009) showed that, for symmetric shapes viewed monocularly, maximum compactness and minimum surface priors could be used to recover the perceived 3D

shape from the 2D image. Specifically, the 3D shape that maximizes the ratio V/S^3, where V is the volume and S is the surface area of the polyhedron, is very close to what human observers perceive. Therefore, for symmetrical shapes in the monocular condition, we used the same model as the one presented in Li et al. (2009). The ratio V/S^3 will hereafter be referred to as modified compactness. This model has no free parameters.

For asymmetrical shapes in monocular condition, the model consists of a weighted sum of asymmetry and modified compactness. In our experiment, the subject could recover an asymmetrical shape, and therefore, the asymmetry term representing the symmetry prior is explicitly included in the cost function, along with the modified compactness prior used in Li et al. (2009). The cost associated with a shape H is given by:

$$C(H) = \boldsymbol{W_S} \times Asym(H) - \frac{MCmp(H)}{MaxCmp(H)} \tag{2.8}$$

Here, $Asym(H)$ is the asymmetry measure of shape H, $MCmp(H)$ is the modified compactness of shape H, W_S is the weight of the asymmetry term, and $MaxCmp(H)$ is the maximum value of modified compactness that can be achieved by a shape, belonging to the family of shapes, that can be generated from shape H, by applying the 3D affine transformation in Equation (2.5). The weight decides the importance of the asymmetry relative to modified compactness of the shape. According to the model, the shape which minimizes this cost is the perceived shape. To determine the weight of the asymmetry term, which is the only free parameter, the subject's data were used. Let the shapes recovered by the subject and the model be referred to as subject shape and model shape, respectively. The idea is to find the weight that minimizes the average shape difference between the subject shape and the model shape. In order to do this, a metric to perform shape comparison is required. The asymmetry metric can be modified to obtain a shape difference metric. To compare two shapes, H_1 and H_2, Equation (2.6) can be used. The set $A = \{\alpha_1, \alpha_2, \cdots, \alpha_N\}$ now represents all the possible angles of shape H_1 and the set $A' = \{\beta_1, \beta_2, \cdots, \beta_N\}$

represents the corresponding angles of shape H_2. Using this shape difference metric, the best weight found for each subject in monocular viewing of asymmetrical shapes is listed in Table 2.2. An exhaustive search was used to determine the weights.

Table 2.2. Weights for the asymmetry term in monocular condition

Subject	W_S
EP	5.1
VJ	3
ZP	4

Note that our shape dissimilarity metric, based on comparing angles, is consistent with the conventional definition of shape, because it is invariant under rigid motion and size scaling. However, we are not claiming that the human visual system uses our formula, but examination of our results strongly suggests that whatever the human visual system uses is likely to be correlated with our measure.

In the binocular condition, for both symmetrical and asymmetrical shapes, the cost function is a weighted sum of asymmetry, binocular depth order and modified compactness. Li et al. [1] established the importance of the depth order cue in 3D shape perception. The asymmetry and modified compactness terms represent the monocular priors, and the binocular depth order term represents the likelihood (or the data term). The depth order score for a shape H is computed as follows. For each pair of visible vertices of shape H, a score is assigned using the function below:

$$Scr(v_i, v_j) = \frac{1}{1 + e^{(k_{i,j} \times 0.5 \times \delta_{i,j})}} \tag{2.9}$$

where v_i and v_j are two visible vertices of the shape H, $\delta_{i,j}$ is the absolute value of binocular disparity (in minutes of arc) between the corresponding vertices in the reference 3D shape, $k_{i,j} = -1$ if the two vertices have the same depth order as the corresponding vertices in the reference 3D shape, and $k_{i,j} = 1$ if the depth order of the vertices v_i and v_j is different from that of the reference shape. Once such a score

is assigned to each visible pair of vertices, the depth order score for the shape H is computed as:

$$DpOrd(H) = \sqrt[|\Omega|]{\prod_{(v_i,v_j)\in\Omega} Scr(v_i, v_j)} \tag{2.10}$$

where Ω represents the set of all unique pairs of visible vertices of shape H, and $|\Omega|$ represents the number of elements in the set Ω. Equation (2.9) uses a logistic function to represent visual noise, whereas Li et al. [1] used a cumulative Gaussian distribution function.

The cost function associated with a shape H in the binocular condition is now given by:

$$C(H) = \boldsymbol{W_S} \times Asym(H) - \boldsymbol{W_D} \times DpOrd(H) - \frac{MCmp(H)}{MaxCmp(H)} \tag{2.11}$$

where $\boldsymbol{W_S}$ and $\boldsymbol{W_D}$ are the weights of the asymmetry term and depth order terms, respectively. The same cost function is used to model both the symmetrical and asymmetrical cases but with different weights. Data from the experiment were used to estimate the weights. The optimal weights found by exhaustive search for each subject are listed in Table 2.3. The shape which minimizes this cost function represents the model's prediction of the perceived 3D shape. Note that the weights for symmetrical shapes in binocular viewing are similar across the three subjects. However, for asymmetrical shapes, there are quite large differences. Li et al. [1] also found some individual differences in binocular viewing: one subject, whose stereoacuity threshold was very high, had binocular performance similar to monocular performance. However, Li et al. only tested symmetrical shapes. As Table 2.3 shows, our three subjects were all quite similar with symmetrical shapes. It is the asymmetrical shapes that revealed larger individual differences. Specifically, EP's and ZP's asymmetry term has weight which is an order of magnitude greater than the weight of the depth order cue, while VJ's weights for these two terms are similar. Furthermore, VJ's depth order term has weight greater than the weight of compactness, while ZP's depth order term

has weight smaller than the weight of compactness. It has to be pointed out that one cannot draw any conclusions about the relative importance of the three terms in a cost function by just comparing weights, simply because the three terms are computed using different units. However, one can interpret the relative contribution of the three terms across subjects, as we did just above.

Table 2.3. Weights for shapes in binocular condition

Subject	W_S (Symmetric Case)	W_D (Symmetric Case)	W_S (Asymmetric Case)	W_D (Asymmetric Case)
EP	9.7	8.6	10.0	1.0
VJ	7.6	11.5	4.0	5.8
ZP	9.0	14.0	5.0	0.3

To test the accuracy of the model, we started with a control psychophysical experiment, which is described next.

2.4 Control Experiment

In the control experiment, the same set of 3D shapes, as used in the experiment described above, were used. The subject was shown a stationary 3D shape in the center and two rotating 3D shapes were shown on the right. One of the rotating shapes was the shape the subject recovered (subject shape) in the main experiment. The other 3D shape was the shape recovered by the model (model shape). The two rotating shapes were shown side by side and the placement of these shapes (left vs. right) was random from trial to trial. The subject was asked to choose the 3D shape from the two rotating 3D shapes which best corresponded to his percept of the stationary 3D shape. The models described above can be considered good models if the subject has chance performance (50/50). Two of the subjects, EP and VJ, participated in the control experiment.

2.5 Results

The results in Table 2.4 show that the models described above do a good job in predicting the shape perceived by the subjects because the subject's performance was not far from chance. This also means that the shape metric which was used to build the models is a good measure of the shape dissimilarity between shapes. The fact that EP chose the model shape in favor of his own shape almost 70% of trials in monocular viewing could be related to the operation of visual and motor noise when the subject was recovering shapes in the main experiment. The easiest way to see evidence of this is to check the data points in Figure 2.2 representing symmetrical shapes perceived as symmetrical (red diamonds). The asymmetry of the subject shapes is not zero as it should be, since they were judged by the subjects in the beginning of the trial as symmetrical. Our models, on the other hand, did not have any noise except for the finite, but small, steps in the three adjusted parameters when the cost function was minimized. So, if the models are correct, they may recover 3D shapes that match the percept better than the shapes recovered by the subject, himself.

An important observation here is that, in the monocular condition, a simple combination of asymmetry and modified compactness, which are global properties of the shape, can effectively model the abstract shapes that we used. Without using any other priors, the model was able to account for the subject's percept. In the binocular condition, the only additional information required was the binocular depth order derived from the binocular disparity. Even for shapes which are not perfectly symmetrical (near-symmetrical shapes), combining the binocular depth order information with symmetry and modified compactness priors leads to a good model of shape perception.

Table 2.4. Results from the control experiment

Subject	Condition	Number of times (out of 90) the model shape was chosen
EP	Monocular	62(68.8%)
EP	Binocular	48 (53.3%)
VJ	Monocular	40 (44.4%)
VJ	Binocular	46 (51.1%)

We already know that the models are good because they produce shapes that are similar to what the subjects produce. This was shown in the control experiment described just above. This can also be directly seen by examining the 3D shapes on the website (`https://lorenz.ecn.purdue.edu/~vthottat/shapeexp/chooseshp.php`). Note that the same set of 90 shapes was used, in monocular and binocular viewing, for all three subjects. This allows more direct comparison of one subject to another. The website shows the view that was presented in the psychophysical experiment (in binocular viewing this was one of the two images projected to the left and right eyes). Next to this view we show the actual reference 3D shape presented to the subject, the 3D shape recovered by the subject and the 3D shape recovered by the model. The website contains 540 trials like this (3 subjects $\times$ 90 shapes $\times$ 2 viewing conditions: monocular and binocular). We do not expect the reader to look through all 540 trials. However, we felt that the reader should have an opportunity to view as many as they wish, because no summary statistics will do justice to what was actually observed. This would also provide the reader with an opportunity to judge the goodness of the models and that of the metrics we used.

We began the quantitative evaluation of the subject's and model's recovery with symmetrical shapes. Already, Li et al. [1] showed that this recovery is extremely accurate in binocular vision. However, their subjects dealt with a simpler task, in which all shapes were symmetrical and the only characteristic that was adjusted was the aspect ratio. In our experiment, we used both symmetrical and asymmetrical

shapes and the subject adjusted three parameters which allowed them to recover symmetrical or asymmetrical shapes.

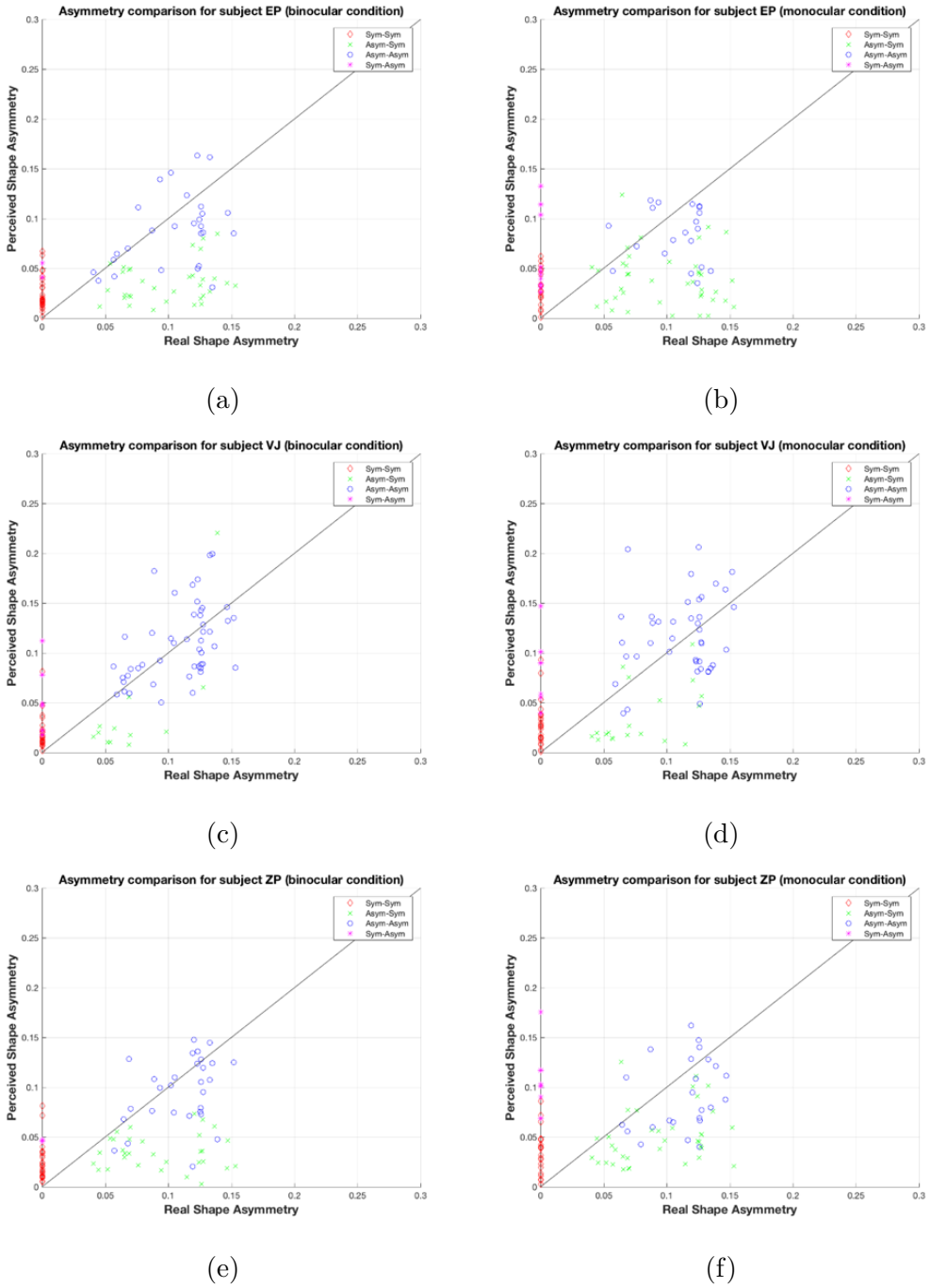
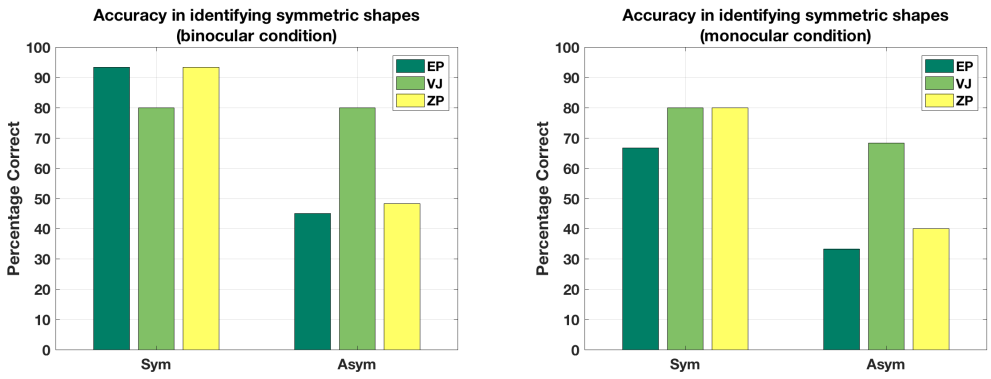


Fig. 2.2. Perceived vs. real asymmetry of shapes for (a) subject EP (binocular), (b) subject EP (monocular), (c) subject VJ (binocular), (d) subject VJ (monocular), (e) subject ZP (binocular) and (f) subject ZP (monocular)

Figure 2.3 shows the accuracy of identifying symmetrical and asymmetrical shapes correctly in monocular and binocular viewing by the three subjects. This graph is based on the binary responses that the subject produced before performing 3D recovery. Note that we are not computing d' for this discrimination task, although this in principle could be done. The number of trials is small (30 symmetrical and 60 asymmetrical shapes), so any estimate of d' would be quite unreliable. The proportion correct on symmetrical and asymmetrical trials, which is what we plotted on Figure 2.3, is likely affected by response bias. As a result, we will not be drawing any strong conclusions based on these proportions. It can be seen in Figure 2.3 that symmetrical shapes are almost always (80–95% of the time) identified as symmetrical in binocular viewing. They are also frequently (65–80% of the time) correctly identified as symmetrical in monocular viewing. The differences among the three subjects were pretty small. With asymmetrical shapes, VJ was clearly more correct (by a factor of 2) than the other two subjects. It seems that VJ put more emphasis on the depth order information, as evidenced by the weights in the models. EP and ZP's binocular percepts, on the other hand, were affected more by compactness and symmetry priors.

These two priors pushed the binocular percept away from the true, asymmetrical reference shape and towards symmetrical and compact shapes. When recovering the shapes, VJ spent twice as much time as EP and ZP. This led to higher precision in recovering the 3D shapes. This will show up in smaller shape dissimilarities between the shapes recovered by VJ and the reference shapes, as compared to these dissimilarities for subjects EP and ZP.



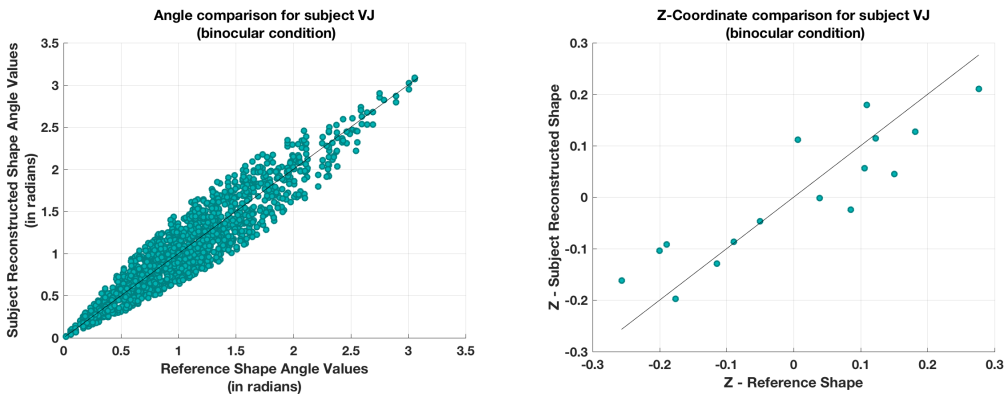
(a) (b)

Fig. 2.3. Accuracy in identifying symmetrical and asymmetrical shapes in (a) binocular and (b) monocular condition. Recall that there were 30 symmetrical and 60 asymmetrical shapes.

Next, look at Figure 2.2, which shows scatterplots of perceived shape asymmetry (i.e., asymmetry of the shape recovered by the subject) vs. reference shape asymmetry for four types of trials. The trials have been grouped into four groups based on the binary responses, symmetrical vs. asymmetrical, that were collected in the beginning of each trial. The diagonal line represents the points where perceived shape and reference shape asymmetries are equal. The label 'Sym-Sym' indicates symmetrical shapes perceived as symmetrical, 'Asym-Sym' indicates asymmetrical shapes perceived as symmetrical, and so on. Red diamonds indicate symmetrical shapes that were classified as symmetrical. It can be seen that the perceived asymmetry of these shapes is not zero. This represents the limited precision of the visual system, as well as limited precision of the motor system. The adjustment task was not easy because the interaction of the three parameters was not intuitively obvious. So, reducing asymmetry of the shape required coordinated change of all three parameters. VJ was the most patient from the three subjects—he also spent most time during the task. Still, most of the symmetrical shapes that were classified as symmetrical resulted in a subject shape with asymmetry not larger than 0.05. Note that a value of 1 for shape asymmetry corresponds to an average difference of 180° (π radians) be-

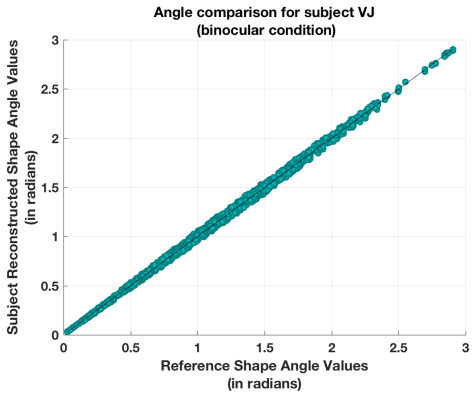
tween corresponding angles in the left and right half of the shape (see Equation (2.6)). So, a value of 0.05 corresponds, on average, to a 9° difference between corresponding angles. Examination of the 3D shapes on the website shows that the symmetrical shapes were almost always recovered as shapes that were not far from symmetrical.

Next, look at Figure 2.4, left column, which shows the scatterplots illustrating the relation between the angles in the recovered shape and the angles in the reference shape for three symmetrical shapes recovered by VJ in binocular viewing. The first row in Figure 2.4 shows the plots for the shape in (row = 2, column = 5) in set 2 for subject VJ in binocular viewing on the website. Note that both row and column indices start at one. The second row in Figure 2.4 represents the shape (row = 2, column = 6) in set 5, and the third row in Figure 2.4 represents the shape (row = 3, column = 3) in set 3 on the website. These three trials illustrate the range of correlations that we observed. The scatterplot on top shows one of the weakest correlations we observed. Each scatterplot has $16 \times 15 \times 14/2$ data points because we considered all unique angles formed by all triplets of vertices. All scatterplots for symmetrical shapes were within the range shown in Figure 2.4. The correlation between perceived and actual angles is strong, and all data points are clustered around the diagonal. Figure 2.4, right column, also shows the analogous scatterplots illustrating the relation between the depths of points in the recovered shape and depths of points in the reference shape for the same three symmetrical shapes viewed binocularly by VJ. These scatterplots have only 16 data points, which is the number of vertices in our 3D shapes. The Z coordinates of the recovered and the reference shapes were taken when the shapes had the same 3D orientation relative to the viewer. More precisely, both 3D shapes, reference and subject, produced the same orthographic image. The fact that all data points are clustered around the diagonal means that there was no systematic distortion of binocularly viewed shapes along the depth direction. This makes sense on rational grounds. Stretching or compressing a 3D symmetrical shape along the depth direction, when the shape is not viewed from perfectly degenerate direction, would have destroyed the symmetry of the 3D shape. Since symmetrical

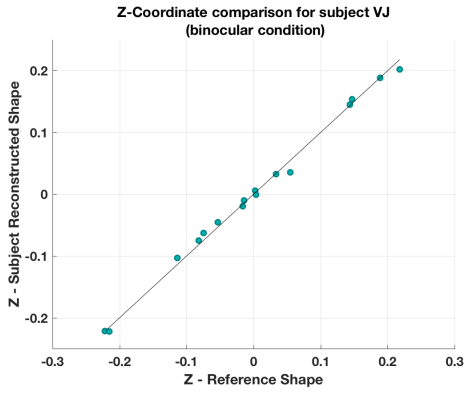


(a)

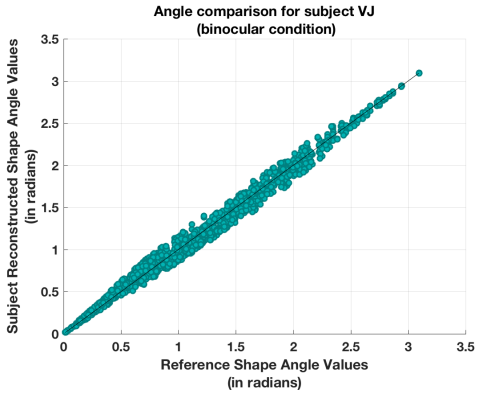
(b)



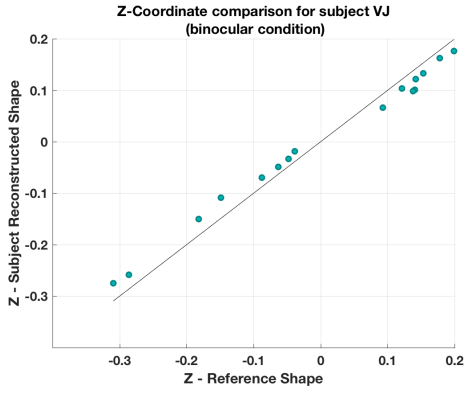
(c)



(d)



(e)



(f)

Fig. 2.4. Perceived vs. real angles (first column, in radians) and depth (second column) for three different symmetric shapes. Each row represents the corresponding plots for a particular shape. (a) and (b) represent the plots for the shape in (row = 2, column = 5) in set 2, (c) and (d) represent the plots for the shape in (row = 2, column = 6) in set 5 and (e) and (f) represent the plots for the shape in (row = 3, column = 3) in set 3 on the website

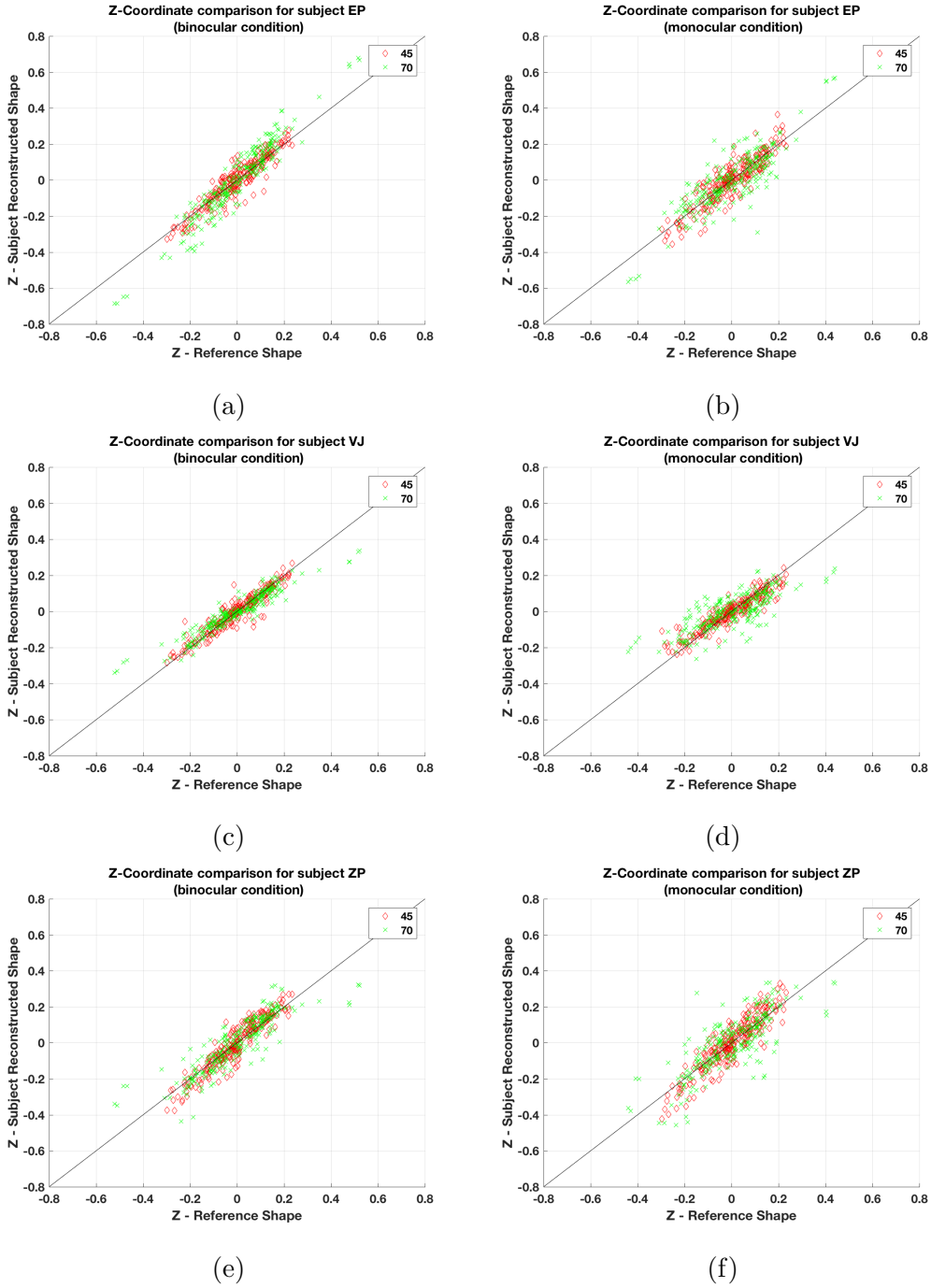
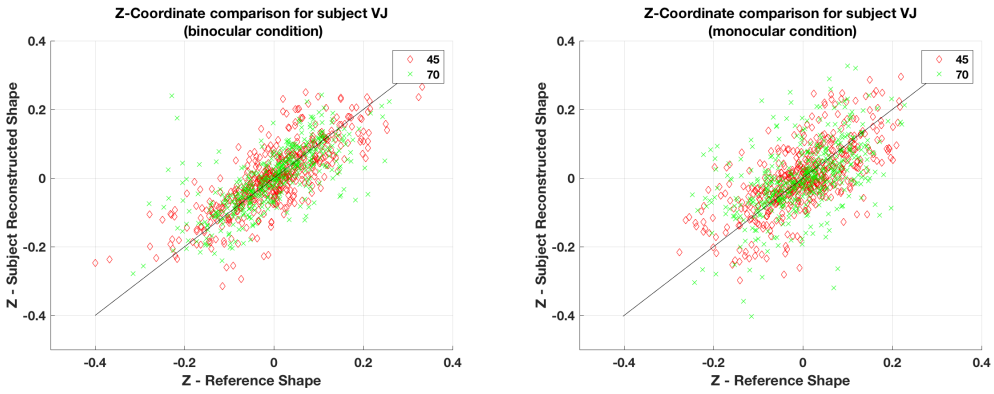


Fig. 2.5. Subject shape vs. reference shape depth plots for symmetrical shapes for (a) subject EP (binocular), (b) subject EP (monocular), (c) subject VJ (binocular), (d) subject VJ (monocular), (e) subject ZP (binocular) and (f) subject ZP (monocular). The numbers 45 and 70 indicate viewing directions. The green x marks include both 20° and 70° viewing directions.

shapes are almost always perceived as symmetrical, there is not much space for depth distortion.

Figure 2.5 shows scatterplots of the depths of points for all symmetrical shapes for the three subjects for monocular and binocular viewing. Note that in the monocular condition, to ensure that there was no binocular disparity information available, the shapes were displayed at a depth of 1 km after scaling up the shapes by a factor of 1000. In Figures 2.5 and 2.6, the Z-coordinate values for the monocular condition is scaled down by a factor of 1000 for better comparison with the binocular condition. It can be seen that all data points in Figure 2.5, except a few, are close to the diagonal line. This means that there are no systematic distortions of depth: no systematic under- or overestimation. Figure 2.6 shows analogous graphs for all asymmetrical shapes for subject VJ in monocular and binocular conditions (graphs not shown here, and illustrating the other two subjects are similar). The scatterplots for asymmetrical shapes are substantially noisier in comparison to the corresponding plots for the symmetrical shapes. Notice that the distortion (noise) does not seem to be systematic. The most natural explanation for greater veridicality with symmetrical shapes, as compared to asymmetrical, is the influence of the priors, especially the symmetry prior. The strong effect of a symmetry prior implies that the percept of symmetrical shapes should be more veridical than the percept of asymmetrical ones. This indeed is the case.



(a) (b)

Fig. 2.6. Perceived vs. real depth plots for asymmetric shapes for subject VJ in (a) binocular and (b) monocular condition.

Figure 2.7 shows shape dissimilarity for all subjects, with the bars grouped into different cases. The label 'Sym' represents the average shape dissimilarity for symmetrical shapes. Similarly, 'Asym' refers to the average shape dissimilarity for asymmetrical shapes. The label 'All' indicates the average shape difference for all 90 shapes in that condition.

Note that the binocular percept is closer to veridical than the monocular percept is, as can be seen from the subject vs. reference shape dissimilarity values (yellow bar in the group labeled 'All'). Similarly, perception of symmetrical shapes is closer to veridical in comparison to asymmetrical shapes. This can also be verified qualitatively by examining the shapes on the website. Finally, model shapes are closer to the subject shapes than the reference shapes are. This means that, despite small errors between the subject shapes and the reference shapes, the model is a good predictor of what the subjects see.

