Planar Section Representations of 3D Shape

by

James McCrae

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

© Copyright 2014 by James McCrae

Abstract

Planar Section Representations of 3D Shape

James McCrae
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2014

Planes are fundamental geometric primitives. Evidence in art and science shows that planar sections are useful – structures composed purely of planar primitives are employed in architecture, design and manufacturing, where one can with relative ease physically create a model from planar parts. In medical visualization, patient-aligned planes may suffice to represent important features within a volumetric dataset. Many different applications involving 3D shape can geometrically benefit from a contextually-appropriate piecewise planar representation. This thesis systematically studies planar section representations of 3D shape.

We show that humans are consistent in abstracting existing 3D shapes using a minimal number of planar 3D sections. We provide geometric insights that allow us to replicate human planar abstraction of 3D shapes algorithmically, and evaluate the algorithm's performance with a shape recognition study.

We study how humans create planar shape representations interactively from scratch. Observations of their interaction patterns are used to develop a streamlined interactive drawing system. The system is further augmented to facilitate procedural creation, incorporate existing geometry and aid fabrication. A physical simulation approach identifies structurally weak models before they are fabricated, and allows interactive refinement.

We then study the perceptual efficacy of a range of planar representations of 3D shape in a large-scale user study. The analysis of results reveals geometric sources responsible for error, and we use these sources to create predictive linear models of error. With this knowledge, we demonstrate we are able to improve the quality of algorithmically-created representations.

Finally we present a number of applications of planar section representations. Some applications are artistic in nature, such as paper statues, puppets, detailed compositions and scanimations. Other applications are more scientific, including surface reconstruction and compression, 3D annotation, volumetric data visualization, 3D navigation, and poly-postors.

Overall, the dissertation provides a comprehensive treatment of various aspects of planar section representations of 3D shape and their applications. We provide both qualitative and quantitative perceptual validation that planar section representations are indeed effective proxies of 3D shape, useful in both artistic and scientific settings.

Contents

Chapter 1

Introduction

1.1 Motivation

Over the last few decades, significant progress been made in the acquisition and modelling of 3D geometry. The underlying shape representation is typically a collection of points or polygons, popular for their generality, rendering efficiency, and amenability to geometry-processing algorithms. Unfortunately, such a representation can be expensive and, in itself, neither conveys the essence of the depicted object nor aids in our understanding of the represented shape.

Minimalist object representations or *shape-proxies* that spark and inspire human perception of shape remain an incompletely understood, yet powerful aspect of visual communication. We explore the use of planar sections for creating shape representations, motivated by their popularity in art and engineering.

What is a *planar section representation*? Mathematically, a planar section representation is a non-empty set of planar sections. Each *planar section* is a (possibly disconnected) compact (closed and bounded) subspace of $\mathbb{R}^2$ with a planar embedding in $\mathbb{R}^3$. A planar section is defined geometrically by a unique 3D-embedded plane and one or more closed boundary curves that lie on the plane. No curve self-intersects and there are no intersections between any two coplanar curves.

Intuitively, we consider a planar section as being (in general, close to) the result of intersecting a plane with a hidden, abstracted surface, which acts as an enveloping shape to define some interior volume. We discuss the examples presented in Figure 1.1. For the first example (the fish), the planar section representation can be thought to define a skeleton of some smooth shape, which is perhaps slightly offset and contains the skeleton like a skin. For the second example (the articulated human), the planar sections have boundaries that largely lie on the surface except at specific regions of articulation (e.g., the hips) where the sections have been cut and closed. For the third example (the ant), the planar section boundaries lie precisely on the abstracted surface.

All three are examples of planar section representations. Examples where planar section boundaries lie precisely on an abstracted surface are the focus of Chapters 4 and 6, where

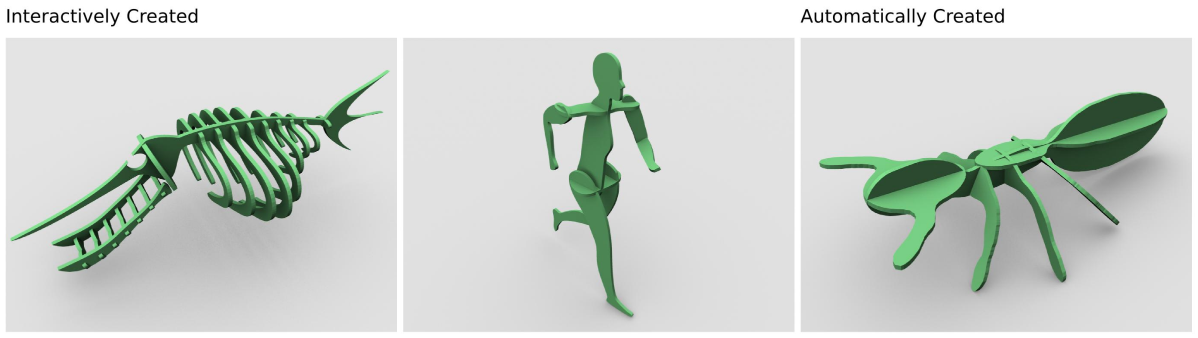


Figure 1.1: Three examples of planar section representations. Two examples (left, centre) were created interactively, the last (right) was created algorithmically.

we cover automatic creation and surface perception. The first two examples contain planar sections typical of an interactive creation process, and where the representation is meant to be fabricated. Such interactively created planar section representations are the focus of Chapter 5.

In geometry processing, model simplification [25] and variational shape approximation [28] remain the dominant modes of shape abstraction. Recently, there has been work on shape abstraction using curve networks [32, 100]. These methods and their variants evaluate the quality of an approximation by the geometric deviation of the proxy from the original shape. In contrast, we aim to explore shape abstractions that are based on human perception of form. Given a shape S, our goal is to produce a proxy $\mathcal{S}$ using planar sections of the shape, such that perceptually S and $\mathcal{S}$ are comparable, while $|\mathcal{S}| \ll |S|$. Note that $\mathcal{S}$ and S are likely to be quite different from a purely geometric or topological standpoint. The fundamental difficulty in proposing or evaluating algorithmic solutions to such a problem is the lack of a computational model of our perceptual response system.

The fascinating aspect of planar section representations is that they are an effective form of shape representation in spite of their minimalism. Planar section representations are geometrically very distant from the original surface in a Euclidean sense, yet often we can still clearly recognize the object being represented. Thus the planar section representation must touch upon a deep, but incompletely-understood pathway by which the human brain interprets shape. The question of how the human mind interprets 3D shape in a high-level, abstract way is one of the key questions in the area of visual perception. While the content presented here focuses on planar section representations within a computer graphics context, we detail perceptually motivated original contributions, observations and insights that extend beyond computer graphics and into the area of perceptual psychology.

Evidence suggests that symbolic abstractions [39] dominate our mental model of objects. Thus across cultures, we both recognize shape proxies quickly and tend to communicate objects by drawing them as symbolic abstractions. Artists and sculptors have long explored minimalist shape proxies, including planar section representations, to highlight defining aspects of familiar objects and examples of such representations can be found throughout the world. One of the

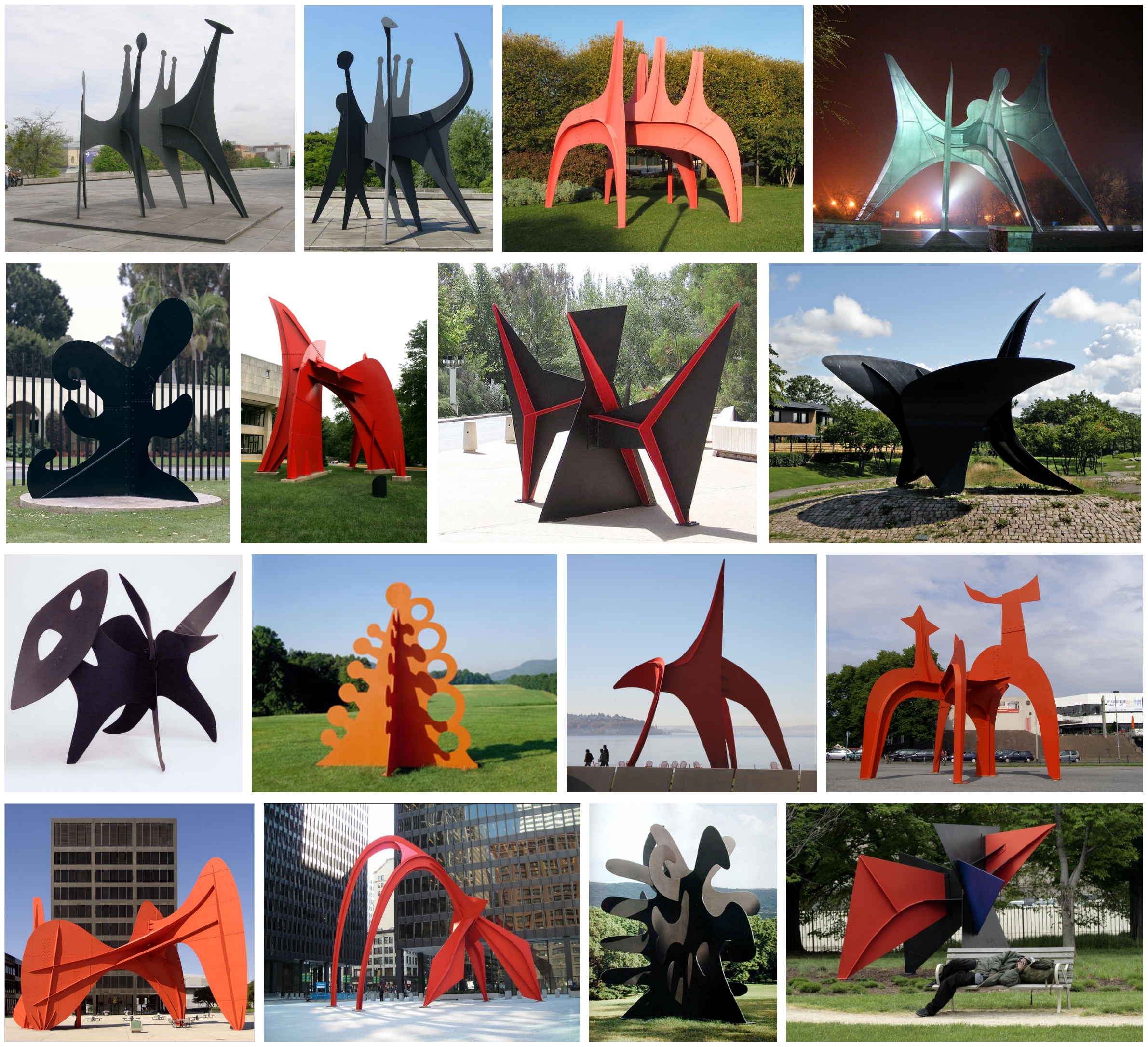


Figure 1.2: Examples of planar sections in art (©Alexander Calder).

more prominent creators of such artistic creations in the past century is the well-known sculptor Alexander Calder, and some examples of his work are shown in Figure 1.2.

Sculptures that consist of principally planar sections can also be found in many contemporary settings. One reason in particular for this is that planar parts can be fabricated with ease using computer numerical control (CNC) machines and the computer aided design and manufacturing programs available. Figure 1.3 shows artistic sculptures that are also functional furnishings that could be found in a home.

Figure 1.4 shows other decorative examples that could be wall mounted. As humans, we can with little effort perceive the underlying shapes even from such sectional representations, despite the fact that they greatly differ from their surface representations. Also, the sparseness of these representations allows us to see otherwise occluded details. The absence of unnecessary detail makes such art forms attractive, fascinating, and sometimes mysterious.

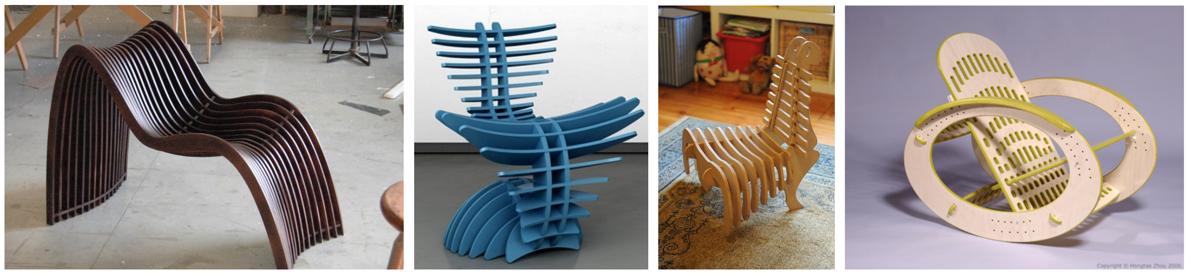


Figure 1.3: Examples of planar section sculptures in design (left section chair ©Hebert Franco, right rocking chair ©Hongtao Zhou).

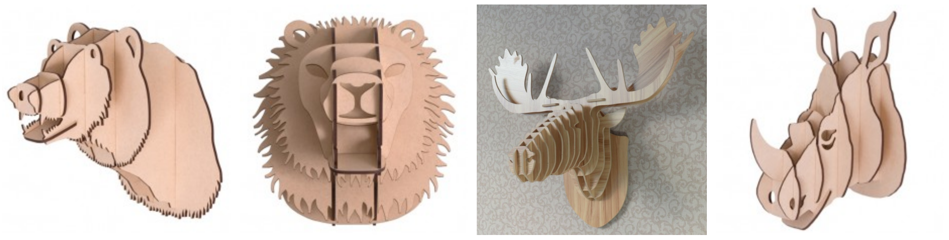


Figure 1.4: Decorative planar section sculptures suitable for wall mounting.

There are also many architectural examples of interior design where planar sections together define the structure and shape of the room itself (see Figure 1.5). Such interior designs are surprisingly economical to produce and offer the designer considerable artistic freedom, as even smooth undulating surfaces can be represented.

Planar section proxies are also motivated by medical and engineering visualization where section planes are used to illustrate the interior details of complex shapes (see Figure 1.6). These planar sections often pass through anatomic landmarks or engineering features – such as the axis of revolution of the output shaft of an internal combustion engine, or in the case of medical visualization the planes will be defined along coronal, sagittal or transverse (axial) directions, which refer to the basis vectors of a patient-aligned referential frame. This observation reaffirms our use of high-level shape features like segments, symmetries, ridges and valleys that we come to rely on in order to automatically define planar sections in Chapter 4.

In interactive drawing applications, planes are heavily used as the drawing canvas. This choice is likely rooted in the fact that most input devices and displays themselves are planar (see Figure 1.7). Planar section representations then are potentially of great use in any applications where users interact with 3D shape.

To motivate the work on a more technical level, we consider a number of open research questions involving planar section representations. Some questions relate to the feasibility and performance of automatic creation of planar section representations using algorithmic means:

- Can a computer be used to automatically generate planar section representations of surfaces representing common objects?

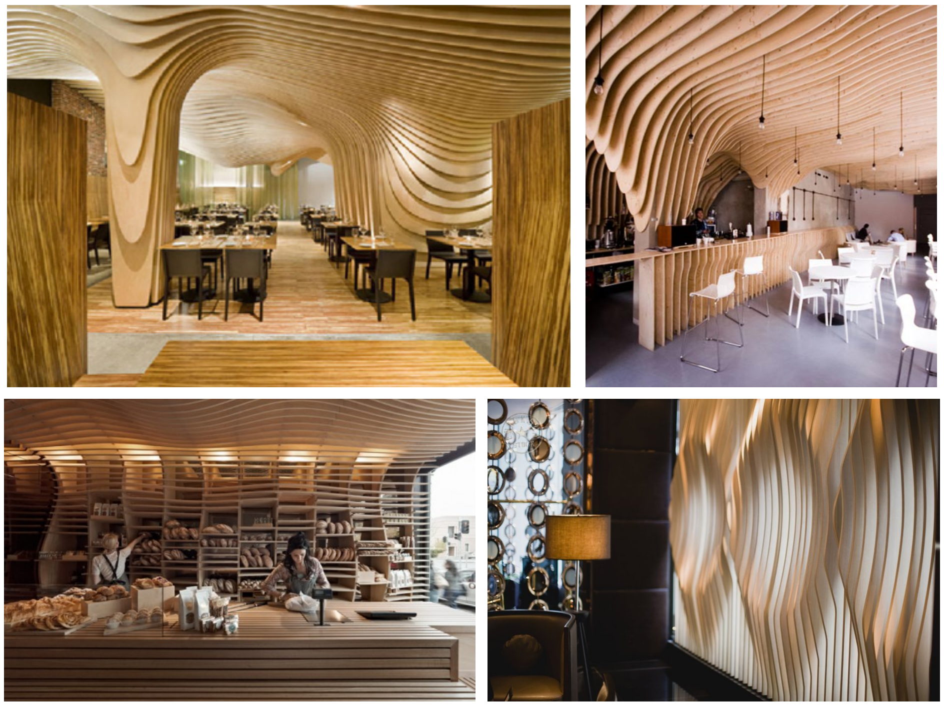


Figure 1.5: Architectural interior design examples that consist of planar sections of wood.

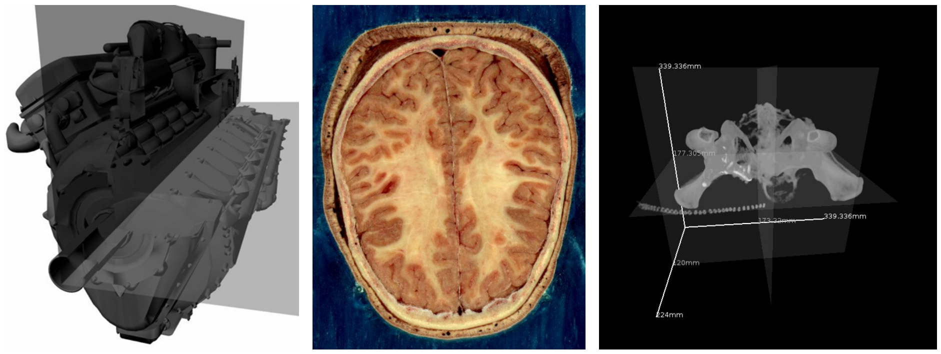


Figure 1.6: Planar sections can be used to take one or more "slices" from a volumetric dataset to aid in visualization.

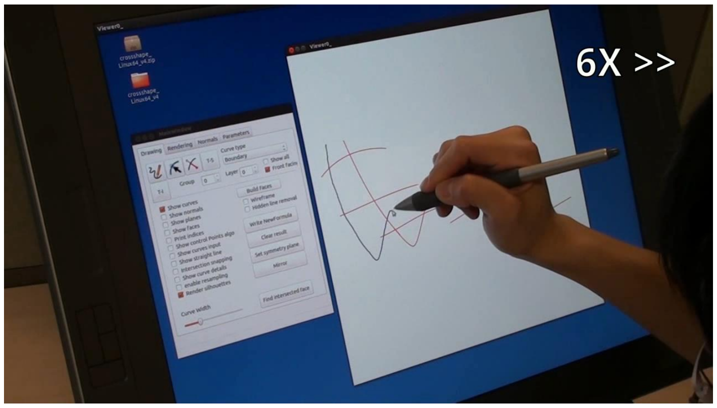


Figure 1.7: A user interacts with the computer using a touchscreen device and a stylus. The display and input device are both examples of planar devices.

- Can computer-generated abstractions be made that mimic how humans create planar section representations (that is, assuming humans even agree)?
- Do humans recognize the objects represented by planar section representations, even when they are computer-generated?

Other questions motivate work on the interactive systems and techniques that can be used to create planar section representations:

- Can we learn from human-drawn examples?
- Are planar sections useful primitives for geometric modelling?
- Can planar section representations be used to represent shapes that do not consist of principally planar sections or are highly articulated?

Some questions motivate research to deepen understanding of the visual perception of planar section representations:

- How well is the surface represented by a planar section representation perceived?
- What underlying geometric properties relate to errors in perception of the represented surface?
- Can the errors in human perception be predicted?
- How can planar section representations be improved?

Finally, we might ask:

- What are the potential practical applications of planar section representations?

The above questions cover a broad range of topics relating to planar section representations and serve to motivate the work presented in this dissertation.

1.2 Thesis and Contributions

Our thesis is the following:

> Evidence in both art and science suggests that planar section representations can be effective as a shape abstraction and for understanding 3-dimensional shape. Planar section representations are effective abstractions of shape, they can be easily constructed interactively or algorithmically, and they are applicable to a variety of domains in art and science.

The work contained in this dissertation has both introduced and stimulated research on planar section representations in the computer graphics community. Our intention in the dissertation is to explore the planar section representation deeply, while covering the breadth of critical subjects such as creation, perception and application thus establishing the truth of our thesis. Specifically, the contributions made in this dissertation surround (i) the first perceptually-driven approach to the automatic creation of recognizable planar section representations; (ii) the first study on the perception of planar section representations by measurement of represented surface properties; (iii) the design of an interactive system that allows planar section representations to be modelled effectively, using novel interaction techniques unique to planar sections; and (iv) presenting practical applications of planar section representations across a variety of disciplines to convey their usefulness, some of which we believe to be novel.

1.3 Overview

We first provide the necessary background. In Chapter 2, we review the related work in a number of different sub-fields, including: visual perception of shape, shape abstraction, shape modelling, and fabrication of piecewise planar objects. In Chapter 3, we provide the necessary technical background for reading the remainder of the thesis, such as: the mathematical representations we employ for planes, curves, surfaces and other geometric structures, and details on the geometric features of a surface we consider and how to derive them. We distinguish between 3 important categories of shape representation: *low-level* shape representations, *high-level* shape representations, and shape *abstractions*.

We cover in detail two fundamental approaches toward the creation of planar section representations (see Figure 1.8). In Chapter 4, we focus on the task of automatic, algorithmic construction of planar section representations given an input surface. Chapter 5 explores the interactive, manual construction of planar section representations; we consider interactive techniques specifically for those situations where either no input surface is provided (a creation "from scratch"), or the situation where an input surface is provided that can be used as a base template the designer is free to deviate from. We additionally provide a treatment for input surfaces that are not directly suitable for representation using planar sections.

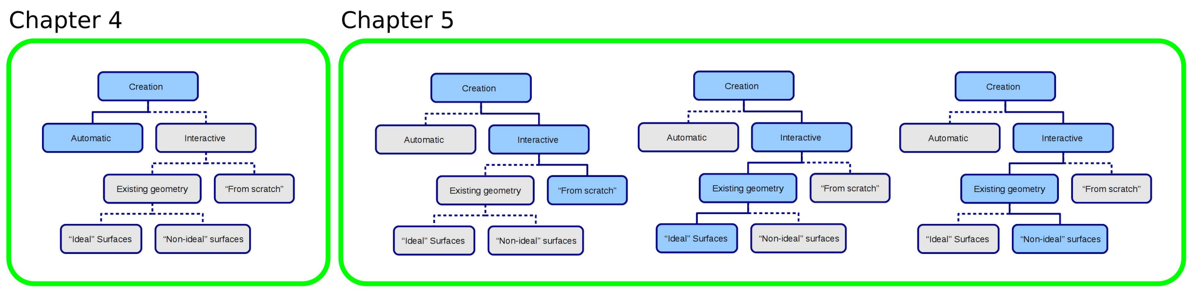


Figure 1.8: Four possible paths to creating planar section representations that are covered in Chapters 4 and 5.

We change focus in Chapter 6 to explore the visual perception of created planar section representations in much greater detail. At the core of this chapter is a large-scale user study that involves an established gauge figure setting task from the area of perceptual psychology [81]. We discover a number of geometric properties that relate to one's ability to perceive surface shape. We employ these observed relationships in predictive models that greatly improve the prediction of perception error over a baseline estimate, and further demonstrate that this knowledge can be used to improve the *process* of creating planar section representations, to improve surface perception.

In Chapter 7, we consider and demonstrate the wide variety of potential applications for planar section representations. We apply planar section representations as a solution to generic problems in data visualization, real-time rendering, and manufacturing, and domains beyond computer graphics such as in medicine, engineering, architecture and the visual arts.

Finally, in Chapter 8, we conclude the dissertation and review the contributions made. We discuss the limitations of the approaches presented, and suggest possible directions for future work.

Chapter 2

Related Work

In this chapter, we review work broadly relating to planar section representations. Each section roughly categorizes the previous work – but the subjects of perception, representation and abstraction of shape are all connected and there is significant overlap. We review existing approaches that, like planar section representations, perform the task of shape abstraction (Section 2.1). The *process* of creating planar section representations is of great importance, so we review many of the existing systems and interfaces designed for the task of modelling 3D geometry (Section 2.2). We review perception and vision research that strongly motivates our desire to represent 3D shape using planar sections (Section 2.3). Finally, the ease of fabrication of planar section representations is a major strength. We consider previous work relating to fabrication and manufacturing, primarily by using planar primitives (Section 2.4).

2.1 Shape Abstraction

Curve shape proxies

Curve shape proxies have been used as a shape abstraction in traditional art for centuries, and for decades now designers have used networks of curves to effectively model 3D shapes. Inspired by traditional design drawing [40], much of interactive 3D conceptual design has been influenced by techniques to build sparse 3D curve representations of shape [4, 124]. Design curves are perceptually consistent [132] and design cross-sections in particular are in fact planar and perceptually important for conveying 3D shape.

Mehra et al. [100] detail an approach to effectively compute curve abstractions from 3D objects of man-made shapes. The approach, which involves the computation of an enveloping surface using a voxel-based method, smooths narrow concavities, fills small holes and generally acts to suppress model detail. A three-step iterative approach performs matching, deformation and mesh regularization of the enveloping surface. The result of this iterative process is a manifold triangle mesh that approximates the topology and geometry of the input model, up to a desired resolution. A curve network is derived using a VSA approach [28] on the enveloping surface to find a near-planar segmentation. The curve network is regularized and simplified, and surfaces are fit to each closed loop in the geometric reconstruction.

Skeletons

Skeletons are shape abstractions formed by computing the medial axis of a surface. The original work of Blum [14] uses the analogy of a grass fire that spreads. The medial axis is defined using all points where the derivative of the signed Euclidean distance to the surface is discontinuous. A more robust method to compute the skeleton of a shape was recently developed by Giesen et al. [48], known as the scale axis transform, which yields a hierarchy of successively simplified skeletons. Another commonly-used approach is to use the Voronoi diagram (or dual Delaunay tetrahedralization [135]) of a set of surface points to approximate the medial axis. Tagliasacchi et al. [145] show that skeletons can be computed even from incomplete point clouds.

Tam and Heidrich [146] demonstrate a medial axis-based approach to simplify shape. The medial axis is decomposed into parts that are selectively removed to form a simplified medial axis. A simplified surface is reconstructed from the simplified medial axis.

Fatih Demirci et al. [45] show that shape skeletons are useful for object categorization. A many-to-many correspondence can be established between segments of two shape skeletons. This correspondence allows for understanding the articulating parts of an object, and since object categorization requires prototypical models that are invariant to changes in shape (e.g., articulation) this skeleton-based approach can find use.

Planar surface approximations

Planar surface approximation addresses the problem of finding planes as good proxies for a 3D shape, but perhaps the earliest use of planar surface approximation has been to simplify 3D surface geometry.

Motivated by a lack of existing surveys, Cignoni et al. [25] survey and compare the performance of a collection of mesh simplification algorithms. Their taxonomy includes six categories of approach: coplanar facet merging, decimation, energy optimization, clustering, intermediate hierarchical representation, and other approaches.

Numerous approaches based on variational shape analysis have been proposed, which effectively optimize the geometric difference between simplified planar pieces and the original surface. The variational shape approximation approach [28] clusters faces using $\mathcal{L}^2$ or an introduced $\mathcal{L}^{2,1}$ metric, and planes or other primitives are fit to each cluster. The variational approach proposed by Krayevoy and Sheffer [83] attempts to decompose shape into perceptually meaningful components using a convexity metric. The variational mesh decomposition approach of Zhang et al. [167] uses a formulation that simultaneously handles both computing the segmentation and segmentation boundary smoothing, which in previous work were handled separately.

Some approaches use geometric or view estimates to project surface geometry onto a small number of planar polygons for lightweight rendering. The billboard clouds technique by Décoret et al. [35] produces a set of textured planar polygons that approximate the shape of an input surface. A greedy approach iteratively chooses planes that approximate large quantities of faces, and planes are favoured that are tangential to the surface. The approach has also been shown to produce suitable lightweight shadows. The poly-postors technique by Kavan et al. [75] is a 2D view-dependent representation of 3D geometry. Poly-postors are constructed manually, by decomposing a character into articulating regions – arms, legs and torso. Though poly-postors are a view-dependent representation, they can be animated efficiently by translating the vertices of the representation. The real-time rendering of large crowds consisting of tens of thousands of humans is demonstrated. In Chapter 7, we demonstrate that planar section representations are also directly applicable to the automatic generation of lightweight impostors.

Hierarchical bounding volumes

Hierarchical bounding volumes globally approximate shape, and have been explored extensively in the context of performing efficient collision detection between rigid objects. The choice of primitive used for the bounding volume hierarchy influences the tightness of the boundary, which can lead to better efficiency. However there is a trade-off, as for simpler primitives collision tests can be performed faster and less memory is required for each primitive.

Gottschalk et al. [52] use oriented bounding box trees (OBB trees) that are constructed by recursively partitioning geometry and defining oriented bounding boxes to bound each partition. An approach by van den Bergen et al. [158] efficiently performs collision detection using axis-aligned bounding box trees (AABB trees), with performance comparable to previously existing oriented bounding box methods. Klosowski et al. [78] use convex polyhedra for representation (known as k-DOPs or Discrete Orientation Polytopes). O'Rourke and Badler [108] present a method to convert from a surface representation to a sphere-based volume representation that is demonstrated to add realism to the rendering of articulating models [3]. Sphere trees are a hierarchical bounding volume approach that globally approximates shape with spheres at a specified level of detail. Numerous methods have been proposed to construct sphere trees from surfaces, such as the octree method, Hubbard's method [59] (which uses the object's medial axis), and the method of Bradshaw et al. [17]. Hierarchies of swept-sphere volumes [86] have also been used for collision and approximate distance computation – a swept-sphere volume is formed by the union of spheres whose centres lie within some region such as a line or polygon.

Primitive fitting

Samet et al. [120] propose the bintree, which creates a hierarchy through recursive subdivision of space and time, and generalizes the quadtree and octree. Snyder's work on generative modelling [140] allows the specification, rendering and analysis of a variety of shapes including 3D curves, surfaces and solids. Shapes are specified in a language that builds multi-dimensional parametric functions that serve as shape primitives, and operators (e.g., sweep, reparameterize) are defined on these functions. Cuboid shape proxies [168] have been used within a high-level approach to image manipulation. A recent approach GlobFit [89] fits cylindrical, planar and spherical shape primitives using global shape relations such as coaxial, coplanar, and parallel shape regions. Thiery et al. propose Sphere-Meshes [150], where spheres with varying radii are interpolated to approximate a surface. A related work by Bærentzen et al. [6] produces quad-dominant meshes from a skeleton where sphere radii are defined at bone endpoints. Simari and Singh [138] propose ellipsoidal representations, and present techniques to perform remeshing from this representation. In the co-abstraction work [165], a *spectrum* of abstractions is generated, where volume-based primitives are fit to progressively capture finer detail. The co-abstraction method chooses an optimal abstraction from the spectrum that minimizes a weighted sum of entropy-based measurements.

2.2 Sketch-based Modelling

Perhaps the most historic and notable example of interactive shape modelling (computer-aided design) is Sutherland's "Sketchpad" [143]. In this dissertation, we specifically survey *sketch-based* modelling systems, as we later design a sketch-based modelling system to create planar section representations in Chapter 5.

The numerous sketch-based modelling systems we review mostly provide a single 3D perspective view for interaction. The reasoning behind such a choice is important: the aim of these systems is to capture aspects of a workflow that artists and designers will already be familiar with – namely, sketching with a pen and paper. These systems are intended to be used by designers or artists who may have little experience with professional 3D modelling software such as Maya, CATIA, Blender, etc. Sketch-based modelling systems aim to bridge the gap between conceptual design with sketches and the final digital modelling of 3D surfaces, while addressing the important goal of feeling natural to use, even for 3D modelling novices.

The problem of inferring 3D curves from sketched 2D strokes is perhaps the most fundamental problem in all sketch-based 3D interfaces. Many sketch-based systems rely upon an existing 3D sketch surface or canvas that a 2D stroke will be projected upon. A study by Schmidt et al. [123] explored this specific issue, revealing that even among expert users there is a relationship between bias and the amount of foreshortening of the sketch surface.

A survey of sketch-based modelling systems [107] presents various ways to categorize and distinguish existing sketch-based modelling systems. They are:

(i) The creation method (e.g., iconic, template, free-form);

(ii) the underlying geometric representation (e.g., parametric surface, polygonal mesh, implicit surface);

(iii) the editing operations that are supported (e.g., surficial or additive modifications, oversketching, CSG/Boolean operations);

(iv) features of the interface (e.g., suggestive, gestural).

In this review of sketch-based modelling systems, we broadly categorize the systems based on the underlying geometric representation. This approach makes sense since the focus of our own work is the introduction and development of the planar section representation – a form of geometric representation distinct from those used in existing sketch-based modelling systems. We present a large number of systems, discussing details and novel features of each.

Parametric curve and surface representations

Stroke beautification is the process of taking a noisy, imprecise sketched stroke (that the user was not likely to be intending to draw), and computing the curve the user was intending to draw. Often the user will intend to draw smooth curves, free of noise, which take the shape of simple primitives such as line segments or circular arcs. Beautification can also occur at the drawing level, where the relationship between existing strokes is considered – for instance, two curves meeting roughly at a right angle can be beautified to meet precisely at a right angle.

Banks and Cohen [7] describe a stroke beautification method that fits 2D or 3D splines to a sketched input curve in real-time. The method reduces the data collected from an input stroke, and constructs a curve that faithfully represents the sketched data.

The Pegasus system [61] allows users to create precise geometric 2D diagrams, incorporating interactive beautification and predictive drawing. Interactive beautification [62] refers to the process of computing various possible beautifications, and then allowing the user to select the intended curve from a list of candidates. The Pegasus system considers existing curves similar to the sketched curve when suggesting beautified curves. This feature makes it predictive, since previously-sketched curves are treated as beautified exemplars.

CHATEAU [60] builds upon these ideas in a 3D setting, supporting the modelling of planar polygons and line segments within a suggestive and gestural interface. The designer provides a number of geometric hints to the system by highlighting relevant geometric components, and CHATEAU infers a number of possible operations based on these hints, presenting the operations as small thumbnails at the bottom of the interface. The interface is extensible and a wide range of operations are supported, including: the creation of closed polygons, cuboid and pyramid shapes, chamfers and corner cuts, copying and beautification such as rotational symmetry.

Cohen et al. [27] created a sketch-based system to create 3D curves. Since the position of points of the curve along the depth axis are ambiguous, this system introduces the idea of drawing the drop shadow of the curve on a ground plane, in order to disambiguate depth along the curve. The system is able to adjust the profile of the drawn shadow to match the curve even in the event they do not correspond well, and the case where the shadow may not define a unique 3D curve is also considered.

Some sketch-based systems facilitate the creation of *fair* curves. Digital French curves [139] moves physical French curves into the design software. A designer uses the non-dominant hand on a “puck” to manipulate the position of a digital French curve, whose boundary can be traced along using a pen or other pointing device with the other hand. Another system [98] demonstrates an approach to fits parts of a French curve to an input 2D stroke, to create fair curves automatically. The Elasticurves system [149] shows how stroke dynamics and inertia can be used to perform real-time smoothing of sketched input, using the analogy of a weight connected to the tip of the drawing instrument by an elastic force.