2.6 Discussion

It has to be pointed out that the 3D shape recovery is quite robust to perturbation of the weights around their optimal values. A plot of the shape difference between the

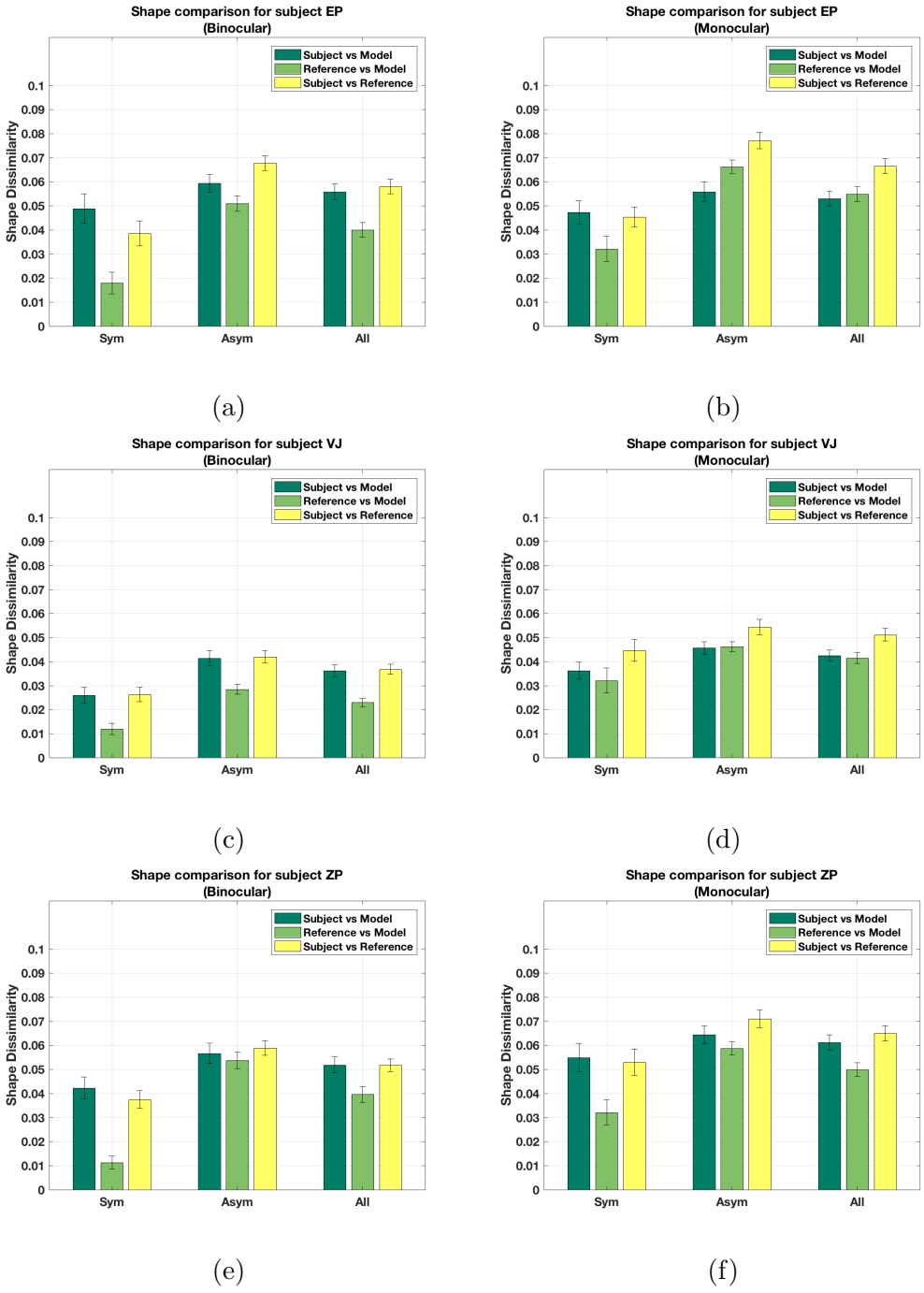
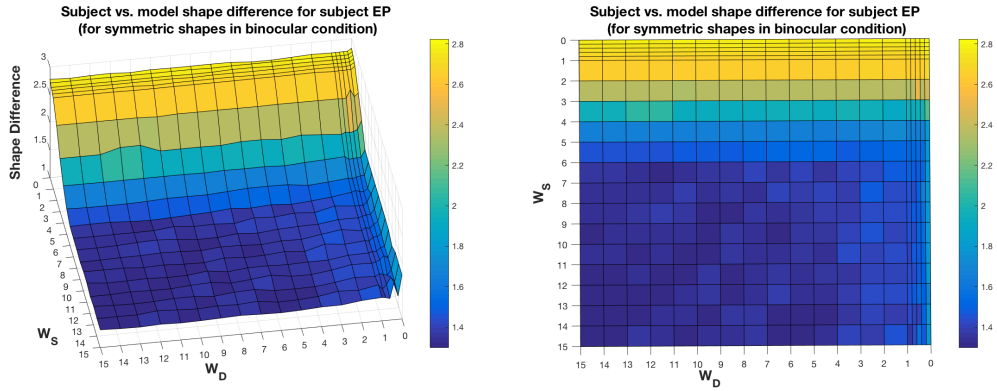


Fig. 2.7. Shape dissimilarity for (a) subject EP (binocular), (b) subject EP (monocular), (c) subject VJ (binocular), (d) subject VJ (monocular), (e) subject ZP (binocular) and (f) subject ZP (monocular).

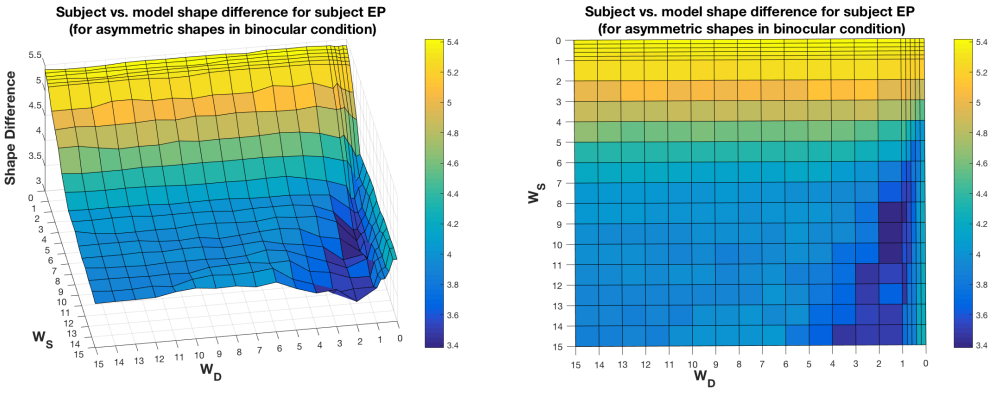
shape recovered by the subject and the shape recovered by the model, as a function of the weights of the model, shows that there are other weight values which could be chosen, without increasing the shape difference substantially. As shown in Figure 2.8, there are flat regions around the optimal weight values for the symmetrical shapes in the case of subject EP in the binocular condition. This is true for the other subjects, too. There are obviously some weight values, like choosing $\boldsymbol{W_S} < 5$, which would lead to large errors (shape differences), but there is a range of other weight values with errors close to the optimal error. However, for other cases, like for asymmetrical shapes in the binocular condition for subject EP (Figure 2.9), there seems to be a narrow range of weight values that minimize the error. This is also true for subject ZP but not for subject VJ, i.e., for subject VJ, there are flat regions around the optimal weights, just like in the case of symmetrical shapes. In short, in some cases, there is some flexibility in choosing the weights for a good model. The issue of computational stability of our models in the neighborhood of optimal values of the parameters will be studied in our future work.



(a) (b)

Fig. 2.8. Subject vs. model shape difference, as function of the model weights, for symmetrical shapes, for the subject EP, in the binocular condition. (a) and (b) represent two views of the same shape difference plot.

Next, it has to be pointed out that, even though different models are used for symmetric objects in monocular and binocular conditions, the relative importance of



(a) (b)

Fig. 2.9. Two views ((a) and (b)) of the subject vs. model shape difference, as function of the model weights, for asymmetrical shapes, for the subject EP, in the binocular condition.

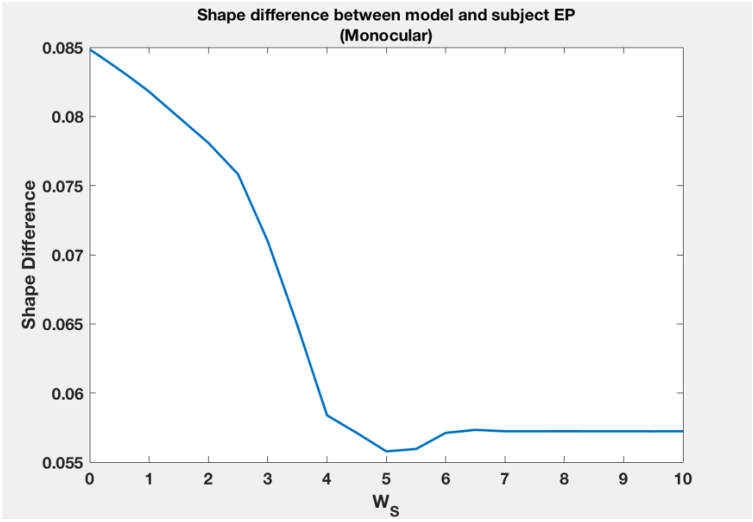


Fig. 2.10. Shape difference for asymmetric shapes in the monocular condition as a function of the weight of the symmetry term.

symmetry in comparison to modified compactness can be the same in both viewing conditions. That is, if we minimize Equation (2.8), with the weight $\boldsymbol{W_S}$ equal to that of the symmetric case in the binocular condition, the model will produce a shape very close to the one obtained by using the model presented in Li et al. [32]. For instance, the average shape difference between the shape obtained with the model presented in Li et al. [32] and the shape obtained by minimizing Equation (2.8) with $\boldsymbol{W_S} = 7.6$ (the optimal weight for subject VJ) is 1.9×10^{-5}, and the maximum shape difference between the corresponding shapes is 2.3×10^{-4} . A value of 0.01 for the shape difference metric corresponds to an average difference of 1.8° between corresponding angles of the two shapes being compared. Therefore, a shape difference metric value of 1.9×10^{-5} implies an average difference of 3.4×10^{-3} degrees between corresponding angles of the two shapes. Furthermore, a shape difference metric value of 2.3×10^{-4} corresponds to an average difference of 4.1×10^{-2} degrees between corresponding angles of the two shapes. These are extremely small values, and, therefore, the shapes generated by the two models are essentially the same.

For asymmetrical shapes, the weight of the asymmetry term relative to modified compactness is similar in monocular and binocular conditions for subjects VJ and ZP. For asymmetrical shapes, if the binocular model weights for VJ are used for the monocular model of VJ, the average subject shape vs. model shape difference would go up by 0.0094. Doing the same for subject ZP would lead to an increase in the average shape difference of 0.0022 between the subject and the model. These are not large changes. For asymmetrical shapes for subject EP though, the weight of the asymmetry term seems to be very different across the two conditions. However, as shown in Figure 2.10, assigning $\boldsymbol{W_S} = 10$, for asymmetrical shapes in the monocular condition for subject EP, will not lead to any substantial change in shape difference with subject shapes. Therefore, for all three subjects, the weight of asymmetry relative to modified compactness is similar across conditions.

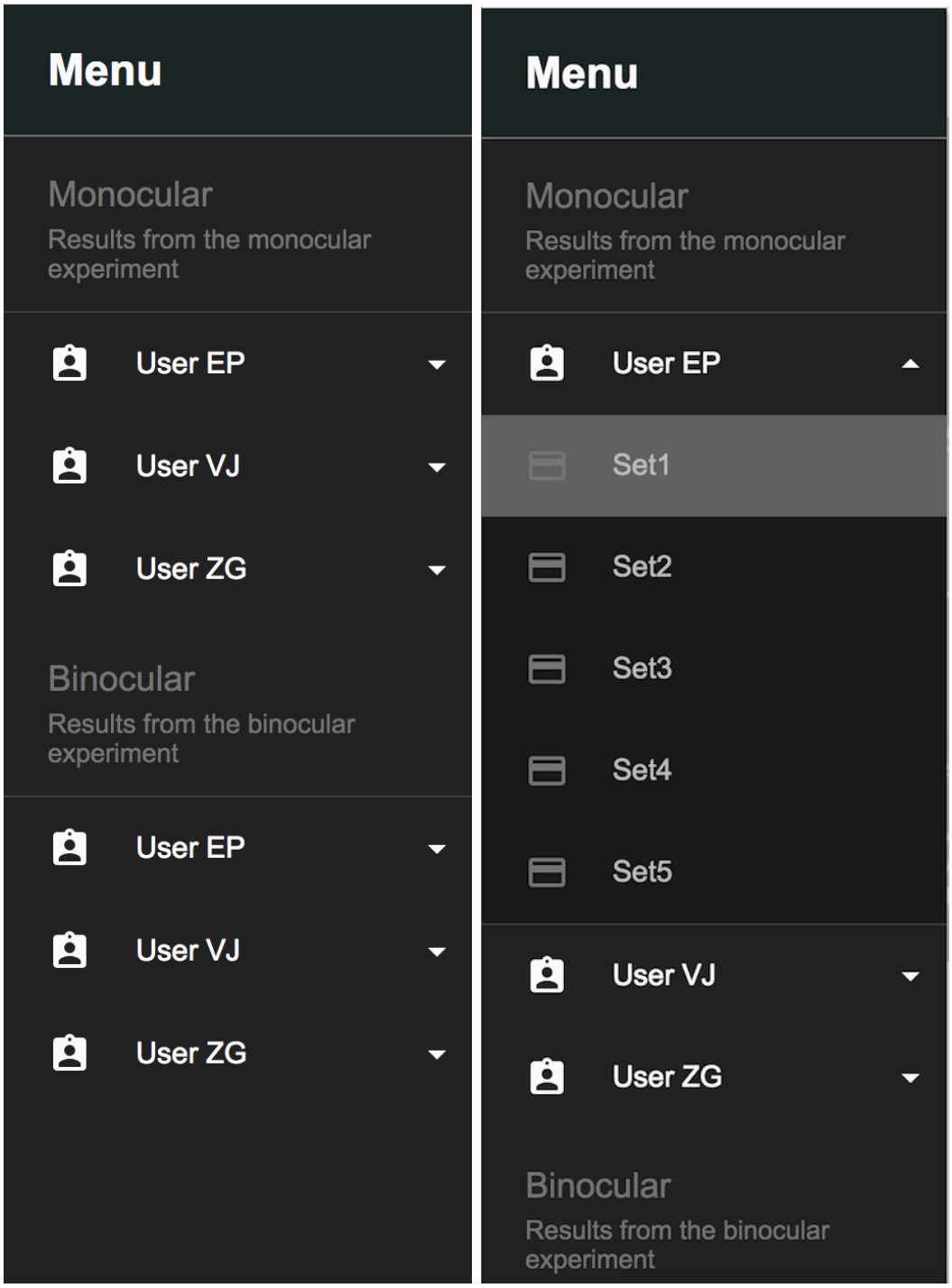
To summarize the above analysis, we do not need separate models for monocular and binocular viewing. However, we need two different binocular models per subject:

one for symmetrical shapes and the other for asymmetrical shapes. The monocular model is obtained from either of them by simply dropping the binocular depth order term. In natural viewing, this "dropping" happens naturally when the effectiveness of stereoacuity drops with increasing viewing distance and decreasing object size. Now, how realistic is it to have two different binocular models (cost functions) in the visual system: one for symmetrical and the other for asymmetrical shapes? How does the visual system distinguish between symmetrical and asymmetrical shapes in the first place? If the 3D shape is asymmetrical, the binocular depth order cue can inform the visual system about the asymmetry. For example, once the 3D symmetry correspondence is established in a single 2D retinal image, assuming that the 3D shape is symmetrical (which is always possible—see [38]), stereoacuity can reject this 3D hypothesis. Asymmetrical perception could also result from other biases (priors). For instance, the visual system might prefer an asymmetrical shape with planar faces instead of a symmetrical shape that violates planarity of the faces. Next, given that the importance (weight) of compactness relative to depth order goes up with asymmetrical shapes, compactness could be another factor leading to an asymmetrical percept [33]. We have not done any simulations yet to examine this issue, but it seems reasonable to assume that planarity, compactness and binocular depth order could all be involved when the visual system decides whether to use the cost function for symmetrical shape or asymmetrical shape.

2.7 Website

The idea behind the website is to provide interested readers an opportunity to view the user and model reconstructed shapes. The menu, shown in Figure 2.11, is used to navigate the website. Figure 2.11 (a) shows the first level, which allows user to choose the condition and the user. Figure 2.11 (b) shows the second level, which lets the user select one of the five data sets used in the experiment. Note, the experiment used five sets of eighteen objects. Each set represents a session. Once a

set belonging to one of the two conditions for a particular user is chosen, the user is shown all the eighteen objects in that set. Figure 2.12 shows such a list of shapes. The user can now hover over a shape using the mouse pointer to get information regarding the asymmetry and compactness of that shape. On clicking a shape, the user navigates to the page where the user can see the 2D view shown to the subjects and rotating 3D shapes corresponding to the real shape, the user reconstructed shape and the model reconstructed shape. Figure 2.13 shows such a page. The asymmetry, compactness and modified compactness values are displayed under each shape. The page also shows the shape difference metric values for all pairs of shapes. This page allows the readers to do a qualitative analysis of the effectiveness of the model and the asymmetry and shape comparison metrics that we employed.



(a) (b)

Fig. 2.11. The menu for navigating the website

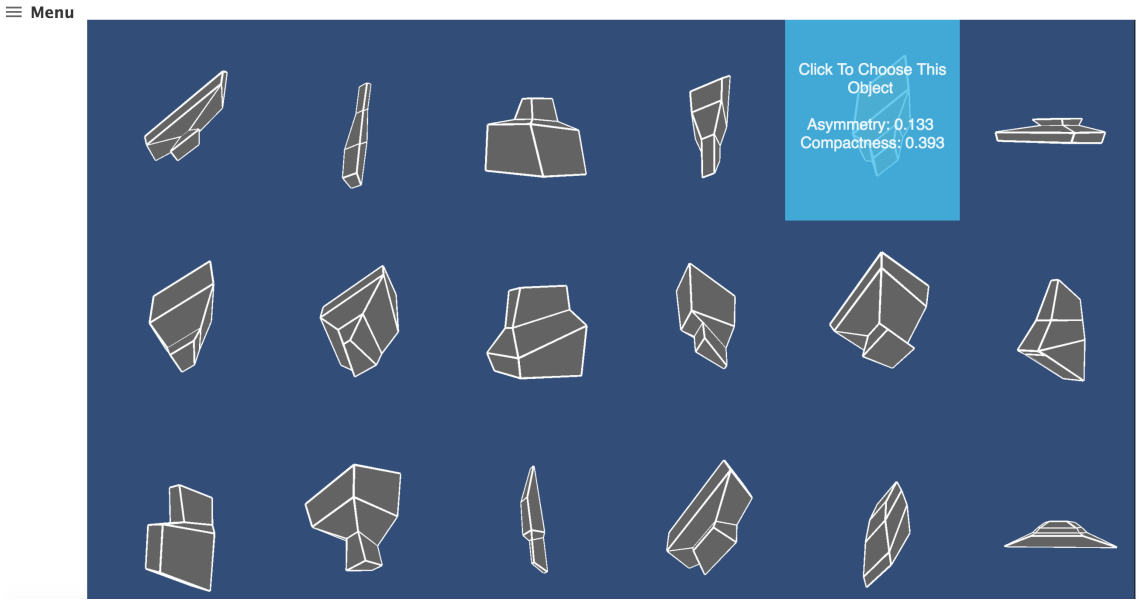


Fig. 2.12. All shapes from a particular set are displayed

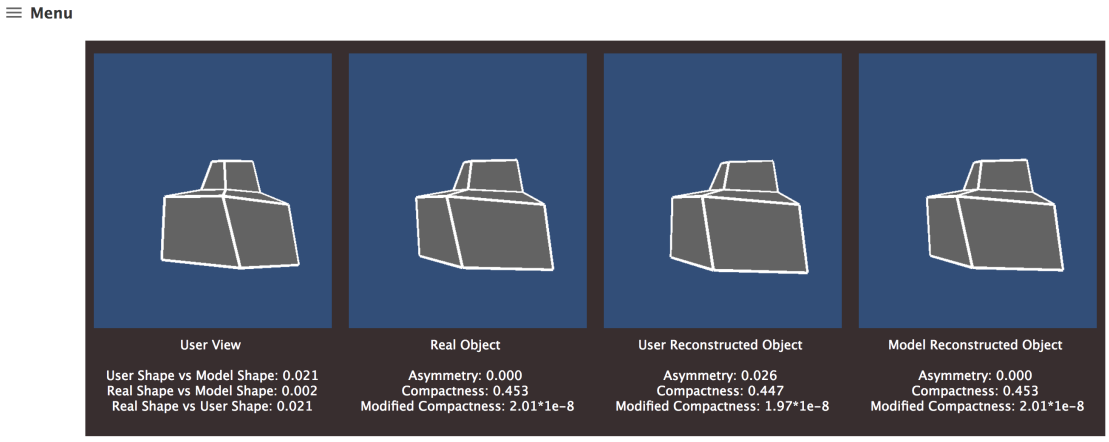


Fig. 2.13. The shape comparison page

3. 3D SHAPE RECONSTRUCTION FROM A SINGLE IMAGE

3.1 Introduction

Recovering shapes from a single view has advantages compared to algorithms based on binocular disparity. Establishing symmetry correspondence (i.e., identifying which two pixels in the image are projections of 3D symmetric points) leads to more accurate reconstructions in comparison to establishing binocular correspondence, and then using binocular disparity for reconstruction. Reconstructions based on binocular disparity loose accuracy quickly as the distance between the object and camera increases. However, reconstructions based on symmetry are more robust to changes in distance. Moreover, 20 years ago Zabrodsky and Weinshall [39] showed that using a symmetry prior can substantially improve the accuracy of reconstructions from multiple views.

Symmetry has been used in the past for 3D reconstruction of objects and scenes; however, some of these methods [40–42] require extensive user intervention, like manually establishing symmetry correspondences. Methods described in [43, 44] concentrate on dense 3D reconstruction of scenes, rather than shape reconstruction from curves. Sinha et al. [45] considered the symmetric curve matching problem; however, their dynamic programming algorithm only works with restricted views of the 3D object. Xue et al. [46] used symmetry to obtain depth maps; however, they used synthetic images that have planar surfaces bound by straight lines. In our work, we are interested in estimating a 3D shape representation of the object in the scene. Though we use a planarity constraint, we are also interested in obtaining shape representations for objects with approximately planar surfaces. We accomplish this by using a planarity measure that counts the number of planes required to approximate

the object. For instance, we can approximate the furniture shown in Figure 3.1 with four planes. Three of these planes are shown in Figure 3.1, and the fourth plane, not shown, is opposite to the orange plane. This concept is made more clear in the following sections. In the next section we provide an overview of the algorithm, and the following sections describe each step in detail. This is followed by results and conclusions.

3.1.1 Overview

The edge map of the image of an object is often a reasonable representation of its 2D shape, and so it can be used to establish symmetry correspondence by identifying pairs of points on the edge map that are projections of 3D symmetric points. Points that symmetrically correspond in an image must be co-linear with the vanishing point. However, as shown in Figure 3.2 (b), more than two edge pixels can be co-linear with the vanishing point, and therefore, we need a way to discriminate correct and incorrect correspondences. It is advantageous to work with smooth curves (if such curves can be extracted), because it reduces the complexity of the problem (the number of curves is often much less than the number of edge pixels). Additionally, curves have shape which can be utilized. Therefore, in this framework, we establish symmetry correspondence for pairs of 2D curves, instead of just working with pairs of 2D points.

The first step to solving symmetry correspondence is estimating the position of the vanishing point in the camera image. There are several methods available in the literature for estimating vanishing points from monocular images [47–56]. Most methods involves detection and clustering of oriented edge points [50, 54, 56] or line segments [47,51,53,57] in images. However, we use the estimates obtained by Michaux and Pizlo [58], because it is more reliable as it uses binocular information. Note that if one can identify higher order features like long curves or corners, it is also possible to partially solve symmetry correspondence without estimating the vanishing point

first. Establishing correspondence for one or more pairs of features will lead to the vanishing point, which can then be used to solve correspondence for the remaining points and parts of the image. Once the vanishing point is known, the next step is extracting long meaningful curves, where the word meaningful implies that the curve would make sense to a human observer. We have evidence that the human visual system extracts long curves by solving the shortest (least-cost) path problem in the image [59]. We incorporate this method in our algorithm. Specifically, we minimize the cost of a path, where the cost is a weighted combination of the interpolations and turning angles. From now on, we use the term correspondence to denote a pair of 2D curves that symmetrically correspond to each other in the 3D representation. The next step is identifying some candidate correspondences and candidate planes (planes that could be used to approximate the 3D shape of the object). Though we could start off by assuming that any long curve extracted could correspond to any other curve, we use a few criteria to reject some unlikely correspondences from the list of all possible correspondences, resulting in what we refer to as candidate correspondences. Once we have the candidate correspondences, we evaluate which correspondences lead to a 3D shape recovery that can be approximated by a minimum number of planes. This is achieved by converting the problem into a binary integer program and solving it using the Gurobi solver [60]. Each of these steps are explained in detail in the next sections.

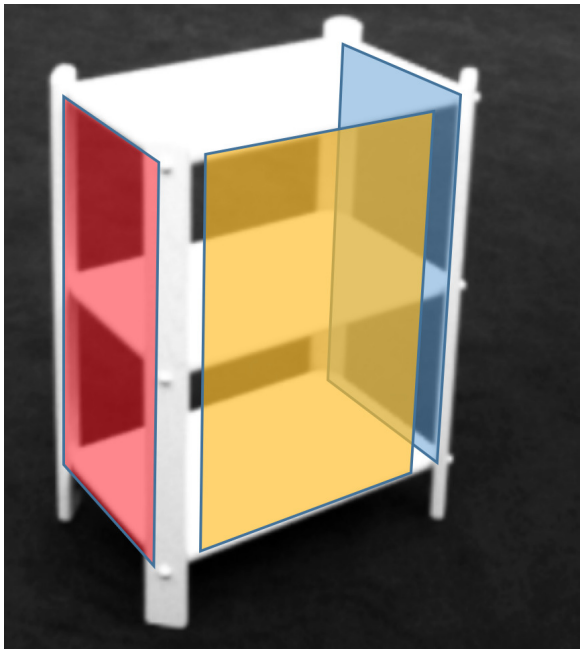
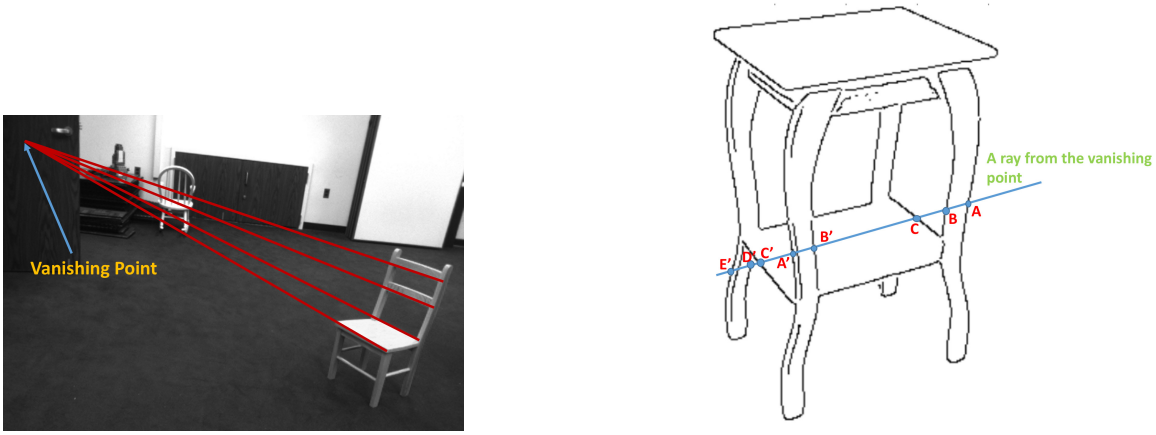


Fig. 3.1. Planar approximation by using minimum number of planes.



(a) (b)

Fig. 3.2. (a) Vanishing Point (b) Symmetry Correspondence Problem.

3.1.2 Curve extraction

The first step in curve extraction is edge detection. As mentioned earlier, the image edge map serves as a representation of the 2D shape of the object. The canny operator is used with an adaptive threshold to form an edge map. Connected components in the edge map are then identified and are broken down, based on gradient orientation, to get short pieces of curve. The idea is to split the connected components at high curvature points, like junctions, to obtain short and smooth pieces of curves. Figure 3.3 (a), shows short curve pieces obtained for the image of a piece of furniture. Longer curves are obtained by combining these short curve pieces. This is achieved by finding the shortest paths between all pairs of short pieces of curve with a cost function that penalizes spatial separation and large turning angles. To determine the turning angle and the spatial separation, the end points of the short curve pieces are first computed. The closest endpoints of two curves decides how the curves connect, which in turn decides the distance and the turning angle between them. For instance, consider curve combination a) in Figure 3.3 (b). An approximate 145 $^{\circ}$ turn is required to continue from the blue curve to the red curve. So what we are calculating is literally the turning angle. A point to note here is that when joining curves, rather than straight lines, the direction of the curve (used to calculate turning angle) is represented by a few pixels near the vicinity of the connecting end points. After the pairwise distances and turning angles are computed for all curve combinations, curve combinations with very high turning angles, or very large interpolated distances, are rejected. I.e., combining such curves is forbidden. The turning angle values and distances are then normalized separately, by subtracting the mean and dividing by the standard deviation. This can result in a negative cost for some curve combinations, so the absolute value of the minimum is added to avoid this. Turning angles are weighted one and a half times in comparison to distances. As shown in Figure 3.3 (b), smooth curves are assigned lower costs. Although shortest paths between all pairs of short curve pieces are computed, we only use those whose

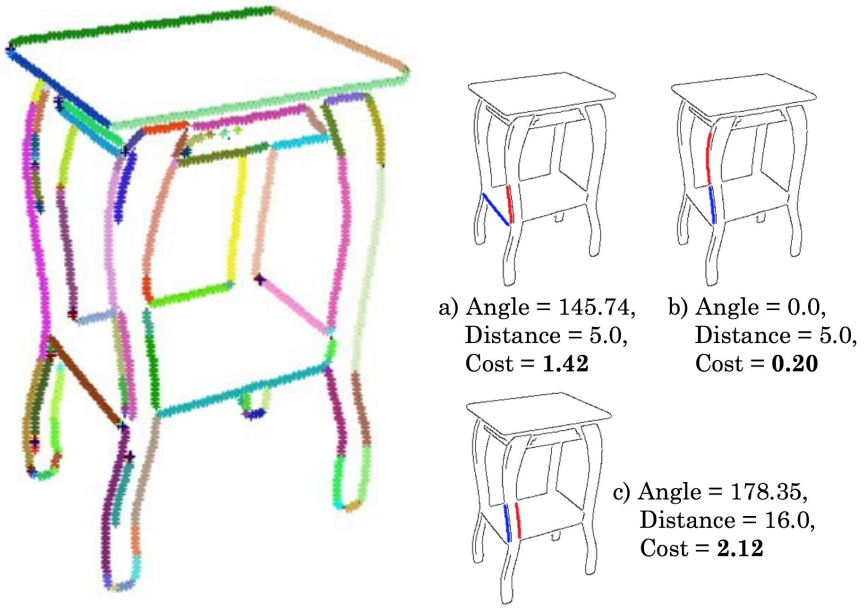
cost is lower than a threshold in the next step. An example of such a low cost path is shown in Figure 3.3 (c). These long curves are then used to identify candidate correspondences and candidate planes.

3.1.3 Identifying Candidate Correspondences and Planes

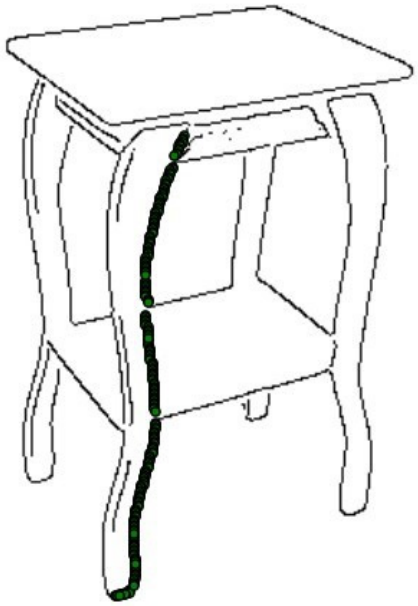
The idea, as mentioned earlier, is to view the correspondence problem as a curve matching problem. I.e., given a curve, say curve A, we would like to identify another curve from the set of extracted curves, which is the symmetrical counterpart of A. If curve B is the symmetric counterpart of curve A, then curve B is said to correspond to curve A. The set (A, B) is called a correspondence. We have a set of long curves at the end of the curve extraction step; however, we do not know the correspondences. The problem of identifying correspondences can be converted into a binary integer program (BIP). However, candidate correspondences must first be identified in order to formulate the problem as a BIP. Ideally, the candidate correspondences form a superset of which the correct correspondences are a subset. Let $s_1, s_2, \ldots, s_{N_c}$ represent N_c extracted long curves. Let S_A represent the set of all possible pairs of curves (correspondences), i.e., $S_A = \{(s_i, s_j) \mid i \neq j\}$. Most of the correspondences in this set are incorrect, and can be rejected based on criteria described later. The idea here is to select a set of correspondences, S_C, such that $S_C \subset S_A$, and ensure at the same time that the true correspondences are included in S_C. In order to accomplish this, we use two types of criteria. One type of criteria applies to curves in the 2D image, and the other applies to the 3D reconstruction of the curves.

2D and 3D Criteria

We use three 2D criteria when choosing candidate correspondences. The first two criteria deal with necessary conditions, and the third criterion is a heuristic. The first criterion used to judge whether a correspondence (s_i, s_j) should be a part of S_C, is to look at the overlap between s_i and s_j when viewed from the vanishing point. As



(a) (b)



(c)

Fig. 3.3. (a) Different pieces of curves are represented by different colors. (b) Costs for combining short curves. (c) A low-cost long curve extracted by the shortest path algorithm.

shown in Figure 3.4 (a), curves that truly correspond have a large overlap. Images of symmetric 3D curves have 100% overlap when viewed from the vanishing point, and therefore correspondences with low overlap can be safely rejected.

Shape dissimilarity between two 2D curves is another criterion for rejecting correspondences. As long as the 3D symmetric curves are approximately planar, their projected images have similar shape [61]. Psychophysical experiments show that when the 2D curves are arbitrarily different, then they are not perceived by observers as 3D or symmetrical [20]. The shape similarity is evaluated for polygonal approximations of the curves, where the polygonal approximations are obtained by sampling the curves using rays from the vanishing point (as shown in Figure 3.4 (b)). Comparing the turning angles at each of the sampled points serves as a shape match metric, which can be used to decide whether a correspondence should be part of S_C. Images of two planar symmetric curves either always turn the same way, or always turn the opposite way at each sampled point [20]. For instance, the symmetric curves that are part of the object in Figure 3.4 (a) turn the opposite way, while those in Figure 3.4 (b) turn the same way. The signed turning angles are either subtracted or added, depending on whether the curves turn the same way or the opposite way. The ambiguity in whether turning angles are added or subtracted is resolved by counting the number of times that the curves turn in the same way, or the opposite way (at the sampled points), and then choosing the direction with the maximal count. This measure leads to a low shape cost for the images of planar symmetric curves, but not necessarily for non-symmetric curves.

For pairs of straight lines, the shape similarity criterion is ineffective, because lines always have zero turning angles, up to pixelation error in the image. In such cases their relative edge orientation can be used to choose candidate correspondences. I.e., corresponding straight lines usually have similar edge orientations. We use K-Means to cluster the edge orientations for the entire image into three clusters. We use three clusters because rectangular objects have three dominant directions. We then remove edge pixels that are more than one standard deviation away from their

cluster center. This results in four clusters of edge pixels (as shown in Figure 3.4 (c)). Three clusters correspond to the three cluster centers (shown in red, green and blue) and another set of unclustered edge pixels belonging to none of the clusters(shown in black). This is done to ensure that pixels with ambiguous edge orientations (i.e., their edge orientations cannot be assigned to any group with confidence) are not forced into being part of one cluster or another. When looking for candidate correspondences between two curves, we insist that of those pixels that are clustered, more than half of the pixel-wise correspondences belong to the same cluster.

There is one 3D criterion for choosing candidate correspondences: we assume that 3D curves are approximately planar. Therefore, the approximate planarity of a pair of curves can be used to decide whether or not to include or exclude a correspondence. To produce a planarity score, we first assume that the two curves correspond and reconstruct them in 3D. Planes are then fit using RANSAC. The goodness of the fit tells us if the 3D curves are approximately planar. Curve correspondences that produce highly non-planar 3D reconstructions can be rejected.

Identifying Candidate Planes

We derive a set of candidate planes, that are used to approximate the 3D object, from the set of candidate correspondences. First we reconstruct the 3D curve pairs from the candidate correspondences, and consider each curve pair in turn. These two 3D curves may be co-planar. In this case, we can fit a single plane to both curves and add it to the list of candidate planes. When curve pairs are not coplanar, we fit planes to both the 3D curves separately, and add them to the set of candidate planes.

As mentioned before, we assume that the approximate direction of gravity is known. It is reasonable to assume that most planes used to approximate a real world object are vertical [20], and this is useful for finding additional candidate planes. When the individual curves in a curve pair are 3D lines, then we also add the candidate vertical planes that pass through each 3D line. This is accomplished by minimizing

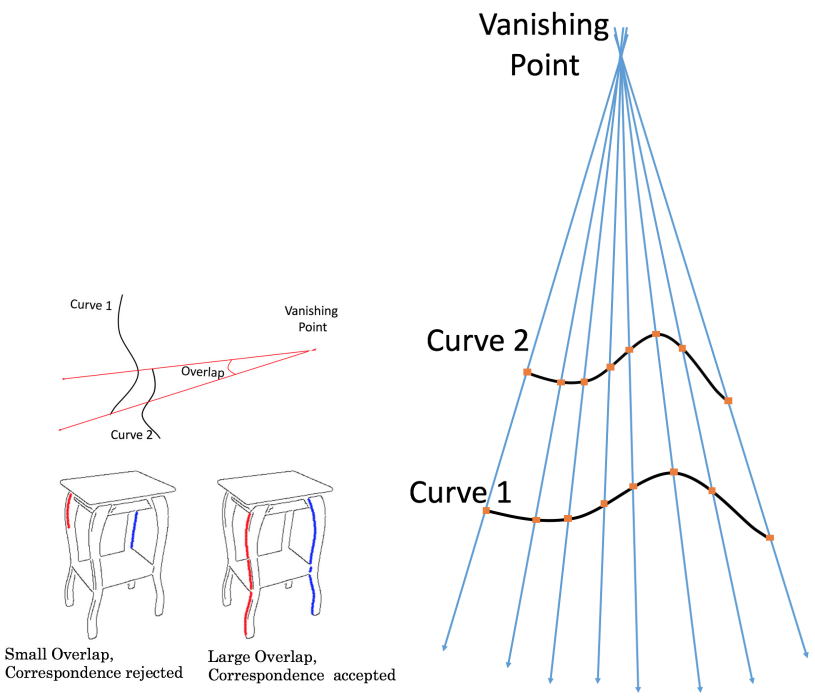
the function: $\sum_{i=1}^{N}(a\,x_i + b\,y_i + c\,z_i - d)^2 \;+\; \alpha(a\,g_1 + b\,g_2 + c\,g_3)^2$, where $[x_i, y_i, z_i]$ represents the points on the 3D line, $[a, b, c, d]$ represents the plane we are seeking, $[g_1, g_2, g_3]$ represents direction of gravity, and α is a weight factor.

We now have a large set of candidate planes; however, it is very likely that many planes are redundant. Therefore, mean shift clustering is performed to reduce the number of planes. Once we have identified the candidate planes and candidate correspondences, we frame the problem of choosing the correct curve correspondences from S_C as a binary integer program as described in the next section.

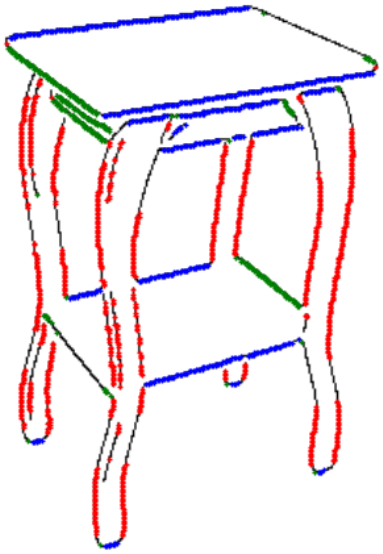
3.1.4 Choosing the Correct Correspondences

Let $c_1, c_2, \ldots, c_N \in S_C$ be the set candidate correspondences, and $\pi_1, \pi_2, ..., \pi_M \in \Pi_C$ be the set candidate planes, identified in the previous steps. In the next step we identify, the symmetric correspondences, and the planes, used to approximate the 3D shape of the object. In other words, we choose a subset of the correspondences in S_C, and a subset of planes in Π_C, such that the resulting 3D reconstruction uses a minimal number of planes, while ensuring that a substantial portion of the object is still reconstructed. This problem can be formulated as a binary integer program (BIP).

The table on the left in Figure 3.5 shows the variables involved in the BIP. Each row represents curve correspondences, while each column represent candidate planes. Every variable ${x^k}_{ij}$ is a binary variable. Two things are implied when a variable ${x^k}_{ij}$ is set (i.e., when ${x^k}_{ij} = 1$). First, the curve correspondence i is chosen as part of the subset of true correspondences. Second, the 3D curve obtained from this reconstruction is assigned to plane j, which in turn means that plane j is chosen as one of the planes used to approximate the 3D object. As shown in Figure 3.5, each correspondence is repeated twice. This is because each correspondence involves two 2D curves from which two 3D curves are reconstructed, and these two 3D curves need not be assigned to the same plane. That is, we must be able to assign a single correspondence to one



(a) (b)



(c)

Fig. 3.4. (a) Overlap from Vanishing Point, (b) Polygonal approximation for the shape match metric, and (c) Clustered edge orientations.

or two separate planes, and hence, in the BIP, each correspondence is repeated twice. For example, let us say that the i^{th} correspondence, $c_i = (s_p, s_q)$, then ${x^1}_{im} = 1$ and ${x^2}_{in} = 1$ means that the 3D curve resulting from the reconstruction of 2D curve s_p, was assigned to plane π_m, and the 3D curve resulting from the reconstruction of the 2D curve s_q was assigned to plane π_n.

Variables

	Plane 1 π_1	Plane 2 π_2	....	Plane M π_M
Correspondence 1 c_1	${x^1}_{11}$	${x^1}_{12}$	....	${x^1}_{1M}$
Correspondence 2 c_2	${x^1}_{21}$	${x^1}_{22}$	....	${x^1}_{2M}$
. . .	. . .	. . .	. . .	. . .
Correspondence N c_N	${x^1}_{N1}$	${x^1}_{N2}$	....	${x^1}_{NM}$
Correspondence 1 c_1	${x^2}_{11}$	${x^2}_{12}$	....	${x^2}_{1M}$
Correspondence 2 c_2	${x^2}_{21}$	${x^2}_{22}$	....	${x^2}_{2M}$
. . .	. . .	. . .	. . .	. . .
Correspondence N c_N	${x^2}_{N1}$	${x^2}_{N2}$	....	${x^2}_{NM}$

Cost

	Plane 1 π_1	Plane 2 π_2	....	Plane M π_M
Correspondence 1 c_1	${w^1}_{11}$	${w^1}_{12}$	....	${w^1}_{1M}$
Correspondence 2 c_2	${w^1}_{21}$	${w^1}_{22}$	....	${w^1}_{2M}$
. . .	. . .	. . .	. . .	. . .
Correspondence N c_N	${w^1}_{N1}$	${w^1}_{N2}$	....	${w^1}_{NM}$
Correspondence 1 c_1	${w^2}_{11}$	${w^2}_{12}$	....	${w^2}_{1M}$
Correspondence 2 c_2	${w^2}_{21}$	${w^2}_{22}$	....	${w^2}_{2M}$
. . .	. . .	. . .	. . .	. . .
Correspondence N c_N	${w^2}_{N1}$	${w^2}_{N2}$	....	${w^2}_{NM}$

Fig. 3.5. BIP formulation.

The table on the right in Figure 3.5 shows the cost associated with setting each variable ${x^k}_{ij}$. I.e., ${w^k}_{ij}$ represents the cost of choosing correspondence i, and assigning (associating) the k^{th} 3D curve resulting from the correspondence i to plane j. Here, $k \in \{1, 2\}$, and refers to either the first or second reconstructed 3D curve from correspondence i. The weight, or cost ${w^k}_{ij}$, consists of two terms, and is given by, ${w^k}_{ij} = exp({d^k}_{ij}) + \beta\,\delta_i$. The first term is the exponential of the mean distance ${d^k}_{ij}$ of the k^{th} 3D curve (reconstructed based on the correspondence i) to plane j. The second term depends on the shape match δ_i (Figure 3.4 (b)) between the two 2D curves involved in the correspondence, and is measured as the average difference in turning angles at the sampled points. Correspondences, whose reconstructed 3D curves are far from the plane, or whose shape match is poor, are removed. The remaining values are normalized, and then a weighted combination (represented by

β) of the two terms is taken. The problem of picking the right correspondences and planes can now be framed as a constrained BIP as shown below.

$$minimize \quad \boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{\mu}\boldsymbol{y}$$

subject to

$$\boldsymbol{x^T l} \geq \left(\frac{p}{100}\right) L \tag{C_1}$$

$$\boldsymbol{Ux} \leq \boldsymbol{1_N} \tag{C_2}$$

$$\boldsymbol{Rx} = \boldsymbol{0_N} \tag{C_3}$$

$$\begin{bmatrix} \boldsymbol{E} & \boldsymbol{-I_M} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \leq \boldsymbol{0_M} \tag{C_4}$$

$$\sum_{k=1}^{M} y_i \ \geq \ 2 \tag{C_5}$$

$$x^k{}_{ij} \in \{0,1\} \ \forall \ i,j,k \tag{C_6}$$

$$y_i \in \{0,1\} \ \forall \ i \tag{C_7}$$

where,

$\boldsymbol{x} = ({x^1}_{11}, {x^1}_{12}, .., {x^1}_{1M}, {x^1}_{21}, {x^1}_{22}, ..., {x^1}_{2M}, ..., {x^1}_{NM}, {x^2}_{11}, {x^2}_{12},$

$.., {x^2}_{1M}, {x^2}_{21}, {x^2}_{22}, ..., {x^2}_{2M}, ..., {x^2}_{NM})$

$\boldsymbol{w} = ({w^1}_{11}, {w^1}_{12}, .., {w^1}_{1M}, {w^1}_{21}, {w^1}_{22}, ..., {w^1}_{2M}, ..., {w^1}_{NM},$

${w^2}_{11}, {w^2}_{12}, .., {w^2}_{1M}, {w^2}_{21}, {w^2}_{22}, ..., {w^2}_{2M}, ..., {w^2}_{NM}) \in \mathbb{R}^{MN}$

$\boldsymbol{y} = (y_1, y_2, ..., y_M), \quad \boldsymbol{l} = (l_1, l_2, ..., l_{MN}) \in \mathbb{R}^{MN}$

$\boldsymbol{U} \in \mathbb{R}^{N\times(NM)}, \quad \boldsymbol{R} \in \mathbb{R}^{N\times(NM)}, \quad \boldsymbol{E} \in \mathbb{R}^{M\times(NM)}$

$\boldsymbol{I_M}$ is the identity matrix of order M

$\mathbf{1_N} = (\underbrace{1, 1, ..., 1}_{N \text{ elements}})^T, \quad \mathbf{0_N} = (\underbrace{0, 0, ..., 0}_{N \text{ elements}})^T, \quad \mathbf{0_M} = (\underbrace{0, 0, ..., 0}_{M \text{ elements}})^T$

$\boldsymbol{\mu} \in \mathbb{R}^M, \ \mu_i \in (0, \infty) \ \forall \ i$

The elements of vector $\boldsymbol{y}$ are binary variables which indicate whether a plane is selected or rejected. I.e., if $y_i = 1$ then plane i is selected as a plane that is used to approximate the 3D object. Constraint C_4 is devised to ensure that the elements of $\boldsymbol{y}$ are indicator variables for including or excluding planes. As mentioned before in Figure 3.5, the columns represent planes. Hence, if any of the variables (${x^k}_{ij}$) are set in a column, say j, then constraint C_4 ensures that plane j is chosen and y_j is set. This can be achieved by insisting that:

$$\sum_{k=1}^{2} \sum_{i=1}^{N} \frac{{x^k}_{ij}}{2N} \leq y_j$$

The variable μ represents the cost for each included plane, thus biasing the solution towards fewer planes. Another important constraint that needs to be imposed is that if one of the variables in row i is set, then one variable in row $i + N$ has to be set too. This is because we cannot assign one 3D curve (from correspondence i) to a plane, and not assign the other 3D curve (from correspondence i) to a plane. This can be achieved by imposing the following constraint for row i:

$$\sum_{j=1}^{M} {x^1}_{ij} = \sum_{j=1}^{M} {x^2}_{(i+N)j}$$

Constraint C_3 imposes this condition for all the correspondences (half the rows). Hence matrix $\boldsymbol{R}$, used to represent this set of constraints, has N rows.

The trivial solution – setting $\boldsymbol{x}$ and $\boldsymbol{y}$ to the zero vectors – is not interesting, since there is no reconstructed shape. Constraint C_1 ensures that the trivial solution is not selected. Element $l_{(iM+j)}$ of vector $\boldsymbol{l}$ represents the length of the curves involved in correspondence i. Hence, the dot product $\boldsymbol{x^T l}$ gives the total length of all the curves that are part of the selected correspondences. Constraint C_1 ensures that this length is at least $p\%$ of L, where L is the total length of all the distinct curve pieces in the image.

Constraints C_2 ensure that the correspondences are unique. The idea here is that every edge pixel has a unique symmetric counterpart in the 3D object, and this constraint has to be explicitly imposed in our formulation of the problem. For instance, in Figure 3.6, correspondences (a) and (b) could be chosen at the same time, but correspondences (a) and (c) cannot, because of the angular overlap from vanishing point. For each correspondence, we can identify other correspondences that have a substantial angular overlap from the vanishing point. This information is used to add a constraint that only one among those with substantial overlap is selected. Constraints C_2 is a matrix where each row represents this constraint for a given correspondence.

Since, we do not expect a single plane to approximate any object, we ensure that at least 2 planes are selected by enforcing constraint C_5. We also add a preference for planes approximately parallel or perpendicular to the symmetry plane, by decreasing the weight, μ_i, of such planes to eighty percent of that of other planes.

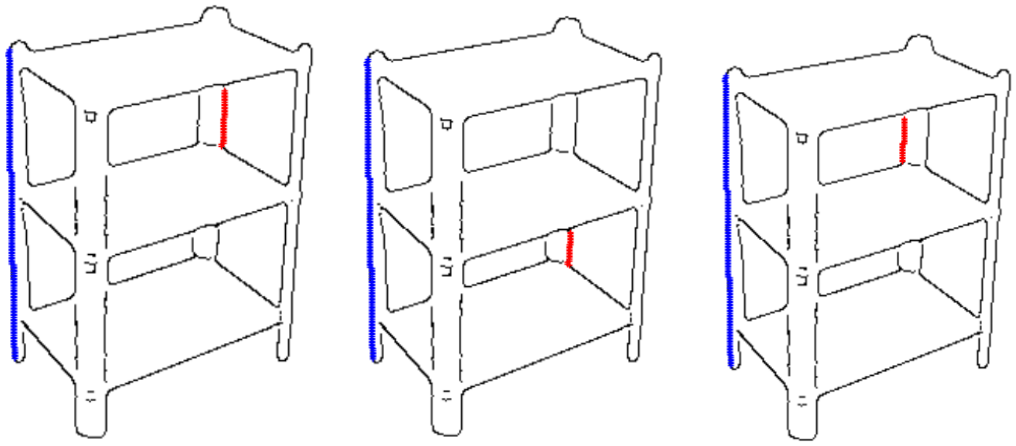
We refer to straight line edges in the edge map that approximately pass through the vanishing point as self-symmetric lines. For example, in Figure 3.4 (c), the blue lines are self-symmetric. They are referred to as self-symmetric because, the symmetric counterparts of points on such lines lie on the same line. Since we frame the

problem as a curve matching problem, and since we do not expect any other curve/line to be the symmetric counterpart of a self-symmetric line, we remove these lines from the edge map prior to all processing. These lines can be fit later to the 3D reconstruction that was obtained from optimizing the BIP. The 3D orientation of these lines is the same as the normal of the symmetry plane. The neighborhood information from the 2D image is used to determine the exact position (and extent) of these lines in 3D. A small neighborhood of pixels in the 2D image, around the endpoints of the self-symmetric line, is considered. After the BIP optimization is complete, the 3D position of some of these neighborhood pixels may be available depending on whether the optimization process was able to find matches for them. If 3D positions are available for pixels in both neighborhoods (corresponding to both end-points of the self-symmetric line), then we consider all possible lines that go between points in one neighborhood with points in the other neighborhood. We then pick the 3D line that is most aligned with the normal of the symmetry plane, to obtain the 3D line corresponding to the self-symmetric 2D line.

As mentioned before, the BIP is solved using the Gurobi solver [60]. In order to account for occlusions in the image, we ask the algorithm to reconstruct only 70% of the pixels in the edge map. Specifically, the value of p in constraint C_1 is set to 70. If a large part of the object is occluded, this value will be too high, and the problem is infeasible. The value of p is initialized to 70, and is automatically decremented if the Gurobi solver detects that the problem is infeasible. In practice it does not take more than a couple of attempts to find a feasible value of p.

Though a large number of correspondences obtained by the optimization process are correct, it was observed that a few mistakes were made. One of the reasons for the loss of accuracy comes from the clustering of candidate planes. In order to obtain better plane estimates, once the optimization process converges and a solution to the problem is obtained, the planes associated with the correspondences chosen by the optimization process are identified. Keep in mind that the candidate planes were obtained from candidate correspondences in the first place, and hence for each

chosen correspondence, the planes it added/contributed can be identified. (I.e., the planes before clustering.) The optimization process is then rerun with these planes as candidate planes. When the process is rerun, we consider if the new set of candidate planes requires clustering. It may not, because we expect the new set of candidate planes to be much smaller than the initial set of candidate planes. Even when the number of planes is large enough to require clustering, the accuracy of the clustered planes should be much better. In practice, rerunning the optimization process with the new set of planes corrects some of the errors made in the first run.



(a) (b) (c)

Fig. 3.6. Correspondences (a) and (b) could be chosen simultaneously, but correspondences (a) and (c) cannot, because of the angular overlap from vanishing point.

3.1.5 Results

Figures 3.7 and 3.8 show some of the results obtained. Some of these images were taken by us using a Point Grey Bumblebee2® stereo camera, and about half were obtained from the internet. For images taken with the Bumblebee2®, we estimated the vanishing point and the direction of gravity using the algorithm described in [58]. The image from the left camera was then used by our algorithm as input, along with

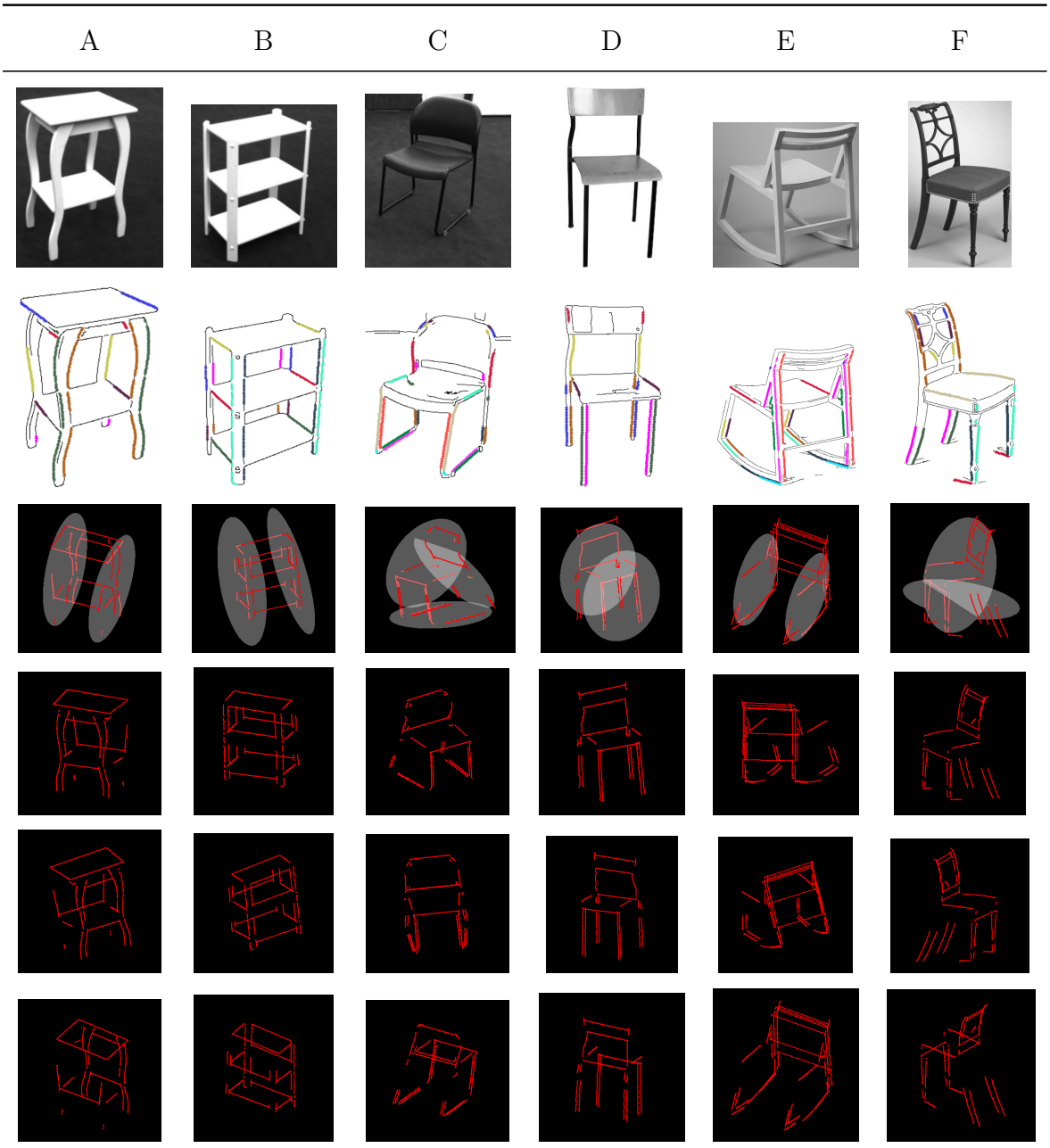


Fig. 3.7. Results for objects A-F: Original Image is shown in row one, row two shows the symmetric correspondences detected with corresponding curves shown in same color, the planes selected are shown in row three, and rows four through six show three different views of the reconstructed object.

the estimates for the vanishing point and the direction of gravity. The focal length and the principal point were read off of the camera's firmware. For the internet images we tried to obtain the estimates of camera calibration parameters and vanishing points

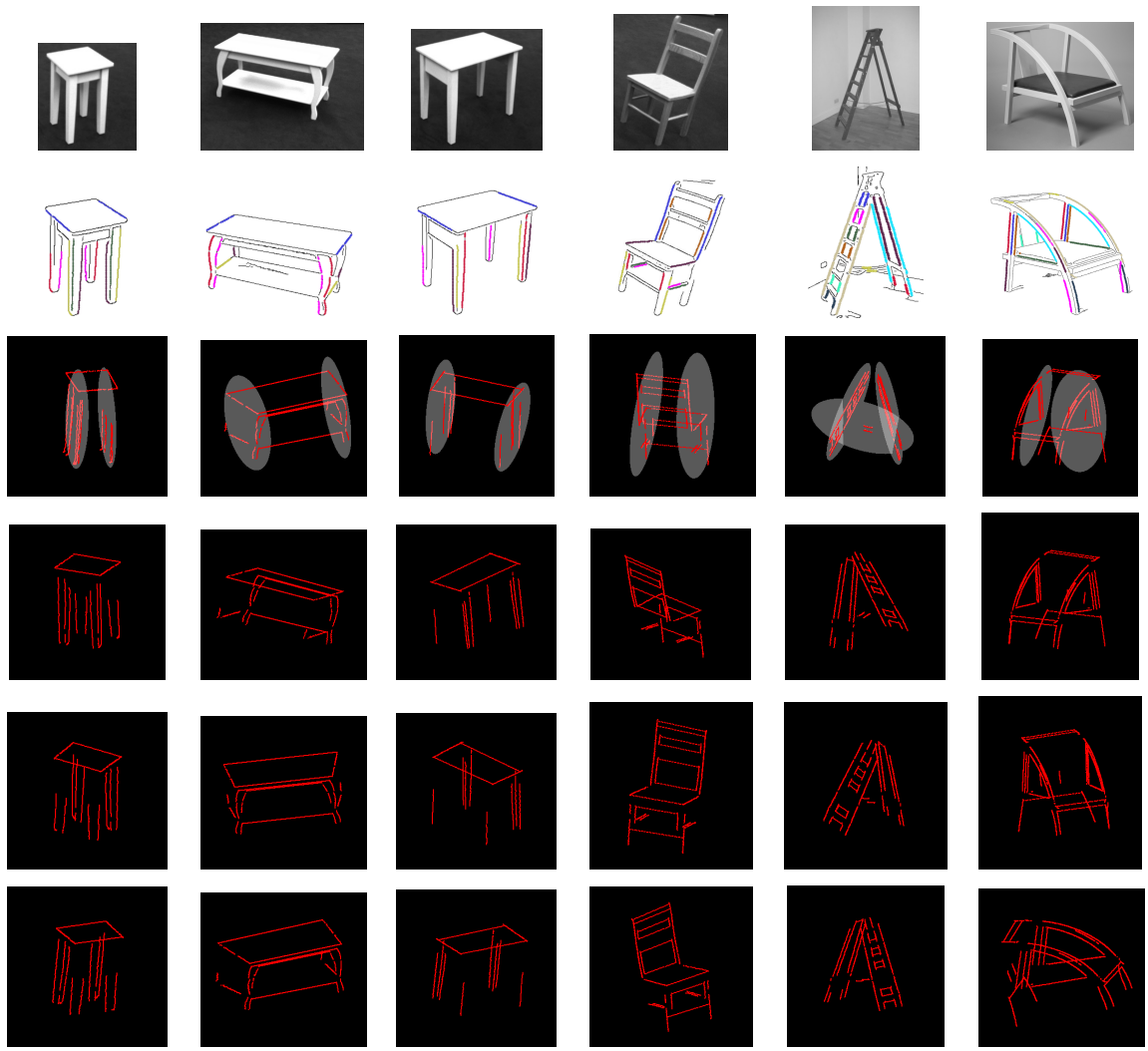
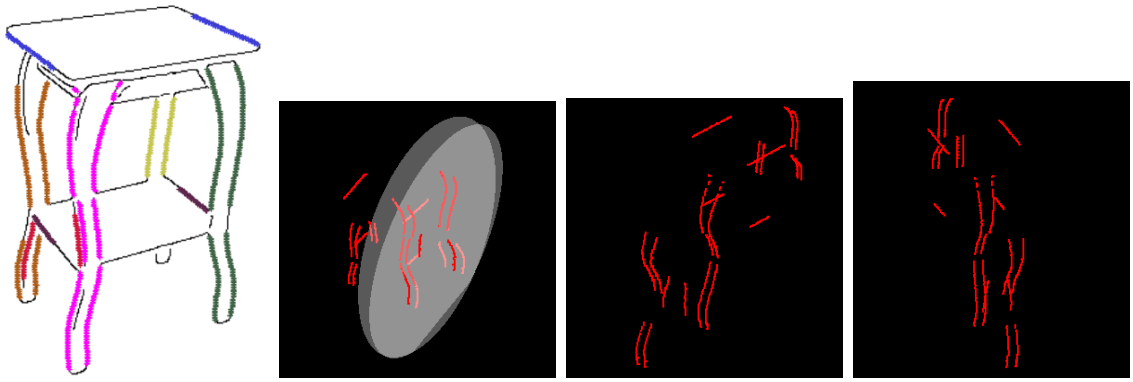


Fig. 3.8. Results for objects G-L: Original Image is shown in row one, row two shows the symmetric correspondences detected with corresponding curves shown in same color, the planes selected are shown in row three, and rows four through six show three different views of the reconstructed object.

using existing algorithms [48–50]. These estimates were not reliable. In fact, under the Manhattan world assumption [54], used by most of these algorithms, two of the three vanishing points correspond to the direction of gravity, and the normal of the symmetry plane. But it is difficult to apply to Manhattan world assumption to objects like E, K, and L, because not all the required vanishing points are salient in the images.



(a) (b) (c) (d)

Fig. 3.9. (a) Wrong correspondences resulting from allowing to curves very close to each other to correspond, (b) the planes picked by the algorithm, (c) and (d) different views of the reconstruction.

This is perhaps the reason why the automatic vanishing point estimating algorithms find these images difficult to handle. Therefore, for such images from the internet, two vanishing points (representing orthogonal directions in 3D) were estimated by hand. Since, these vanishing points represent orthogonal directions in the 3D space, the focal length can be obtained if the principal point is assumed – typically the center of the image. The solution to $x_{V1}\,x_{V2} + y_{V1}\,y_{V2} + f^2 = 0$ gives the focal length, where (x_{V1}, y_{V1}) and (x_{V2}, y_{V2}) are the two estimated vanishing points, and f is the focal length. Though estimating the direction of gravity can be challenging, our algorithm only needs a crude estimate, and hence this is not a problem in general.

The results show that the algorithm was able to obtain a reasonably good representation of the 3D shape of the objects. The 3D shape representation is accurate, in most cases, if we take into consideration that the algorithm was not designed to take care of occlusions in the image. Shape is a spatially global property and so is symmetry and planarity. Therefore, enforcing these constraints should lead to good shape recovery for objects where such regularities (at least approximately) exist. As an example, for object L, the hind legs are occluded, but the algorithm reconstructs

it from a wrong correspondence, and it is consistent with a good shape representation because of the regularities imposed on the 3D reconstruction. Similarly, the correspondences obtained for the top part of the hind legs (represented by red, brown and dark blue lines) for object L are not accurate, but the shape representation is still good.

One of the problems we faced while performing reconstructions, is that when 2D curves that are very close to each other are allowed to correspond, it often leads to bad reconstructions, as shown in Figure 3.9. To deal with this issue, we use a distance threshold to prevent close 2D curves from corresponding. The distance is measured as the median distance between corresponding points on the two 2D curves. To decide on the threshold, we first note that the shape of an object is defined mostly by curves close to the 3D convex hull of the object. Hence, reconstructing 3D curves close to the hull is more important. Self-symmetric lines are used to dynamically decide this threshold. For non-degenerate views, the length of self-symmetric lines is a good estimate of the distance between curves that actually correspond. We choose a value of $0.4\,l_{ss}$ as the distance threshold, where l_{ss} is the length of the longest self-symmetric line. This value works for all images except for object K, for which it had to be set to $0.2 * l_{ss}$. Using such a threshold means that some of the internal details of the shape of the object may not be reconstructed, as seen with object F. But this method can still capture the most important aspects of the 3D shape. A better solution to this problem is to view 3D shapes as composed of 3D parts, and correspondences should be found between the images of parts rather than the images of curves. The object in Figure 3.9 can be thought of as being composed of six parts: four legs, and two flat surfaces parallel to the ground. Parts either have symmetric counterparts, or they are self-symmetric. (In Figure 3.9, the legs have symmetric counterparts, and the flat surfaces are self-symmetric.) Dealing with parts, if they can be reliably detected, would simplify the problem and make it more robust. Moreover, some work [62, 63] towards part detection is already available and can be used.