Tsang et al. [154] developed a suggestive system to model and position 3D planar curves, whose shape is guided by a collection of scanned images taken from the conceptual stage. Gestures allow for the selection, cutting and deletion of sketched strokes. A gluing operation allows for snapping to only occur along specific regions, and a pinning operation specifies a point the drawn curve should pass through. In addition to snapping, two types of suggestions made to users are either to (i) close the curve, or (ii) create an extrusion curve. During modelling, the interface suggests curves based on both the construction images, as well as from a database of existing 3D curves. A curve matching algorithm compares the input curve to a database of “curve signatures”. The top three candidates suggestions are shown to the user, directly within the space they are modelling.

Masry et al. [96] reconstruct depth for a collection of 2D strokes by analyzing the stroke connectivity and angular distribution of strokes. The angular distribution of strokes is used to determine an orthogonal 3D axis system, whose projection correlates with the stroke orientations. A plausible depth for each vertex is then computed within this coordinate system. An additional optimization procedure is used to reconstruct curved strokes in the original sketch, where the curves are assumed to be planar. However, this approach is suitable only for sketches that are not drawn in perspective, and whose curves conform to an overall orthogonal axis system.

The Mental Canvas system [38] uses a collection of planes (or “canvas assemblies”) situated in the 3D space to act as sketching surfaces. The user sketches within the 2D window, and the system projects the sketched curve onto a selected canvas to create a planar curve situated in the 3D space. Interestingly, perspective projection can be used to move strokes from one canvas to another. The authors demonstrate a number of architectural sketches made using this system. Another illustrative system [16] has in common the idea of sketching curves onto planar surfaces positioned in 3D, and considers silhouette curves drawn from a number of distinct view directions. The authors also demonstrate that the system can be used to perform 3D annotation.

The system of Chen et al. [23] also explores a sketching workflow in the early stages of architectural design. Sketches are reconstructed as a set of planar and (symmetric) curved surfaces, utilizing information from the junctions, edges and faces provided by the sketch. To add detailed geometry, an approach matches partial sketches to “index drawings” of detailed geometry, such as doors, windows and other fixtures.

In the ILoveSketch system [4], symmetry is exploited to create parameterized 3D curves in perspective. The interface organizes designs using the analogy of a virtual sketchbook, which artists and designers will feel familiar with. Further, 2D controls continue with the analogy of manipulating the scale and orientation of paper. Multi-stroke curve sketching is supported by averaging among cubic Bézier curves fit to each stroke, and curves can be edited by over-sketching [8]. The interface has no visible menus, relying on a set of gestures for command input and a few buttons to set navigation mode. Audio feedback is used to confirm gestures. There are two methods to sketch 3D curves that are both based on epipolar geometry. The first is aimed at expert designers and a pair of symmetric curves are drawn within a single view relative to a given centre plane. The second method is a two-view method, where the designer draws the curve from two unique viewpoints to define it in 3D. Designers can also draw curves onto orthographic planes or along surfaces that are extrusions of already-specified curves. The authors also introduce the measure of *sketchability* of a view, which refers to the ratio of the areas of two faces of the bounding box of a reference curve. The system incorporates an auto-tumble feature which automatically rotates the viewpoint to a new position where the sketchability exceeds a pre-defined threshold. An evaluation study involving a professional designer revealed a positive opinion of all features except for automatic paper rotation (a rotation of the view plane).

EverybodyLovesSketch [5] extends ILoveSketch [4] by adding a number of novel interactions to make the system more usable for non-experts, who may be unfamiliar with sketching in perspective. An insight behind the approach is that most people intuitively understand the notion of a “sketch surface” – the surface that the sketched curves are projected upon. These sketch surfaces (or “3D canvases”) may be sets of planes with various spatial configurations (e.g., orthographic and axis-aligned, radial about a vertical axis), or possibly surfaces that are extruded curves or even freeform meshes. In addition to crossing menu selection, *ticks* (or tick marks) are used to define position constraints on existing lines and curves. A horizontal sketch plane is defined by using a single tick across a 3D curve or intersection, where the plane passes through the point the tick was made. Two tick marks defines a vertical plane that passes through the ticks, and three tick marks can be used to define an arbitrary sketch plane. As with ILoveSketch, an axis widget can be invoked using a small lasso gesture positioned along the 3D curve, from which extruded sketch surfaces can be formed. Temporary NURBS sketch surfaces can be created by lasso selecting two to four curves and performing an area-fill gesture. Interactive grids can be used to aid the precise sketching of straight lines, circular and elliptical arcs. An evaluation with 49 high school students revealed that almost all (47) were able to create meaningful 3D shapes.

In the analytic drawing system of Schmidt et al. [124], a linear 3D scaffold is interactively created within a single perspective view to facilitate the creation of a 3D curve network. Techniques are presented to derive 3D constraints for sketched curves based on the scaffold, such as points of intersection and tangent directions at intersection points. Since sketched line segments and curves may be ambiguous in their 3D interpretation, the system infers the most probable interpretation. The approach to infer the 3D interpretation is based on the result of a defined fitness function, which incorporates information about the stroke, constraints implied by the scaffold, existing geometry in the scene, and other prior assumptions such as curve planarity.

The sketch-based modelling system by Cherlin et al. [24] produces parametric surfaces using a minimal number of input strokes. The two types of parametric surfaces are rotational and cross-sectional blending, which are inspired by illustration techniques. Based on the "spiral method" for sketching, the rotational blending surface is specified by the boundary curves, centre curve, and one or more circular slices of the surface. Based on the illustrative "scribbling method", cross-sectional blending surfaces are specified by boundary curves and one or more cross-sectional curves that define variation along the depth axis. The system includes two methods of deformations for created surfaces: orthogonal deformations that deform the central axis of the shape, and cross-sectional over-sketches that allow folds and creases to be formed on the surfaces.

The structured annotations system [49] allows sketch-based modelling using ellipsoids and generalized cylinders as primitives to create free-form surfaces. Importantly, the system uses a single fixed viewpoint throughout creation of the model, alternate 3D viewpoints are used solely for verification. Generalized cylinders are added simply by sketching the central axis; properties such as cross-section, symmetry, tilt and scale are initialized to a default setting. A number of annotations can also made to the primitives so that the 3D model appears consistent even for an inconsistently-sketched 2D image. Annotations allow for the connecting of primitives, mirroring a primitive about a plane, and specifying spatial constraints such as equal length, tilt and scale of two primitives.

The 3-Sweep system [21] uses an input image in combination with sketch-based input to guide the fitting of a generalized cylinders and cuboids in 3D space. The sketched stroke consists of three steps that define the diameter, slant, and central axis for the geometric primitive. The system is able to computer the varying diameter of the shape based on information from the image. Complex objects can be created out of a combination of cylinders, and the system can automatically create 2D textures for the parametric surfaces, using the image content. Using the system to explore variations of geometry, and creating new images through composition are both demonstrated. Although the approach is restricted to cylindrical and cuboid primitives, an impressive collection of different types of objects can be quickly brought to 3D using this technique.

Polygonal mesh representation

Teddy [63] is a sketch-based interface used to create 3D polygonal meshes. The designer sketches a 2D polygon that is closed and is intended to be the silhouette of the shape. A constrained Delaunay triangulation is performed on the 2D polygon. A chordal axis is defined using edges of the dual of the Delaunay triangulation graph, and edges are removed that join “fan triangles” to create a spine. A surface is formed by re-triangulating the 2D polygon, using the points of the spine. The flat surface is then “inflated” by translating vertices along the depth axis an amount proportionate to the distance of each interior vertex from the boundary of the surface. The Teddy system provided numerous intuitive techniques for modelling – scribbling to erase, cutting strokes, extrusion strokes, smoothing and deformation (along a curve, or by over-sketching).

FiberMesh [103] is another sketch-based modelling system to produce 3D polygonal meshes. Similar to Teddy [63], the designer sketches the silhouette of the desired geometry. In FiberMesh however, functional optimization is used to fit a round surface that matches the silhouette. The 3D curves once sketched remain with the surface, acting as handles for controlling the geometry. The FiberMesh system contains four operations previously introduced [63]: creation, cut, extrusion, and tunnel, as well as an additional operation: add-control-curve. The authors note that their system is able to handle specific cases such as sharp tips and open sharp curves that are difficult to represent using implicit surfaces as the underlying representation.

The system by Kara and Shimada [71] creates polygonal meshes using a two-step process: in the first step, 3D curves are defined using sketch-based operations; in the second step, interpolating surfaces are fit to the curves, which can be interactively modified and boundary conditions can be specified. The system supports multi-stroke curve sketching, where B-spline curves are created, and over-sketching is supported to modify the curve after creation. A template surface (such as a rectangular prism) is used as a sketching surface to define the initial position the curves in 3D. Designers can then modify the profile of these curves by sketching the curve as it should appear from the given viewpoint. Triangular meshes are fit to a selected set of curves that form a loop, and Laplacian smoothing is applied to the triangulated interior. Control curves can be sketched onto these surfaces and manipulated, and the surface is deformed to match the shape of the control curve. The system also supports gesture-based commands, where gestures are classified using a neural network approach.

SmoothSketch [73], demonstrated that other systems [63, 72, 122] can benefit from an improved inflation algorithm. Rather than sketching the boundary contour of a silhouette, all visible contours can be drawn. Such a sketch may include T-junctions, which imply hidden contours. The SmoothSketch approach allows for the creation of a broader collection of surfaces to be sketched.

Implicit surface, volumetric and CSG representations

A system by Karpenko et al. [72] demonstrates the sketch-based editing of free-form shapes using variational implicit surfaces [155], whose functions are based on thin-plate interpolation where data points are interpolated such that squared second derivatives are minimized. Geometry is organized hierarchically into a tree-like data structure, where implicit surfaces are the leaves, and groups and linear transformations are represented by the internal nodes, allowing modifications to be applied to more than one object at once.

ShapeShop [122] uses hierarchical implicit volumes (or "BlobTrees" [161]) as the underlying representation. This choice of representation allows for models to be created of arbitrary topology. Like other systems [63, 103], shapes are created by sketching the boundary contour of the silhouette. A number of sketch-based operations are defined that allow the combining of shapes using blending and traditional CSG operations. Since the hierarchy of volumes defines a construction history, non-linear editing and removal of sketched components is supported. Compared to previous methods that blend surfaces [72], ShapeShop is the first to support blending at an interactive rate. The interface is also suggestive, in the sense that an "expectation list" allows for the selection of a specific shape-modelling operation amongst of a number of different methods.

The SKETCH system [166] introduced the use of gestures to specify geometric primitives. For example, the gesture for a cube is to sketch three segments such that they meet at a common point, and each is roughly parallel to the projection of the major axes. Another gesture creates a "duct" by specifying a closed curve for its cross section, and another curve to define its path of extrusion. The placement of primitives could be specified by the sketching of the drop shadow, as well as more traditional click-and-drag operations. SKETCH supports a range of primitives, including: cones, cylinders, spheres, objects of revolution, prisms, extrusions, ducts and superquadrics. More complicated shapes are produced in SKETCH by using a combination of these primitives.

2.3 Shape Perception

A significant body of perceptual psychology literature addresses human understanding of 3D shape from pictures [81]. The surface information conveyed by contours [79, 142] have been studied in various pictorial contexts. It has been shown that when viewing 3D curves, a planarity assumption allows us to better resolve a mental 3D curve from its projected 2D image [142, 153]. This allows us to better understand curves that are planar sections of a shape, and prevents a common misinterpretation of non-planar curves as planar curves with the same view projection. Curvature along surface contours [80] is known to capture important shape information [46] and influence our segmentation of shape into salient parts, suggesting that in reverse, both part segmentation and curvature extrema are important cues in the creation of planar section representations by humans.

In the context of stylized sketching from 3D models, Cole et al. [29, 30] study aspects of human perception relating to depicting shapes using line drawings and consistency among line strokes used by humans. Traditionally, these perceptual studies have been carried out from static viewpoints providing us with a rich understanding of shape from pictures but with little insight to our perception of an imagined 3D surface induced by a 3D abstraction.

Since planar section representations are shaded planar constructs, we refer to prior art on surface perception of shaded objects [116]. Numerous studies have been conducted on the perception of shape under a variety of view, material and illumination conditions [19, 104, 109, 129], sometimes with contradictory evidence, such as the whether specular highlights aid or hamper surfaces perceived under Lambertian shading [41]. Parallel and perpendicular planar sections of a shape have been shown to provide a better understanding of the position and orientation of a surface compared to the shaded surface itself [144] and might explain the aesthetic appeal of planar section-based design. Once again, these studies are either performed from static viewpoints or at most provide automatic view oscillation about a fixed viewpoint as a motion cue to shape perception. We take the findings of this body of research into account when designing the viewing, lighting and material parameters in our own perception studies that we detail in Chapter 6.

2.4 Fabrication

There has a been a recent surge in research on the processing of 3D shape for fabrication, especially involving designs created by slicing existing 3D objects [54, 99, 127]. Hildebrand et al. [54] detail a method to automatically slice a 3D object into a set of planar sections that can be assembled. Schwartzburg and Pauly [127] further formalize and detail geometric constraints necessary for the assembly of connected 3D planar section structures. Holroyd et al. [56] present a method that converts a 3D model into a parallel stack of 2D images within a semi-transparent medium, which provide a strong sense of the object's 3D shape. The Autodesk 123D Make system [2] can be used to create regularly-spaced parallel or orthogonally arranged planar sections from imported surface geometry that are suitable for fabrication.

Fabricated surface patches are often required to be (nearly) developable, and the fabrication process is greatly simplified when patches are planar. Yamauchi et al. [164] present a mesh segmentation algorithm that uses integrated Gaussian curvature to evaluate the developability of a region. Similarly, Julius et al. [68] propose D-charts, a technique to segment a given mesh into quasi-developable regions, using an introduced metric for developability of a region. (The term "quasi-developable" refers to surfaces that are nearly developable – that is, the distortion relating to non-zero Gaussian curvature is bounded.) The D-chart approach is applied to variety of surfaces, where papercraft and stuffed toy examples are fabricated. An approach by Rose et al. [117] shows that developable surface patches can be computed for 3D curve boundaries that have been sketched arbitrarily. Fair and predictable surface patches are created for a variety of examples, ranging from architectural design to garments. Kilian et al. [77] develop a computational framework for design of curved folds using a planar sheet, where no stretching, tearing or cutting takes place. This approach allows for a variety of fascinating and elegant shapes to be made. Liu et al. [91] show how to optimize a quadrilateral mesh so that its faces are made planar (a PQ mesh). A hierarchy of PQ meshes is obtained by iterative application of Catmull-Clark subdivision and PQ perturbation (translation of vertices so faces are planar). This optimization allows for the creation of developable subdivision surfaces from arbitrary quad meshes with non-planar faces.

Manufactured planar sections have been used to make V-style pop-ups [88]. Volumes can be sketched to allow the design of radial section lampshades based on 3D curve input [159]. Hart [53] presents "modular kirigami", which is defined as the "symmetric assemblage, with artistic intent, of multiple copies of cut paper shapes". A number of intricate examples with icosahedral symmetry are presented, where duplicate pieces are interlocked using slits cut into each shape.

The Sketch It, Make It system by Johnson et al. [65] allows a designer to create planar sections within a sketch-based 2D interface, using pen-based interactions that include selection [66], dimensioning, and specifying collinearity and orthogonality constraints. The SketchChair system by Saul et al. [121] consists of a 2D sketch-based interface for the rapid creation of chair designs that are 3D extrusions of a 2D sketch.

An interactive furniture design system by Umetani et al. [156] incorporates physical simulation and offers suggestions, models consist of rectangular planks that are nailed together. A complementary work by Lau et al. [87] converts surface representations of furniture models into fabricable parts and connectors. The Chopper system by Luo et al. [95] partitions a large piece of furniture into smaller pieces suitable for 3D printing. The pieces can be assembled together in a structurally-rigid manner.

The Make It Stand system by Prévost et al. [114] facilitates the creation of 3D sculptures that can stand once fabricated, the algorithm alternates between carving out volumes and performing deformation to reach a balanced result.

Numerous recent works have emerged to create 3D puzzles from input surfaces. Xin et al. [163] treat the input surface as a volume and propose a method to create 3D Burr puzzles, which consist of interlocking pieces held together by a single key piece. Lo et al. [93] present a method to create Polyomino puzzles from the surface, relying on a parameterization that creates a quad-based tiling. Séquin and his class [131] create puzzles using dissections. A grid based approach is shown to produce Burr puzzles, and a helicoidal approach is shown to create puzzles whose pieces separate in a helical screw motion.

Planar section representations themselves can be thought of as puzzles, as the order that pieces interlock may be non-trivial. In Chapter 3, among a number of topics, we explore in further detail some necessary conditions for the assembly of a planar section representation.

Chapter 3

Geometric Preliminaries

With a large body of related work presented, we now cover preliminary topics that will be used throughout the remaining chapters. The chapter material starts with Section 3.1, detailing the steps required to create planar sections from input surfaces, including: obtaining the planar section contours, their topology in the plane and their triangulation, and extrusion to add thickness. Then, in Section 3.2 we then cover some fundamental material on differential geometry: specifically curvature for the cases of space curves, curves on surfaces, and surfaces themselves. We define watertight surfaces (that we assume all input surfaces to be throughout this work, unless otherwise noted) and touch upon the Gauss-Bonnet theorem. We also discuss the ridge and valley surface features that are defined by measurements of differential geometry – the surface features are of critical importance in our approach to the problem of automatic creation of planar section representations.