The algorithm runtime averages about 15 secs on a 2.8 GHz Intel® Core i7 quad core processor with 16 GB RAM. The code is written in Python. Though BIP is an NP-Hard problem, the number of variables involved in the BIP is usually not more than 2000. The Gurobi solver can easily handle such problems and usually converges within a second or two. Setting up the integer program accounts for the bulk of the runtime. There is ample scope for performance improvements, if it is a priority.

3.1.6 Conclusion

We have designed an algorithm that can effectively reconstruct 3D shapes from single camera images by employing symmetry and planarity priors. It demonstrates how these priors can be used to convert an ill-posed problem into a well-posed one. The results demonstrate the effectiveness of the idea of viewing the reconstruction problem as a constrained curve matching problem. By shape, some sort of regularity is implied, and by using regularities like symmetry and planarity we have successfully reconstructed simple shapes that can explain the given images.

4. SKELETON EXTRACTION FROM 3D POINT CLOUDS BY DECOMPOSING THE OBJECT INTO PARTS

4.1 Introduction

Decomposing a complex 3D shape into its components is an important problem that has applications in many fields including computer graphics and computer vision. Psychological studies have shown that human shape perception and recognition is based on decomposing complex shapes into simple primitives [22]. Therefore, characterization of a shape by its parts and the connections between them is only natural, especially for applications involving human interaction. A good example in this regard is the work by Funkhouser e. al [64], which aims at constructing detailed 3D models by assembling parts extracted from existing models. Part based approaches also have applications in other area like character animation [65] and surface reconstruction [66–68].

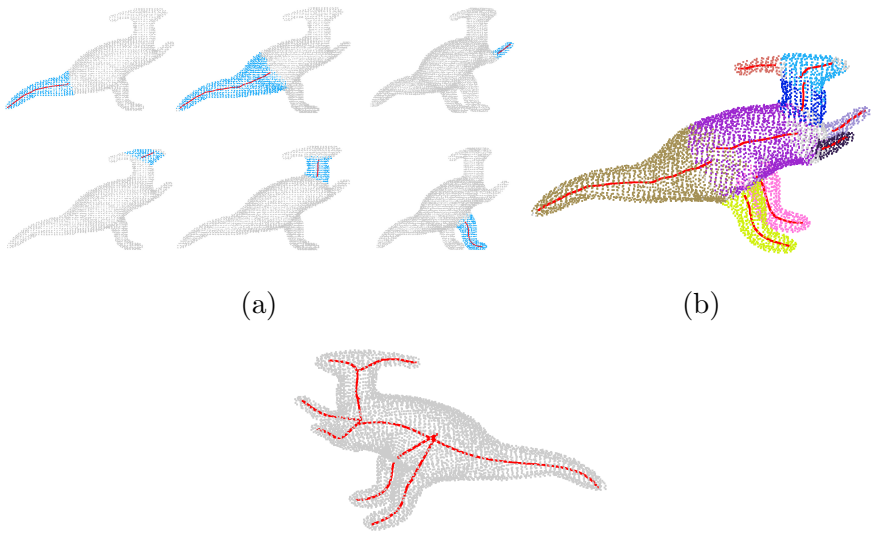
To decompose a shape into its parts, we first need a good definition for a part. A good definition of a part would be general enough to capture the wide variety of part types available in the real world and at the same time be able to decompose a shape into meaningful units. In this context, generalized cylinders (GCs) introduced by Binford [69] would serve as a very good definition of a part. A wide variety of natural as well as man made objects have parts that can be well modeled by GCs. GCs are formed by sweeping a planar cross-section along a 3D axis, with uniform size scaling applied to the cross-section as it moves along the axis. An illustration of this is shown in Fig. 4.2. Translational symmetry is the fundamental property that defines a GC [20]. In this work, we think of parts of a 3D shape as GCs and employ translational symmetry to extract them.

Decomposing a shape into its components can be thought of as an inverse problem which is often times also ill-posed. It is an inverse problem because the forming of a complex 3D shape by interconnecting parts can be thought of as the forward problem and the ill-posedness comes from the fact that there is no unique solution to the problem. Solution to ill-posed inverse problems are obtained by imposing additional constraints (aka priors). In this context, translational symmetry can be thought of as the prior we use to identify parts which eventually leads to the decomposition of shapes.

Skeleton extraction methods often times lead to shape decomposition [70–72]. But in our case it works in the opposite direction. The 3D-axis of a GC, which is integral to its definition, would serve as the skeleton of the part. Therefore, a GC based decomposition would naturally lead to skeletonization of the shape. There exist a method by Zhou et al. [73] which address the problem of decomposing shapes into GCs. As we point out later, though our method has similarities to their approach, there are significant differences stemming from the fact that our method works on unorganized point clouds and theirs on meshes.

As shown in Fig. 4.1, the algorithm has three main steps. In the first step, a few candidate parts are generated. Not all candidate parts are ideal to represent the actual parts of the object, and there is often overlap between different candidate parts. Therefore, an optimal subset of parts are then selected from these candidate parts in the next step. Note that, every part has an axis associated with it, which serves as its skeletal representation. Therefore, in the final step, the individual skeletons of the selected parts are linked appropriately to form the complete skeleton.

It needs to be pointed out that there isn't a definition of curve skeletons that is agreed upon by everyone. The medial axis based definition of a skeleton, introduced for 2D shapes, would result in medial surfaces when extended to 3D shapes. Defining curve skeletons precisely is even more challenging when 3D shapes are presented as unorganized point clouds. Analytical methods that attempt to extract curve skeletons do not usually subscribe to any proper definitions of curve skeletons. However, a curve



(a) (b)

(c)

Fig. 4.1. An overview of the algorithm. (a) Candidate parts are first generated. The points representing the part are shown in blue and the skeletal representation of parts are shown in red. (b) An optimal subset of parts, that can represent the entire point cloud, are selected from the candidate parts. Different parts are shown in different colors. Also shown are the individual skeletons of each part. (c) Appropriate connections are made between skeletons of individual parts to form the final skeletal representation.

skeleton can be intuitively defined as a good 1D representation of the 3D shape. In this context, if we model each part of a shape as a GC, we can obtain a 1D skeletal representation of the 3D shape that is not only intuitive but also well defined. Though such a definition will only lead to a skeletal representation for the individual parts, and obtaining the complete skeletal representation from the part skeletons could be challenging, a few simple post-processing steps can be employed to obtain the complete skeleton as we show later.

In order to generate candidate parts, we first detect the cross-section of parts and then grow parts starting from this initial cross-section to obtain complete parts. An interesting fact to notice here is that growth explains why many things in nature, like limbs of animals and tree trunks, exhibit translational symmetry [20]. We employ translational symmetry to grow parts. Specifically, we start with a initial

cross-sectional cluster (a small cluster of points representing a thin cross-section) and grow this cluster by identifying similar clusters in its neighborhood via 3D point set registration. By combining ideas from [74] and [75] we present a new registration algorithm that is tailor made for our purpose of growing parts via registration. The proposed algorithm makes use of the orientation information of the point clouds (i.e., the point normals) and this, as we show in the results section, helps it to achieve greater accuracy in the registration process. In the results section, we also compare our method to two state of the art methods for skeleton extraction and point out the advantages of using a part based method. We also present a graphical user interface (GUI) to demostrate the advantage of using a part based approach to skeletonization. Specifically, we show the ease with which mistakes, if any, can be corrected with minimal user interaction using the GUI.

In the next section we review the past literature relevant to our work. In section 4.3, we explain the procedure to generate candidate parts. Method for selecting optimal subset of parts and the method for linking selected parts are explained in Sections 4.4 and 4.5 respectively. The GUI is described in section 4.6 and the results are presented in section 4.7. This is followed by sections on the implementation details and conclusion.

4.2 Related Work

We organize the literature review into two sections. First, we discuss methods from the literature used for curve skeleton extraction. This is followed by a review of methods for shape decomposition.

Skeleton Extraction

There is a vast body of prior work addressing the problem of curve skeleton extraction. Methods exist to extract curve skeletons from complete surface models, like polygonal meshes, as well as unorganized point clouds. Methods belonging to the lat-

ter category are the most relevant to our work and hence we focus on these methods for the review. The reader may refer to [76] or [77] for a comprehensive review of the various skeleton extraction methods that are available.

There exist a class of methods that rely on defining a field on the voxelized internal volume of shapes. In such methods, the ridges of the field are of particular interest when it comes to skeleton extraction. For example, in [78], the authors identify valleys, in the generalized potential field, which connect several seed points, to derive the axis of the GC representation of the shape. In [79], Siddiqi et al. extract skeletons by using the fact that the divergence of the euclidean distance transform would be zero everywhere except at the skeletal points. Song et al. [80] improve upon the results obtained by Huang et al. [81] by using the distance field to guide the L1-median skeleton extraction. Note that the appropriate voxelization of unorganized point clouds is a non-trivial task. Apart from this, as we will demonstrate in the results section, these methods will not generalize well to certain shapes in comparison to a part-based approach.

By defining an appropriate real valued function on a compact manifold and then tracking the evolution of its level sets, Reeb graphs, which capture the topology of the manifold, can be created. Methods described in [82–84] employ Reeb graphs to extract skeletons. Authors in [85, 86] extract skeletons by employing Voronoi diagrams. Voronoi diagrams can be used to approximate medial surface of shapes and these can be further pruned to obtain curve skeletons. Tagliasacchi et al. [87] proposed a method, based on the rotational symmetry axis, to extract skeletons from incomplete point clouds of generally cylindrical shapes. Sharf et al. [70] proposed a method that is based on the evolution of a deformable model inside the point cloud. They obtain a first approximation to the skeleton by tracing the growing fronts and perform further filtering to obtain the final curve skeleton. Cao et al. [88] extend the mesh contraction method of Au et al. [71] to point clouds and use Laplacian based contraction followed by topological thinning to obtain curve skeletons. In the results

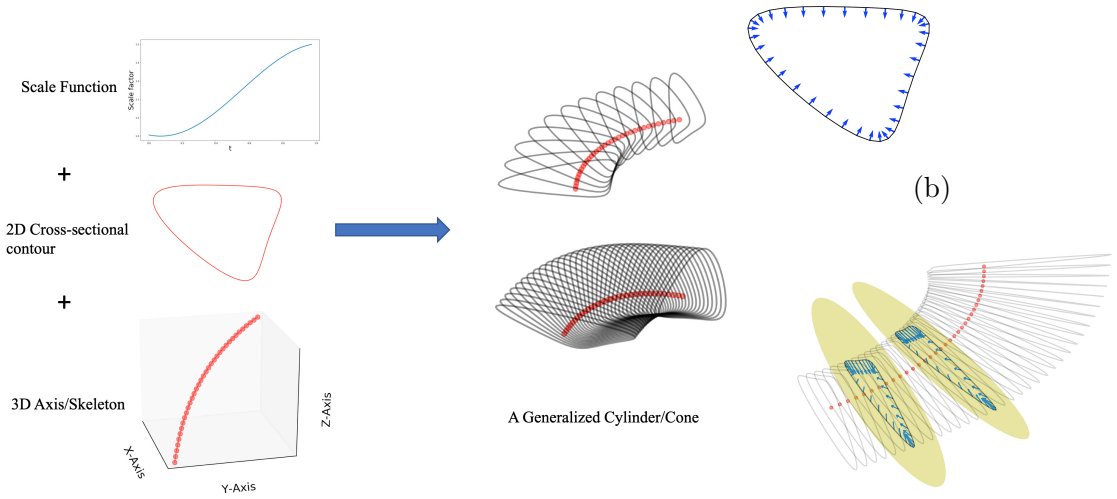


Fig. 4.2. Our definition of a part is based on translational symmetry. (a) The three fundamental properties of 3D Axis, 2D cross-sectional contour and scale function define a part . Parts are formed by sweeping a planar cross-section through 3D space along a defined axis (a space curve) and simultaneously applying size scaling as the cross-section is swept along the axis. (b) The normals to the contour at the various points on the contour are shown. (c) At each point along the axis, the plane containing the corresponding 2D contour is perpendicular to the axis. Or in other words, the normal of the plane represents the tangent to the axis at that point. The normals of the 2D contour lie on the cross-sectional plane and hence they are perpendicular to the normal of the plane.

section, we compare our method with that of Cao et al. and point out the drawbacks of using a contraction based approach for skeleton extraction.

Shape Decomposition

A variety of methods address the problem of segmentation and semantic labelling of point clouds [89]. Skeleton extraction methods can some time lead to decomposition of shapes [70–72]. But model fitting methods, which addresses the problem of shape decomposition, are the most relevant to our work. Most primitive detection and fitting methods employ simple primitives for the purpose. For instance, Schnabel et al. [90] describe an efficient RANSAC based method capable of fitting simple primitives like

planes, spheres, cylinders, cones and tori. Attene et al. [91] present a method to fit simple primitives to triangle meshes using a hierarchical face clustering approach. GCs, compared to these simple primitives, can represent a much broader range of shapes.

Lit et al. [92] present an early work on decomposing polygonal meshes into parts that resemble GCs. They extract approximate curve skeletons by mesh edge contraction and then identify the critical points (points where the topology or geometry changes) by sweeping a plane, perpendicular to the skeleton branches, over the mesh. The parts of the mesh between consecutive critical points are then extracted as components. Though it is not stated explicitly, this method uses the notion of translational symmetry to identify components/parts. One of the two methods to grow parts that we describe later has similarities with this method. However, unlike this method we operate on unorganized point clouds and hence cannot employ their method of skeleton extraction. Moreover, the second method to grow parts that we describe later is more sophisticated as it is based on the very definition of a GC (i.e., based on translational symmetry), and is more suitable for point clouds. In [78], the authors first derive a skeleton of the shape using a potential-based skeletonization approach and then generate the cross-sections with the help of seed points. But if the shape has to be decomposed into parts, each of which is a GC, the seed points have to be manually chosen.

There are a number of automatic as well as interactive surface reconstruction methods that use a GC representation [66–68]. However, they do not explicitly aim at obtaining an optimal decomposition of the shape into GCs. With regards to obtaining a GC based decomposition, the work of Goyal et al. [93] and Zhou et al. [73] are the most relevant to our work. Both these methods work on meshes unlike our method which takes unorganized point clouds as input. Goyal et al. obtain a direct parameterization of 3D meshes in terms of sets of locally prominent cross-sections (PCS). Each set of PCS represent a different sweep component. Zhou et al. propose a metric to measure cylindricity of a GC. They first extract an over-complete

set of local cylinders using the method proposed by Tagliasacchi et al. [87] and then combine these local cylinders to form longer candidate cylinders. The set of candidate cylinders is still over-complete and the partition of the shape into GCs is obtained by solving the exact cover problem. Our approach is similar to that of Zhou et al in the sense that we too first generate an overlapping set GCs and we choose the best subset of GCs that cover the shape. However, there are some significant differences between the two approaches. The most signifacnt difference being that our method operates on point clouds and not polygonal meshes. Our method to extract GCs is rooted in the very definition of a GC which is in turn based on translational symmetry. Their method of scoring the GCs is not applicable to us because we operate on point clouds. Therefore, we introduce metrics to score GCs based on factors like their self-similarity (i.e., based on how much one portion of the GC is similar to another portion of the same GC), length, straightness of axis etc. And finally, we frame the problem of choosing the final subset of GCs as a binary integer program.

4.3 Generating Candidate Parts

As shown in Fig. 4.2, a GC is formed by sweeping a planar cross-section along a 3D axis, with size scaling applied to the cross-section, as it moves along the axis. Such objects were referred to as generalized cones by Binford [69]. As per this definition, every part has an axis. This axis serves as the skeletal representation of the part. According to this definition of a part, there is a cross-sectional plane associated with each point on the axis. And as shown in Fig. 4.2 (c), the normal of the cross-sectional plane associated with a point on the axis, is the tangent to the axis at that point. This important observation will be employed later in detecting local cross-sectional planes.

Fig. 4.3, shows the various steps involved in generating candidate parts. Given a point cloud, the first step in the algorithm is to estimate the point normals for each point in the cloud. The point normals are nothing but the local surface normals.

At the end of this step, we would obtain an oriented set of points and the point normal orientations, as we show later, would play a very important role throughout the algorithm. In the next step, we derive local thresholds. The key idea here is that whenever thresholds are required to be applied (like estimating the closeness of a point in the cloud to some plane or deciding if two points in the point cloud can be considered as neighbors), locally adaptive thresholds need to be employed. For instance, the sparsity/density of the point clouds need not be uniform through out the cloud. To deal with this issue, we assign a number, which represent the local distance threshold, to each point in the cloud.

In the next step, we detect what we call "initial cross-sections". An initial cross-section is a cluster of points in the cloud that are likely to represent one of the cross-sections of a part. In Fig. 4.4, cluster 0, represents such an initial cross-section. Such clusters serve as the starting points to "grow" parts. Keep in mind that all points in the point cloud are assumed to come from parts which are generalized cylinders. Therefore, each point in the cloud corresponds to one of the cross-sectional planes (like the yellow planes in Fig. 4.2 (c)) of one of the parts. Therefore, in this step, given any point in the cloud, we would like to be able to estimate the orientation of the cross-sectional plane, passing through that point. Provided, we can find the orientation of the cross-sectional plane corresponding to a given point in the cloud, all points sufficiently close (the notion of being sufficiently close will be made more concrete later) to this cross-sectional plane would constitute what we call an initial cross-section.

Assume for a moment that given such an initial cross-section, we will be able to extract the part that the initial cross-section belongs to. For instance, in Fig. 4.4, given cluster 0, let's assume we can grow the part and extract the torso of the dinosaur. We will explain how this can be done later. Since, we would like to extract all parts of an object we would need at least as many initial cross-sections as there are parts. In the example of the dinosaur (Fig. 4.4), if we assume that the number of parts constituting the object is nine (i.e., four limbs, a tail, torso, neck, horn and face), we

need at least nine initial cross-sections to extract all the parts. Hence, we cluster the point cloud into M clusters and find the initial cross-section for each of these clusters. A large value is chosen for M (close to hundred) to ensure that we detect initial cross-sections for all the parts. It is better to have some redundant parts than to miss out on extracting some parts. As shown in Fig. 4.3, the next step is "growing" parts out

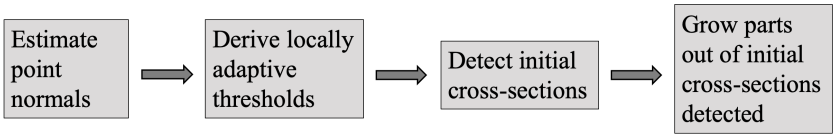


Fig. 4.3. Block diagram showing the different steps involved in generating candidate parts.

of these initial cross-sections. To understand this process consider cluster 0, in Fig. 4.4, which represents an initial cluster. Once we have such a cluster of points, we search in the neighborhood, of that cluster, for a similar group of points which could represent the neighboring cross-section of the initial cluster. Cluster 1 in Fig. 4.4 represents such a matching cluster. Note, we will make the notion of similarity/match concrete later. If we find a matching cluster, we repeat this process of searching for matching clusters. For instance, cluster 2 is found to be a match for cluster 1. Keep in mind that our parts are defined by translational symmetry. Therefore, for each cross-sectional cluster, we will find similar clusters in its neighborhood. Or in other words, the procedure just described for growing parts from initial cross-sections is inspired by the definition of parts as generalized cylinders. This process of growing a part is repeated until there are no more matches to be found. Note, the parts are to be grown, from the initial cluster, in both directions. In Fig. 4.4 , the clusters in the two directions are represented by positive and negative cluster numbers respectively.

The three fundamental components of a part is its 2D cross-sectional contour, the scaling function and the 3D Axis. Since we are dealing with point clouds, the concept of a 2D contour lying on a plane does not make sense. Hence, the initial cross-section is a cluster of points and these thin clusters play the role of the 2D cross-sectional

contour. In the growing stage we are attempting to determine matches for the thin cross-sections. Finding such matches would give us an estimate of the scaling function and also the estimate of the axis of the part. The procedure of finding matches and how such matches would lead to estimates of the scaling function and the axis would be explained in the sections below. But the key point to keep in mind is that the part detection algorithm is attempting to identify parts by using its very definition. Each of the steps in Fig. 4.3 are explained in greater detail in the sections below.

4.3.1 Estimating Point Normals

Before extracting the candidate parts, we estimate the local surface normal of the points in the 3D point cloud. We use the software CloudCompare to do this [94]. A quadratic surface is fit on to the local neighborhood to estimate the normal direction. But the orientation of these normals (i.e., whether it points towards the inside or outside of the object) is still ambiguous. To ensure that the normals are oriented in a consistent fashion, the normals are re-oriented by propagating the normal orientation starting from a random point with the help of a Minimum Spanning Tree (MST). We use $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^N\}$ to represent the point cloud. Note, $\mathbf{x}^i = (\mathbf{x}_p^i, \mathbf{x}_n^i)$, where $\mathbf{x}_p^i \in \Re^3$ represents the position of the point and $\mathbf{x}_n^i \in \Re^3$ is a unit normal representing the orientation of the normal at the point.

4.3.2 Deriving Locally-adaptive Thresholds

We use minimum spanning tree (MST) to derive local distance thresholds. The distance to neighboring points in an MST, constructed on the 3D point cloud, is a good indication of the local sparsity of the point cloud. Therefore, to derive a locally adaptive distance threshold for a point in the cloud, the distances of the point to its neighbors, as per the MST constructed on the cloud, can be used. To compute the MST, we must have an initial graph on the point cloud. Instead of considering a fully connected graph as input to MST, we construct a graph where every point

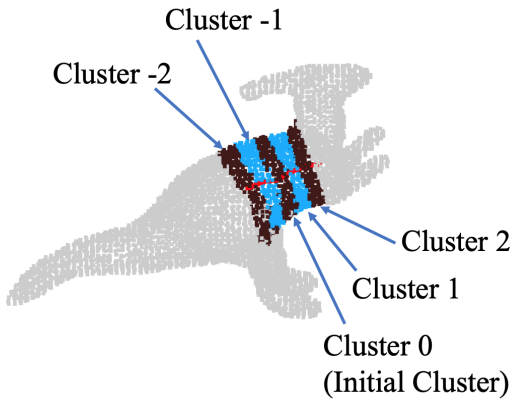


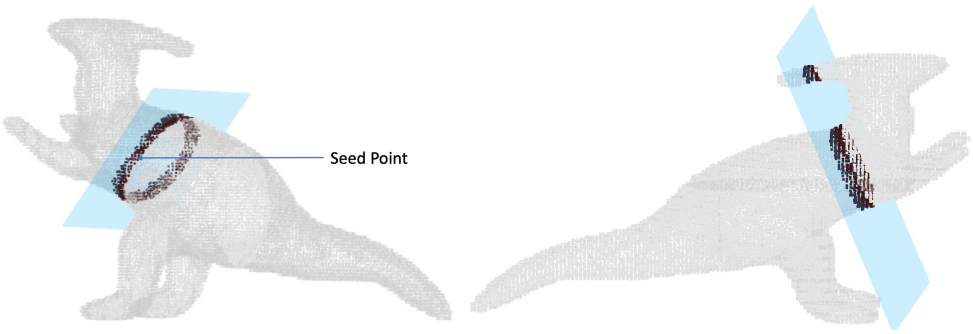
Fig. 4.4. Depiction of how a part is grown from an initial cross-sectional cluster (Cluster 0). Neighboring cross-sectional clusters are shown in alternating colors. The red curve represents the axis of the part.

is connected to its hundred nearest neighbors. Though a higher number of nearest neighbors could be considered, in practice, we found it be sufficient to just consider the nearest hundred neighbors. The reason for not considering a fully connected graph is to reduce the time for MST computation. The time complexity for MST is $\mathcal{O}(E \log V)$, where V is the number of nodes/vertices in the graph and E is the number of edges. A fully connected graph would have a time complexity of $\mathcal{O}(V^2 \log V)$, whereas considering a fixed number of nearest neighbors, say hundred in our case, will only have a time complexity of $\mathcal{O}(V \log V)$. We found that using a fixed, albeit large enough, neighborhood to create the input graph to MST produced considerable savings in computation time in practice. We use the k-d tree data structure to speed up neighborhood queries. Specifically , we use the k-d tree implementation from the Scikit-learn package [95]. Kruskal's algorithm implemented in the SciPy package [96] is used to compute MST.

As mentioned before, the locally adaptive threshold for points in the cloud is employed, for various purposes, throughout the algorithm. One purpose of the locally adaptive threshold is to create a connectivity matrix on the point cloud. Or in other words, the locally adaptive threshold derived for a point in the cloud is used to determine its neighbours. Therefore, once the MST is computed, we assign a distance threshold for each individual point in the cloud. I.e., for each point $\mathbf{x}^i$ in the point cloud, we set a distance threshold $\delta^i_{cnct} = 1.5 d^i_{max}$, where d^i_{max} is the maximum among the distances to all the neighbors the point $\mathbf{x}^i$ is connected to, in the MST. Two points $\mathbf{x}^i$ and $\mathbf{x}^j$ in the point cloud are considered to be connected if $d_{eucl}(\mathbf{x}^i, \mathbf{x}^j) \leq$ maximum$(\delta^i_{cnct}, \delta^j_{cnct})$, where $d_{eucl}(\mathbf{x}^i, \mathbf{x}^j)$ is the euclidean distance between the points. Based on this criterion a connectivity graph, $\mathbf{G}_{cnct}$, is constructed on the point cloud.

4.3.3 Detecting Initial Cross-sections

As stated earlier, parts are formed by scaled versions of a 2D cross-sectional contour translated along an axis. And since thin cross-sectional clusters serve as an

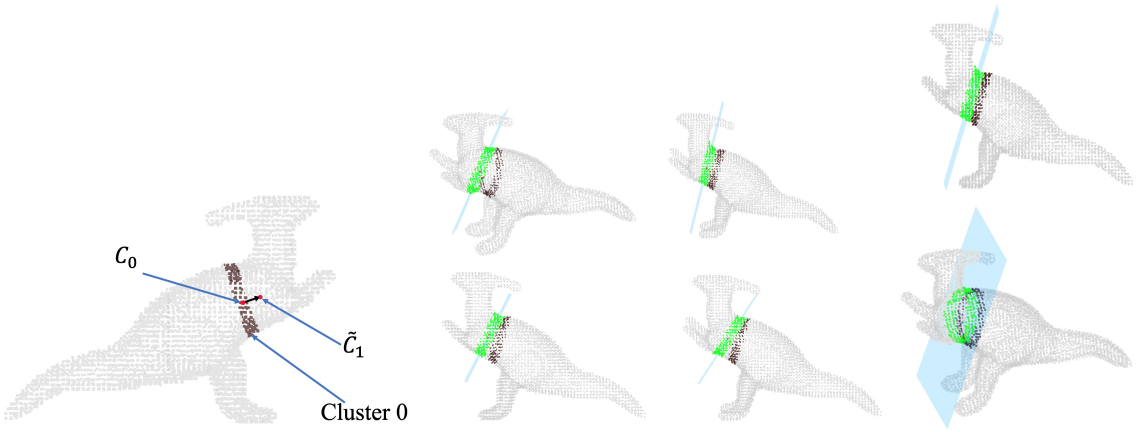


(a) (b)

Fig. 4.5. (a) The cross-sectional plane is shown in blue, the thin cross-section associated with the plane is shown in brown and seed point is shown in red. (b) All points close to the cross-sectional plane, but not necessarily connected to the seed point.

approximation to the 2D contour of a part in a point cloud, we would like to detect them and then use them to grow parts. Given any point in the cloud, we need an algorithm to detect a thin cross-sectional cluster corresponding to that point. For instance, in Fig. 4.5 (a), given the red point, we need the algorithm to return the brown points. The red point is referred to as the seed point as it serves as the starting point for growing parts. As shown in Fig. 4.5 (a), detecting the orientation of the cross-sectional plane (shown in blue) passing through the seed point would give us the thin cross-sectional cluster we are looking for. Because, once we estimate the orientation of the cross-sectional plane passing through the seed point, all points lying sufficiently close to that plane would represent the thin cross-sectional cluster. We need to consider only points lying close to the plane and at the same time connected to the seed point. As shown in Fig. 4.5 (b), considering all points near the plane that are not necessarily connected to the seed point could lead to the inclusion of some wrong points as part of the cross-section. Note that the approach described above is similar to that of Tagliasacchi et al. [87].

We need to detect initial cross-sectional clusters at multiple locations on the point cloud to be able to extract all the parts. As mentioned before, in the example of the dinosaur in Fig. 4.4, we need at least nine initial cross-sections. Therefore, the point



(a) (b) (c)

Fig. 4.6. Growing parts by method 1. (a) Step 1: take a small step along the normal of cross-sectional plane of Cluster 0 to obtain an estimate of the neighboring axis point $\widetilde{\mathbf{C}}_1$. (b) Step 2: Consider a set of planes (planes in $\mathbf{A}_\theta \bigtimes \mathbf{A}_\phi$) whose orientation is close to the orientation of the cross-sectional plane of Cluster 0. Assign a cost to each of these planes using the same cost function as the one in algorithm 4.1. Only four planes (blue color) in the set $\mathbf{A}_\theta \bigtimes \mathbf{A}_\phi$ are shown. The green points represent inliers of the corresponding plane. (c) Step 3: choose the plane from the set $\mathbf{A}_\theta \bigtimes \mathbf{A}_\phi$ which minimizes the cost computed. This plane represents the cross-sectional plane of the adjacent cross-section and the inlier set of this plane represents the adjacent cross-sectional cluster (Cluster 1 in our example). Top and bottom shows two views of the chosen plane.

cloud is clustered into M clusters using k-means clustering [95]. These clusters would determine the locations at which we would detect thin cross-sectional clusters from which candidate parts are grown. The idea is to pick one seed point per cluster and then find the thin cross-section to which the seed point belongs and finally grow parts out of this thin cross-section. For each cluster, we pick the point in the cluster which is closest to the k-means cluster center as the seed point. To find the cross-sectional plane that passes through a seed point, $\mathbf{s}^i$, we do an exhaustive search. Algorithm 4.1 shows the steps involved in searching for a cross-sectional plane that passes through a seed point. Note, the algorithm takes as input the connectivity matrix defined on the clusters, $\mathbf{G}_{clust}$. If any member of one cluster is connected to any member of the other cluster, according to $\mathbf{G}_{cnct}$, then two clusters are considered to be connected.

Spherical coordinates are used to sample the unit sphere at regular angular intervals to obtain the normal directions for the planes. Approximately 30 planes are considered and these planes are denoted by $\mathbf{A}_\theta \bigtimes \mathbf{A}_\phi$. Note, all the planes pass through the seed point, so picking a normal direction for the plane fully defines it. For each plane an inlier set, $\mathbf{B}$, is first computed. This inlier set, as shown in Fig. 4.5 (a), contains all points that lie close to the plane and are connected to the seed point. A cluster dependent distance threshold, $\delta^i_{pd} = median\big(\{d^j_{max} \mid \mathbf{x}^j \in \text{cluster } i\}\big)$, is used to determine if a point is considered close enough to a plane. We use the adaptive threshold derived earlier in section 4.3.2 here to decide which points lie close to the plane.

A cost is then assigned to each plane. The cost, represented by $c(\theta, \phi)$, is based on the defenition of the part and measures the average length of projection of the point normals of the points in $\mathbf{B}$ on to the normal of the plane. Ideally, the point normals will be perpendicular to the normal of the cross-sectional plane and therefore will have a zero length projection on the normal of the cross-sectional plane. As shown in Fig. 4.2 (c), the cross-sectional plane (shown in yellow) contains the contour. The normals to the contour lie on the plane and hence are parallel to the plane and have projection length of zero on to the normal of the plane. This implies that the plane that minimizes this cost, for a given seed point, would be the best estimate for the cross-sectional plane for that seed point. After the costs associated with each plane is computed the plane that minimizes this cost is selected as the cross-sectional plane. The inlier set of that plane defines the thin cross-section.