We can roughly distinguish between shape representations that are either *low* or *high*-level. Low and high-level shape representations can be thought to exist within a continuum. This continuum can be defined by how localized the understanding of the geometry of the shape is required for elements of the representation. Low-level shape representations, which we cover in Section 3.4, consist of the most basic geometric primitives that can be used in the representation of a shape. For instance, a triangular mesh is a low-level shape representation that consists of vertices and edges to define triangular planar facets; a point cloud is an even lower level of representation that consists solely of vertices. In contrast, high-level representations, which we cover in Section 3.5, consist of primitives that are less localized, individually capturing a greater amount of information about the shape. An example of a high-level representation are the planar facets used in variational shape approximation – the facet captures an entire surface patch, and its boundaries are defined by the neighbouring surface patches.

Beyond shape representation, we introduce the notion of a shape *abstraction* in Section 3.6. Recalling the continuum of shape representations again, we place shape abstractions at an even higher level than the high-level shape representations, where we extend the continuum to include not just local geometric understanding of shape, but global shape understanding where the perceptual qualities of the shape may also be considered. Thus shape abstractions can be thought of as high-level representations that aim to capture the “essence” of the shape. To clarify this distinction, consider regular axis-aligned planar sections of a volume, versus an interactive tool where an artist can place planar sections individually. The regular planar sections will form a high-level representation but not an abstraction – only local understanding of shape is required to create each of the regular axis-aligned planar sections – the process by which the planes themselves are selected is agnostic to any global structure or perceptual qualities of the given shape. In contrast, the artist-authored planar section representation is a true shape abstraction since both global shape knowledge and perceptual qualities are considered in the process of selecting planar sections.

3.1 Planar Section Construction

3.1.1 Representation of a plane

A plane is defined by the following equation:

$$N \cdot (P - P_0) = 0, \tag{3.1}$$

where N is the plane's normal and P_0 is a point on the plane. The plane is defined by all points P where the equation is true. While this suggests that 6 values are required for the representation of a plane (three for N, and three for point P_0), a plane can be defined compactly using only three parameters:

(i) a spherical coordinate polar angle ϕ for the plane's normal, $\phi \in [-\pi/2, \pi/2]$;

(ii) a spherical coordinate azimuth angle θ for the plane's normal, $\theta \in [0, 2\pi)$;

(iii) the distance d of the plane from the origin, $d = N \cdot P_0$.

We use the 3-parameter representation in Chapter 4, since our approach computes feature volumes and performs selection in the plane parameter space. Thus it is crucial to reduce the dimensionality of the plane representation as much as possible.

In contrast, for our interactive system presented in Chapter 5, we explicitly represent the normal with a 3-dimensional vector, and store a point on the plane (P_0) directly. We also store two orthogonal basis vectors for each plane, which together with the normal and plane point, define a local 3D coordinate system. Having all of this data is useful for spatial transformations, for instance for the procedural modelling operations we present in Chapter 5.

We note that this parameterization of planes in 3-space is not homogeneous – the volume of parameter space occupied by a plane is potentially changed when the plane is translated or rotated. This is an important point, as in Chapter 4 we discretize (or “bin”) the plane parameter space. Consider a plane whose normal is parallel to the polar axis (e.g., $\phi = -\pi/2$, $\theta = 0$, $d = 1$); this plane will occupy a large volume (“many bins”) since all values of θ define the same plane. In contrast, a plane whose normal is perpendicular to the spherical poles (e.g., $\phi = 0$, $\theta = 0$, $d = 1$) will occupy relatively little volume ("few bins") since any changes to θ define a different plane. In Chapter 4, we fully detail our treatment for the lack of homogeneity of the parameter space.

3.1.2 Planar section contours

A *planar section* is defined by a plane and one or more planar section *contours* lying in that plane. The contours constitute the boundary of a planar section, which is a compact subspace (intuitively, a finite area on the plane that includes the boundary contours). It is possible that the planar section area may be disconnected, or that boundary contours can define an annulus (or hole, in fact many holes are possible). Both possibilities are shown in Figure 3.1. (Contour self-intersections and pairs of contours intersecting are degenerate cases. Consider the planar section formed by a large circle enclosing a contour in the shape of a figure 8, where the interior of the lobes are hollow. The treatment of such potential degenerate cases is ignored, as they did not occur amongst the hundreds of input surfaces used in this work.)

The planar section contours can either be interactively specified, or derived from an input surface. When they come from an input surface, an intersection is made between the surface and the plane. If the plane and surface intersect, this yields one or more contours.

In practice, our input surfaces are triangular meshes. Other polygonal meshes, and all other shape representations of surfaces can be converted into triangular meshes (at least, with a notion of bounded error), so our method can apply to them as well. Even volumetric data can be converted into triangular meshes, by treating the data as a scalar field to define isosurfaces.

For each triangular face, we perform an intersection test between its edges (line segments) and the plane. We assume that the triangle is non-degenerate meaning no two edges are collinear and all edges have length greater than zero. If the plane's intersection with the edges results in exactly 2 unique points, a line of intersection is formed along the triangular face. If there are not exactly 2 unique points, the intersection is discarded. (Theoretically, it is impossible for only one edge to intersect. Zero length edges, where the plane intersects two edges at a shared vertex, are also discarded since the points are not unique.)

The next step is to chain the lines of intersection together (to arrange them in sequence so they define a set of piecewise linear closed contours). To do this reliably, we can use the face adjacency information from the triangular mesh. Alternatively, in the absence of face adjacency information, the distances between the line segment endpoints can be used to link them together. Theoretically, line segments that link should have zero distance between their common endpoint, but in practice an upper bound on distance between endpoints ϵ needs to be used.

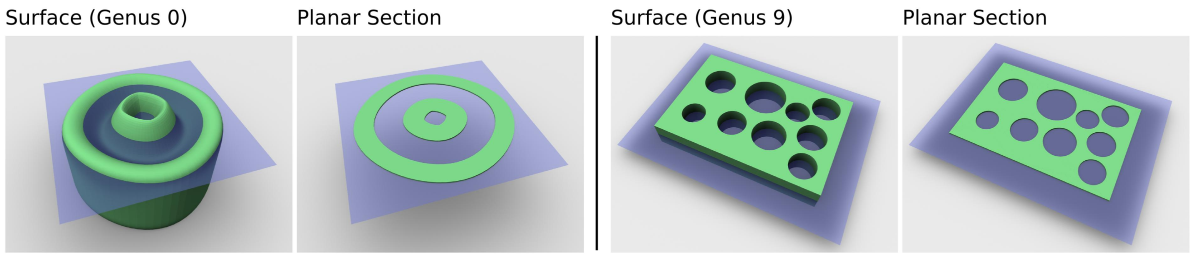


Figure 3.1: (Left) A planar section is formed between a plane and a surface of genus zero. The area is both unconnected and contains holes. (Right) A planar section formed between a plane and surface of genus 9.

3.1.3 Identifying holes

The input surface, though watertight, may be of any genus – meaning it may possess one or more holes. In fact, even a surface with zero genus may produce an area on the plane with one or more holes (see Figure 3.1 for examples).

Some contours delineate *holes*, and we call such contours *hole contours*. A hole contour will always be contained by a non-hole contour, but a hole contour may contain other contours as well (see Figure 3.1, left).

The direct method to determine whether a contour is a hole contour is to examine the surface normals along it. The surface normals (assuming the surface is orientable and normals have been correctly oriented) will point toward the outside of the volume at all points. Another way to examine this: for a march along one of the contours in a clockwise direction (viewed from "above"), if the surface normals appear to point to the right, then the contour is a hole contour. Thus we can check the surface normal at any point along a contour to determine whether it delineates a hole or not.

However, even in the absence of normal information we can still compute which contours are hole contours (assuming we have a proper watertight surface). A *ray-casting algorithm* can be used, where rays originate outside all contours (we assume the surface bounds some finite volume and thus the area beyond the contours is not the interior region). Moving along a ray on the plane, each time the ray intersects a contour, it alternates between a state of classifying contours as non-hole or hole contours (the first contour hit is always a non-hole contour, and the first time contours are hit along the sequence of intersections is when they are classified).

3.1.4 Triangulation

We now have a set of non-hole and hole contours, and must triangulate the interior region(s). The set of contours define what is known as a *planar straight line graph* (or PSLG). There are a number of existing algorithms that perform triangulations of PSLGs, which we briefly discuss (see [10] for a more thorough survey on mesh generation and triangulation).

The problem of triangulation becomes much simpler when there is only a single contour

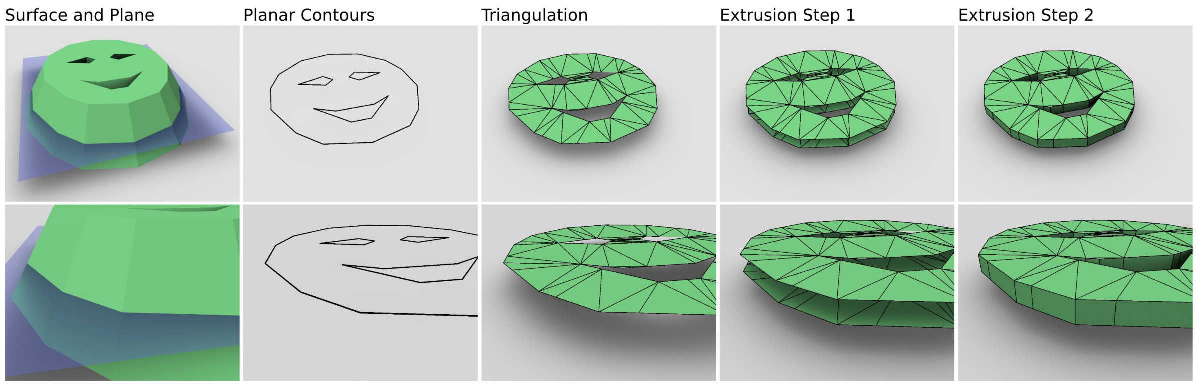


Figure 3.2: The steps to creating a planar section. (From left to right) Specifying the plane; surface-plane intersection and formation of planar section contours; triangulation; step 1 of the extrusion: duplicating and translating the triangulations; step 2 of the extrusion: creating quads to connect the triangulations along the planar section contours. In this example, the section thickness has been exaggerated to better convey the geometry.

or *simple polygon* to triangulate. For this case, there exist numerous algorithms that can triangulate quickly, the most recent algorithms performing with $O(n \log^* n)$ time complexity [26, 130, 147]. However, these algorithms do not apply in the general case where holes are expected.

The implementation we use in this work is based on the Triangle library [134], a C implementation by Jonathan R. Shewchuk. The concepts surrounding his approach to triangular mesh generation and Delaunay refinement in this library are detailed in [135].

3.1.5 Section thickness

Our planar section representations are intended to be fabricable, physical objects that will be constructed from materials that have some amount of thickness. We would like our digital planar section representations to be visually consistent with their real-world counterparts. Thus, we create thickness, performing an *extrusion* of each planar section for a specified thickness t along the direction of the plane normal N. This consists of three steps: (i) duplicating the triangulation from the previous step and flipping the orientation of the triangles; (ii) translating each triangulation by $-\frac{1}{2}Nt$ and $\frac{1}{2}Nt$; (iii) triangulating the space between the planar section contours: contour line segments are connected by a quad (see Figure 3.2).

3.1.6 Intersection with slits

We require some means to interlock a pair of intersecting planar sections. Our approach is to cut rectangular-shaped *slits* into each planar section along their shared axis of intersection (see Figure 3.3). The width of the rectangular slit is equal to the section thickness. For the purpose of fabrication, the section thickness will be set to the material thickness.

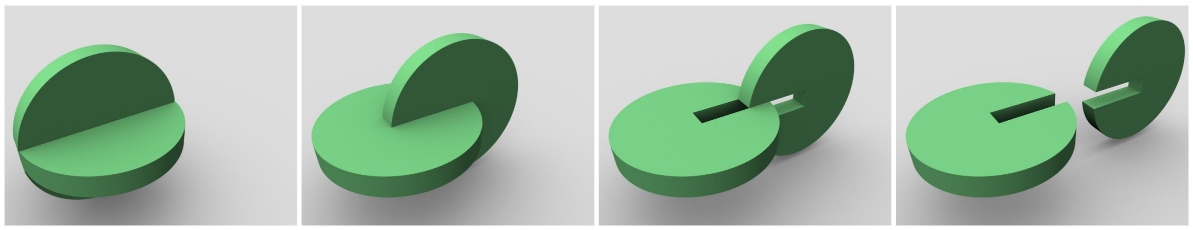


Figure 3.3: A rectangular *slit* is cut into all planar sections where they orthogonally intersect each other. For the two discs shown that intersect (left), the length of the slit made for each planar section is exactly *one half* the total length of the intersection between them. This approach to interlocking makes the planar sections rigid once slid together – no rotation can occur and translation is constrained to the direction of the slit axis.

Schwartzburg and Pauly [126, 127] have thoroughly investigated assembly constraints of orthogonally-intersecting planar sections, arriving at a number of insights that we briefly review. It is useful to consider the structure of the intersecting planar sections as a graph, where the planar sections are graph vertices and the lines of intersection between planar sections are graph edges. For instance, the minimum size of a cycle that is both physically assemblable and where no two planar sections are parallel is seven. For an assemblable cycle of size six or less, there must exist at least two planar sections that are parallel due to constraints on rotational freedom (see Figure 3.4 for examples).

In general, assemblable cycles of any length must contain two parallel slits (lines of intersection). The intuition is that the two parallel slits divide the cycle into two halves, which are able to slide apart along the common direction. An informal inductive proof is the following: Consider $n \geq 3$ planar sections that intersect to form a cycle, but where no two intersections are parallel. The base case: no single planar section can move because it has two non-parallel intersections (it would be forced to translate in two different directions simultaneously, which is impossible). The inductive step: We consider an adjacent planar section, and attempt to move this planar section simultaneously with the planar sections we attempted to move previously, as an entire assembly. The entire assembly also does not move, since the two intersections formed with planar sections not in the assembly we attempt to move constrain translation along two non-parallel directions. No sub-assembly of the assembly can move either, since this was previously attempted. Therefore, no subset of the n planar sections can move.

A final possibility is that all n planar sections somehow move simultaneously. No individual planar section can translate along both intersection line directions at once since they are non-parallel, so we consider the case of each planar section translating along one intersection line direction (the direction for each planar section is unique, otherwise we are attempting to move a sub-assembly together along one direction, which we have already shown to be impossible). However, because the planar sections form a cycle, each constrains the translation of a neighbour along a direction that is different than the direction the planar section must move. Therefore, only n-cycles where two intersections are parallel can be assembled.

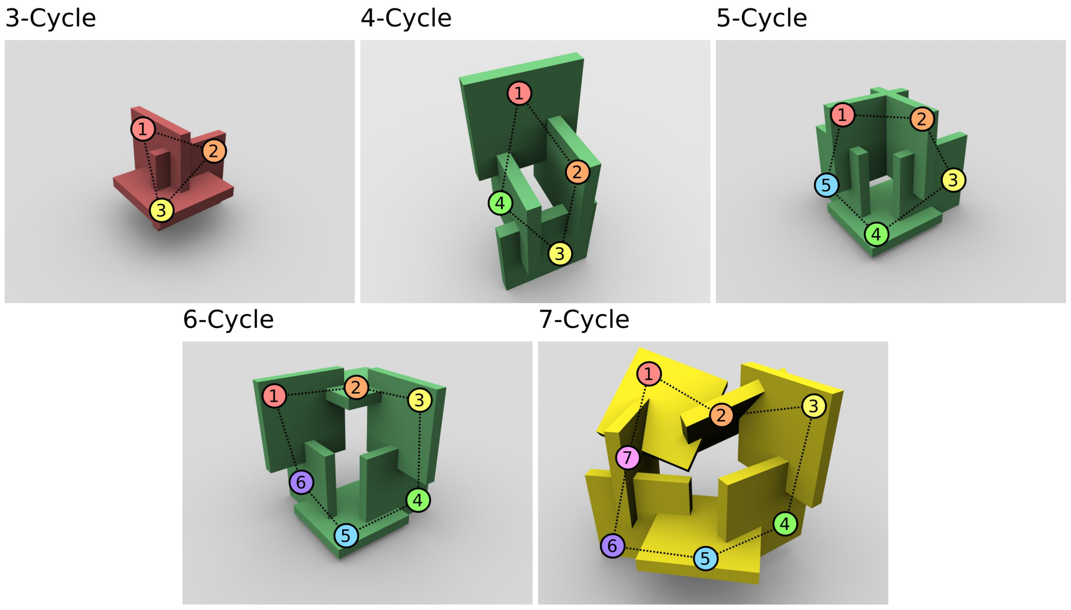


Figure 3.4: Various sizes of cycles formed by orthogonally-intersecting planar sections. A cycle of size three (red sections) cannot be assembled because there are no two parallel lines of intersection. A cycle of size seven (yellow sections) can be assembled with no parallel planar sections; cycles of size four, five, or six (green sections) cannot. Overlaid onto each cycle is a numeric labelling of sections and the graph intersection edges.

The minimum cycle size that can be assembled is 4. For a cycle of three planar sections, since they all intersect each other, the three slits formed are also orthogonal to each other. Since there must be two parallel slits, the cycle of length three cannot be assembled. Cycles of length 4 and greater can be assembled (see Figure 3.4).

All of this information on planar section assembly becomes important in Chapter 5, where we present a system to facilitate the interactive creation of orthogonally-intersecting planar sections. Since the representations created with the system are meant to be fabricated, it is essential to understand the requirements for physical assembly. Note that we have not considered all possible requirements – for instance, we have not considered the *global collision* of planar sections, as they are slid into place during assembly.

3.1.7 Assembly with foldable material

In general in this work, we assume the use of rigid, non-foldable materials for fabrication. Given this assumption, there many spatial configurations of planar sections that cannot be assembled. For example, consider 4 planar sections that form a rectangular prism and adding a fifth planar section which intersects each orthogonally – the interior of the prism cannot be reached. Or consider three disc-shaped planar sections that meet orthogonally to represent a sphere – one section must be split into two.