4.3.4 Growing Parts

In the previous step, we identified M initial cross-sectional clusters. Starting from these initial cross-sectional clusters, parts need to be grown. Two methods are used to grow parts from initial cross-sections. Both methods are again based on the definition of a part as a GC. We would refer to these methods as method 1 and method 2 for

Algorithm 4.1 Algorithm for finding the cross-sectional plane

Input:

Point cloud : $\mathbf{X}$

Connectivity graph on $\mathbf{X}$: $\mathbf{G}_{cnct}$

Connectivity graph on the clusters: $\mathbf{G}_{clust}$

Distance Threshold: δ^i_{pd}

Seed point of i^{th} cluster: $\mathbf{s}^i = (\mathbf{s}^i_p, \mathbf{s}^i_n)$

Output: θ, ϕ

function GET_INLIERS($\mathbf{X}, \mathbf{G}_{cnct}, \delta^i_{pd}, \mathbf{s}, \hat{\mathbf{n}}$)

  $d \leftarrow \hat{\mathbf{n}} \cdot \mathbf{s}^i_p$ ▷ "$\cdot$" represents dot product

  $\mathbf{A}$ is the set of points close to the plane

  abs(): absolute value

  $\mathbf{A} \leftarrow \{\mathbf{x}^i \mid \text{abs}(\ \hat{\mathbf{n}} \cdot \mathbf{x}^i_p - d\ ) \leq \delta^i_{pd}\}$

  $\mathbf{B} \leftarrow \{\mathbf{x}^i \in \mathbf{A} \mid \mathbf{x}^i$ is connected to $\mathbf{s}$ as per $\mathbf{G}_{cnct}\}$

  return B

end function

$\mathbf{H} \leftarrow \{\mathbf{s}^j \mid$ cluster j neighbor of cluster i as per $\mathbf{G}_{clust}\}$

for $\theta \in \{0, \pi/6, 2\pi/6, \cdots, 11\pi/6\}$ **do**

  for $\phi \in \{0, \pi/6, \pi/3, \pi/2\}$ **do**

    $\hat{\mathbf{n}} \leftarrow [cos(\theta)sin(\phi), sin(\theta)sin(\phi), cos(\phi)]$

    $\mathbf{B} \leftarrow$ GET_INLIERS($\mathbf{X}, \mathbf{G}_{cnct}, \delta^i_{pd}, \mathbf{s}^i, \hat{\mathbf{n}}$)

    $c(\theta, \phi) \leftarrow$mean$\big(\{\text{abs}(\hat{\mathbf{n}} \cdot \mathbf{x}^i_n) \mid \mathbf{x}^i \in \mathbf{B}\}\big)$

  end for

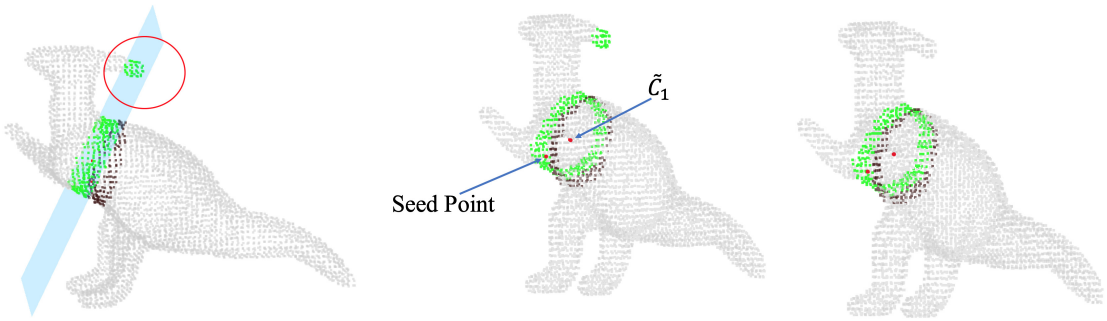
end for

return $\arg\min_{\theta,\phi} c(\theta, \phi)$

clarity. While method 1 is very simple, method 2 is much more sophisticated. Before we describe the methods, we would describe three assumptions that these methods make. The first assumption is that the scaling function of the GC is smooth and the second assumption is that the 3D axis of the GC is smooth. Note that we are looking for matching cross-sectional cluster for the initial cross-sectional cluster in its neighborhood to grow a part. In this context, the first assumption implies that there would not be a sudden jump in the scale of a cross-sectional cluster compared to its neighboring cross-sectional cluster. And the second assumption would mean that the orientation of the cross-sectional planes of neighboring cross-sectional clusters would be similar. As an example, in Fig. 4.2 (c), the two yellow planes represent cross-sectional planes of two close-by cross-sections and therefore have similar orientation. Also, note that the scale function in Fig. 4.2 (a) is smooth. The third assumption that both methods make is that the mean of all the points in a cross-sectional cluster of a part represents a point on the axis of the part. For instance, in Fig. 4.5 (a), if we take the centre of mass of all brown points, we would obtain a point in the interior of the torso of the dinosaur. The torso of the dinosaur is a GC and the centre of mass of the points in the cross-sectional cluster (shown in brown) is assumed to lie on the axis of this GC. We refer to the centre of mass of points in the cluster as the cluster center. The axis of the torso, shown in red in Fig. 4.4, is obtained by joining the cluster centers of the blue and brown clusters. While this assumption is not generally true, we make such a simplifying assumption to estimate the axis points, given a cross-sectional cluster. We will demonstrate later that this assumption is a reasonable assumption for many real-world parts. The following sections will explain the two methods in detail.

Method 1

As stated above, this is a simple method for growing parts from initial cross-sectional clusters. Keep in mind that, given an initial cross-sectional cluster, we are



(a) (b) (c)

Fig. 4.7. Identifying a seed point. (a) All points lying close to the plane are shown in green. The brown points represent members of cluster 0. The points within the red circle are unwanted points, not part of the true cross-sectional cluster. (b) A seed point is identified as the point closest to $\widetilde{\mathbf{C}}_1$, from among the points lying close to the plane. The seed point and $\widetilde{\mathbf{C}}_1$ are shown in red. (c) Considering only points connected to the seed point removes the unwanted points and gives us the right cross-sectional cluster.

looking for the neighboring cross-sectional cluster to grow the part. For instance, in Fig. 4.4, we are looking for cluster 1, given the initial cluster, cluster 0. Fig. 4.6, shows the main steps in this method. As depicted in the figure, the first step is to obtain a rough estimate of the cluster center of the neighboring cluster, cluster 1. Let $\mathbf{C}_0$ denote the centre of cluster 0 and the normal of the cross-sectional plane corresponding to cluster 0 be denoted by $\hat{\mathbf{n}}_0$. By definition, the cross-sectional plane of cluster 0 is perpendicular to the axis of the GC (the torso of the dinosaur being the GC here) at $\mathbf{C}_0$. Or in other words, $\hat{\mathbf{n}}_0$ is the tangent to the axis of the GC at $\mathbf{C}_0$. Therefore, taking a small step along $\hat{\mathbf{n}}_0$ would give us a rough estimate of the axis point corresponding to cluster 1. Keep in mind that the tangent of a curve provides a linear approximation to the curve locally. Let this estimate of the neighboring axis point be denoted by $\widetilde{\mathbf{C}}_1$. I.e., $\widetilde{\mathbf{C}}_1 = \mathbf{C}_0 + \delta_{step}\hat{\mathbf{n}}_0$, where, δ_{step} represents a small step size. Fig. 4.6 (a), depicts this process.

In the next step we estimate the orientation of the cross-sectional plane through $\widetilde{\mathbf{C}}_1$. An exhaustive, but local search is used to estimate the orientation of the cross-sectional plane. The exhaustive search is identical to the one described in algorithm

4.1, but here we only consider planes whose normal orientation is close to $\hat{\mathbf{n}}_0$. This is because, according to the second assumption, the orientation of the cross-sectional plane of cluster 1 is close to the orientation of the cross-sectional plane of cluster 0 (because the 3D axis is smooth). To obtain planes whose orientation is close to $\hat{\mathbf{n}}_0$, an angle range, Δ_{ang}, and a step number, k_{step}, is first chosen. Let θ_0 and ϕ_0 represent the azimuth and the zenith angles corresponding to $\hat{\mathbf{n}}_0$. Let $\mathbf{A}_\theta$ represent k_{step} equally spaced angles in the range $(\theta_0 - \Delta_{ang}, \theta_0 + \Delta_{ang})$ and $\mathbf{A}_\phi$ represent k_{step} equally spaced angles in the range $(\phi_0 - \Delta_{ang}, \phi_0 + \Delta_{ang})$. All planes represented by the cartesian product, $\mathbf{A}_\theta \bigtimes \mathbf{A}_\phi$, are considered. We use $\Delta_{ang} = 12.5$ and $k_{step} = 3$. This ensures that all the planes considered have normals which are similar in orientation to $\hat{\mathbf{n}}_0$. The best plane is chosen by using the same criterion as the one described in algorithm 4.1. I.e., for each plane in the set $\mathbf{A}_\theta \bigtimes \mathbf{A}_\phi$, we compute its inlier set (see algorithm 4.1) and then choose the plane which minimizes the average length of projection of the point normals of the points in its inlier set on to the normal of the plane.

Considering all points lying close to a plane could lead to problems like the one shown in Fig. 4.7 (a). To avoid this issue, we need to only consider points lying close to the plane and connected to a seed point. As shown in Fig. 4.7 (b), the point closest to $\widetilde{\mathbf{C}}_1$, from amongst the points close to the plane (the green points), is chosen as the seed point. Choosing only those (green) points that are connected to the seed point would fix the issue of unwanted points being included in the cross-sectional cluster (Fig. 4.7 (c)).

The steps described above can be used to find the neighboring cross-sectional cluster, cluster 1, given the initial cluster. This process can be repeated to obtain the neighboring cross-sectional cluster, cluster 2 in Fig. 4.4, of cluster 1. There needs to be a stopping criterion for this process. Or in other words, there needs to be some criterion that can be checked to see if we have reached the end cluster of a part. Here is where the first assumption, mentioned earlier, comes into play. Since, the scale function of a part is smooth, the two neighboring cross-sections would have similar scale factors. A sudden jump in the scale factor of a neighboring cross-section would

indicate that we have reached a junction. For example, in Fig. 4.4, the search for the neighboring cross-sectional cluster for cluster 2 would fail as all the potential clusters considered would have a substantially different scale factor.

A very simple metric is employed to measure the scale factor of a cross-sectional cluster. A local coordinate system is constructed, using PCA, for the cluster. Let the axes of this coordinate system be represented by $\mathbf{a}_1, \mathbf{a}_2$ and $\mathbf{a}_3$, with $\mathbf{a}_1$ representing the maximum variance direction and $\mathbf{a}_3$ representing the direction with the least variance. The axis $\mathbf{a}_3$, would have an orientation that is very close to the normal of the cross-sectional plane of the cross-sectional cluster because the cross-sectional cluster is thin and close to planar. Let the eigenvalues corresponding to axes $\mathbf{a}_1$ and $\mathbf{a}_2$ be e_1 and e_2 respectively. These eigenvalues serve as a good representation for the scale of the cluster. I.e., bigger cross-sectional clusters would have larger e_1 and e_2 values and vice-versa. Let i and j represent two neighboring cross-sections. We treat the combination of the two eigenvalues, $[e_1, e_2]$, as a vector. I.e., for cross-section i, we have a vector $[e_1^i, e_2^i]$ representing its scale. Similarly the vector $[e_1^j, e_2^j]$, represents the scale of cross-section j. We stop growing parts, if $abs\big(1 - \frac{d_{eucl}([e_1^i, e_2^i],[e_1^j, e_2^j])}{\left\|[e_1^i, e_2^i]\right\|_2}\big) > \Delta_{eg}$, where $d_{eucl}([e_1^i, e_2^i], [e_1^j, e_2^j])$ represents the euclidean distance between the vectors, $abs()$ represents the absolute value and Δ_{eg} represents an acceptable threshold for scale difference between neighboring clusters. The fraction is measuring the percentage change in the scale between the cross-sectional clusters. A value of one for the fraction would mean that the size of the cross-section did not change (according to our metric). If the percentage change in scale of the next cross-sectional cluster compared to the current cross-sectional cluster is greater than a threshold, it implies that we have reached a junction and the growing process stops. The growing process also terminates if there are no further points to be added to the part. For instance, the growing process of the tail of the dinosaur halts in one of the directions due to this reason. The growth of the tail stops in the other direction because the junction where the legs meet the tail is detected.

As mentioned earlier, this method is a very simple one in which we depend on the smoothness of the scale function to grow parts. The termination criterion for the growth process as well as the metric to compute the scale of a cross-sectional cluster are also simple. In the next section, we describe a sophisticated method, which we refer to as method 2, to grow parts which directly takes inspiration from the process in which GCs are formed from its components (scale function, cross-sectional contour and 3D axis). Though we use method 2 as the primary method to grow parts, there are scenarios in which method 1 would be more suitable. Such scenarios will also be pointed out in the following sections.

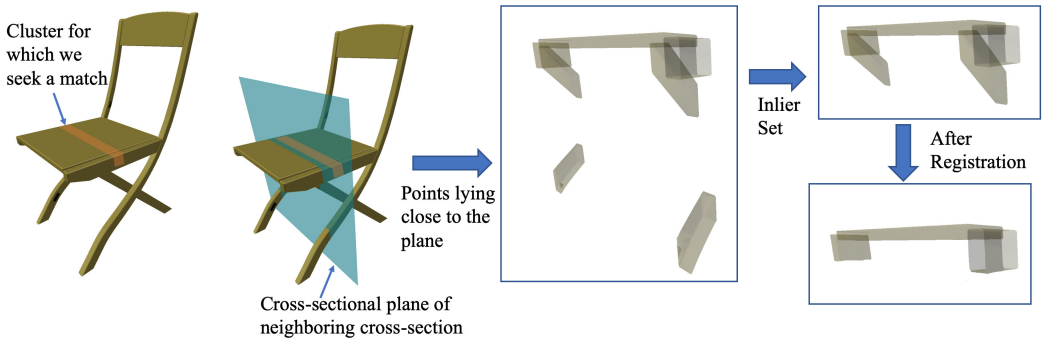


Fig. 4.8. Registration can help in removing some unwanted points from the neighboring cross-section.

Method 2

Given a scale function, cross-sectional contour and 3D axis, a GC is formed by a similarity transformation (i.e., rigid transformation and uniform scaling) of the contour. I.e., the action referred to as sweeping the 2D contour along an axis, consists of three transformations (translation, rotation and uniform scaling) applied to the 2D contour. This implies that the relationship between adjacent cross-sections is defined by a similarity transformation. In method 2, like in method 1, parts are grown, starting from an initial cross-sectional cluster (like cluster 0 in Fig. 4.4), by finding its neighboring cross-sectional cluster. Since, the relationship between neighboring

clusters is defined by a similarity transformation, one way to search for a similar cluster of points is to do registration between the initial cross-section and the points in its neighborhood. I.e., through registration, we can estimate the similarity transform that relates the two neighboring cross-sections. In Fig. 4.4, registration between cluster 0 and the points in its neighborhood would lead to the identification of cluster 1 as its neighbor. Similarly registration between cluster 1 and its neighborhood points would identify cluster 2 and so on.

Fig. 4.8, demonstrates why method 2, which relies on registration to grow parts, is necessary to discover meaningful parts in the point cloud. The saffron colored portion represents the cross-section for which we seek to find a match. The plane shown in blue represents the actual cross-sectional plane of the neighboring cross-section.The points lying close to the cross-sectional plane, include many unwanted points as shown. Deriving a seed point (like in Fig. 4.7 (b)) and looking for the inlier set (alll points connected to the seed point) will help remove some unwanted points like the portions of the chair's front legs. But portions of the chair's hind legs still remain in the cluster. Performing a registration would tell us that these points belonging to hind legs have no match in the cluster for which we seek a match. Such points for which we fail to find a match can be removed to exactly identify points belonging to the neighboring cross-section.

In [74], Myronenko and Song present a solution to the registration problem by treating it as a probability density estimation problem. Note, though the authors call their method rigid registration, their method actually estimates the similarity transform that relates the two point clouds being registered. Myronenko and Song, however, only consider the location of the points being registered. In our case though, we have an oriented point set with each point having a position and an orientation (representing the local surface normal). As we demonstrate later, considering the orientation of the points is important in our case. Therefore, for the registration of two sets of oriented points, we modify the method in [74] by drawing in ideas from [75]. We first introduce some additional notation and then explain the registration process

Algorithm 4.2 Algorithm to estimate parameters $\mathbf{R}$, s, $\mathbf{t}$,α and σ

Input:

Point set 1: $\mathbf{X}$

Point set 2: $\mathbf{Y}$

Output: $\mathbf{R}$, s, $\mathbf{t}$, α, σ

Set $\mathbf{R} = \mathbf{I}_3$, $s = 1$ and $\mathbf{t} = (0, 0, 0)^T$

while convergence not achieved **do**

Compute $\mathbf{P}$:

$$p_{ji} = \frac{exp\Big(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_p^i - s\mathbf{R}\ \mathbf{y}_p^j - \mathbf{t}\right\|_2^2 + \alpha(\mathbf{x}_n^i)^T\mathbf{R}\mathbf{y}_n^j\Big)}{\sum_{m=1}^{M} exp\Big(-\frac{1}{2\sigma^2}\left\|\mathbf{x}_p^i - s\mathbf{R}\ \mathbf{y}_p^m - \mathbf{t}\right\|_2^2 + \alpha(\mathbf{x}_n^i)^T\mathbf{R}\mathbf{y}_n^m\Big)}$$

Estimate the parameter values, θ, using the above value of $\mathbf{P}$:

$$\hat{\mathbf{X}} = \mathbf{X}_p - \tfrac{1}{N}\mathbf{1}\big(\mathbf{X}_p^T\mathbf{P}\mathbf{1}\big)^T$$

$$\hat{\mathbf{Y}} = \mathbf{Y}_p - \tfrac{1}{N}\mathbf{1}\big(s\mathbf{R}\mathbf{Y}_p^T\mathbf{P}\mathbf{1}\big)^T$$

$$\mathbf{A} = \hat{\mathbf{X}}^T\mathbf{P}^T\hat{\mathbf{Y}}$$

$$\mathbf{B} = \mathbf{X}_n^T\mathbf{P}^T\mathbf{Y}_n$$

$$K_1 = tr\big(\hat{\mathbf{Y}}^T d\big(\mathbf{P}\mathbf{1}\big)\hat{\mathbf{Y}}\big)$$

$$K_2 = tr\big(\hat{\mathbf{X}}^T d\big(\mathbf{P}\mathbf{1}\big)\hat{\mathbf{X}}\big)$$

Use the BFGS algorithm with the gradient equations defined in equation 4.16, to compute optimal values for $\mathbf{R}$, s, α and σ.

$$\mathbf{t} = \tfrac{1}{N}\big(\mathbf{X}_p^T\mathbf{P}\mathbf{1} - s\mathbf{R}\mathbf{Y}_p^T\mathbf{P}\mathbf{1}\big)$$

end while

return $\mathbf{R}$, s, $\mathbf{t}$, α, σ

between two oriented sets of points in the next section. Later sections explain how this registration process can be employed to grow parts.

Registartion of Two Oriented Point Sets

Let $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^N\}$ and $\mathbf{Y} = \{\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^M\}$ represent the two point sets to be registered. As mentioned before, $\mathbf{x}^i = (\mathbf{x}^i_p, \mathbf{x}^i_n)$, where $\mathbf{x}^i_p$ represents the location and $\mathbf{x}^i_n$ represent the point normal of $\mathbf{x}^i$. Similarly, $\mathbf{y}^i = (\mathbf{y}^i_p, \mathbf{y}^i_n)$. Let $\mathbf{X}_p = (\mathbf{x}^1_p, \mathbf{x}^2_p, \cdots, \mathbf{x}^N_p)^T$ be a $N \times 3$ matrix representing the positions and $\mathbf{X}_n = (\mathbf{x}^1_n, \mathbf{x}^2_n, \cdots, \mathbf{x}^N_n)^T$ be a $N \times 3$ matrix representing the point normals of $\mathbf{X}$. Similarly, let $\mathbf{Y}_p$ and $\mathbf{Y}_n$ represent the positions and point normals of points in $\mathbf{Y}$ respectively. The diagonal matrix formed from a vector $\mathbf{v}$ is represented by $diag(\mathbf{v})$. A column vector of all ones is represented by $\mathbf{1}$ and a $P \times P$ identity matrix is represented by $\mathbf{I}_P$. The trace of a matrix $\mathbf{A}$ is represented by $tr(\mathbf{A})$.

In [74], the authors consider $\mathbf{Y}$ as the centroids of a Gaussian mixture model (GMM) and $\mathbf{X}$ as the data points generated by the GMM. The probability density function for the GMM is then given by:

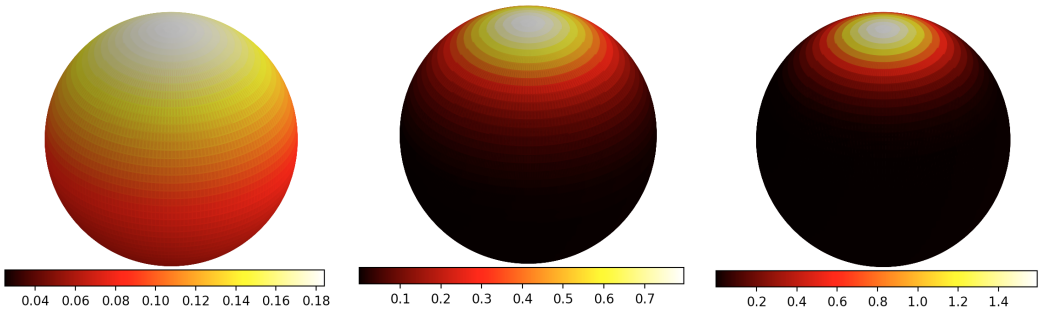
$$p(\mathbf{x}) = \sum_{j=1}^{M} P(j) p(\mathbf{x} \mid j) \tag{4.1}$$

where $p(\mathbf{x} \mid j) = \frac{1}{(2\pi\sigma^2)^{3/2}} exp\big(-\frac{1}{2\sigma^2} \big\| \mathbf{x}_p - \mathbf{y}^j_p \big\|_2^2 \big)$ and $P(j) = \frac{1}{M}$. In our case, we need the density to also depend on the point normal orientations. Therefore, we modify $p(\mathbf{x} \mid j)$ with an additional term that depends on the orientation as shown below:

$$p(\mathbf{x} \mid j) = \frac{\alpha \; exp\big(\alpha \mathbf{x}^T_n \mathbf{y}^j_n\big)}{2\pi(exp(\alpha) - exp(-\alpha))} \frac{exp\big(-\frac{1}{2\sigma^2} \big\| \mathbf{x}_p - \mathbf{y}^j_p \big\|_2^2 \big)}{((2\pi\sigma^2)^{3/2})} \tag{4.2}$$

The above distribution is obtained by combining the original Gaussian distribution with another distribution, referred to as the Von Mises-Fisher distribution in directional statistics. The idea of combining these two distributions was first introduced

by Billings and Taylor in [75]. However, they do not involve a scaling parameter in their transformation. Since, our definition of a part involves uniform scaling, we cannot leave out the scale factor parameter from the formulation. Unlike [74], we do not include a uniform distribution to account for noise. Fig. 4.9, shows the effect of the value of α on the distribution. We place an upper limit of ten on the value of α to maintain a good balance between the importance of position and orientation of points.



(a) $\alpha = 1$ (b) $\alpha = 5$ (c) $\alpha = 10$

Fig. 4.9. The effect of the value of α on the Von Mises-Fisher distribution. The location of the north pole on the sphere represents the "mean direction". The values on the color-bar represent probability density values. The greater the value of α, the greater is the concentration of the distribution around the mean direction.

The central idea behind this approach to registration is to parameterize the distribution defined by $\mathbf{Y}$ with a rotation matrix $\mathbf{R}$, uniform scale factor s and a translation vector $\mathbf{t}$, and then estimate these parameters using maximum likelihood estimation (MLE) by treating the point set $\mathbf{X}$ as the observed data. After the parameterization, $p(\mathbf{x} \mid j)$ is given by:

$$p(\mathbf{x} \mid j) = \frac{\alpha \; exp\big(- \frac{1}{2\sigma^2} \big\| \mathbf{x}_p - s\mathbf{R} \; \mathbf{y}_p^j - \mathbf{t} \big\|_2^2 + \alpha \mathbf{x}_n^T \mathbf{R} \mathbf{y}_n^j \big)}{2\pi (exp(\alpha) - exp(-\alpha))(2\pi\sigma^2)^{3/2}} \tag{4.3}$$

Like in [74], we define $P(j \mid \mathbf{x}^i) = \frac{P(j)p(\mathbf{x}^i|j)}{p(\mathbf{x}^i)}$ as the correspondence probability between points $\mathbf{x}^i$ and $\mathbf{y}^j$. Using the i.i.d data assumption, the negative log-likelihood is given by:

$$\mathcal{L}(\theta) = \mathcal{L}(\mathbf{R}, s, \mathbf{t}, \sigma, \alpha) = -\sum_{i=1}^{N} log\left(\sum_{j=1}^{M} P(j)p(\mathbf{x}^i \mid j)\right) \tag{4.4}$$

Where,

$$p(\mathbf{x}^i \mid j) = \frac{\alpha\ exp\big(-\frac{1}{2\sigma^2}\big\|\mathbf{x}_p^i - s\mathbf{R}\ \mathbf{y}_p^j - \mathbf{t}\big\|_2^2 + \alpha(\mathbf{x}_n^i)^T\mathbf{R}\mathbf{y}_n^j\big)}{2\pi(exp(\alpha) - exp(-\alpha))(2\pi\sigma^2)^{3/2}} \tag{4.5}$$

The Expectation Maximization (EM) algorithm is used to estimate the parameters that minimize the negative log-likelihood. The algorithm alternates between the E-step and the M-step until convergence. In the E-step, the posterior probability of the mixture components are computed as shown below:

$$P^k(j \mid \mathbf{x}^i) = \frac{exp\Big(-\frac{1}{2\sigma^2}\big\|\mathbf{x}_p^i - \mathcal{T}_p(\mathbf{y}_p^j, \theta^{(k-1)})\big\|_2^2 + \alpha(\mathbf{x}_n^i)^T\mathcal{T}_n(\mathbf{y}_n^j, \theta^{(k-1)})\Big)}{\sum_{m=1}^{M} exp\Big(-\frac{1}{2\sigma^2}\big\|\mathbf{x}_p^i - \mathcal{T}_p(\mathbf{y}_p^m, \theta^{(k-1)})\big\|_2^2 + \alpha(\mathbf{x}_n^i)^T\mathcal{T}_n(\mathbf{y}_n^m, \theta^{(k-1)})\Big)} \tag{4.6}$$

Where $\mathcal{T}_p(\mathbf{y}_p^j, \theta^{(k-1)}) = s^{(k-1)}\mathbf{R}^{(k-1)}\mathbf{y}_p^j - \mathbf{t}^{(k-1)}$ and $\mathcal{T}_n(\mathbf{y}_n^j, \theta^{(k-1)}) = \mathbf{R}^{(k-1)}\mathbf{y}_n^j$. The superscript $(k-1)$ indicates that the parameter values being used are the estimates from the previous iteration of the EM algorithm. In the M-step, the parameters need to be estimated by minimizing complete-data negative log-likelihood Q, which is given by:

$$Q = -\sum_{i=1}^{N}\sum_{j=1}^{M} P^k(j \mid \mathbf{x}^i) log\big(P(j)p(\mathbf{x}^i \mid j)\big) \tag{4.7}$$

Considering only terms that depend on θ, we can rewrite Q as:

$$\begin{aligned} Q(\theta) = \Bigg(\sum_{i=1}^{N}\sum_{j=1}^{M} P^k(j \mid \mathbf{x}^i)\Big(\frac{1}{2\sigma^2}\big\|\mathbf{x}_p^i - s\mathbf{R}\ \mathbf{y}_p^j - \mathbf{t}\big\|_2^2 - \alpha(\mathbf{x}_n^i)^T\mathbf{R}\mathbf{y}_n^j\Big)\Bigg) + \\ \frac{3N}{2}\log(\sigma^2) - N\log(\alpha)\ + N\log\big(exp(\alpha) - exp(-\alpha)\big) \end{aligned} \tag{4.8}$$

Like in [74], taking the partial derivative of $Q(\theta)$ with respect to $\mathbf{t}$ and setting it to zero gives:

$$\mathbf{t} = \frac{1}{N}\big(\mathbf{X}_p^T\mathbf{P}\mathbf{1} - s\mathbf{R}\mathbf{Y}_p^T\mathbf{P}\mathbf{1}\big)$$

where $\mathbf{P}$ is a M $\times$ N matrix with elements $p_{ji} = P^k(j \mid \mathbf{x}^i)$. Substituting $\mathbf{t}$ back into the equation (4.7) and setting $\hat{\mathbf{X}} = \mathbf{X}_p - \frac{1}{N}\mathbf{1}\big(\mathbf{X}_p^T\mathbf{P}\mathbf{1}\big)^T$ and $\hat{\mathbf{Y}} = \mathbf{Y}_p - \frac{1}{N}\mathbf{1}\big(s\mathbf{R}\mathbf{Y}_p^T\mathbf{P}\mathbf{1}\big)^T$, we get:

$$
\begin{aligned}
Q(\theta) = \frac{1}{2\sigma^2}\big[tr\big(\hat{\mathbf{X}}^T d\big(\mathbf{P}^T\mathbf{1}\big)\hat{\mathbf{X}}\big) - 2s\ tr\big(\hat{\mathbf{X}}^T\mathbf{P}^T\hat{\mathbf{Y}}\mathbf{R}^T\big) \\
+\ s^2 tr\big(\hat{\mathbf{Y}}^T d\big(\mathbf{P}\mathbf{1}\big)\hat{\mathbf{Y}}\big)\big] - \alpha\ tr\big(\mathbf{X}_n^T\mathbf{P}^T\mathbf{Y}_n\mathbf{R}^T\big) \\
+ \frac{3N}{2}\log(\sigma^2) - N\log(\alpha) + N\log\big(exp(\alpha) - exp(-\alpha)\big)
\end{aligned}
$$

Let $\mathbf{A} = \hat{\mathbf{X}}^T\mathbf{P}^T\hat{\mathbf{Y}}$ and $\mathbf{B} = \mathbf{X}_n^T\mathbf{P}^T\mathbf{Y}_n$. Using the invariance of trace under cyclic matrix permutation, the fact that $\mathbf{R}$ is orthogonal, we can rewrite $Q(\theta)$ as:

$$
\begin{aligned}
Q(\theta) = \frac{1}{2\sigma^2}\big[tr\big(\hat{\mathbf{X}}^T d\big(\mathbf{P}^T\mathbf{1}\big)\hat{\mathbf{X}}\big) - 2s\ tr\big(\mathbf{A}^T\mathbf{R}\big) \\
+ s^2 tr\big(\hat{\mathbf{Y}}^T d\big(\mathbf{P}\mathbf{1}\big)\hat{\mathbf{Y}}\big)\big] - \alpha\ tr\big(\mathbf{B}^T\mathbf{R}\big) + \frac{3N}{2}\log(\sigma^2) \\
- N\log(\alpha) + N\log\big(exp(\alpha) - exp(-\alpha)\big)
\end{aligned}
\tag{4.9}
$$

The form of equation (4.9) does not allow us to directly apply the method in [74] to estimate $\mathbf{R}$. Therefore, we use the BFGS algorithm to compute the parameters. For $\mathbf{R}$ to be a proper rotation matrix, it has to satisfy two constraints. I.e., $\mathbf{R}$ has to be an orthogonal matrix whose determinant is one. To avoid having to deal with these constraints during optimization, a reparameterization of $\mathbf{R}$ is necessary. We use the quaternion based parameterization of $\mathbf{R}$ based on [97]. In [97], the authors describe two different ways to parameterize a rotation matrix. We use the second method which stereographically projects a 3D hyperplane onto the 4D unit quaternion sphere. Such a parameterization not only removes the constraints but also provide rational expressions for the derivative of R, which becomes significant when iterative methods like BFGS are employed during optimization. The relevant equations from [97] are provided below.