In this subsection, we relax the assumption of using rigid materials and consider foldable

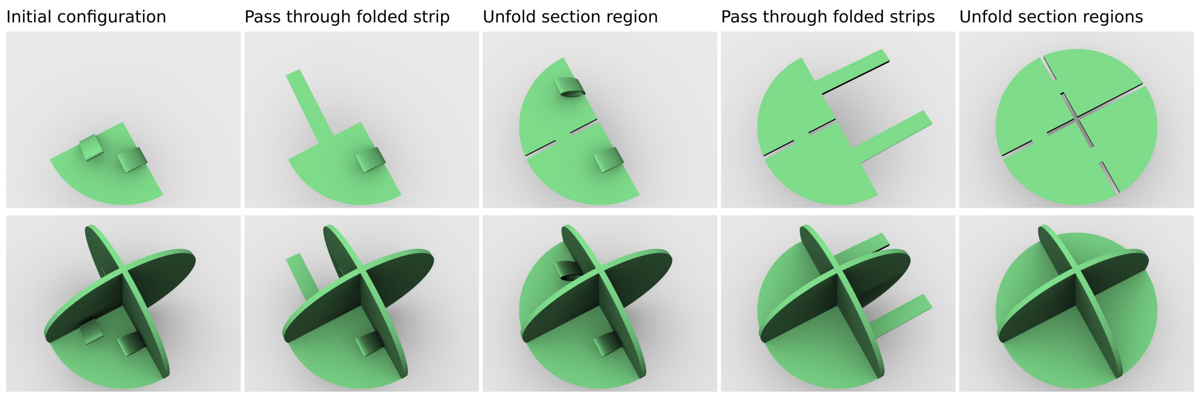


Figure 3.5: We visualize our assembly approach for a foldable material, which involves folding section regions into rectangular strips and passing them through holes cut into the existing planar sections. This ordering imposes a recursive binary partitioning of the planar section that is added.

materials (such as paper) for fabrication. We show that if the material used for planar sections is foldable, we can assemble any planar section representation. Intuitively, our technique is to fold unassembled regions of the new section into rectangular strips that are passed through holes cut into the existing sections. To prove this technique works, we use induction on k, the number of planar sections used in the representation.

The base case ($k = 1$): consider one planar section. Trivially, this is assemblable.

The induction step: Assume that a planar section representation with $k-1$ planar sections is assemblable. We show that an assembly with k planar sections can be assembled.

We start with the first $k-1$ planar sections already assembled (assumption). We now describe how to add the k^{th} planar section to the assembly of $k-1$ planar sections. Consider the pattern of line segments on the k^{th} section that mark the intersection lines with the existing assembly. These line segments partition the k^{th} section into *regions* (see Figure 3.5 (top right)). Construct a graph with a vertex v for each region and edges (u, v) connecting regions u, v that are separated by a common line of intersection $l(u, v)$.

We build a spanning tree of this graph, again using induction, on the number of edges in the graph.

The base case ($m = 0$): if there are no edges, there is only one region and thus the section is connected.

The induction step: Assume we can find a spanning tree for sections whose graph has $m-1$ or fewer edges. We show that a spanning tree can be found for a graph with m edges.

Pick any edge (u, v). We cut the k^{th} section in two along $l(u, v)$ except for a small *bridge* somewhere in the middle of the common segment of the line between region u and region v. We cut the corresponding hole for this bridge in the existing assembly. Now fold up the k^{th} section perpendicular to $l(u, v)$ into a strip, so it can be passed through a hole made in the assembly of $k-1$ sections, and unfolded on either side. Each of the two sides has a sub-graph of regions with no more than $m-1$ edges. For each of the two sub-graphs, we can thus compute the spanning tree recursively, applying the same folding and pass through approach between regions (see Figure 3.5, which visualizes this process).

3.2 Curvature

We first consider the curvature of a *space curve*, and in this work we consider curves embedded in 3D but the analysis applies to higher dimensions. We then examine the curvature of a curve lying on a surface, and then the curvature of surfaces. We also introduce ridge and valley surface features, which are defined using derivatives of surface curvature. Throughout the following, we use lower case symbols to denote scalar values, and capitalized symbols to denote vector quantities.

3.2.1 Curve

We first consider the curvature for a 3D embedded curve that is C^2. We define a curve $C = C(s)$, where s is the arc length [1] measured from some initial position on C (we assume that C is continuously differentiable). The *tangent* T along a curve C at s will be the first derivative of C with respect to s:

$$\frac{dC}{ds} = T. \tag{3.2}$$

We may also define a *normal* direction, also known as the *principal normal* of the curve. The normal of C will be orthogonal to the tangent T along C. Intuitively, the normal N captures the direction of *acceleration* orthogonal to T that a a physical body would experience moving along C. The normal N is obtained from the second derivative of C (or from the tangent derivative):

$$\frac{d^2C}{ds^2} = \frac{dT}{ds} = \kappa N. \tag{3.3}$$

Equation 3.3 is one of three *Frenet-Serret* formulas. The coefficient κ is known as the *curvature* of C. It captures the amount of non-tangential acceleration (along N) at s. Augustin-Louis Cauchy, a mathematician who formed and proved theorems of infinitesimal calculus, considered the normals $N(s_1)$ and $N(s_2)$ along C at two points $C(s_1)$ and $C(s_2)$ that are infinitesimally separated. If C is locally straight, lines along $N(s_1)$ and $N(s_2)$ never intersect, however if C exhibits any curvature – implying variation of the normal direction – then the lines will intersect at some common point. This common point defines the centre of an *osculating circle* that intersects C at $C(s_1)$ and $C(s_2)$. The osculating circle has a specific radius r, where:

$$\kappa = \frac{1}{r}. \tag{3.4}$$

[1] Note that given parameterizations for C other than arc length s, such as position at time t, we can re-parameterize the curve or normalize the derivative quantities directly. For example, if C is parameterized by t which is interpreted as time, then we would normalize measurements by the velocity: $T = \frac{dC}{dt} \left|\left|\frac{dC}{dt}\right|\right|_2^{-1}$. This concept provides the distinction between geometric and parametric continuity of a curve.

A quantity closely related to curvature that exists for 3D embedded curves is *torsion.* To examine torsion we first consider the tangent and normal directions T and N at $C(s)$. Using T and N, we can define a third orthogonal direction B, which is known as the *binormal*:

$$T \times N = B. \tag{3.5}$$

The torsion measures the rotation of the binormal B about C at s, where both:

$$\frac{dB}{ds} = -\tau N \tag{3.6}$$

and

$$\frac{dN}{ds} = \tau B - \kappa T \tag{3.7}$$

hold. Equations 3.6 and 3.7 are the two remaining Frenet-Serret formulas, where the coefficient τ is the torsion value. Intuitively, the torsion captures the amount of *twisting* that C has to escape the *plane of curvature* defined by the point $C(s)$ and directions T and N.

Since this body of work is concerned with planar section representations, we note that surface geometries representing space curves with non-zero torsion, such as a surface representing a thin volume in the shape of a helix, as being a worst-case scenario. We propose a reasonable approach for this case in Chapter 5 to create an aesthetically-pleasing planar section representation. (Note that a thin volume with *varying* torsion will cause the distribution of planar sections used to represent it to also vary accordingly – the distribution of planes as they are positioned in space will not be regular.)

3.2.2 Curve on surface

We consider a C^2 curve C that lies on a C^2 surface S. At each point of the curve $C(s)$, the surface S has a *surface normal* N_S, which can be used to derive other quantities: *normal curvature*, *geodesic curvature* and *geodesic torsion.*

The directions of the Frenet-Serret frame were introduced in Subsection 3.2.1; the directions T, N and B were defined for any point along the curve C. Here we introduce the Darboux frame, which has directions T, N_S and $B_S = T \times N_S$.

The *normal curvature* of the curve is the projection of the curvature component κN onto the axis along N_S. Another way to phrase this, is that if one were to intersect the surface S with the plane defined by point $C(s)$ and directions T and N_S to produce a new curve C_S, the curvature of curve C_S will be equal to the normal curvature of C on S. The *geodesic curvature* of the curve is the projection of the curvature component κN onto the axis along B_S. This geometric relationship between curvature, normal curvature and geodesic curvature is illustrated in Figure 3.6. Finally, the *geodesic torsion* measures the variation of N_S about the tangent T. These quantities are quite similar to how curvature and torsion were derived before, the key difference being the use of the surface normal N_S instead of the curve normal

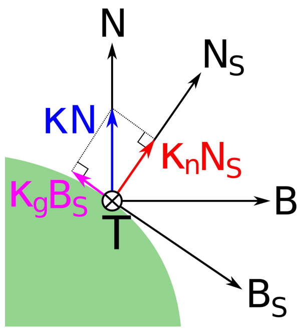


Figure 3.6: A view parallel to the tangent direction of C illustrates the geometric relationship between curvature κ, normal curvature κ_n and geodesic curvature κ_g. The vector quantity κN (blue) when projected onto the axis defined by the surface normal N_S or surface binormal B_S defines vector quantities $\kappa_n N_S$ (red) and $\kappa_g B_S$ (purple).

N.

The following relates the normal curvature, geodesic curvature and geodesic torsion, using the directions of the Darboux frame:

$$\begin{pmatrix} \frac{dT}{ds} \\ \frac{dB_S}{ds} \\ \frac{dN_S}{ds} \end{pmatrix} = \begin{pmatrix} 0 & \kappa_g & \kappa_n \\ -\kappa_g & 0 & -\tau_r \\ -\kappa_n & \tau_r & 0 \end{pmatrix} \begin{pmatrix} T \\ B_S \\ N_S \end{pmatrix}, \tag{3.8}$$

where κ_n is the normal curvature, κ_g is the geodesic curvature and τ_r is the geodesic torsion. Note that when the surface normal and curve normal coincide (that is, $N_S = N$ and thus $B_S = B$) then the normal curvature and curvature are equivalent ($\kappa_n = \kappa$), the geodesic curvature is zero ($\kappa_g = 0$) and the geodesic torsion and torsion are equivalent ($\tau_r = \tau$). This makes sense since the Darboux frame and Frenet-Serret frame share the same directions, and thus Equation 3.8 yields the three Frenet-Serret formulae introduced in Subsection 3.2.1.

3.2.3 Surface

Recall that T is one of infinitely many tangents on the tangent plane of S at point $C(s)$, and the normal curvature along T is κ_n. One could compute κ_n for all possible tangents of S at $C(s)$. To do so would yield a maximal and minimal value for the normal curvature κ_1 and κ_2; these values are known as the *principal curvatures*. The two tangent directions associated with κ_1 and κ_2, which we denote U_1 and U_2, are known as the *principal directions* and they are orthogonal. The curvature of any tangent T can be computed using the formula for the *second fundamental form*, a binary quadratic form:

$$II(T,T) = \kappa_1(T \cdot U_1)^2 + \kappa_2(T \cdot U_2)^2. \tag{3.9}$$

Note that if the normal curvatures along all tangents of S at $C(s)$ are equal, then the two principal curvatures κ_1 and κ_2 are equal and every tangent is a principal direction. Such surface points are known as *umbilic points.* For example, an ellipsoid has four umbilic points. The sphere is an example of a surface where every point is an umbilic point.

The *mean curvature* κ_m is the mean of the two principal curvatures:

$$\kappa_m = \frac{\kappa_1 + \kappa_2}{2}. \tag{3.10}$$

A surface with zero mean curvature is known as a *minimal surface*, such surfaces locally minimize their area – an intuitive example of this are the soap film surfaces that form when a wire frame is dipped into a soap solution. A helicoid is an example of a minimal surface.

The *Gaussian curvature* κ_G is the product of the two principal curvatures (not the geodesic curvature κ_g previously introduced):

$$\kappa_G = \kappa_1 \kappa_2. \tag{3.11}$$

An important difference between mean curvature and Gaussian curvature is that mean curvature is an *extrinsic curvature* while Gaussian curvature is an *intrinsic curvature.* This distinction relates to the embedding of a surface in 3D space. For instance, consider a surface S that is a 2-dimensional sheet with a Cartesian grid pattern. S could be embedded into space as either a flat plane, or as a cylinder in a way that is *locally isometric* (meaning the distances and angles of the grid pattern are preserved). The mean curvature for the planar embedding will be zero, but non-zero for the cylindrical case. In contrast, the Gaussian curvature for both embeddings will be zero.

A *developable surface* is a surface with zero Gaussian curvature at every point (that is, every point on a developable surface can be thought to lie on at least one straight line). A *ruled surface* is a surface created by moving a straight line (curve) along a path in space. While all developable surfaces are ruled (e.g., a cone), not all ruled surfaces are developable (e.g., a hyperboloid).

In our work, practically we must compute curvature not from smooth, continuously differentiable surfaces, but from low-level shape representations like triangle meshes. To accomplish this, we use the method and implementation by Rusinkiewicz [118].

3.2.4 Watertight surface

We introduce the property of a surface being *watertight.* The intuition behind a surface being watertight is that if the surface were immersed into a liquid, would the interior volume bounded by the surface fill with water? A more formal definition a watertight surface by [36]: "A 2-complex embedded in $\mathbb{R}^3$ whose underlying space is the same as the boundary of the closure of an open 3-manifold embedded in $\mathbb{R}^3$." In other words, a watertight surface is the boundary of a solid volume. We use the concept of watertight surfaces throughout the thesis but in particular in Chapters 4 and 6, as our planar sections capture slices of the interior volume bounded by the surface.

The Gaussian curvature is an important quantity, since its integration over the surface S provides understanding of the topology of S, specifically the *Euler characteristic* of S. This relationship is known as the Gauss-Bonnet theorem:

$$\int_S \kappa_G \, dS + \int_{\partial S} \kappa_g \, ds = 2\pi\chi(S). \tag{3.12}$$

Within our context of planar section representations, we are concerned with watertight surfaces that have no boundary and so the term involving geodesic curvature $\int_{\partial S} \kappa_g \, ds$ does not apply. For instance, for any (watertight) surface S topologically equivalent to a sphere:

$$\int_S \kappa_G \, dS = 4\pi, \tag{3.13}$$

and therefore $\chi(S) = 2$. In theory, we can use the Gauss-Bonnet theorem as a means to verify that a given surface is watertight by examining its Gaussian curvature. It is an interesting result that small, localized measurements can be brought together to produce global understanding.

While watertight surfaces are ideal for our planar section creation approaches detailed in Chapters 4 and 5, we note that they are not necessary. For instance, given a surface whose intersection with a plane yields one or more open contours, we can link them using any smooth interpolation approach (e.g., cubic Hermite segments) to create a closed planar section contour.

3.2.5 Ridge and valley surface features

Ridges and *valleys* are curvature-dependent surface features. The automatic approach to planar section selection we later present in Chapter 4 is heavily dependent on these surface features, so they are of importance. The method to mathematically derive ridges and valleys from a surface is detailed by Ohtake et al. [106], which we now review.

Ridges and valleys are isocurves representing the maxima and minima of principal curvature. More formally, a point P on a surface S will be classified as a ridge point if the following conditions on the first and second derivatives of principal curvature κ_1 with respect the principal direction U_1 are true:

$$\frac{\partial \kappa_1}{\partial U_1} = 0, \quad \frac{\partial^2 \kappa_1}{\partial U_1^2} < 0, \quad \text{and} \quad \kappa_1 > |\kappa_2|. \tag{3.14}$$

A point P on a surface S will be classified as a valley point if the following conditions on the first and second derivatives of principal curvature κ_2 with respect the principal direction U_2 are true:

$$\frac{\partial \kappa_2}{\partial U_2} = 0, \quad \frac{\partial^2 \kappa_2}{\partial U_2^2} > 0, \quad \text{and} \quad \kappa_2 < -|\kappa_1|. \tag{3.15}$$

The equations define curvature *extrema*, which locally varies quadratically. This is why the

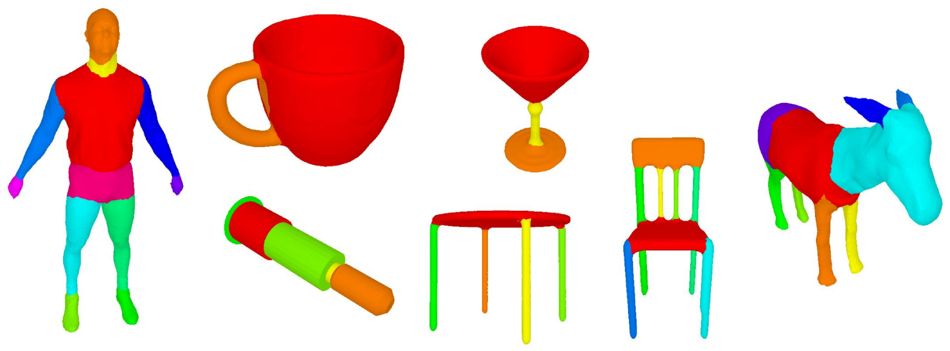


Figure 3.7: Segmentations of a few different surfaces (polygonal meshes). Each segment for a surface is shown with a unique colour.

first and second derivatives of curvature are used – as with finding the maximum or minimum value of a parabola, it is the point where the first derivative evaluates to zero, and deciding whether the point is a minimum or maximum is based on the sign of the second derivative. Note that if the orientation of the surface is reversed, ridges become valleys and vice versa.

3.3 Segmentation

A *segmentation* of a surface decomposes it into a set of parts (see Figure 3.7 for examples). A segmentation is also useful (but not required) for our automatic approach to creating planar section representations. Since each segment of a surface represents some important *part* of the represented shape, a planar section representation should capture the shape of each segment with at least one planar section.

In a recent work "A Benchmark for 3D Mesh Segmentation" [22], segmentation data was collected from a large number of users who manually segmented meshes representing a wide range of objects. This human-authored data was found largely to be consistent between participants. The human-authored data was then clustered together and used as ground truth in the evaluation of the performance of a collection of seven different existing segmentation algorithms:

(i) k-means [136], which performs k-means clustering of faces;

(ii) random walks [85], a two-face procedure consisting of over-segmentation and hierarchical clustering;

(iii) fitting primitives [1], clustering based on fitting primitives;

(iv) normalized cuts [51], clustering using area-normalized cuts;

(v) randomized cuts [51], clustering using a minimal normalized cut from a set of randomized cuts;

(vi) core extraction [74], which consists of 4 parts: multidimensional scaling, feature point extraction, spherical mirroring, and boundary refinement;

(vii) shape diameter function [133], which uses a Gaussian Mixture Model fit to the histogram of a shape diameter function that measures local thickness at each face.

The randomized cuts algorithm [51], though taking the most computation time (approximately 84 seconds on average, over 380 surfaces), produces the best segmentations according to the measurements used in the study on average. However, the authors also note that no single algorithm is ideal across all categories of object – but do state that algorithms that employ non-local shape properties tend to do better.

A recent segmentation method not included in the benchmark study is that of Kalogerakis et al. [69], which uses a bag-of-features approach and learns from user data. In Chapter 4, our own bag-of-feature approach is similar in spirit – we can extend our feature set and and we also learn the feature weights from human-authored planar sections. Ideally, as input to our automatic algorithm we have a segmentation generated by [69] or randomized cuts [51] – but any reasonable segmentation of an input surface will help to improve the quality of the planar section representations created.

3.4 Low-level Shape Representations

We survey some existing low-level representations of shape. Recalling our notion of a continuum for shape representation, the individual elements of these low-level shape representations rely relatively upon very localized shape information. Low-level representations serve the fundamental purpose of shape representation, but are largely divorced from the goal of high-level shape understanding. Shape representations presented in this section are covered in greater detail in other existing publications such as those by Farin [42, 43, 44] for curve and surface design (including Bézier, B-spline and NURBS), Botsch et al. [15] for polygonal meshes, and Bloomenthal and Bajaj [13] for implicit surfaces.

3.4.1 Point cloud

Point clouds consist of a set of *vertices* $\mathcal{V} = \{v_1, \ldots, v_n\}$, where each element is a (typically 3-dimensional) point in space. Often, point clouds are generated as the result of a scanning process where a physical object is sampled at various positions on its surface, yielding a (sometimes dense) collection of 3D points that sample the surface. However, in the case where a point cloud represents a volume rather than a surface, tetrahedralization is performed. Sometimes, extra information is associated with each point sample, such as colour or surface albedo, or possibly a surface *normal*, as is the case for an *oriented* point cloud.

Creating polygonal meshes from point clouds has been an active area of research. Concerning planar sections, given a point cloud as input, there are two approaches: (i) we may either

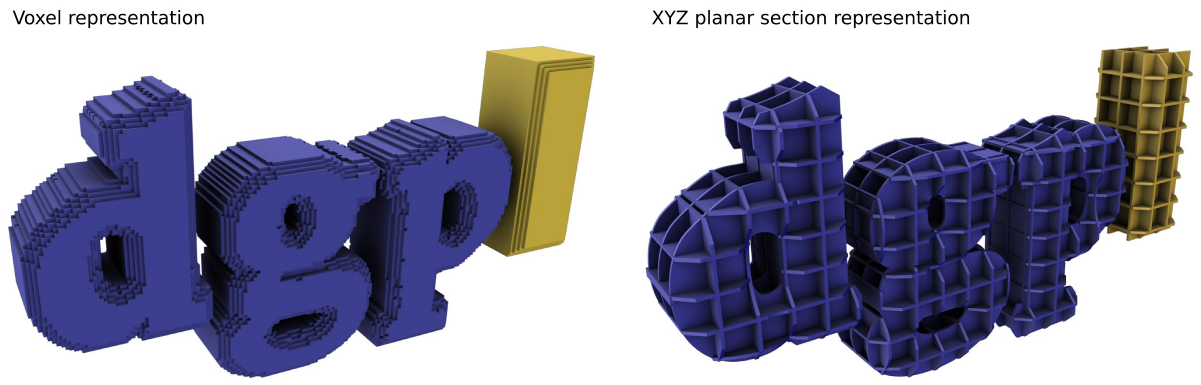


Figure 3.8: (Left) Rendering of a volumetric dataset consisting of voxels. In this example, each voxel is associated with a Boolean value that specifies whether a cell is inside or outside of the volume. (Right) A planar section representation of the same volume, where planar sections are regularly-spaced and oriented along XYZ directions. (Note that the voxels have a much smaller spacing than the planar sections.)

convert the point cloud to a surface representation such as a polygonal mesh and perform a plane intersection with the surface, or (ii) project points near the plane onto it, and perform planar curve fitting with these points. Concerning the first approach, fitting a surface to a point cloud, numerous techniques have been proposed. These techniques broadly either use a Voronoi-based method such as Delaunay tetrahedralization [36, 82], or volumetric approaches that use the zero level-set of a distance function [57, 58] and reconstruct the surface using a technique such as Marching Cubes [94].

3.4.2 Voxels

Voxels are very much like the pixels or texels that discretely sample a 2-dimensional space. A 3-dimensional space can be divided into a set of cube-shaped cells or *voxels* using a Cartesian grid, where each voxel has a uniform width across each of the 3 dimensions (see Figure 3.8 (left)). Technically, a voxel of dimension n could be defined, but 3 dimensions are used for representing volumetric data, and such datasets are the ones most typically encountered.

As voxels together form a discrete sampling of volumetric data, increasing the number of voxels in the representation can improve the quality but at high cost – to double the resolution along each dimension results in 8 times the number of voxels. This cubic relationship between resolution and the number of voxels prohibits the use of high-resolution datasets in some situations. The voxel shape representation is closely related to a planar section representation where cuts are made regularly along each of the XYZ axes, as the planar sections partition geometry into a set of voxel-shaped cells (see Figure 3.8 (right)).

Each voxel element is assigned a (typically scalar) value. For instance, to represent the interior volume represented by a surface using volumes, one may assign to each voxel the interior volume at each cell: if the voxel is fully enclosed by the surface, the voxel is assigned a value of 1; if the voxel is fully outside the surface, it is assigned a value of 0; for those voxels whose cube-shaped cells intersect the surface, the precise volume within the surface's interior can be calculated and assigned to the voxel.

In Chapter 4, we represent the plane parameter space effectively using a voxel-based approach, as we subdivide the parameter space into uniform regions. At each voxel position, we store a list of the features and a coverage value for the feature. In our visualizations, we render only those voxels that cover one or more of the features.

3.4.3 Polygonal mesh

A polygonal mesh is arguably the most common surface representation used in computer graphics. Meshes consist of *vertices*, *edges* and *faces*. For an n-vertex mesh, the vertices will be a set $\mathcal{V}$ of (usually 3D) points, $\mathcal{V} = \{v_1, \ldots, v_n\}$. A set of edges $\mathcal{E}$ connect vertices, each edge $e_{ij} = \{v_i, v_j\}$ connects a pair of vertices from $\mathcal{V}$ with indices i and j.

A set of faces $\mathcal{F}$ defines every polygonal face of the mesh, where each element of $\mathcal{F}$ is an ordered set of edges. A mesh is said to be *triangular* if all faces consist of three edges. Any face f for a triangular mesh would have the form $f_{ijk} = \{e_{ij}, e_{jk}, e_{ki}\}$. The *orientation* of a face is given by whether the edge traversal appears to be in a clockwise or counter-clockwise direction. Two adjacent faces that share a consistent orientation share a common edge, but the *direction* of traversal of the edge will be reversed (one face may contain edge e_{ij} and the other e_{ji}). If every edge of the mesh is shared by exactly two faces, the surface is said to be *closed.* If every edge is shared by two or fewer faces, the surface is said to be a *pseudomanifold.* A surface is said to be a *manifold* if, for every vertex v, the set of faces connected to v can be organized into a single cycle where adjacent faces in the cycle share a common edge. (For an open manifold, at each vertex the connected faces form either a single cycle or single chain.)

It is common for polygonal meshes to contain polygonal faces with a varying number of edges. Any polygonal mesh can be converted into a triangular mesh without any change to the shape of the surface, assuming that each polygonal face is planar. Each polygon with 4 or more edges can be split into constituent triangles naively using a *triangle fan* approach, as long as the polygon is convex.

Polygonal meshes may have issues such as self-intersections, which are problematic for defining an interior volume. A *degenerate element* is a mesh polygon with zero area – this can occur when edges are collinear or have zero length. $T - junctions$ are a related issue, where a vertex may be positioned within another edge. The surface of a mesh with a T-junction may appear to be closed, but topologically this will not be the case since some edges only have one adjacent face. Inserting a degenerate face into the T-junction closes the surface but this is not an ideal solution; the best solution is to remove the vertex within the edge.

Various data structures for meshes, such as the *half-edge* and *winged-edge* structures, are useful for when an operation on a mesh requires traversal. For example, these structures simplify the process of iterating through the faces adjacent to a given face.

3.4.4 Coons patch

A Coons patch [31] is a surface that interpolates between four boundary curves. Denote the curves $C_0(u)$, $C_1(u)$, $D_0(v)$ and $D_1(v)$ for surface parameters u and v. These four curves meet at the four corners of the patch with a parameterization such that $C_0(0) = D_0(0)$, $C_0(1) = D_1(0)$, $C_1(0) = D_0(1)$ and $C_1(1) = D_1(1)$. (Note that an existing flowline-based approach [11] can be used to segment a given closed 3D curve into sets of four boundary curves, from which quadrangulation of the patch interiors can be performed using Coons patches or another approach.)

We define two functions: L_C performs linear interpolation between curves C_0 and C_1, and L_D performs linear interpolation between D_0 and D_1:

$$\begin{aligned} L_C(u,v) &= (1-v)C_0(u) + vC_1(u) \\ L_D(u,v) &= (1-u)D_0(v) + uD_1(v). \end{aligned} \tag{3.16}$$

L_C and L_D each define a ruled surface. A bilinear interpolation B of the corners of the patch define another surface:

$$B(u,v) = C_0(0)(1-u)(1-v) + C_0(1)u(1-v) + C_1(0)(1-u)v + C_1(1)uv. \tag{3.17}$$

Combining the previously defined surfaces, a bilinearly blended Coons patch S parameterized on the unit square ($u, v \in [0,1]$) can be expressed as the following:

$$S(u,v) = L_C(u,v) + L_D(u,v) - B(u,v). \tag{3.18}$$

Unfortunately, using linear interpolation causes tangent discontinuities between the surfaces of two adjacent patches along their shared boundary (the surface formed by two adjacent linearly interpolated Coons patches is guaranteed to be C^0, not C^1). To fix this, the functions L_C and L_D, which perform linear interpolation, can be replaced by two other interpolating functions H_C and H_D. The replacement functions incorporate two of the four cubic Hermite basis functions H_0^3 and H_3^3, to perform *partial bicubic blending*:

$$\begin{aligned} H_C(u,v) &= H_0^3(v)C_0(u) + H_3^3(v)C_1(u) \\ H_D(u,v) &= H_0^3(u)D_0(v) + H_3^3(u)D_1(v), \end{aligned} \tag{3.19}$$

where $H_0^3(u) = 2u^3 - 3u^2 + 1$ and $H_3^3(u) = -2u^3 + 3u^2$. Note that the surface will be smooth across the boundary of adjacent patches, since the derivatives of H_0^3 and H_3^3 with respect to u are 0 for $u = 0$ and $u = 1$. We demonstrate the use this formulation to create surfaces from curve networks formed by planar section contours as an application in Chapter 7. We refer the interested reader to Farin [43] for the details of (non-partial) bicubically blended Coons patches, where tangent directions can be specified along patch boundaries, and flat spots at patch corners can be avoided.

3.4.5 Other curve and surface representations

We briefly survey a collection of other low-level curve and surface representations. For more details on these representations, we refer the interested reader to Farin's literature [42, 43, 44].

Bézier curve and surface

Technically, in 1959 Paul de Casteljau was the original creator of the Bézier curve, but it was Pierre Bézier who came to popularize it having published his work first. The two both worked for separate automotive companies in France, and developed the method for the design of the surfaces of automobile bodies. Bézier curves and surfaces can be defined to be of any order (e.g., linear, quadratic), but most common are cubic. The order of the Bézier curve or surface will determine the number of *control points* required for its construction; an n-th order Bézier curve has $n+1$ control points $P_0, \ldots, P_n$ and curves take the following general form:

$$C(u) = \sum_{i=0}^{n} B_i^n(u) P_i, \tag{3.20}$$

where $u \in [0,1]$ is the interpolation parameter. The coefficient used for each control point in the summation is a *Bernstein polynomial*:

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}. \tag{3.21}$$

Bézier curves have a parametric representation where derivatives can be computed analytically. This makes measurements of curvature, for example, more convenient than for a polyline or polygonal mesh.

A *Bézier spline* consists of multiple Bézier curves joined by sharing common endpoints. Conveniently, to maintain C^1 (tangent vector) continuity between adjacent curves at the endpoints, all that is required is that the adjacent control points and the endpoint need to be collinear (see Figure 3.9). Note that higher-order geometric continuity (C^2 and beyond) can be achieved by using higher-order Bézier curves with enough degrees of freedom to satisfy the additional geometric constraints. For instance, C^2 continuity can be achieved with the use of fifth-order curves.

A *Bézier surface* naturally extends from a Bézier curve by increasing the dimensionality by one, extending the curve along an additional dimension to form a square-shaped patch. In the same fashion that Bézier curves can be linked to form Bézier splines, the boundaries of Bézier surface patches can be joined with continuity at the patch boundaries to create larger, more complex surfaces.

We use cubic Bézier splines to represent planar section contours in our interactive system presented in Chapter 5 because they are easy to implement, efficient to evaluate, fairly intuitive to manipulate, and there are straightforward approaches to fit Bézier splines to sketched input polylines.

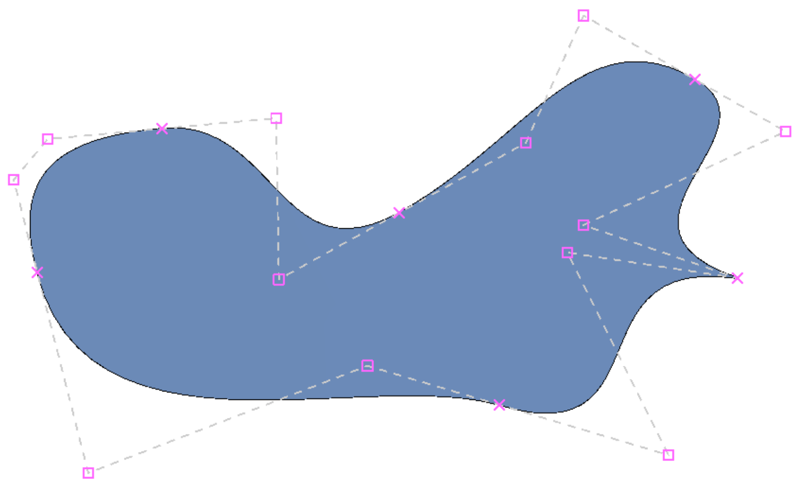


Figure 3.9: A closed cubic Bézier spline, used to represent the boundary contour of a planar section (blue). The spline's control points are shown (purple); control points that define curve endpoints are shown as X's and control points that define tangents are shown as squares. The control polygon (dashed grey) is also shown. Note the control point at the far right, as the tangents on either side are not parallel, an intentional G^1 discontinuity is created.

B-spline curve and surface

B-splines (or *basis splines*) are a generalization of Bézier curves. Each control point in a B-spline is associated with a recursively-defined basis function. The basis functions come from a formula known as the Cox-de Boor recursion formula. The basis functions of a B-spline depend on the order and the *knot vector* values. If the values in the knot vector do not have a constant spacing, the B-spline is said to be *non-uniform.* Like Bézier curves, B-splines can be extended to form a surface patch that is parameterized over the unit square, by using B-spline curves along each of the two directions.

Non-uniform rational B-spline (NURBS) curve and patch

NURBS (or non-uniform rational B-splines) are commonly used in computer graphics to represent the shape of curves and surfaces. Like Bézier curves, NURBS are defined by their order and control points. Like B-splines, they are also defined by a (possibly *non-uniform*) knot vector. NURBS differ in that control points are *weighted*, and the formulation involves a *rational* expression (see [44] for technical details). NURBS, like regular B-splines, can be extended to surfaces with a unit square parameterization.

Subdivision curve and surface

Given a control polygon consisting of an ordered set of vertices, the positions of the current vertices can be used to determine new vertex positions, a process called "subdivision". In

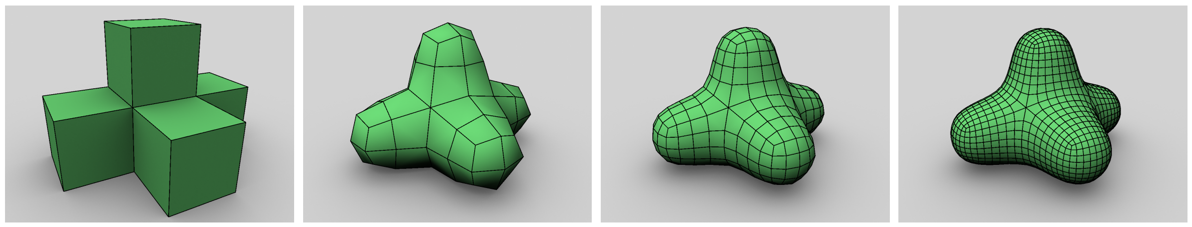


Figure 3.10: (Left to right) A polygonal mesh, and three iterations of subdivision applied to it.

general, for every current vertex, two new vertices will be created; this means that as subdivision continues the curve becomes progressively more refined. This concept has been extended from curves to surfaces (see Figure 3.10). *Doo-Sabin subdivision* [37] creates a smooth surface from a control point mesh, which at the limit yields bi-quadratic uniform B-spline surfaces. Perhaps the most popular subdivision scheme in use is *Catmull-Clark subdivision* [20], which yields bi-cubic B-spline surfaces. The limit surfaces for various subdivision schemes can be evaluated directly without requiring iterative refinement [141].