The rotation matrix corresponding to an unit quaternion $\mathbf{q} = [q_0, q_1, q_2, q_3]$ is given by:

$$
\mathbf{R}(\mathbf{q}) = \begin{bmatrix}
q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\
2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\
2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2
\end{bmatrix}
\tag{4.10}
$$

The partial derivatives of the rotation matrix with respect to the components of the quaternion is given by:

$$
\begin{aligned}
\frac{\partial \mathbf{R}(\mathbf{q})}{\partial q_0} &= 2 \begin{bmatrix} q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \\
\frac{\partial \mathbf{R}(\mathbf{q})}{\partial q_1} &= 2 \begin{bmatrix} q_1 & q_2 & q_3 \\ q_2 & -q_1 & -q_0 \\ q_3 & q_0 & -q_1 \end{bmatrix} \\
\frac{\partial \mathbf{R}(\mathbf{q})}{\partial q_2} &= 2 \begin{bmatrix} -q_2 & q_1 & q_0 \\ q_1 & q_2 & q_3 \\ -q_0 & q_3 & -q_2 \end{bmatrix} \\
\frac{\partial \mathbf{R}(\mathbf{q})}{\partial q_3} &= 2 \begin{bmatrix} -q_3 & -q_0 & q_1 \\ q_0 & -q_3 & q_2 \\ q_1 & q_2 & q_3 \end{bmatrix}
\end{aligned}
\tag{4.11}
$$

As mentioned earlier, a stereographic projection is employed to project each point, $[x, y, z]$, on a 3D hyperplane onto a 4D unit quaternion sphere (see Figure 1 in [97]). This mapping provides a way to represent a unit quaternion using a 3D vector. Given a point, $[x, y, z]$, on the hyperplane, the corresponding unit quaternion is given by:

$$
\begin{aligned}
\beta^2 &= x^2 + y^2 + z^2 \\
\mathbf{q} &= \frac{1}{(\beta^2 + 1)} \Big(2x, 2y, 2z, 1 - \beta^2 \Big)
\end{aligned}
\tag{4.12}
$$

The projection of a unit quaternion $\mathbf{q} = [q_0, q_1, q_2, q_3]$ onto the hyperplane is given by:

$$
\begin{aligned}
\beta^2 &= \frac{1 - q_3}{1 + q_3} \\
\big(x, y, z\big) &= \frac{\beta^2 + 1}{2} \big(q_0, q_1, q_2\big)
\end{aligned}
\tag{4.13}
$$

The partial derivatives of the quaternions with respect to the point $\psi = (x, y, z)$ is given by:

$$\frac{\partial \mathbf{q}(\psi)}{\partial x} = \frac{1}{(\beta^2+1)^2}\begin{bmatrix} 2(\beta^2+1)-4x^2 \\ -4xy \\ -4xz \\ -4x \end{bmatrix}$$
$$\frac{\partial \mathbf{q}(\psi)}{\partial y} = \frac{1}{(\beta^2+1)^2}\begin{bmatrix} -4xy \\ 2(\beta^2+1)-4y^2 \\ -4yz \\ -4y \end{bmatrix} \tag{4.14}$$
$$\frac{\partial \mathbf{q}(\psi)}{\partial y} = \frac{1}{(\beta^2+1)^2}\begin{bmatrix} -4xz \\ -4yz \\ 2(\beta^2+1)-4z^2 \\ -4z \end{bmatrix}$$

The partial derivative of the rotation matrix, $\mathbf{R}$, with respect to ψ can now be written as:

$$\frac{\partial \mathbf{R}(\mathbf{q}(\psi))}{\partial x} = \sum_{j=0}^{3} \frac{\partial \mathbf{R}(\mathbf{q})}{\partial q_j}\frac{\partial q_j}{\partial x}$$
$$\frac{\partial \mathbf{R}(\mathbf{q}(\psi))}{\partial y} = \sum_{j=0}^{3} \frac{\partial \mathbf{R}(\mathbf{q})}{\partial q_j}\frac{\partial q_j}{\partial y} \tag{4.15}$$
$$\frac{\partial \mathbf{R}(\mathbf{q}(\psi))}{\partial z} = \sum_{j=0}^{3} \frac{\partial \mathbf{R}(\mathbf{q})}{\partial q_j}\frac{\partial q_j}{\partial z}$$

The partial derivatives required for optimizing $Q(\theta)$ using the BFGS algorithm are provided below:

$$
\begin{aligned}
K_1 &= tr\big(\hat{\mathbf{Y}}^T d\big(\mathbf{P1}\big)\hat{\mathbf{Y}}\big) \\
K_2 &= tr\big(\hat{\mathbf{X}}^T d\big(\mathbf{P}^T\mathbf{1}\big)\hat{\mathbf{X}}\big) \\
\frac{\partial Q(\theta)}{\partial \sigma} &= \frac{-1}{\sigma^3}\big(K_2 - 2s\ tr(\mathbf{A}^T\mathbf{R}) + s^2 K_1\big) + \frac{3N}{\sigma} \\
\frac{\partial Q(\theta)}{\partial s} &= \frac{-1}{\sigma^2} tr(\mathbf{A}^T\mathbf{R}) + \frac{s}{\sigma^2} K_1 \\
\frac{\partial Q(\theta)}{\partial \alpha} &= -tr(\mathbf{B}^T\mathbf{R}) - \frac{N}{\alpha} + N\frac{exp(\alpha) + exp(-\alpha)}{exp(\alpha) - exp(-\alpha)} \\
\frac{\partial Q(\theta)}{\partial x} &= \frac{-s}{\sigma^2} tr(\mathbf{A}^T\frac{\partial \mathbf{R}}{\partial x}) - \alpha\ tr(\mathbf{B}^T\frac{\partial \mathbf{R}}{\partial x}) \\
\frac{\partial Q(\theta)}{\partial y} &= \frac{-s}{\sigma^2} tr(\mathbf{A}^T\frac{\partial \mathbf{R}}{\partial y}) - \alpha\ tr(\mathbf{B}^T\frac{\partial \mathbf{R}}{\partial y}) \\
\frac{\partial Q(\theta)}{\partial z} &= \frac{-s}{\sigma^2} tr(\mathbf{A}^T\frac{\partial \mathbf{R}}{\partial z}) - \alpha\ tr(\mathbf{B}^T\frac{\partial \mathbf{R}}{\partial z})
\end{aligned}
\tag{4.16}
$$

The steps involved in the registration algorithm are summarized in algorithm 4.2.

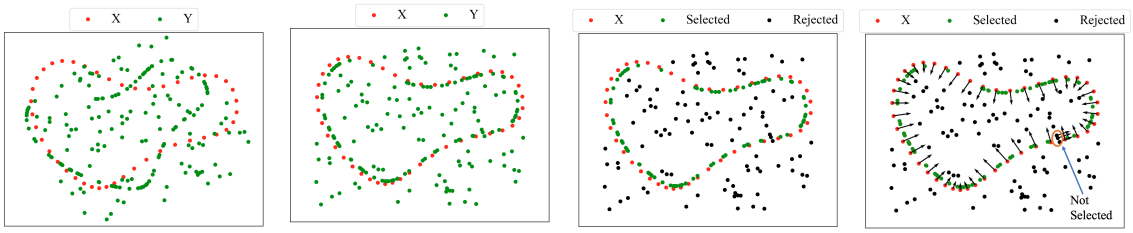
Growing Parts Using Registration

In method 2, like in method 1, given an initial cross-sectional cluster, we grow parts by finding its neighboring cross-sectional cluster. Method 2 uses registration to find the neighboring cross-sectional cluster. From the definition of a part as a GC, we know that two cross-sectional clusters of a part are related to each other by a similarity transformation. In method 2, we use registration to estimate this transformation. For instance, there exists a rotation $\mathbf{R}$, uniform scale factor s and a translation $\mathbf{t}$ that can transform cluster 0, in Fig. 4.4, to cluster 1. Looking at the two cross-sectional clusters, we can see that the rotation matrix in this case would be close to the identity matrix, the scale factor would be close to, but a little less than, one and the translation vector would point from the center of cluster 0, towards the center of cluster 1. The registration algorithm takes two point sets as its input. One of the point sets, $\mathbf{X}$, would be the cluster for which we are seeking a match. In our example, this would be cluster 0. The idea is to now provide points in the

neighborhood of cluster 1 to the registration algorithm as the second point set, $\mathbf{Y}$. The registration algorithm would then find matches for points in cluster 0 in its neighborhood leading to the detection of cluster 1. In method 1, we did consider a group of planes whose orientation is close to the orientation of the cross-sectional plane of cluster 0. The second point set is formed by all points that are part of the inlier set of any of these planes (belonging to the set $\mathbf{A}_\theta \bigtimes \mathbf{A}_\phi$). As illustrated in Fig. 4.7 (b), these inlier points of various planes in set $\mathbf{A}_\theta \bigtimes \mathbf{A}_\phi$ will contain points in cluster 1. Some additional point, not part of cluster 1, are also contained in the inlier sets. But the registration algorithm is capable of dealing with the presence of some additional points.

Given cluster 0 and the points in its neighborhood, the registration algorithm estimates the parameters, θ. After registration, we still have to choose points from the second point set $\mathbf{Y}$ to form the neighboring cluster, cluster 1. Fig. 4.10, illustrates the procedure of identifying points belonging to neighboring cluster, post registration. A 2D illustration is used for simplicity. As shown, the point $\mathbf{y}^j \in \mathbf{Y}$ is chosen only if it has at least one point $\mathbf{x}^i \in \mathbf{X}$ sufficiently close to it and if the point normal orientations $\mathbf{y}^j$ and $\mathbf{x}^i$ are sufficiently close. Points $\mathbf{x}^i$ and $\mathbf{y^j}$ are considered sufficiently close if $d_{eucl}(\mathbf{x}_p^i, \mathbf{y_p^j}) \leq 1.5\sigma$, where σ refers to the standard deviation of the GMM just estimated using the registration process. The point normal orientations are considered sufficiently close if the angle between them is less than 15°.

As described above, the registration process can be used to detect cluster 1 given cluster 0. This process can be repeated to find a matching cross-sectional cluster for cluster 1, resulting in the detection of cluster 2. This process of growing a part stops when a good match cannot be found for a cross-sectional cluster. As mentioned above, $P(j \mid \mathbf{x}^i)$ is defined as the correspondence probability between points in the two point sets. For each point in $\mathbf{X}$, we can find the best match in $\mathbf{Y}$ using the correspondence probability, $P(j \mid \mathbf{x}^i)$. The average angular difference between the best matches is used to decide whether the growing process needs to be terminated.



(a) (b) (c) (d)

Fig. 4.10. Illustration of point selection after registration. (a) The two point sets, $\mathbf{X}$ representing the cluster for which we are seeking a match (cluster 0 in our example) and $\mathbf{Y}$ representing the neighboring points, before registration. (b) The transformed $\mathbf{Y}$, $\mathcal{T}(\mathbf{y}^j) = (s\mathbf{R}\mathbf{y}_p^j + \mathbf{t}, \mathbf{R}\mathbf{y}_n^j)$, after registration. (c) Chosen and rejected points, after threshold based selection, shown in different colors. (d) Having a member of $\mathbf{X}$ close by, after transformation, is not enough to be selected, the point normals should also closely match. The point normals of all the red points are shown. Also shown are the point normals of three green points (within the circle) that were rejected. The point normal orientations of the green points are too different from that of its neighboring red points, and hence these points are not chosen.

If the average angular difference between best matches is greater than 15°, the growing process terminates.

As depicted in Fig. 4.8, method 2 is better equipped for dealing with junctions and therefore is the default method to grow parts. But when the cross-sectional cluster is sparse, the registration process cannot be relied on. The point normal estimation is itself unreliable in such a scenario. Therefore, method 1 is employed only when the cross-sectional cluster is sparse. In practice, if the cross-sectional cluster has less than hundred points in it, method 1 is employed.

4.4 Optimal Part Selection

In the previous section, methods to grow parts were described. The point cloud was clustered into M clusters and one part per cluster was grown. Given these M parts, we first assign costs to each part and then select the best subset of parts that cover the whole point cloud based on these scores. The overall cost has four components. Below, we first describe each of these components. The process of

combining these components to obtain a single cost per part is then described. And finally we explain how the optimization process, which selects the best parts, is set up.

4.4.1 Cost Components

The first component is the registration cost. For a good part, the adjacent cross-sectional clusters would be a good match for each other. Remember, the relationship between adjacent cross-sectional clusters is defined by a similarity transform and via the registration process, we have estimated the best similarity transform parameters. After transformation (with the parameters estimated by the registration algorithm), the two point sets should be a good match to each other as illustrated in Fig. 4.10 (b). Once we perform the registration between adjacent clusters of a part, for each point in the point set $\mathbf{X}$, we can find the best match in point set $\mathbf{Y}$ using the correspondence probability, $P(j \mid \mathbf{x}^i)$. The average angular difference between the best matches is used as the registration cost. Therefore, the registration cost would be low if the corresponding points in the two point sets have similar orientation. If a part is made up of k cross-sectional clusters, we would obtain $k-1$ registration costs from pairs of adjacent clusters. To obtain the registration cost, we take the mean of these $k-1$ costs.

The second component, referred to as the fit cost, measures the goodness of fit of the skeleton to the points that constitute the part. This cost is computed in the same manner as the cost, $c(\theta, \phi)$, in algorithm 4.1. As explained before, and as depicted in Fig. 4.2 (c), the point normals are expected to be perpendicular to the normal of the local cross-sectional plane. The fit cost is designed to penalize any deviations to this. To compute the fit cost, for a part with k cross-sectional clusters, we compute the cost, $c(\theta, \phi)$, for each cross-sectional cluster and take their mean value.

The third component of the overall cost is the length of the part. The length here refers to the length of the skeleton. Since, skeletons are represented by cluster centers

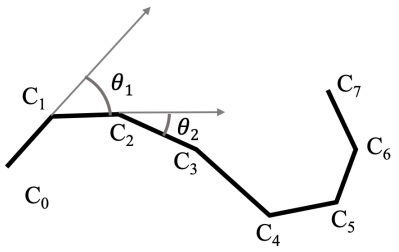


Fig. 4.11. Skeletons are formed by joining cluster centers. The black lines represent the skeleton. The length of the skeleton is computed as the sum of the euclidean distances between cluster centers when traveling from one end (C_0) to the other end (C_7) of the part. The turning angles at cluster centers C_1 and C_2 are also shown.

of the cross-sectional clusters, the length is computed as the sum of the euclidean distances between the centers as you travel from one end to the other end of the part. For instance, in Fig. 4.11, $\sum_{i=0}^{6} d_{eucl}(C_i, C_{i+1})$ represents the length of the skeleton. Longer parts are preferred over shorter parts.

Since, we prefer smooth curves as skeleton, the fourth component is the total turning angle of the skeleton. Fig. 4.11, shows how turning angles are computed. In the example shown, the total turning angle cost would be $\frac{\sum_{i=1}^{6} \theta_i}{6}$.

To obtain the overall cost, each individual cost component is first normalized separately by subtracting the mean and dividing by the standard deviation. Let c^i_{reg}, c^i_{fit}, c^i_{len} and c^i_{ang} represent the normalized registration, fit, length and turning angle costs of part i respectively. The overall cost for part i is then given by: $c^i_{ovr} = c^i_{reg} + c^i_{fit} - c^i_{len} + c^i_{ang}$. Note, during optimization we minimize the cost and since we prefer longer parts over short parts, we subtract the length component instead of adding it.

4.4.2 Optimal Parts Selection

Once we assign a cost to each part, the optimal subset of parts are chosen by minimizing the following constrained binary integer program.

$$\begin{aligned} &minimize \quad \boldsymbol{c}^T\boldsymbol{x} \\ &\text{subject to} \end{aligned} \tag{4.17}$$

$$\boldsymbol{x^T a} \geq \left(\frac{k_1}{100}\right) N \tag{C_1}$$

$$\boldsymbol{x^T Q\ x} \leq \left(\frac{k_2}{100}\right) N \tag{C_2}$$

where,

$\boldsymbol{x} = (x^1, x^2, \cdots, x^M) \in \{0,1\}^{M\times 1}$,

$\boldsymbol{c} = (c^1_{ovr}, c^2_{ovr}, \cdots, c^M_{ovr}) \in \mathbb{R}^{M\times 1}$

$\boldsymbol{a} \in \mathbb{N}^{M\times 1},\ \boldsymbol{Q} \in \mathbb{N}^{M\times M}$

The optimization is carried out over the vector $\boldsymbol{x}$, and individual components of $\boldsymbol{x}$ indicate whether a particular part is selected or rejected. I.e., $x^i = 0$ implies that part i is rejected and $x^i = 1$ implies that it is selected. Vector $\boldsymbol{c}$ represents the costs associated with individual parts. Therefore, by minimizing $\boldsymbol{c}^T\boldsymbol{x}$, we are minimizing the overall cost. The trivial and uninteresting solution for $\boldsymbol{x}$ is the vector of all zeros. But we are interested in solutions that cover almost all the points in the point cloud. Note, a point in the cloud is said to be covered if the optimization process selects a part, of which the point is a member. In order to force the optimization process to cover most of the points in the point cloud, we impose the constraint C_1. The i^{th} component of vector $\boldsymbol{a}$ gives the number of points belonging to part i. Therefore, imposing constraint C_1 would ensure that at least $k1\%$ of the total number of points in the cloud are covered. The total number of points in the cloud is indicated by N.

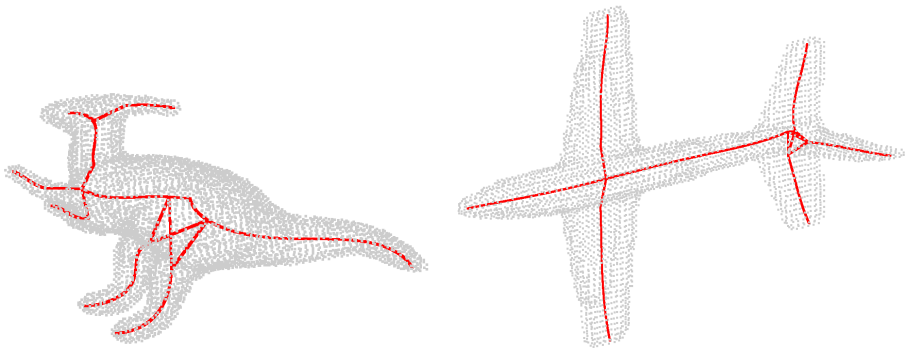
But if two of the selected parts share some points, these points would be counted twice. Note, a point in the point cloud can be a member of multiple parts. In Fig. 4.1 (a), for example, two parts (in the top-left corner) covering the tail portion have substantial overlap. If part i covers n_i points, part j covers n_j points and if the two parts share n_s points, then the total number of points covered by selecting both parts i and j is $n_i + n_j - n_s$. To avoid too much overlap between parts, constraint C_2 is imposed. The element q_{ij} of $\boldsymbol{Q}$, represents the number of points that parts i and j share. Thus constraint C_2 ensures that the sum of pair-wise overlap (in terms of number of points) between parts is less than $k_2\%$ of the total number of points in the cloud. To see this, note that, the element q_{ij} is relevant only when both parts i and j are selected, i.e., when $x^i = 1$ and $x^j = 1$. Note, the lower triangular part of $\boldsymbol{Q}$ is zero to ensure that the pair-wise overlaps are not penalized twice. In short, the constraints ensure that most points in the point cloud are covered with little overlap between parts and the objective function ensures that this is done at a low cost (meaning the best parts are selected).

The constrained binary integer program, described above, is solved using the Gurobi solver [60]. Since, the value of M, the number of candidate parts, is usually small (150 or less), the optimization algorithm converges fast, within a few seconds in most cases. The result of the optimization procedure is depicted in Fig. 4.1 (b). Once the individual parts that cover the point cloud are identified, the next step is to link them to obtain a skeletal representation of the shape. This procedure is explained in the next section.

4.5 Linking Part Skeletons

The first step in linking skeletons is to determine potential connections among the various parts. For instance, the legs of the dinosaur in Fig. 4.1 (b) could be linked to the parts representing the torso or the tail but not to the parts representing the horn or neck of the dinosaur. I.e., a part can only be linked to a subset of other parts and

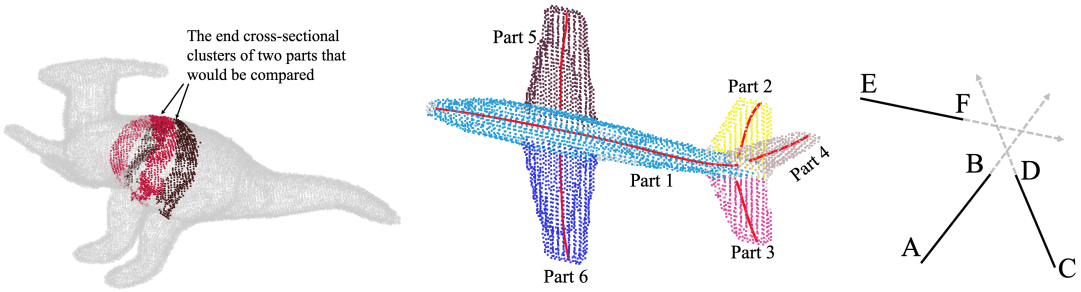
the first step determines this subset of potential links for each part. To determine if two parts can be potentially linked to each other, a very simple observation is used. Two parts, say part A and part B, can be potentially linked only if there is at least one point in A who is the neighbor of some point in B. To determine if any two points in the point cloud are neighbors, the connectivity matrix G_{cnct}, described in section 4.3.2, is used.



(a) (b)

Fig. 4.12. Linking all parts that can be potentially linked according to G_{cnct} can lead to unnecessary additional connections. (a) The legs can potentially connect to both the torso and the tail parts. (b) The many links near the tail of the airplane are unnecessary.

Linking all the parts that can be potentially linked would lead to strange additional connections like the one shown in Fig. 4.12. Note, the parts extracted for the dinosaur and the airplane are shown in Fig. 4.1 (b) and Fig. 4.13 (b) respectively. In the case of the dinosaur, the problem is that parts that could be combined, stay as separate parts. The skeletons for parts representing the tail and the torso of the dinosaur can be combined to form a single smooth curve. This process, of combining skeletal curve where it is possible, will be helpful later when we attempt to connect the legs of the dinosaur to the body. On the other hand if the tail and torso remain as separate parts, interconnecting the two legs, the torso and the tail could be confusing. Therefore, combining skeletons wherever it is possible would be a good first step. To decide if two parts can be combined, we check if the cross-sectional clusters at the



(a) (b) (c)

Fig. 4.13. Steps for linking parts. (a) The cross-sectional clusters at the end of parts are compared to see if the two part skeletons needs to be combined. (b) All parts identified by the algorithm are shown for a point cloud of an airplane. Parts 1, 2, 3 and 4 form a clique where the end points of the corresponding parts meet. In this scenario, the algorithm looks for a common point to connect the four skeletons together. (c) AB, CD and EF represent three part skeletons that form a clique of size three. Rays $\overrightarrow{AB}$, $\overrightarrow{CD}$ and $\overrightarrow{EF}$ are obtained by extending the corresponding skeletons.

end are a good match. This is done by registering the end cross-sectional clusters from the two parts. The end cross-sectional clusters for the torso and the tail of the dinosaur are shown in Fig. 4.13 (a). After registering the two cross-sections, the registration cost (the same as the one defined in section 4.4.1) is computed. Keep in mind that a lower registration cost means that the points in one cross-sectional cluster, after registration, were able to find, near it, a similarly oriented match from the other cross-sectional cluster. If the registration cost, measured in terms of angular difference, is less than a threshold of 35° and the scale factor difference from one is less than 0.5, it is decided to combine the two parts. These thresholds were chosen after experimenting with a few values.

In the next step, we deal with parts that form connected components. If a part is only connected to one other part, there is no confusion as to how to link the parts. In Fig. 4.13 (b), for example, part 5 has only part 1 as a potential link. Therefore, all that needs to be done is to connect par 5 to part 1. The case with part 6 is also the same. But when there are connected components, it is not immediately clear as to how to interconnect them. For instance, in Fig. 4.13 (b), parts 1, 2, 3 and 4

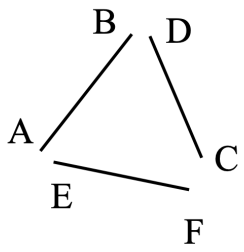


Fig. 4.14. An example of a clique of size three for which finding a junction point to interconnect the parts would not be appropriate. The key point is to see if it's the same end point of a part that connects to all other parts in the clique.

form a connected component. The ideal way to interconnect them would be to find a junction point, and connect all parts to that single point forming one junction. To find the junction point, rays formed by extending the part skeletons are considered. Fig. 4.13 (c), shows three part skeletons and the rays obtained by extending them. For each ray, the point that minimizes the sum of distances to the other rays is found. In the example in Fig. 4.13 (c), for ray $\overrightarrow{AB}$, we find the point on $\overrightarrow{AB}$ that minimizes the sum of distances to rays $\overrightarrow{CD}$ and $\overrightarrow{EF}$. The same is done for rays $\overrightarrow{CD}$ and $\overrightarrow{EF}$. Among these points (we get one point per ray), the point with minimum sum of distances to the other rays is chosen as the junction point.

Finding a junction point to connect together all the members of the connected component would not be appropriate in the case when both end points of a part are involved in the clique formation. An example of such a scenario is shown in Fig 4.14. Here, each end point of a part has only one potential link and therefore liking these parts together is straightforward. Hence, finding a junction point to link all the members of the connected component, is only appropriate when for each part involved, it is the same end point that connects to all the other parts in the component. Once, the issues depicted in Fig. 4.12 are addressed, all remaining potential connections are made to obtain the final skeleton.

4.6 User Interface

One significant advantage of a part based skeleton extraction algorithm is the ease with which users can interact with the algorithm. For point clouds that are not noisy, user interaction is unnecessary. But the part based algorithm can handle very noisy point clouds with little user interaction. To demonstrate the ease of interaction, a graphical user interface (GUI) was developed. Fig. 4.15, shows the GUI with some annotations. Since the GUI is not the focus of this work, only a brief overview is provided and only the relevant features are explained. To begin the processing, the user can use the file browser to load the point cloud. Once the point cloud is loaded, the main display would show the rotating point cloud. Note, after each processing step the main display will update to show the corresponding results. The toolbar button labelled "Grow Parts" can be used to grow parts from initial cross-sections. I.e., this button executes steps described in sections 4.3.2, 4.3.3 and 4.3.4. The grown parts are then displayed on the two-by-two grid display on the right side of the main display. Note, all the displays (the main display and the grid displays) allow user interaction. I.e., the users can play/pause the rotation by double clicking on the display and also manually rotate the display using the mouse/touchpad. The button labelled "Part Selection", can then be used to run the optimization routine that selects the optimal subset of parts. And finally, the "Link Part Skeletons" button can be used to link the individual part skeletons to obtain the final skeleton. The user may also choose to simply click the "Skeletonize" button to run all the steps (of generating parts, selecting optimal parts and linking selected parts) at once. The message box and the progress bar will provide the user with feedback on the status of the program execution.

The two buttons labelled "Browse Parts", can be used to browse through all the parts generated. The left and right arrows can be used to view the previous and next four parts respectively. After the parts are generated (by using the "Grow Parts" button), the users, instead of running the optimization routine, may choose to

manually select the parts that they think are the best parts. For this purpose, each part shown in the grid display is accompanied with a checkbox that will allow the user to select that part. The users can instead run the optimization routine, which selects the best subset of parts, and then choose to correct mistakes, if any, made by the routine. For all parts that the optimization routine selects, the corresponding checkbox in the grid display would be checked. The user can browse through all the parts, after the optimization routine is run, and decide to reject the parts chosen by the routine by unchecking the corresponding checkbox. The user can also add to the currently selected list of parts by checking the checkbox corresponding to any unselected part. This procedure, allows the user to easily modify the selection of the optimizing routine with a few clicks. The user also has the option to permanently remove noisy parts from the available list of parts by clicking the "Remove" button.

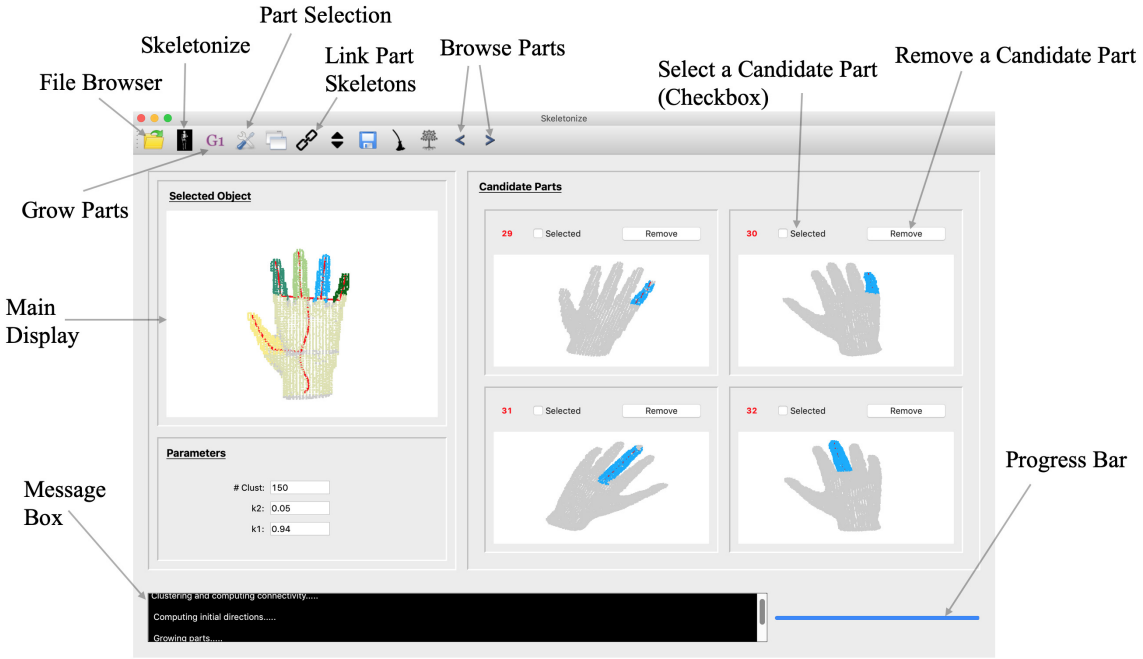


Fig. 4.15. The Graphical User Interface

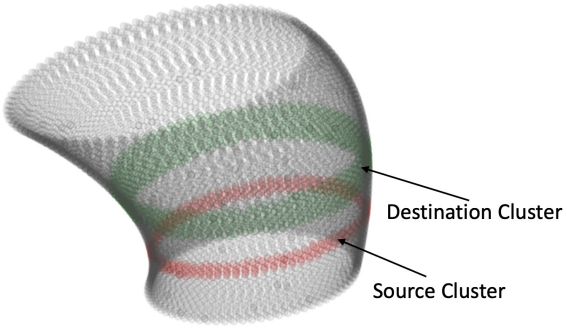


Fig. 4.16. A synthetically generated GC.

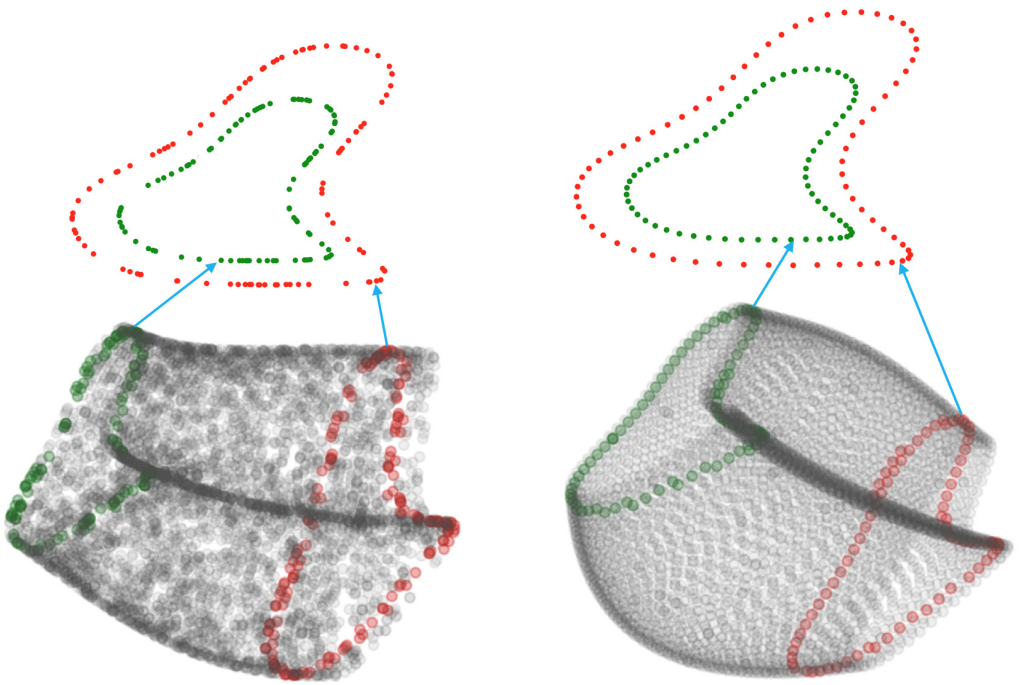
4.7 Results

The results are divided into two sections. We first present the results for comparison of the our newly proposed registration algorithm with that of Myronenko and Song [74]. Next, the results of using the proposed algorithm for skeleton extraction are presented.

4.7.1 Registration Results

As we described above, we use the registration algorithm to grow parts from initial cross-sectional clusters. Therefore, it is appropriate to use a similar setting to test the newly proposed registration algorithm. I.e., we first generate synthetic point clouds that represent GCs and then measure the accuracy of the registration algorithms in registering a random cross-sectional cluster with another randomly chosen neighborhood. For instance, in Fig. 4.16, the point cloud corresponding to a GC is shown. To measure the accuracy of the registration algorithms, we register the source cluster with the destination cluster. Since, the point cloud was generated by us, we can measure the accuracy by comparing the registration results with the ground truth.

There are a couple of challenges that the registration algorithm has to overcome in order to accurately estimate the registration parameters. Understanding these challenges is important in understanding why it is appropriate to test the registration



(a) Random sampling (b) Regular sampling

Fig. 4.17. The random sampling of the point cloud could lead to inaccuracies in the registration if position alone is used to evaluate the fit.

algorithms in the setting described above. One of the two challenges is related to the issue of random sampling. I.e., the points on a point cloud can be thought of as random samples from the surface of the shape. Or equivalently, they are random samples from a scaled rotated and translated 2D cross-sectional contour. Samples from two such contours from a point cloud are shown in Fig. 4.17. The point clouds are shown at the bottom with two contours highlighted, and the highlighted contours are shown at the top. The point cloud in Fig. 4.17 (a) is obtained by transforming (i.e., scaling, rotating and translating) 2D contours which are sampled at random, while the point cloud in Fig. 4.17 (b) is obtained by transforming 2D contours which are sampled at regular intervals. It is evident that, in the case of contours in Fig. 4.17 (a), due to random sampling, a simple scaling of points in one of the contours would not produce the other point set exactly. Therefore, using just positional information to estimate the registration parameters would lead to inaccuracies in the estimate as

we would demonstrate later. We would expect the registration algorithm to be able to handle point clouds like the one shown in Fig. 4.17 (a). Adding the orientation information in such a scenario will make the registration process more robust and stable.