Implicit surface

Implicit surfaces are formed by the contours (or isosurfaces) of a defined 3D scalar field. In practice, implicit surfaces are defined by a *field function*, and the surface is formed by evaluating where the function is a specific constant. An early example of implicit surfaces are Blinn's "blobby molecules" [12], whose field function consists of Gaussian basis functions. Later approach, such as Wyvill's "soft objects" [162] and "metaballs" [148], use basis functions that are similar in shape to a Gaussian but are *bounded*: they evaluate to zero beyond a pre-defined *radius of influence*, which leads to more efficient evaluation.

3.5 High-level Shape Representations

We now consider a collection of high-level shape representations where the individual geometric primitives are relatively less localized than for the low-level representations. We note again that all of these shape representations exist within a continuum. Our categorization of representations is somewhat subjective, there is no concrete rule we adhere to in order to assign a representation as being low- or high-level. However, we believe that a rough categorization of a shape representation is possible based on the locality of shape information required by its geometric primitives.

3.5.1 Variational shape approximation

The variational shape approximation approach [28] fits geometric primitives (e.g., planes or ellipsoids [138]) that minimize a given error metric to each of k partitions of the surface.

Specifically, given an error metric E (usually the 2-norm), a number of k shape proxies (primitives, such as planar discs) and an input geometry S, the algorithm first partitions S into a set of k disjoint, connected regions. Since the input geometry S is usually a (triangulated) polygonal mesh, this step essentially performs discrete clustering of the triangular faces, for which there are already existing efficient algorithms. One such algorithm is Lloyd's algorithm [92], which initializes the k clusters with random centroids and iteratively updates these positions by computing new centroid positions from the set of faces closest to each cluster. The algorithm is guaranteed to converge after a finite number of iterations, but the solution may not be globally optimal. See Figure 3.11 for an example of the clustering, and the resulting VSA model.

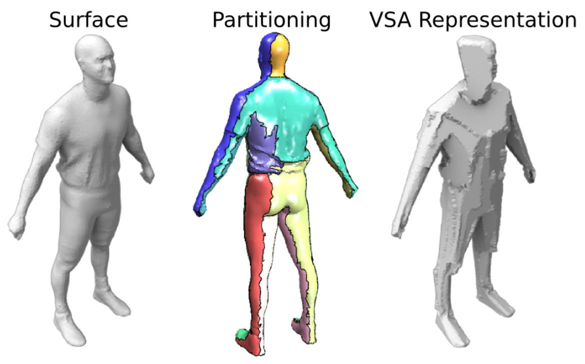


Figure 3.11: (Left) Given an input surface, the variational shape approximation method clusters the triangular faces of an input surface together (middle), where clusters are drawn in unique colours and cluster boundaries are shown. (Right) Geometric primitives (in this case, planes) are fit to each of the clusters, which form an approximating surface.

Concerning planar section representations, of particular interest is the fitting of planar regions to surfaces. However, such a representation in nature is distinctly different from planar sections that treat the surface as a representation of a volume; the aim of variational shape approximation is to produce a faithful representation of the surface itself.

3.5.2 Boundary curve network

A boundary curve network can be thought to generalize a polygonal mesh. For a polygonal mesh, each face is assumed to be planar since the edges which define the face are coplanar. For a boundary curve network, there is no planarity constraint on the edges and the edges need not be straight lines. The interior of each chordless cycle of a boundary curve network represents a surface patch (see Figure 3.12).

A boundary curve network does not explicitly define the shape of each interior patch; for that, an approach to interpolating the surface patch is required, such as a *Coons patch*. To ensure that adjacent patches that share a common boundary have tangential continuity, a *bicubically-blended* Coons patch [43] can be used – this uses a cubic Hermite spline instead of linear interpolation for tangential continuity at the boundary.

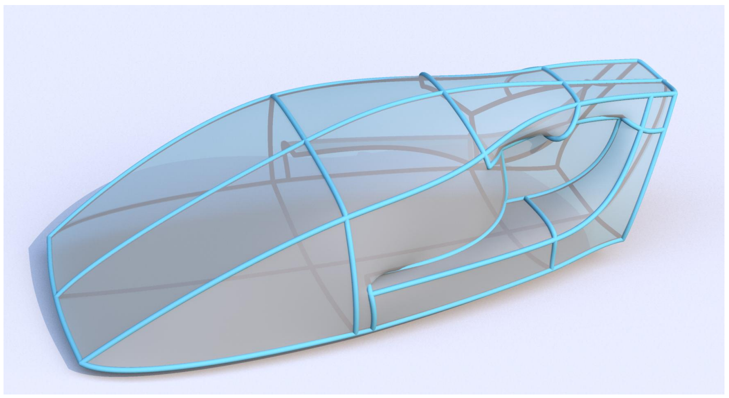


Figure 3.12: Rendering of a boundary curve network (curves shown in blue). A transparent surface for the curve network that interpolates through each boundary is also shown.

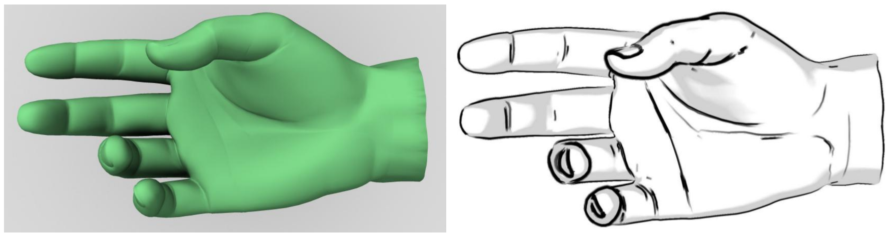


Figure 3.13: (Left) A deforming surface. (Right) A real-time stylized line drawing using the method of [70], which consists of the surface's occluding and suggestive contours [33].

We note that in general, the surfacing of boundary curve networks may not be straightforward, as there can be a number of topological issues, for instance: a patch boundary may be defined by fewer or more than four curves, vertices at patch corners may have high valence, or there may be T-junctions in the curve network.

3.6 Shape Abstractions

We present a number of shape *abstractions*. These shape representations differ fundamentally from all of the previously-presented shape representations in that *global* knowledge and the perceptual qualities of shape are incorporated into the process of selecting the geometric primitives for the representation.

3.6.1 Line drawing

Line drawings are perhaps one of the oldest and still most often used means that one can use to communicate shape, with artists and designers almost always relying on line drawings during early conceptual stages. Even a small collection of lines, quickly sketched onto a 2D surface, often suffice to powerfully convey the shape being represented.

One branch of computer graphics is devoted specifically to such abstract 2D representations

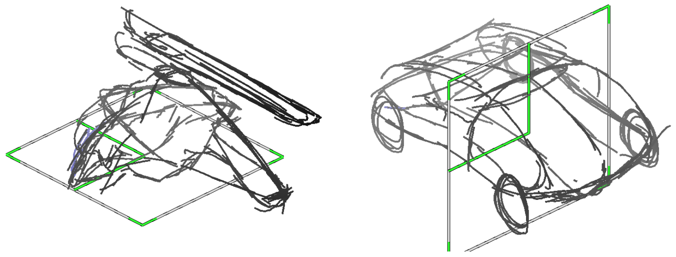


Figure 3.14: 3D planar curve drawings, created with an interactive system we developed for a user study. Also shown is the planar canvas artists would manipulate, upon which curves could be sketched.

of shape – the branch of *non-photorealistic* or *expressive rendering*. Some approaches take a photograph as input, and can automatically compute shape abstractions such as stylized line drawings [34]. Other work takes surface geometry as input and can compute line drawings that include shading effects using hatching [113] or stippling [128]. Some important work has been done to characterize the important lines that artists use in sketches for surface representation, such as: ridges and valleys [106], suggestive contours [33], and apparent ridges [67]. Real-time approaches have been proposed to create curvature-dependent line drawings of deforming objects [70] (see Figure 3.13). Moving beyond surface representations of shape, computing line drawings from volumetric data has also been explored [18].

3.6.2 3D curve drawing

A large number of systems have been developed to allow an artist to create curves that are embedded in 3D space [4, 5, 27, 124, 166]. In our own work, we have also created an interactive system to allow planar curves to be sketched upon a manipulable planar canvas (see Figure 3.14). Such collections of curves can be thought of as generalizing traditional line drawings, since by being brought into 3-dimensional space, their representation is no longer dependent upon a specific viewpoint. However, a negative aspect of this shape representation is that some strategy may be required to deal with the issue of all lines of the representation being visible at once, in contrast to a conventional line drawing where hidden lines are omitted.

3.6.3 Planar section representation

Planar section representations are similar to 3D planar curve drawings, since both use a combination of defined planes, upon which a set of planar curves or contours lie. Unlike 3D planar curve drawings, where each curve is drawn freeform with no constraint other than planarity, for planar section representations each contour must be a closed curve.

In addition, contours do not self-intersect, and any other contours lying on the same plane also will not intersect the contour. However, contours often intersect in the 3D space when they lie on different planes. See Figure 3.15 for examples of planar section representations.

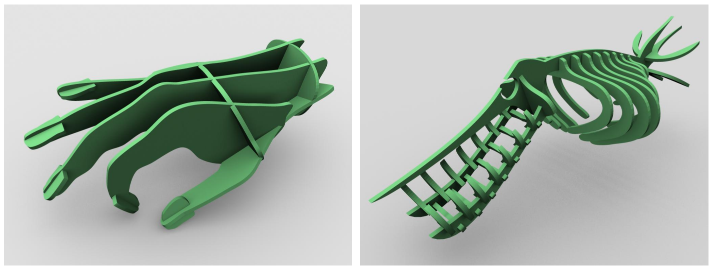


Figure 3.15: Planar section representations of a hand (left) and skeleton of a fish (right). While both are shape abstractions, the hand example is more abstract in the sense that it represents less of the surface of the object.

We note that planar curve drawings, such as those shown in Figure 3.14, may consist of many small strokes that together define a rough contour as shown. This sketching technique may be used, for example, to express intent of a sharp change in curvature in the direction orthogonal to the contour. As the boundaries for planar sections are not represented in this way, there is arguably some loss of information moving from sketched drawing to planar section representation.

3.7 Summary

To summarize, we first presented planar sections, which are the fundamental primitive used for our planar section representations. We discussed how they are represented mathematically, and how they can be constructed from existing geometry. We also discussed how the planar sections can be made to interlock if fabricated, and other conditions for physical assembly. As our focus is on planar section representations, this material will be important throughout the remaining chapters of the thesis.

We then laid the groundwork to explore important geometric features of a surface, which are critical to our bag-of-features approach detailed in Chapter 4 where planar section representations are algorithmically created from input surfaces. We closely examined the concept of curvature – a fundamentally important geometric property that is used to define perceptually important surface features like ridge and valley curves. We also covered the segmentation of surfaces – the segments themselves are also perceptually meaningful geometric features that our automatic approach is also dependent upon.

The various low- and high-level shape representations presented are of interest also from a technical standpoint as our implementations in Chapters 4 and 5 rely on these representations, for instance: input surfaces and even the planar sections themselves are represented as polygonal meshes; we use cubic Bézier splines to represent planar section contours in our interactive system detailed in Chapter 5; and we show the similarity between some existing shape representations and styles of planar section representation such as voxels and the regular XYZ planar sections, which we study in Chapters 4 and 6.

We surveyed a wide range of shape representations that exist on a continuum defined by the amount of local shape knowledge required for primitives in the representation. However, we importantly distinguished many as not being true *shape abstractions* – a category of shape representation where primitives are selected judiciously based on global and perceptual knowledge. Importantly, we note that while all planar section representations are shape representations, not all planar section representations qualify as shape *abstractions*. To be an abstraction, the *process* that planar sections are selected must incorporate global shape knowledge and perceptual understanding. (For the remainder of this dissertation, in lieu of a more appropriate term, this is the meaning we associate with the word "abstraction".) In Chapter 4, our focus is on how to make a computer perform this process – to create true shape abstractions.

Chapter 4

Automatic Creation of Planar Section Representations

We now describe our work on a new scheme to automatically derive perceptually-motivated shape proxies from examples. We first describe a user study conducted to gain insight about how humans define planar section representations of various 3D objects. We observed that humans share a consistent notion of abstraction using planar sections, and further that the chosen planes are correlated to geometric shape features (Section 4.1). Based on this study, one selects a minimal configuration of planes that captures a given set of geometric features of the 3D shape, rather than approximating the object surface. We show this problem is NP-hard, discuss various design possibilities, and propose a solution where planes are progressively selected to maximally capture shape features weighted by their importance (Section 4.2). We then discuss geometric features used in our realization (Section 4.3) and learn their relative importance from the user-study data (Section 4.4). Each selected plane reduces the importance of the features it captures (so that subsequent planes cover different features) and favour new features on the basis of orthogonality and symmetry relationships among planes in the shape proxy. Finally, we evaluate the results of our algorithm on both user-study and novel objects using a second user-study to show that planar proxies are indeed an easily recognizable shape abstraction (Section 4.5). Later, in Chapter 7, we will demonstrate many applications for the extracted planar section shape proxies.

4.1 Planar Section Proxies Created by Humans

We conducted a user study to answer the following questions regarding how humans define planar section proxies:

(i) Can humans, to their perceptual satisfaction, represent common 3D shapes using a small number of planar sections?

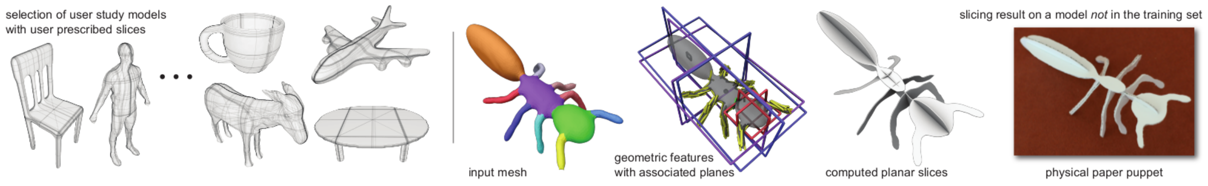

selection of user study models
with user prescribed slices
input mesh
geometric features
with associated planes
computed planar slices
slicing result on a model *not* in the training set
physical paper puppet

Figure 4.1: Based on a user study, where participants abstracted meshes of common objects using a small collection of planar slices, we develop an automatic algorithm to create planar slice based abstractions of (untrained) models. Starting from a set of planar slices, approximating the object's geometric features, the algorithm picks a subset of planes based on relative feature importance learned from the user study.

(ii) Is there a correlation among the planar sections prescribed by different humans, for the same 3D shape?

(iii) How well are the user prescribed planar sections correlated with the geometric features of the 3D shapes?

3D shapes can vary in complexity from a simple sphere, captured by three orthogonal planes (see Figure 4.8-bottom-left), to an intricately complex engine block (see Figure 1.6 in Section 1.1) that may require a large number of planar sections even for a minimal representation. A user study aiming to answer the questions above thus requires a representative set of 3D shapes of manageable and comparable complexity. These models should span various shape categories, such as airplanes, cups or tables, and also have examples capturing shape variability within each category. We found the models from the Princeton 3D segmentation benchmark (PSB) [22] suitable as each model is roughly axis aligned, well sampled, water-tight, and creates well-defined planar sections either on the entire object or per segment (see Figure 4.1). All objects were centred to origin and normalized to unit box.

4.1.1 User study design

In a pilot study, we asked 5 artists and 13 amateurs to interactively explore planar section proxies for presented 3D shapes. Without instruction, some tried to capture shape detail by spending over half an hour to place upwards of 20 planes as regularly placed sections (see Figure 4.4). We empirically found that most shapes in the PSB could be captured well using 5 to 10 planes, requiring about 5 to 10 minutes per model. For the actual user study we thus asked each participant to select planar sections with a soft limit of 10 planes/model over a selected set of 19 models consisting of 5 cups, 5 airplanes, 5 tables, a chair, a machine part (bearing), a biped (human) and a quadruped (donkey) (see Figure 4.3).

A single plane intersecting a closed manifold can produce a number of disconnected contours. In some cases a useful abstraction of the shape includes all the disjoint contours of a planar section. The cross-section of the legs of the table or human feet in Figure 4.3 are examples. Some users, however, specifically marked contours or partial contours of interest for each plane, when given appropriate tools (as in our pilot study). While the resulting proxies can be visually more pleasing, the process is more time-consuming. Hence, for the actual user study, we provided a conceptually simpler and more efficient user interface where users simply picked a unique set of planes and all the intersecting contours formed the planar section proxy.

4.1.2 User interface

The 19 models were displayed to 18 users in random order so that participants were less influenced by prior models when sectioning similar shapes. The users could tumble and zoom the models using a conventional 3D view manipulation interface. Planes were viewed as a plane widget (see supplementary demo) with the section curves overlaid on the model or as a filled planar section without the model. Users often toggled between the views to see the model while placing and evaluating new proxy planes.

Planes could be added, removed, or edited at any time. We provided three means of selecting planes: (i) snapping to axis aligned planes; (ii) starting from one of the axis aligned planes, rotating the plane to refine the azimuth and polar angles, and translating the plane along its normal direction; (iii) enabling a novel widget where a plane is specified by three points than can be interactively moved directly on the surface of the model. The second option was typically used when a *near* axis aligned plane was to be selected, while third option was employed when participants wanted to pick a plane passing through visible features or landmarks. Users reported completing the task in two or three sessions of around 30-45 minutes each and felt the interface provided adequate control for the given task and were satisfied with the abstractions.

4.1.3 User study findings

Do a small number of planes suffice?