To understand the second challenge, keep in mind the process of searching for a match, described in section 4.3.4. Starting from an initial cross-sectional cluster, we search for a similar cluster in its neighborhood. The second cluster, therefore, is obtained from the neighborhood of the initial cluster and almost always has a lot more points than the initial cluster for which we seek a match. For example, in Fig. 4.16, the destination cluster is wider and has a lot more points than the source cluster. Looking just at the positional information of the points, to register the source cluster with the destination cluster, could lead to wrong estimates for the rotation parameter. Here again, penalizing large variations in point normal orientation, in addition to penalizing positional disparity, would significantly improve the estimates of the registration parameters. This is especially true if the neighborhood cluster has a lot more points than the initial cluster. Through experiments described below, we show that adding orientation information can effectively deal with these issues. Note, the random sampling issue and the issue of trying to find a match in a larger point cloud are issues encountered when registration is employed to grow parts. Since, the primary use of registration for us is in growing parts, it is only appropriate to test the registration algorithms in such a setting.The process of generating the point clouds is described next, followed by the description of the experiments and their results.

Point Cloud Generation

As shown in Fig. 4.2, a scale function, a 2D cross-sectional contour and a 3D axis are the three components necessary to generate a GC. The following parametric equation is used to generate the 3D axis of the GC:

$$(x(t), y(t), z(t)) = (C_1 cos(t), C_2 sin(t), C_3 t)$$

Where, C_1, C_2 and C_3 are random real numbers in the interval $(0, 50)$. To obtain a smooth scale function, a sinusoidal curve with a random phase shift is used. Since scale values cannot be negative, a constant offset is added to the phase shifted sinusoid, to ensure that all the values are positive. In fact, the offset added is such that all values in the scale function are greater than or equal to one. To generate the 2D cross-sectional contour, we first generate a set of eight points as shown in Fig. 4.18. The angular intervals between the points are equal and their distances to the origin is randomly chosen. We then fit a smooth, periodic, spline to these points to obtain the 2D cross-sectional contour.

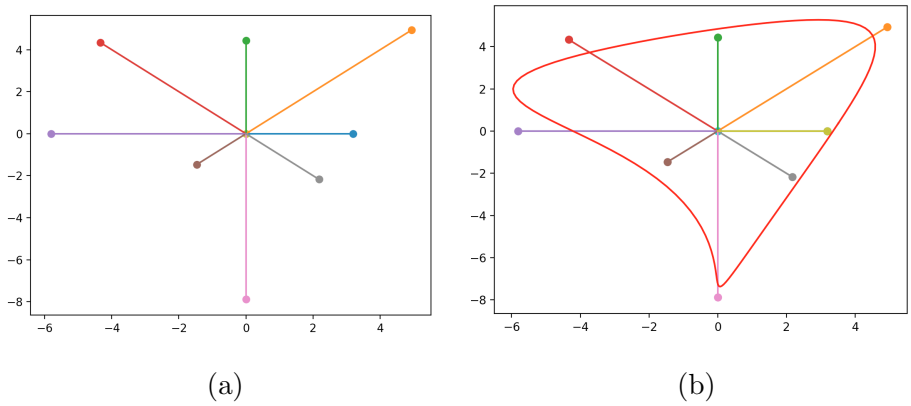


Fig. 4.18. Generating a random 2D cross-sectional contour. (a) Generate a set of points at equal angular interval of $\pi/4$ radians whose distance from the origin is random. (b) Fit a smooth closed contour (shown in red) to these points to obtain the 2D cross-sectional contour.

To obtain a GC, the 3D axis is sampled at regular intervals. In Fig. 4.2, the red blobs on the 3D axis represent this discretization. The key idea in the construction of a GC is to place a rotated and scaled 2D cross-sectional contour at each sample point along the 3D axis. In fact, the 2D cross-sectional contour is translated in addition to rotation and scaling and the translation is determined by the 3D axis. The smooth scale function, defined above, is used to scale the cross-sectional contour along the 3D axis. To obtain a smooth rotation function along the axis, we employ the Frenet-Serret frame. I.e., at each point on the 3D axis, the tangent, normal, and binormal

unit vectors of the axis, determines the rotation that needs to be applied at that point. Therefore, defining the 3D axis simultaneously defines the rotation that needs to be applied to the 2D cross-sectional contour along the 3D axis. Fig. 4.2 (a) shows the GC generated when rotated and scaled 2D cross-sectional contours are placed at the sample points along the 3D axis. Since, we are interested in a point cloud representation of the GC, the 2D contour is sampled before applying the scaling, rotation and translation transformations. Note, the number of samples chosen for the 2D contour and the 3D axis would determine the sparsity/density of the point cloud.

Experiments and Results

As mentioned in section 4.7.1 and as depicted in Fig. 4.16, to compare the registration algorithms, we register a randomly chosen source cluster with a randomly chosen destination cluster within the GC. We consider two different cases for comparing the registration accuracies. In one case we sample the 2D cross-sectional contour, before transforming it, randomly (as shown in Fig. 4.17 (a)) and in the other case we sample the contour at regular intervals (as shown in Fig. 4.17 (b)).

Fig. 4.19, shows the results of comparing the two algorithms for both the cases. The results were obtained by performing five hundred registrations. I.e., a random GC point cloud was generated and a randomly chosen source cluster, within the GC, was registered with a randomly chosen destination cluster. This process was repeated five hundred times to obtain the results. Note, in the plots, "With Normals" refers to our method as it utilizes the point normal information. Similarly, the case in which the 2D contour is sampled at regular intervals is referred to as "Regular" and the other case is referred to as "Random".

In Fig. 4.19 (a), the mean error, from the five hundred registrations, in estimating the rotation matrix is shown. In [98], the author discusses various metrics employed

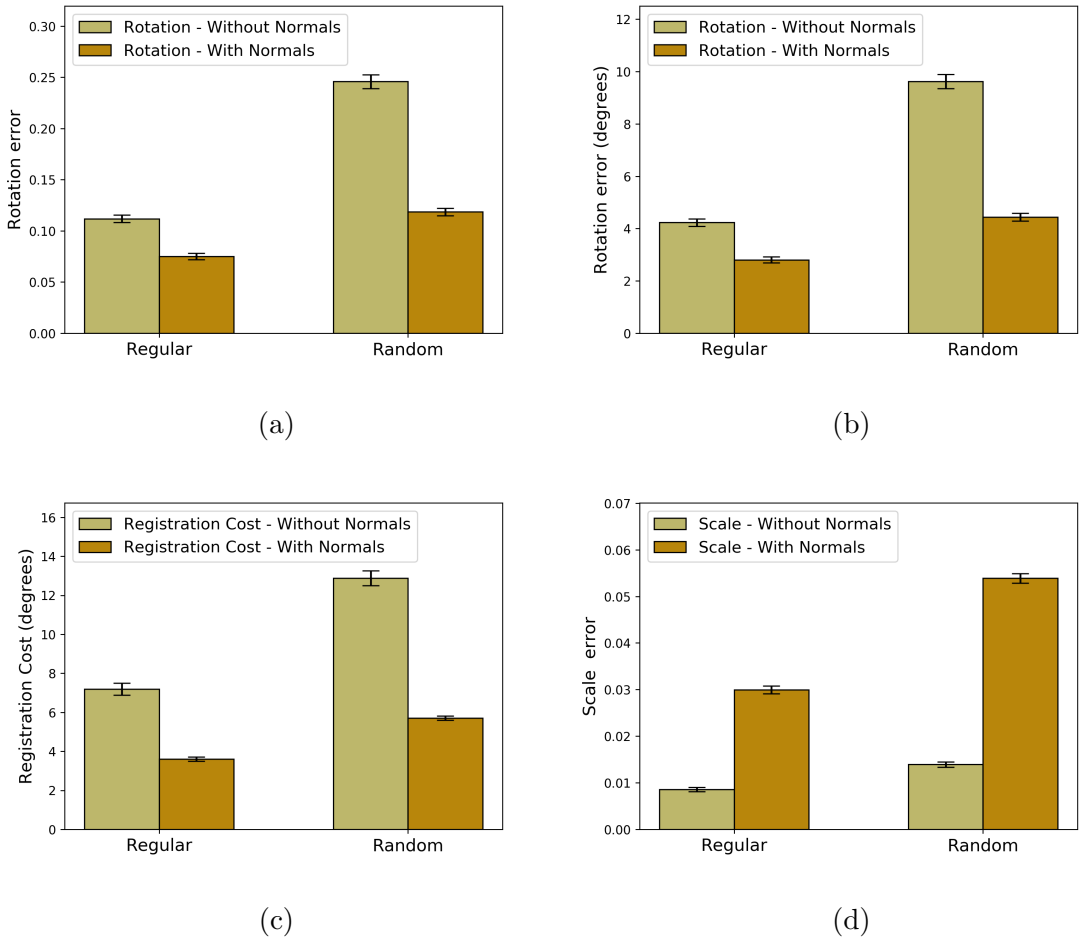


Fig. 4.19. In the legend, "With Normals" refers to the proposed method and "Without Normals" refers to the method in [74]. (a) Error in estimated rotation, expressed in terms of the Frobenius norm of the difference in rotation matrices. (b) Error in the estimated orientation of the cross-sectional plane, expressed as the angular difference between the plane normals in degrees. (c) Comparison of the registration costs (same as the one defined in section 4.4.1). (d) Error in estimation of the scaling parameter.

to measure the difference between rotation matrices. We use the one based on the Frobenius norm which is given by:

$$err_{rot}(\boldsymbol{R}_1, \boldsymbol{R}_2) \;=\; \|\boldsymbol{I}_3 - \boldsymbol{R}_1\boldsymbol{R}_2^T\|_F$$

where $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are the two rotation matrices we want to compare. The value of this metric will be in the range $[0, 2\sqrt{2}]$. As shown in Fig. 4.19 (a), the proposed

method that uses the normal information does better than the one in [74] in both cases. The difference is much higher in the random case. This is expected because an error function based just on positional information is less capable of dealing with the random sampling. Since, it is a little difficult to interpret the metric values for the rotation error, we also plot the error in the estimated orientation of the cross-sectional plane in Fig. 4.19 (b). These plots resemble the plots for the rotation error, but are easier to interpret as the error is measured in terms of the angle between the actual normal and the estimated normal of the cross-sectional plane and is expressed in degrees. In the context of growing parts, the accurate estimation of the cross-sectional plane orientation is crucial because an error in this estimate will accumulate as more registrations are performed to grow the part.

The comparisons for the registration cost, defined in section 4.4.1, are presented in Fig. 4.19 (c). Remember, the registration cost is mean difference in the point normal orientation between a point in the source cluster and its best match (found using the correspondence probability) in the destination cluster. Also keep in mind that, as described in section 4.3.4, if this cost is above the threshold of 15°, the part growth algorithm will decide that it is unable to find a good match in the neighborhood, and will terminate the growth process. Therefore, keeping this cost low is essential to obtain fully grown parts. As shown in Fig. 4.19 (c), proposed algorithm outperforms the other method in both the cases, significantly so in the case of random sampling. Also note that the rise in this cost, when comparing the regular case with the random case, is moderate for our registration method. The use of point normal information leads to a much more stable algorithm.

Fig. 4.19 (d), shows the error in scale estimate. As it can be seen, the other method (i.e., the method in [74]) has a smaller scale error compared to our method. To understand this result better, keep in mind the observation that the estimated value of the parameter, σ, for our method is almost always much larger than the value estimated by the other method. For the other method, that only uses the positional information, to lower the cost, the points in the source cluster has to get

as close as possible to the points in the destination cluster, and at the same time keep the value of σ as small as possible. This can be done by tolerating bigger differences in point normal orientation. But for our method, since it uses both the orientation as well as positional information, getting closer to points in the destination cluster at the expense of larger differences in point normal orientation is not desirable. Moreover, the error in scale estimates is only about five percent on average in the worst case, which is not a significant error and does not have much consequences in the context of growing parts. This is true because the estimated parameters will decide which points in the neighborhood of a cross-sectional cluster would form the adjacent cluster. Therefore, the error is scale estimates is tolerable, as long as the error does not prevent the algorithm from picking the right neighboring points. Keep in mind that the value of the estimated σ is used to determine the neighbors (see the explanation of method 2 in page 101). This makes small errors in the scale estimate insignificant in the context of growing parts, especially because of the higher σ value. As mentioned earlier, the error in rotation parameter estimates will accumulate as we perform multiple registrations to grow parts and therefore it is very significant to get it right. This again shows us why using the point normal orientation information is important in the context of using registrations to grow parts.

And finally, to demonstrate the ability of our algorithm to handle the second challenge of registering the source cluster against a much larger destination cluster, mentioned in section 4.7.1, we ran the following experiment. Like before, we performed five hundred random registration on synthetically generated GC point clouds. For each GC point cloud, we chose a random source cluster and two different neighboring destination cluster from within the GC. One of the neighboring cluster being much larger (more number of points included) than the other. We registered the source cluster against the two neighboring destination clusters for both methods. We then looked at the proportion of cases, from among the five hundred pairs of registrations for each method, where the registration with a larger destination cluster lead to larger errors in the estimated rotation parameter. For our registration method, in

16.2% of the cases, using a larger destination cluster (which is equivalent to looking for a match in a larger neighborhood in the context of growing parts) lead to larger errors in the estimation of the rotation parameter. This proportion was 72.2% in the case of the other method. This simply shows that our method is more robust to variations in the neighborhood size as we search for a matching cross-sectional cluster while growing parts.

4.7.2 Skeleton Extraction Results

Skeleton extraction results for a variety of shapes are shown in Fig. 4.20. For each shape the individual skeletons for the parts identified by the algorithm are shown at the top and the linked skeleton is shown at the bottom. An extended gallery of results for skelton extraction, along with rotating 3D animations, is available at: `https://web.ics.purdue.edu/~vthottat/skel/`. The models were obtained mainly from the McGill 3D shape benchmark [99] and from the 3D object recognition database, ObjectNet3D [100]. As can be seen in Fig. 4.20, the algorithm is capable of handling a wide variety of shapes. Both, flat objects like the circular table top (Fig. 4.20 (f)) and spectacles (Fig. 4.20 (i)), and rounder objects like the unicorn (Fig. 4.20 (h)) or teddy bear (Fig. 4.20 (j)) are handled well.

As shown in Fig. 4.15, there are three parameters that can be tuned to obtain the desired results. The number of clusters from which candidate parts would be grown is one of the parameters. There should be at least as many clusters as there are parts. Remember, the final parts are selected from the candidate part list by the optimization algorithm (section 4.4). Therefore, the number of clusters is usually set to a much higher value. In practice, depending on the shape, we use $50n$ clusters, where, $1 \leq n \leq 4$. The parameters k_1 and k_2 are the optimization parameters that are defined in section 4.4. The value of k_2, which determines the amount of overlap between parts in terms of the number of points they share, was set to 0.05 for all the examples in Fig. 4.20. Keep in mind that it is possible for a point in the point

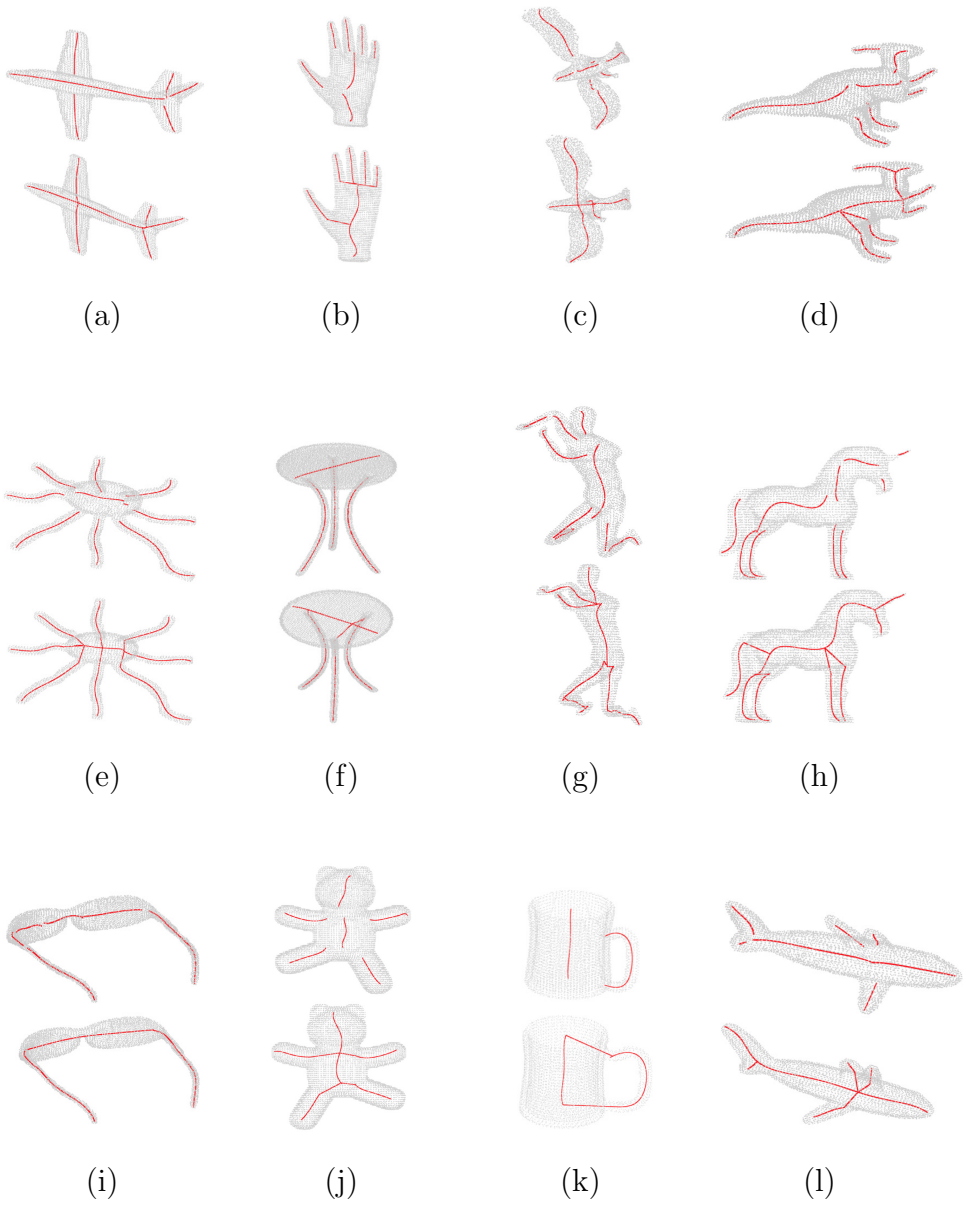


Fig. 4.20. Results for part identification and skeleton extraction. For each shape the individual skeletons corresponding to the parts identified by the algorithm are shown at the top and the linked final skeleton is shown at the bottom.

cloud to be assigned to multiple parts. For example, a point at the intersection of multiple parts would be shared by all parts forming the junction. With $k_2 = 0.05$, less than five percent of all the points in the point cloud could be shared between various parts identified by the algorithm. The value of k_1 will decide the percentage of points

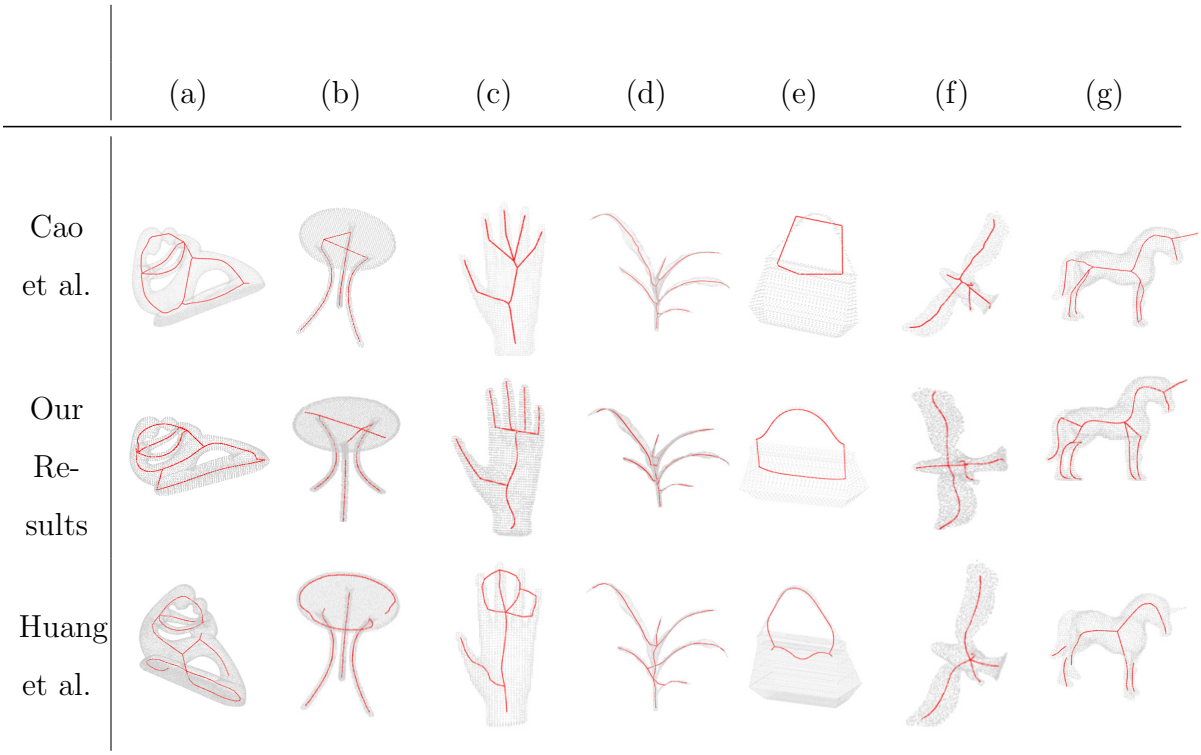


Fig. 4.21. Qualitative comparison of skeletons extracted by our method with the methods by Cao et al. [88], and Huang et al. [81].

covered by the parts chosen by the algorithm. Note that the optimization problem, described in section 4.4 may not be feasible for all values of k_1. For example, it might not be possible to cover 100% of the points in the cloud with only 5% overlap. Since, we would like the parts extracted to cover maximum number of points, the algorithm picks the maximum possible value of k_1. It does so by checking the feasibility, starting at 90% (i.e., $k_1 = 0.90$). If 90% is feasible then it tries a higher value, else it decreases the value of k_1. This process continues until, the maximum feasible value for k_1 is found. However, sometimes picking a value a little less than maximum feasible value for k_1 gives better results. Hence, the user has the option to modify this value through the user interface.

We provide qualitative comparison of our results with the results from Cao et al. [88], and Huang et al. [81] in Fig. 4.21. We use the implementations provided by the authors themselves for the comparison. We found that using the default parameter

settings for methods by Cao et al. and Huang et al., do not produce good results in most cases. Hence, we manually tuned the available parameters to obtain better results and the best results that were obtained are presented in Fig. 4.21. Note, while we and Cao et al. use point normal information in our algorithms, Huang et al. do not use this information. However, keep in mind that the input to our algorithm is just a point cloud. As described in section 4.3.1, we estimate the point normals as a first step of our algorithm. Since the input to all the algorithms is the same, the algorithm of Huang et al. is not at any disadvantage.

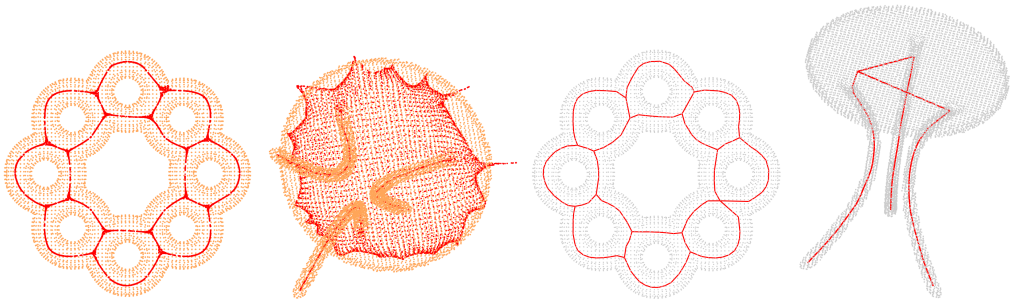
As shown in Fig. 4.21, the method by Huang et al. had difficulty in extracting skeletons for many of the point clouds shown. For the unicorn and the eagle (Fig. 4.21 (g) and (f) respectively), parts of the skeleton were missing. For the hand (Fig. 4.21 (c)), the connection between the fingers are incorrect. Note, Huang et al. draws an initial set of samples from the point cloud to start their skeleton extraction process. We observed that even for the same set of parameters but different initial samples, the skeletons extracted varied widely. The method by Cao et al. did well on most point clouds except for the table (Fig. 4.21 (b)) and the suitcase (Fig. 4.21 (e)). In fact, these two cases demonstrate one of the drawbacks with their approach to skeleton extraction. As shown in Fig. 4.22, when the contracted points closely represent the skeleton of the shape qualitatively, like in the case of the eight inter-connected toruses (Fig. 4.22 (a)), the extracted skeleton (Fig. 4.22 (c)) would correspondingly be good. But in Fig. 4.22 (b), we can see that at the end of the contraction step, the contracted points do not resemble the skeleton of the round table top. Correspondingly, the extracted skeleton fails to provide a good representation for the skeleton of the table top. Note that distance field based approaches would also share this drawback.

As shown in Fig. 4.21 (b), our method was able to extract a nice straight line skeleton for the table top, primarily because we are looking to extract part skeletons first. Similarly, our part based approach was able to extract centered skeletons for the point cloud shown in Fig. 4.21 (e). These examples show why decomposing objects into GCs to detect parts and part skeletons can effectively deal with a wide variety

of shapes. By modeling the parts as GCs we are introducing a prior (or a bias) in the algorithm. But this bias can reduce the variability in the results. Since, our parts are defined by the very generic property of translational symmetry, the algorithm, even with the bias, can still handle a wide variety of shapes with lesser variability. Fig. 4.23 compares the performance of our algorithm to that of Cao et al. in detecting skeletons from noisy point clouds. This point cloud represents a rough 3D shape reconstruction of a rocking chair. Notice that our algorithm successfully identifies the various parts within the noisy point cloud. For instance, it can be seen that our algorithm detected the seat of the rocking chair (shown in light green) as a GC. This demonstrates that using a very basic property like translational symmetry to detect GCs can be a powerful tool in identifying parts and thereby simplifying the skeleton extraction problem. This example also shows that our method can be used as the first step in denoising point clouds and in surface reconstruction.

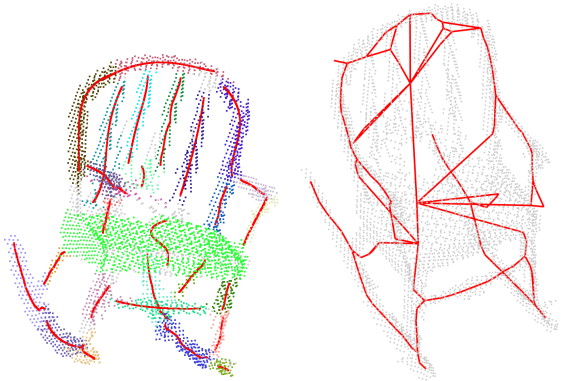
One of the limitations of our algorithm is that the method we use to link part skeletons, to form a complete skeleton of the shape, can sometimes lead to anomalies like the one shown in Fig. 4.24 (b). Keep in mind that we join individual part skeletons together using straight lines. The drawback of this can be clearly seen in Fig. . 4.24 (b), where this approach leads to an unnatural looking overall skeleton. The individual part skeletons of the parts detected by our algorithm shown in Fig. 4.24 (a). Ideally, the point cloud has to be segmented into eight toruses and each torus would be connected to two its neighbors. However, our algorithm does not succeed in extracting individual torus skeletons due to the presence of large number of junction regions where the translational symmetry of a torus breaks. In this case, the algorithm by Cao et al. succeeds in extracting a better skeletal representation of the shape as shown in Fig. 4.21 (c). As pointed out earlier, this is due to the fact that the point cloud at the end of the contraction step closely resembles the skeleton of the shape.

We would like to point out an interesting observation regarding the ambiguity of part skeletons for the same part. The part skeleton extracted for the seat of the point



(a) (b) (c) (d)

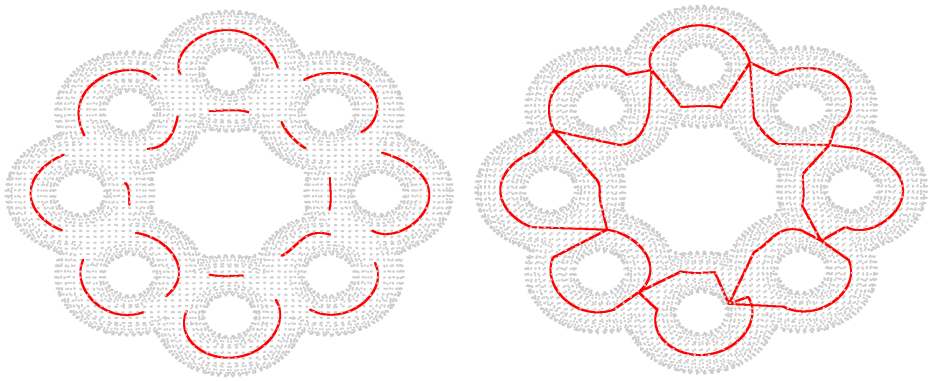
Fig. 4.22. The contracted points, at the end of the contraction step in the algorithm by Cao et al. [88], is shown in red in (a) and (b). The final skeletons extracted by Cao et al., for point clouds shown in (a) and (b), are shown in (c) and (d) respectively.



(a) (b)

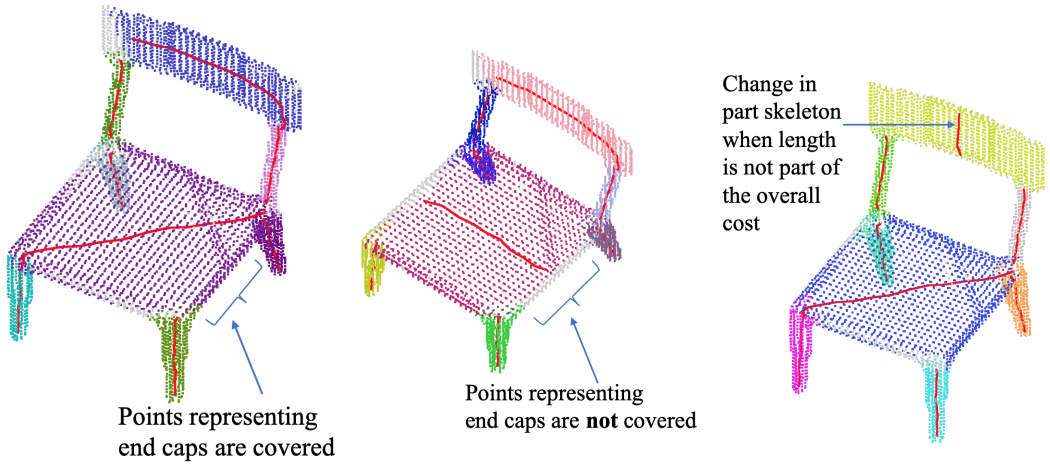
Fig. 4.23. Skeleton extraction with noisy point clouds. (a) Part skeletons extracted by our method with points belonging to different parts shown in different colors. (b) Skeleton extraction results from Cao et al..

cloud of a chair is shown in Fig. 4.25 (a). This result was obtained when the number of clusters was set to 150 and using the maximum feasible value of $k_1 = 0.95$ (in equation 4.17 , constrained C_1) for this point cloud. Note that by setting $k_1 = 0.95$, we are asking the algorithm to cover at least 95% of the points in the cloud. In Fig. 4.25 (b), the results obtained by setting $k_1 = 0.92$ is shown. Notice that the parts extracted are almost the same but the skeleton of the part representing the seat of the chair is now different. In fact, this result (i.e., Fig. 4.25 (b)) is a qualitative better



(a) (b)

Fig. 4.24. (a) Parts extracted for the inter-connected toruses. (b) Final skeleton obtained after linking part skeletons



(a) (b) (c)

Fig. 4.25. Ambiguity in skeletal representation of parts. (a) The different parts identified, by our algorithm, are shown in different colors along with the skeleton shown as a red curve. (b) Parts identified and part skeletons extracted for a vale of k_1 (in equation 4.17 , constrained C_1) slightly lesser than the maximum feasible value. (c) Part skeletons extracted when the length component is not part of the overall cost.

result than the one in Fig. 4.25 (a). Also note that the points at the end cap region of the GC in Fig. 4.25 (b) are not included by the algorithm as part of the seat. This shouldn't be surprising because the translational symmetry based definition of a GC will not capture the end caps. However, the GC can capture the end caps if the skeleton is diagonal as is the case with Fig. 4.25 (a). In short, very similar set of points representing the same part in a point cloud can be well represented by multiple skeletal curves due to the very general definition of a GC. Another example of such ambiguity can be seen by comparing the part skeletons in Fig. 4.25 (a) and (c). Keep in mind that each part was assigned a cost which had four components. One of the components was length of the part skeleton. Longer part skeletons were given preference over shorter ones. Fig. 4.25 (c) shows that removing the length component (i.e., not rewarding or penalizing length of parts) would result in a different set of part skeletons extracted. Note that the parts are almost the same, but the part skeletons are different. To summarize, the same part can be represented by multiple skeletal

representations as per the definition of a GC. When such situations arise, there will be ambiguity in choosing the best skeletal representation. Since, our algorithm extracts parts and lists them as shown in Fig. 4.15, it provides the user with the flexibility of manually choosing the best skeletal representation according to them.

4.8 Implementation and Run Time

The code is implemented in Python and takes about three minutes to run for a point cloud containing five thousand points with the number of clusters set to hundred on a 2.8 GHz Intel Core i7 processor with 16 GB RAM. We use Numba [101] to accelerate the cost function evaluation and gradient calculations used for registering two 3D point clouds. Note that the registration function is called multiple times in order to grow a part. Even with a modest estimate of five calls to the registration function per cluster, the total number of calls would be around five hundred. Hence, it is essential to be able to quickly evaluate the cost function and compute gradients for the registration function. Growing parts is the most time consuming step in the algorithm. Since, two parts can be grown completely independently, this process is amenable to parallelization. Therefore, there is ample scope to speed up the computations by parallelizing the part growing stage.

5. SUMMARY AND FUTURE WORK

The work done in this dissertation is inspired by the fundamental role of priors, especially the symmetry prior, in enabling veridical visual perception. The broad aim, therefore, has been to understand how these priors operate in the process of visual perception and then design algorithms that utilize this information to help machines perceive the world as we do. To this end, we described psychophysical experiments that better our understanding of how these priors are employed by the visual system and we also described a couple of algorithms which directly employ these priors to perform 3D reconstruction and 3D point cloud decomposition into parts.

Through the psychophysical experiments, we showed how symmetry, compactness and minimal surface area priors can be combined with binocular depth order information to model the percept of polyhedral shapes. The effectiveness of the model was demonstrated with the help of a control experiment. We also showed how the monocular model can be obtained from the binocular model by just dropping the binocular depth order term form the binocular model. Metrics were introduced to measure the dissimilarity between polyhedral shapes and to measure the asymmetry of a polyhedral shape. Note that the shape dissimilarity metric we introduced is invariant to rigid motion and size scaling and therefore is consistent with the conventional definition of shape. Using these metrics we showed that the perception of symmetric shapes is more veridical in comparison to asymmetrical shapes and similarly binocular perception is closer to veridical than monocular perception. For symmetric shapes viewed binocularly, we showed the absence of any systematic depth distortion. This is not surprising because symmetric shapes are almost always perceived as symmetric and depth distortions would destroy the symmetry of the shape (the shapes are viewed from non-degenerate directions). A website which shows the results from all 540 tri-

als, which include the 3D reference shape and the 3D shapes recovered by the model and the subjects, was developed and published. This enables the users to judge the goodness of the model and the metrics by themselves.

Note that we used one model for symmetric shapes and a different model for asymmetric shapes. The need to have two separate models per subject needs to be investigated further. If the visual system indeed uses two separate models based on the symmetry of the shape, then there must be a first step where the visual system detects symmetrical shapes. Does the visual system use binocular disparity information to do this ? Or, are there other priors that inform the visual system about the symmetry of shapes ? This needs to be answered in a future study. In the current work, we ignored the role of re-projection error. Keep in mind that all the shapes that the user could generate (by adjusting the three parameters) would produce the same orthographic image as the reference shape but not the same projective image. Though the fact that the models used in the prior study by Li et al. [1] and those used by our current work produced good results indicate that the effect of the re-projection error would be small, it would be still interesting to quantify it. Since, our shapes always had planar faces, planarity was an in-build prior. We know that planarity is a very important prior that the visual system employs. To investigate the significance of this prior with respect to the other priors, it would be interesting to investigate the perception of shapes that do not always have planar faces.

In the second part of the dissertation, an algorithm that employs bilateral symmetry and planarity priors was used to recover 3D shapes from single images. Correspondence problems are common in computer vision when it comes to 3D reconstruction from images. Usually, distinctive feature points like corners are detected in images and the corresponding points are matched in multiple images before proceeding to reconstruct them. Here, to perform 3D reconstruction from a single image, we consider a different kind of correspondence problem. We assume that the shape to be reconstructed is bilaterally symmetric and then we detect symmetrically corresponding pairs of curves in the edge map of the image. If we know the internal camera

parameters and the symmetry plane, the symmetry correspondence between curves would allow us to reconstruct them up to an unknown scale factor. Note, the unknown scale factor is a feature of image based reconstructions and that appears here too. The curves are extracted by finding shortest paths with turning angle penalties in the edge map. We introduced a metric to match curves, i.e., to decide if two 2D curves are the projections of approximately planar 3D curves. Planarity prior in combination with the shape match metric is then employed to choose the right curve correspondences leading to the reconstruction of the shape.

The algorithm can be improved by automatically detecting the vanishing point. If the shape consists of curves, one could use curve matching to derive the vanishing point. Self-symmetric lines, if they can be reliably detected, could also be used for this purpose as they intersect at the vanishing point. Another issue with the algorithm is problem arising when close by 2D curves are allowed to correspond. The current algorithm solves this by imposing a minimum distance threshold between curves that can be considered symmetric counterparts, based on self-symmetric line lengths. But this solution would prevent the algorithm from reconstructing some of the details of the 3D shape. This is because it is possible for two symmetrically corresponding 3D curves to be close to each other (when they are close to the symmetry plane) and then their projections would be close to each other in the image. The distance threshold would prevent the detection of such curves as corresponding curves which would end up in inaccurate reconstructions. A solution to this problem is to match parts rather than curves. Most parts are translationally symmetric and it is possible that human visual system is detecting parts first and then finding their correspondences. An approach based on part detection and matching would exploit the local symmetry (translational symmetry) of parts and global symmetry (bilateral symmetry) of the shape to produce better reconstructions.

In the final part of the dissertation an algorithm to decompose 3D point clouds into its parts was presented. Since, the parts were represented by GCs they have natural skeletal representations. By linking the part skeletons, a skeletal representation of

the 3D shape can be obtained. To extract GCs, we explicitly used translational symmetry, the property that defines GCs. After detecting initial cross-sections, parts are grown (in both directions) from them by performing 3D point set registration with the neighboring points. An improved point set registration algorithm was introduced for this purpose. After extracting a redundant set of parts, each part was assigned a score. Based on this score an optimal subset of parts are selected to obtain an optimal decomposition of the 3D point cloud into GCs. The advantages of a part based approach over other approaches were demonstrated in the results. With the help of a GUI, the ease with which the users can interact with the skeleton extraction algorithm was demonstrated.

As pointed out earlier and depicted in Fig. 4.24, there is scope for improvement for the method used to link parts. A consistant way of dealing with the ambiguity in skeletal representation of parts (Fig. 4.25) can also be take up as future work. The accuracy of the algorithm can be improved by increasing the number of clusters considered. For instance, more detailed skeletons can be extracted by considering smaller clusters. But increasing the number of clusters would substantially increase the execution time because of the additional time spent on registration to grow larger number of parts and also because of the extra time the optimization algorithm might take due to increased number of candidate parts. However there is a way to mitigate this increase in execution time. Currently, there are a number of redundant parts in the list of candidate parts. A way to evaluate if two parts are equivalent and remove one of the parts in case they are equivalent would help reduce the number of parts passed on to the optimization algorithm. Also multiple parts can be grown simultaneously and hence a parallel implementation of this step could vastly improve the execution time. Another way to increase the speed of execution would be to look for appropriate reformulation of the optimization step so that linear relaxations to the binary integer program can be considered.

This dissertation provides further evidence for the important role that priors play in visual perception. With the abundant presence of the various symmetries in nature,

it shouldn't be surprising that the visual system employs this prior to enable veridical perception. By presenting two algorithms (one for 3D reconstruction and the other for 3D point cloud decomposition), the utility of priors, especially the symmetry prior, was demonstrated. If the field of computer vision (including machine learning based vision) aim to create algorithms to perform vision tasks like humans, it seems imperative that these algorithms explicitly try to take advantage of the symmetries that exist in the world.

REFERENCES

REFERENCES

[1] Y. Li, T. Sawada, Y. Shi, T. Kwon, and Z. Pizlo, "A bayesian model of binocular perception of 3d mirror symmetrical polyhedra," *Journal of vision*, vol. 11, no. 4, pp. 11–11, 2011.

[2] M. Singh and D. D. Hoffman, *Natural Selection and Shape Perception*. London: Springer London, 2013, pp. 171–185. [Online]. Available: https://doi.org/10.1007/978-1-4471-5195-1_12

[3] D. Purves, B. B. Monson, J. Sundararajan, and W. T. Wojtach, "How biological vision succeeds in the physical world," *Proceedings of the National Academy of Sciences*, vol. 111, no. 13, pp. 4750–4755, 2014.

[4] D. D. Hoffman, M. Singh, and C. Prakash, "The interface theory of perception," *Psychonomic bulletin & review*, vol. 22, no. 6, pp. 1480–1506, 2015.

[5] J. P. C. Southall, *Helmholtz's Treatise on Physiological Optics: Translated from the Third German Edition*. Dover, 1962.

[6] D. Marr, *Vision*. New York: W.H. Freeman, 1982.

[7] J. Hochberg and E. McAlister, "A quantitative approach, to figural" goodness"." *Journal of Experimental Psychology*, vol. 46, no. 5, p. 361, 1953.

[8] H. Wallach and D. O'connell, "The kinetic depth effect." *Journal of experimental psychology*, vol. 45, no. 4, p. 205, 1953.

[9] J. C. Hay, "Optical motions and space perception: An extension of gibson's analysis." *Psychological Review*, vol. 73, no. 6, pp. 550–565, 1966.

[10] S. Ullman, "The interpretation of structure from motion," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 203, no. 1153, pp. 405–426, 1979.

[11] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, p. 133, 1981.

[12] H. Kopfermann, "Psychologische untersuchungen über die wirkung zweidimensionaler darstellungen körperlicher gebilde." *Psychologische forschung*, vol. 13, no. 1, pp. 293–364, 1930.

[13] D. N. Perkins, "How good a bet is good form?" *Perception*, vol. 5, no. 4, pp. 393–406, 1976.

[14] A. P. Witkin, "Recovering surface shape and orientation from texture," *Artificial intelligence*, vol. 17, no. 1-3, pp. 17–45, 1981.

[15] M. Brady, A. Yuille, and W. Richards, "Inferring 3d orientation from 2d contour (an extremum principle)," *Natural computation*, pp. 99–106, 1983.

[16] K. Sugihara, *Machine interpretation of line drawings*. MIT press Cambridge, 1986, vol. 1.

[17] T. Marill, "Emulating the human interpretation of line-drawings as three-dimensional objects," *International Journal of Computer Vision*, vol. 6, no. 2, pp. 147–161, 1991.

[18] Y. G. Leclerc and M. A. Fischler, "An optimization-based approach to the interpretation of single line drawings as 3d wire frames," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 113–136, 1992.

[19] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," in *Readings in Computer Vision*. Elsevier, 1987, pp. 638–643.

[20] Z. Pizlo, Y. Li, T. Sawada, and R. Steinman, *Making a machine that sees like us*. New York, NY: Oxford University Press, 2014.

[21] I. Rock and J. DiVita, "A case of viewer-centered object perception," *Cognitive Psychology*, vol. 19, no. 2, pp. 280 – 293, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0010028587900132

[22] I. Biederman and P. C. Gerhardstein, "Recognizing depth-rotated objects: evidence and conditions for three-dimensional viewpoint invariance." *Journal of Experimental Psychology: Human perception and performance*, vol. 19, no. 6, p. 1162, 1993.

[23] D. W. Thompson, *On growth and form*, 2nd ed. Cambridge University Press, 1942.

[24] Z. Pizlo and A. K. Stevenson, "Shape constancy from novel views," *Attention, Perception, & Psychophysics*, vol. 61, no. 7, pp. 1299–1307, 1999.

[25] Y. Li and Z. Pizlo, "Depth cues versus the simplicity principle in 3d shape perception," *Topics in cognitive science*, vol. 3, no. 4, pp. 667–685, 2011.

[26] M. Hadamard, "On problems in partial derivatives, and their physical significance," *Princeton University Bulletin*, vol. 13, no. 49-52, p. 28, 1902.

[27] A. Tikhonov and V. Arsenin, *Solutions of ill-posed problems*, ser. Scripta series in mathematics. Winston, 1977. [Online]. Available: https://books.google.com/books?id=ECrvAAAAMAAJ

[28] D. C. Knill and W. Richards, *Perception as Bayesian inference*. Cambridge University Press, 1996.

[29] Z. Pizlo, "Perception viewed as an inverse problem," *Vision research*, vol. 41, no. 24, pp. 3145–3161, 2001.

[30] ——, *3D shape: Its unique place in visual perception*. Mit Press, 2010.

[31] Z. Yang and D. Purves, "A statistical explanation of visual space," *Nature neuroscience*, vol. 6, no. 6, p. 632, 2003.

[32] Y. Li, Z. Pizlo, and R. M. Steinman, "A computational model that recovers the 3d shape of an object from a single 2d retinal representation," *Vision research*, vol. 49, no. 9, pp. 979–991, 2009.

[33] T. Sawada, "Visual detection of symmetry of 3d shapes," *Journal of Vision*, vol. 10, no. 6, pp. 4–4, 2010.

[34] Z. Pizlo, T. Sawada, Y. Li, W. G. Kropatsch, and R. M. Steinman, "New approach to the perception of 3d shape based on veridicality, complexity, symmetry and volume," *Vision research*, vol. 50, no. 1, pp. 1–11, 2010.

[35] J. T. Todd and P. Bressan, "The perception of 3-dimensional affine structure from minimal apparent motion sequences," *Perception & Psychophysics*, vol. 48, no. 5, pp. 419–430, 1990.

[36] M. W. Chan, A. K. Stevenson, Y. Li, and Z. Pizlo, "Binocular shape constancy from novel views: The role of a priori constraints," *Attention, Perception, & Psychophysics*, vol. 68, no. 7, pp. 1124–1139, 2006.

[37] I. Howard and B. Rogers, "Seeing in depth: Depth perception, vol. 1," 2002.

[38] T. Sawada, Y. Li, and Z. Pizlo, "Any pair of 2D curves is consistent with a 3D symmetric interpretation," *Symmetry*, vol. 3, no. 2, pp. 365–388, 2011.

[39] H. Zabrodsky and D. Weinshall, "Using bilateral symmetry to improve 3d reconstruction from image sequences," *Computer vision and image understanding*, vol. 67, no. 1, pp. 48–57, 1997.

[40] A. R. François, G. G. Medioni, and R. Waupotitsch, "Mirror symmetry? 2-view stereo geometry," *Image and Vision Computing*, vol. 21, no. 2, pp. 137–143, 2003.

[41] H. Mitsumoto, S. Tamura, K. Okazaki, N. Kajimi, and Y. Fukui, "3-d reconstruction using mirror images based on a plane symmetry recovering method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 9, pp. 941–946, 1992.

[42] N. Jiang, P. Tan, and L.-F. Cheong, "Symmetric architecture modeling with a single image," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5. ACM, 2009, p. 113.

[43] K. Köser, C. Zach, and M. Pollefeys, "Dense 3d reconstruction of symmetric scenes from a single image," in *Joint Pattern Recognition Symposium*. Springer, 2011, pp. 266–275.

[44] C. Wu, J.-M. Frahm, and M. Pollefeys, "Repetition-based dense single-view reconstruction," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3113–3120.

[45] S. N. Sinha, K. Ramnath, and R. Szeliski, "Detecting and reconstructing 3d mirror symmetric objects," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 586–600.

[46] 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*. IEEE, 2011, pp. 2577–2584.

[47] S. T. Barnard, "Interpreting perspective images," *Artificial intelligence*, vol. 21, no. 4, pp. 435–462, 1983.

[48] J. P. Tardif, "Non-iterative approach for fast and accurate vanishing point detection," in *2009 IEEE 12th International Conference on Computer Vision*, Sept 2009, pp. 1250–1257.

[49] B. Li, K. Peng, X. Ying, and H. Zha, *Simultaneous Vanishing Point Detection and Camera Calibration from Single Images*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 151–160. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17274-8_15

[50] Y. Xu, S. Oh, and A. Hoogs, "A minimum error vanishing point detection approach for uncalibrated monocular images of man-made environments," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, June 2013, pp. 1376–1383.

[51] H. Wildenauer and A. Hanbury, "Robust camera self-calibration from monocular images of manhattan worlds," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2831–2838.

[52] B. Li, K. Peng, X. Ying, and H. Zha, "Vanishing point detection using cascaded 1d hough transform from single images," *Pattern Recognition Letters*, vol. 33, no. 1, pp. 1–8, 2012.

[53] A. Almansa, A. Desolneux, and S. Vamech, "Vanishing point detection without any a priori information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 4, pp. 502–507, 2003.

[54] J. M. Coughlan and A. L. Yuille, "Manhattan world: Orientation and outlier detection by bayesian inference," *Neural Computation*, vol. 15, no. 5, pp. 1063–1088, 2003.

[55] J. Lezama, R. Grompone von Gioi, G. Randall, and J.-M. Morel, "Finding vanishing points via point alignments in image primal and dual domains," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 509–515.

[56] P. Denis, J. H. Elder, and F. J. Estrada, "Efficient edge-based methods for estimating manhattan frames in urban imagery," in *European conference on computer vision*. Springer, 2008, pp. 197–210.

[57] R. T. Collins and R. S. Weiss, "Vanishing point calculation as a statistical inference on the unit sphere," in *[1990] Proceedings Third International Conference on Computer Vision*. IEEE, 1990, pp. 400–403.

[58] A. Michaux and Z. Pizlo, "Two correspondence problems easier than one," in *Computational and Mathematical Models in Vision (MODVIS)*, 2015. [Online]. Available: http://docs.lib.purdue.edu/modvis/2015/session03/5/

[59] T. Kwon, K. Agrawal, Y. Li, and Z. Pizlo, "Spatially-global integration of closed, fragmented contours by finding the shortest-path in a log-polar representation," *Vision research*, vol. 126, pp. 143–163, 2016.

[60] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: http://www.gurobi.com

[61] T. Sawada, T. Li, and Z. Pizlo, "Detecting 3-d mirror symmetry in a 2-d camera image for 3-d shape recovery," *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1588–1606, 2014.

[62] T. Lee, S. Fidler, A. Levinshtein, C. Sminchisescu, and S. Dickinson, "A framework for symmetric part detection in cluttered scenes," *Symmetry*, vol. 7, no. 3, pp. 1333–1351, 2015.

[63] A. Levinshtein, C. Sminchisescu, and S. Dickinson, "Multiscale symmetric part detection and grouping," *International journal of computer vision*, vol. 104, no. 2, pp. 117–134, 2013.

[64] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, "Modeling by example," in *ACM transactions on graphics (TOG)*, vol. 23, no. 3. ACM, 2004, pp. 652–663.

[65] I. Baran and J. Popović, "Automatic rigging and animation of 3d characters," in *ACM Transactions on graphics (TOG)*, vol. 26, no. 3. ACM, 2007, p. 72.

[66] G. Li, L. Liu, H. Zheng, and N. J. Mitra, "Analysis, reconstruction and manipulation using arterial snakes," *ACM Trans. Graph.*, vol. 29, no. 6, p. 152, 2010.

[67] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or, "3-sweep: Extracting editable objects from a single photo," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 195, 2013.

[68] K. Yin, H. Huang, H. Zhang, M. Gong, D. Cohen-Or, and B. Chen, "Morfit: interactive surface reconstruction from incomplete point clouds with curve-driven topology and geometry control." *ACM Trans. Graph.*, vol. 33, no. 6, pp. 202–1, 2014.

[69] I. Binford, "Visual perception by computer," in *IEEE Conference of Systems and Control*, 1971.

[70] A. Sharf, T. Lewiner, A. Shamir, and L. Kobbelt, "On-the-fly curve-skeleton computation for 3d shapes," in *Computer Graphics Forum*, vol. 26, no. 3. Wiley Online Library, 2007, pp. 323–328.

[71] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee, "Skeleton extraction by mesh contraction," in *ACM transactions on graphics (TOG)*, vol. 27, no. 3. ACM, 2008, p. 44.

[72] D. Reniers, J. Van Wijk, and A. Telea, "Computing multiscale curve and surface skeletons of genus 0 shapes using a global importance measure," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 355–368, 2008.

[73] Y. Zhou, K. Yin, H. Huang, H. Zhang, M. Gong, and D. Cohen-Or, "Generalized cylinder decomposition." *ACM Trans. Graph.*, vol. 34, no. 6, pp. 171–1, 2015.

[74] A. Myronenko and X. Song, "Point set registration: Coherent point drift," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 12, pp. 2262–2275, 2010.

[75] S. Billings and R. Taylor, "Generalized iterative most likely oriented-point (g-imlop) registration," *International journal of computer assisted radiology and surgery*, vol. 10, no. 8, pp. 1213–1226, 2015.

[76] A. Tagliasacchi, T. Delame, M. Spagnuolo, N. Amenta, and A. Telea, "3d skeletons: A state-of-the-art report," in *Computer Graphics Forum*, vol. 35, no. 2. Wiley Online Library, 2016, pp. 573–597.

[77] N. D. Cornea, D. Silver, and P. Min, "Curve-skeleton properties, applications, and algorithms," *IEEE Transactions on Visualization & Computer Graphics*, no. 3, pp. 530–548, 2007.

[78] J.-H. Chuang, N. Ahuja, C.-C. Lin, C.-H. Tsai, and C.-H. Chen, "A potential-based generalized cylinder representation," *Computers & Graphics*, vol. 28, no. 6, pp. 907–918, 2004.

[79] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. Zucker, "The hamilton-jacobi skeleton," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2. IEEE, 1999, pp. 828–834.

[80] C. Song, Z. Pang, X. Jing, and C. Xiao, "Distance field guided l_1-median skeleton extraction," *The Visual Computer*, vol. 34, no. 2, pp. 243–255, 2018.

[81] H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, and B. Chen, "L1-medial skeleton of point cloud." *ACM Trans. Graph.*, vol. 32, no. 4, pp. 65–1, 2013.

[82] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii, "Topology matching for fully automatic similarity estimation of 3d shapes," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001, pp. 203–212.

[83] M. Natali, S. Biasotti, G. Patanè, and B. Falcidieno, "Graph-based representations of point clouds," *Graphical Models*, vol. 73, no. 5, pp. 151–164, 2011.

[84] A. Bucksch, R. Lindenbergh, and M. Menenti, "Skeltre," *The Visual Computer*, vol. 26, no. 10, pp. 1283–1300, 2010.

[85] T. K. Dey and J. Sun, "Defining and computing curve-skeletons with medial geodesic function," in *Proceedings of the fourth Eurographics symposium on Geometry processing*. Eurographics Association, 2006, pp. 143–152.

[86] R. Ogniewicz and M. Ilg, "Voronoi skeletons: Theory and applications," in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1992, pp. 63–69.

[87] A. Tagliasacchi, H. Zhang, and D. Cohen-Or, "Curve skeleton extraction from incomplete point cloud," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3. ACM, 2009, p. 71.

[88] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su, "Point cloud skeletons via laplacian based contraction," in *2010 Shape Modeling International Conference*. IEEE, 2010, pp. 187–197.

[89] A. Nguyen and B. Le, "3d point cloud segmentation: A survey," in *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*. IEEE, 2013, pp. 225–230.

[90] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," in *Computer graphics forum*, vol. 26, no. 2. Wiley Online Library, 2007, pp. 214–226.

[91] M. Attene, B. Falcidieno, and M. Spagnuolo, "Hierarchical mesh segmentation based on fitting primitives," *The Visual Computer*, vol. 22, no. 3, pp. 181–193, 2006.

[92] X. Li, T. W. Woon, T. S. Tan, and Z. Huang, "Decomposing polygon meshes for interactive applications," in *Proceedings of the 2001 symposium on Interactive 3D graphics*. ACM, 2001, pp. 35–42.

[93] M. Goyal, S. Murugappan, C. Piya, W. Benjamin, Y. Fang, M. Liu, and K. Ramani, "Towards locally and globally shape-aware reverse 3d modeling," *Computer-Aided Design*, vol. 44, no. 6, pp. 537–553, 2012.

[94] "Cloudcompare (version 2.9.1) [gpl software]," http://www.cloudcompare.org/, 2019.

[95] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python ," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[96] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed ¡today¿]. [Online]. Available: http://www.scipy.org/

[97] G. Terzakis, P. Culverhouse, G. Bugmann, S. Sharma, and R. Sutton, "A recipe on the parameterization of rotation matrices for non-linear optimization using quaternions," Technical report, Technical report MIDAS. SMSE. 2012. TR. 004, Tech. Rep., 2012.

[98] D. Q. Huynh, "Metrics for 3d rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009.

[99] J. Zhang, K. Siddiqi, D. Macrini, A. Shokoufandeh, and S. Dickinson, "Retrieving articulated 3-d models using medial surfaces and their graph spectra," in *International workshop on energy minimization methods in computer vision and pattern recognition*. Springer, 2005, pp. 285–300.

[100] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "Objectnet3d: A large scale database for 3d object recognition," in *European Conference Computer Vision (ECCV)*, 2016.

[101] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, ser. LLVM '15. New York, NY, USA: ACM, 2015, pp. 7:1–7:6. [Online]. Available: http://doi.acm.org/10.1145/2833157.2833162

APPENDICES

A. PROOF: ONE-PARAMETER FAMILY OF 3D SYMMETRICAL SHAPES ARE INCLUDED IN THE FAMILY OF SHAPES THAT THE USER CAN RECONSTRUCT

The XYZ Cartesian coordinate system of a 3D scene and the xy Cartesian coordinate system of a 2D image in the scene are set as follows: (i) the Z-axis of the 3D coordinate system is perpendicular to the image plane π_I and π_I is Z = 0, (ii) the Z-axis passes through the origin of the 2D coordinate system, and (iii) the X- and Y-axes of the 3D coordinate system coincide with the x- and y-axes of the 2D coordinate system, respectively. Under an orthographic projection, a 2D orthographic projection $[x, y]^T$ of a point $[X, Y, Z]^T$ in a 3D scene is $[x, y]^T = [X, Y]^T$. Assume an object with a 3D symmetric shape is given. Its symmetry plane is perpendicular to the image plane π_I and normal to the X-axis. A 3D asymmetric shape used in this study was generated by deforming the 3D symmetric shape using a sub-group of 3D Affine transform:

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} t_{11} & 0 & 0 \\ 0 & 1 & 0 \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{A1}$$

where t_{11}, t_{31}, t_{32}, and t_{33} are positive real numbers. This symmetric shape is referred as the "original" symmetric shape for clarity. After this transformation, the original symmetric shape becomes asymmetric unless $t_{31} = 0$. The "symmetry" plane of the original symmetric shape is not normal to its "symmetry" line-segments anymore and they are referred as an "asymmetry plane" and "asymmetry line-segments" after the transformation. The asymmetry line segments are still parallel to one another and are bisected by the asymmetry plane.

Consider that this asymmetric shape is randomly rotated and is orthographically projected to the image plane π_I along the Z-axis. Assume that the projections of the asymmetry plane and the asymmetry line-segments are "not" degenerate. Namely, the asymmetry plane is not perpendicular to π_I and the asymmetry line-segments are not normal to π_I. If their projections are degenerate, the asymmetry plane is projected to a line in the image and the asymmetry line-segment is projected to a point in the image.

The orientations of the X- and Y-axes of the 3D coordinate system and of the x- and y-axes of the 2D coordinate system are re-set so that the X- and x-axes are parallel with orthographic projections of the asymmetry line segments. Then, an orientation of the asymmetry line segments can be represented by a vector $L_N = [1, 0, Z_N]^T$. Consider two vectors $L_{A1} = [1, 0, Z_{A1}]^T$ and $L_{A2} = [0, 1, Z_{A2}]^T$ that are parallel to the asymmetry plane.

A subject is asked to choose a 3D shape from a sub-group of Affine transform of the asymmetric shape in the experiment. The sub-group of 3D Affine transform is controlled by three parameters and is represented by a matrix T_R:

$$T_R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{A2}$$

where a_{31}, a_{32}, and a_{33} are positive real numbers that the subject can adjust. The vectors L_N, L_{A1}, and L_{A2} are transformed by T_R into $T_R L_N = [1, 0, +Z_T Z_N]^T, T_R L_{A1} = [1, 0, +Z_T Z_{A1}]^T$, and $T_R L_{A2} = [0, 1, +Z_{A2}]^T$ respectively. Note that the orthographic image of the asymmetric shape on π_I is an invariant of T_R. The images of the asymmetry line segments stay parallel to the X-axis under T_R.

In the following part of this appendix, we will show there is always some 3D symmetric shape that can be obtained by transforming the asymmetric shape by T_R. Symmetry line-segments and a symmetry plane of this symmetric shape are transformed from the asymmetry line-segments and the asymmetry plane of the asym-

metric shape. Namely, $T_R L_{A1}$ and $T_R L_{A2}$ are parallel to the symmetry plane, $T_R L_N$ is parallel to the symmetry line-segments, and $T_R L_N$ is perpendicular to $T_R L_{A1}$ and $T_R L_{A2}$ and is normal to the symmetry plane. Note that the asymmetric shape is in the sub-group of 3D Affine transform (Equation A1) of the original symmetric shape and Equation (A2) is another sub-group of 3D Affine transform. Hence, the asymmetry line-segments still connect pairs of points that were symmetrical pairs of the original symmetric shape. The asymmetry line-segments are still parallel to one another and are bisected by the asymmetry plane but the asymmetry line-segments are not normal to the symmetry plane. Hence, T_R transforms the asymmetric shape into a symmetric shape if $T_R L_N$ is perpendicular to $T_R L_{A1}$ and $T_R L_{A2}$.

Recall that the images of the asymmetry line-segments are parallel to the X-axis. Under this condition, the symmetry plane of the symmetric shape transformed from the asymmetric shape is perpendicular to the ZX-plane [33,34]. The Y-axis is parallel to $T_R L_{A2}$.

$$T_R L_{A2} = \begin{bmatrix} 0 \\ 1 \\ a_{32} + a_{33} Z_{A2} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{A3}$$

and is perpendicular to $T_R L_{A1}$ and $T_R L_N$. Then, the orientation of the symmetry plane can be determined by its slant from the Z-axis. The slant σ_N of the symmetry plane can be defined as an angle between the Z-axis and the normal $T_R L_N$ to the symmetry plane:

$$T_R L_N = \begin{bmatrix} 1 \\ 0 \\ a_{31} + a_{33} Z_N \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \frac{1}{tan(\sigma_N)} \end{bmatrix} \tag{A4}$$

Recall that $T_R L_{A1}$ and $T_R L_N$ are perpendicular to one another and $T_R L_{A1}$ is perpendicular to the Y-axis. Hence,

$$T_R L_{A1} = \begin{bmatrix} 1 \\ 0 \\ a_{31} + a_{32} Z_{A1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -tan(\sigma_N) \end{bmatrix} \tag{A5}$$

From Equations (A3), (A4), and (A5), T_R for transforming the asymmetric shape into the symmetric shape can be derived as follows:

$$T_R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{-sin^2(\sigma_N) Z_N - cos^2(\sigma_N) Z_{A1}}{D_T} & \frac{Z_{A2}}{D_T} & \frac{1}{D_T} \end{bmatrix} \tag{A6}$$

where $D_T = sin(\sigma_N) cos(\sigma_N)(Z_N - Z_{A1})$. Such T_R exists unless $Z_N = Z_{A1}$. If $Z_N = Z_{A1}$, the asymmetric line-segments are parallel to the asymmetry plane. Equation (A6) shows there is a one-parameter family of 3D symmetric shapes that can be transformed from the asymmetric shape. The family can be controlled by the slant σ_N of the symmetry plane. These symmetric and asymmetric shapes can be projected to the identical image.

VITA

VITA

Vijai Jayadevan received his bachelor's degree from Cochin University, India in 2008 and his master's degree from The University of Arizona in 2013, both in Electrical and Computer Engineering (ECE). He is currently working toward his PhD in ECE at Purdue University, West Lafayette, and is being advised by Prof. Edward Delp from the ECE Department and Prof. Zygmunt Pizlo from the Department of Psychological Sciences. His research interests include computer vision, computer graphics, machine learning, and signal processing.

Publications

V. Jayadevan, E. Delp, and Z. Pizlo, "Skeleton Extraction from 3D Point Clouds by Decomposing the Object into Parts," *arXiv e-prints*, p. arXiv:1912.11932, Dec 2019.

V. Jayadevan, T. Sawada, E. Delp, and Z. Pizlo, "Monocular and binocular recovery of 3D symmetrical and near-symmetrical shapes," *Journal of Vision*, vol. 18, no. 10, pp. 719–719, 2018.

V. Jayadevan, T. Sawada, E. Delp, and Z. Pizlo, "Perception of 3d symmetrical and nearly symmetrical shapes," *Symmetry*, vol. 10, no. 8, p. 344, 2018.

A. Michaux, V. Kumar, V. Jayadevan, E. Delp, and Z. Pizlo, "Binocular 3D object recovery using a symmetry prior," *Symmetry*, vol. 9, no. 5, p. 64, 2017.

V. Jayadevan, A. Michaux, E. Delp, and Z. Pizlo, "3D shape recovery from real images using a symmetry prior," in *IS and T International Symposium on Electronic Imaging Science and Technology, Computational Imaging XV*, pp. 106–115, 2017.

V. Jayadevan, A. Michaux, E. Delp, and Z. Pizlo, "3-d shape recovery from a single camera image," in *Coputational and Mathematical Models in Vision*, 2016.

A. Michaux, V. Jayadevan, E. J. Delp, and Z. Pizlo, "Figure-ground organization based on three-dimensional symmetry," *Journal of Electronic Imaging*, vol. 25, no. 6, p. 061606, 2016.