Users remarked that they did not feel constrained by the soft maximum, also evidenced by the user study where only 5 planar-section proxies had more than 10 planes, and a maximum of 14 planes in one case (see Figure 4.4). The mean and standard deviation of the number of planes used across all users and models were 4.77 and 2.16, respectively (see Figure 4.2-top blue columns). Thus, we observe that common 3D shapes can be satisfactorily abstracted using a small number of planar sections (question #1). Note that user #15 tended to use more planes than others (mean 8.421, standard error 2.194), but exceeded the soft maximum for only 2 models.

Do users create consistent planar sections?

Figure 4.3-top summarizes the results of our study. Each planar-section chosen by a user is overlaid on the model as a faint curve. Thus, lines of greater thickness and opacity indicate agreement among the planar sections selected by different users to represent the same shape (question #2). Next, we quantify this observation.

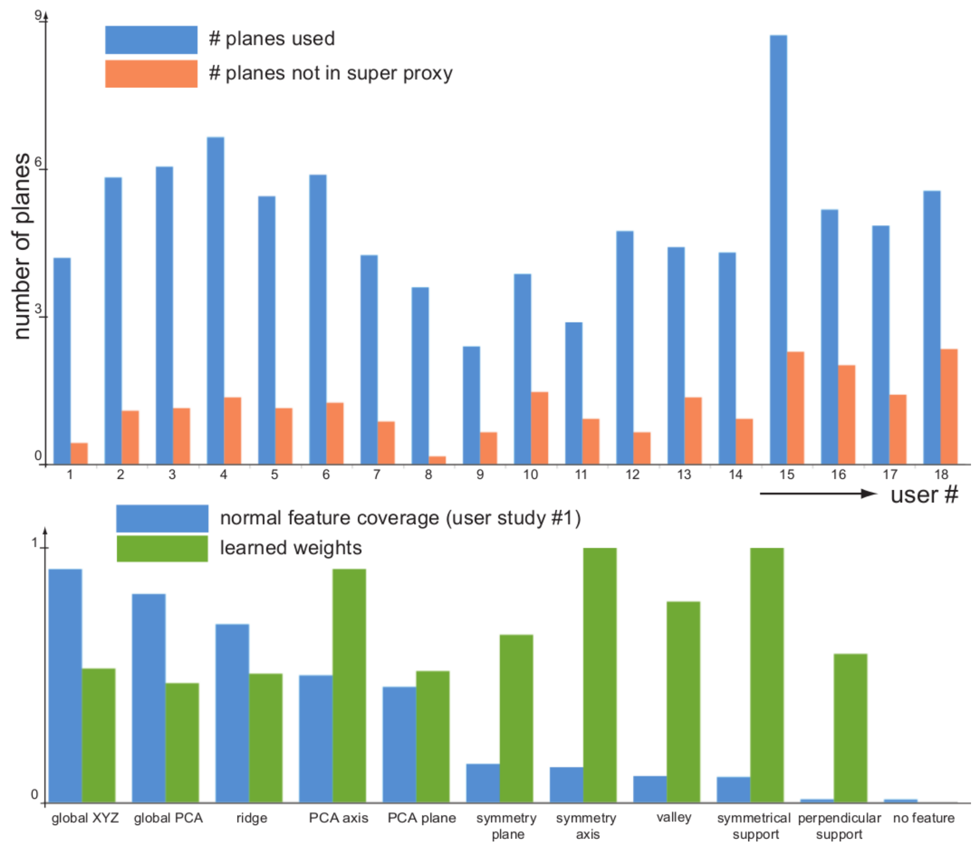


Figure 4.2: (Top) For each of the models, there was a clear consistency among the slicing planes prescribed by the users. (Bottom) We observed a strong correlation among the selected planes with geometric features of the models (blue columns), while the corresponding learned weights (see Sections 4.2 and 4.3) are in green.

First, for each model, we group the planes prescribed by the different users into *equivalent* groups based on their mutual distance (in plane-space), and select a representative from each group. Specifically, for a given model, we define a *plane-likelihood* function as the sum of radial sigmoid functions defined around each plane belonging to the user defined proxies for the model. The fall-off radius is defined by the minimum of a feature size (0.1) and the minimum distance between any two planes belonging to the same proxy over all users (0.059). Groups of visually equivalent planes are then represented by the local maxima of this plane-likelihood function, which we evaluate by discretizing plane-space (see Section 4.2). All models are centred at the origin and normalized to fit within a unit cube.

Next, we define the *super proxy* simply by picking points in the discrete plane-space whose plane-likelihood is above a given threshold (4.5 in our experiments). Figure 4.3-middle shows the super proxy on our input data indeed results in a visual proxy similar to many of the user defined proxies. The colour of the super proxy indicates their plane-likelihood from hot (red) to cold (blue).

Finally, for each user, we measure the deviation from the general collective as the number

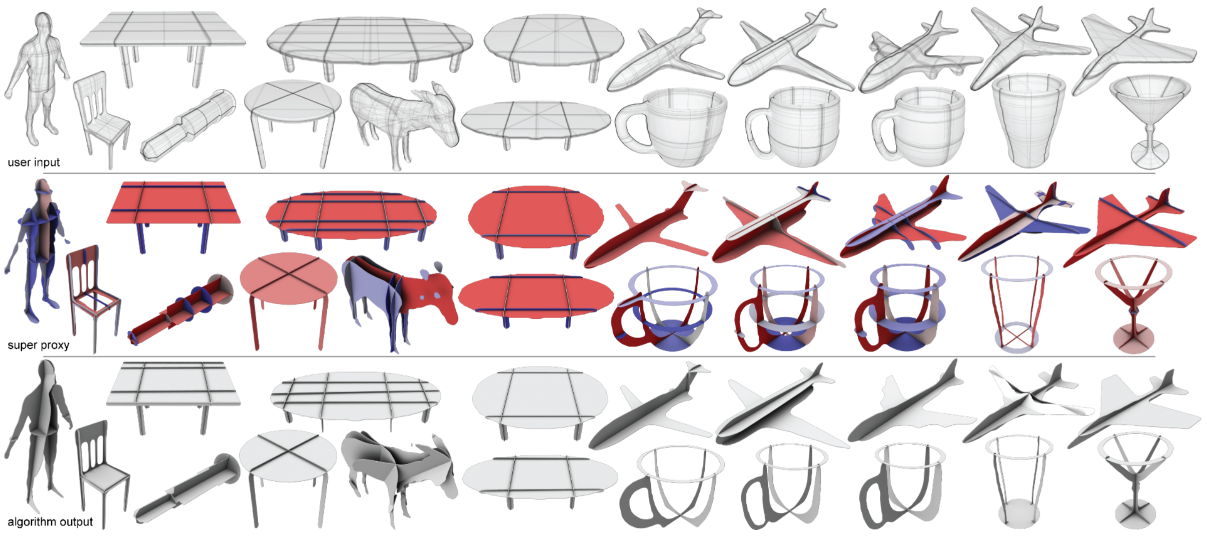


Figure 4.3: User study planar section proxies as faint lines aggregated on the models (top) from which a *super* proxy is computed (middle). The hot (15 users) to cold (4 users) gradient indicates user agreement on the depicted planes. Note the correlation between the proxy created by our algorithm (bottom) and the user agreement on depicted planes (middle).

of planes in their proxy that do not contribute to the super proxy (see Figure 4.2-top orange columns). Overall, we observe that a large number of the selected proxy planes are consistent across users.

Do users capture geometric features?

We hypothesize that users prescribe sectioning planes aligned to geometric features of the shapes (question #3). This was verbally reaffirmed post-study by two of our artist users. We also compute representative planes for various geometric features (see Section 4.3) for the user study objects, and test how well such planes align with the user prescribed planes. Figure 4.2-

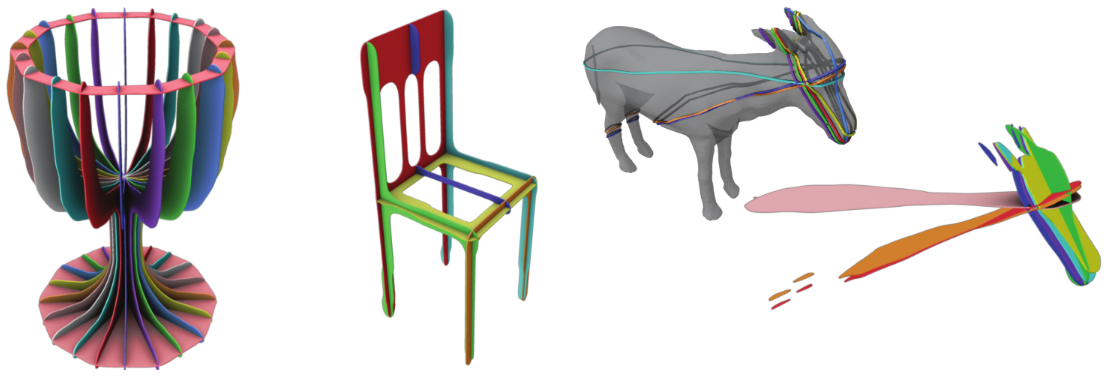


Figure 4.4: User proxy problems: (left) regular sections in our pilot study; (middle) inadvertently capturing concave under side of chair; (right) planes covering no geometric feature on the donkey.

bottom indicates that about 1% of the user prescribed planes (17 of 1630) remain unexplained by the chosen geometric features (note that a sectioning plane can be explained by multiple features, and hence the column sizes add up to more than unity). The heights of the (blue) columns give a qualitative importance of the features. However, such a reasoning ignores that often the choice of planes influences subsequent planes. In Section 4.3, we describe a principled approach to learn the importance of the features, while accounting for the inter-dependencies.

4.1.4 Discussion

Inherent shape knowledge

In certain cases, the participants use their inherent knowledge of the shapes to select the sectioning planes, without any apparent correlation to the geometry of the models. For example, for the chair shown in Figure 4.4, all users simply represented the seat with a single planar section despite the geometric concavity at the bottom of the seat (7 of 16 human-authored planes intersecting the seat had a hole, we believe these users did not realize they captured a concavity as this would require inspecting the chair from below). None of the users marked two planes to indicate both the chair seat and the presence of a concavity below, indicating the use of semantic knowledge of a chair in ignoring the concavity.

Of the 17 human-authored planes in the entire study that capture no geometric features we consider, 8 of these planes are for the donkey model (see Figure 4.4). A number of these user planes roughly pass through the ears, eyes or where one might imagine the donkey's mouth. The variability of these user planes further indicates the use of model semantics and the lack of a clear visual landmark in selection of the planes. These planes may in fact be intersecting multiple distant features simultaneously, but in an approximate manner where the plane is not aligned well with the individual features. Such cases being sparse (see Figure 4.2), we focus only on the geometric aspects of the planar proxies, without requiring additional semantic knowledge.

Design choices

Our study takes about two to three hours per user, and so we only enlisted dedicated users in a controlled experimental setup, instead of devising mechanisms to deal with noisy data with high variance. For this study, we did consider using Mechanical Turk, as did Cole et al. [30], but we were unable to devise a fair experiment that could be completed in a short time. Later in Chapter 6, we devise a large-scale crowd-sourced experiment on Mechanical Turk, where we explore in detail the surface perception of planar section representations.

For models in the PSB dataset, we observed that from around 12 to 15 users the super-proxy set and the corresponding learned weights (see Section 4.5) stabilized. Hence, we limited our user study to 18 users.

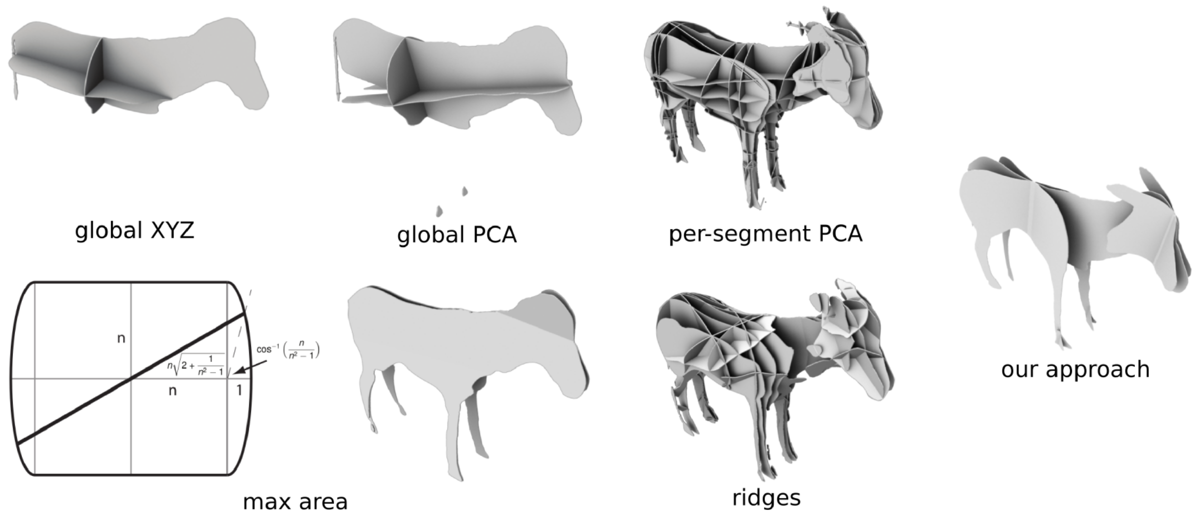

global XYZ
global PCA
per-segment PCA
our approach
n
$n\sqrt{2+\frac{1}{n^2-1}}$
$\cos^{-1}\left(\frac{n}{n^2-1}\right)$
n
1
max area
ridges

Figure 4.5: Simple alternate algorithms give rise to non-intuitive planar slices. (Right) Results from our approach consist of few slices that capture many important mesh features. (Bottom-left inset) A planar section of maximum area seen within a 2D cross-section can cut a shape at an unintuitive angle.

4.2 Algorithm

Our goal is to produce a similar proxy $\mathcal{S}$ of shape S using a small number of planar sections, such that S and $\mathcal{S}$ are perceptually equivalent. This is an ambitious goal given the lack of a suitable computational model of perception. Several seemingly natural formulations turn out to be insufficient (see Figure 4.5): (i) An inefficient but conceptually simple solution is to consider the $\binom{n}{3}$ planes defined by the n mesh vertices (for a poly-mesh model) and choose those maximizing a combination of geometric functions such as, sectional area, length of the section perimeter and total curvature along the section perimeter. While maximizing these functions can locally optimize planar sections, globally they often have no perceptual significance. (ii) Another simple solution is to select axis aligned planes, or principal component (PCA) planes, globally or locally for all segments [22]. The results are unsatisfactory on two accounts. First, such axes and planes do not explicitly capture shape symmetries or geometric landmarks, and second, a simple aggregate of these planes ignores inter-plane relationships, which are especially critical when choosing a minimal set of planes. (iii) Another approach is to compute ridge and valley features for the mesh, making cuts in the direction of the surface normal. Where ridges and valleys are defined some geometric landmarks will be captured, however, once again the set of planes is neither minimal nor accounts for inter-plane relationships.

Instead, we design our algorithm based on the observed consistency between shape features and planes selected in our user study. We hypothesize that choosing a small number of planes to maximally *cover* the characteristic object features produces a good planar proxy. We show the problem of finding the minimum set of planes that covers a given feature set is NP-hard (see

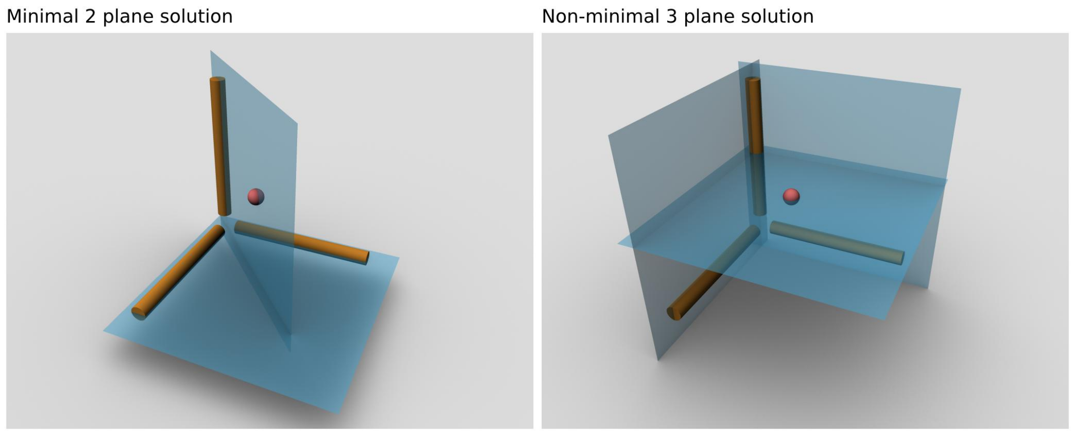


Figure 4.6: (Left) The MPS solution for a shape with one point feature (red sphere) and three axis features (orange cylinders). (Right) The perceptually preferable solution does not have the minimum number of planes.

Subsection 4.2.3). Further, the minimum solution is not always perceptually preferable, since humans also factor in geometric relationships among selected planes when defining a proxy. Figure 4.6 shows a scenario where covering the shape features with three orthogonal planes is perceptually better than the minimum 2 plane solution.

We thus propose a framework where an extendable set of geometric features F (see Section 4.3) are used to populate a parameterized plane-space. We then iteratively select planes corresponding to the most densely populated regions of plane-space, as we did with the computation of the super proxy. Selected planes also *re-adjust* the plane-space with respect to covered features, introducing new features to capture geometric relationships between the selected planes and those that will subsequently be selected. Our algorithm runs in two phases: (i) initialization involving discretizing and populating the plane-space using feature set F, and (ii) iteratively selecting planes to cover any uncovered and populated regions of the plane-space.

4.2.1 Initialization

As introduced in Section 3.1, we place coordinates on plane-space by assigning to a plane P its distance d to the origin and the polar-azimuth coordinates (θ, ϕ) of its unit normal vector, such that $\theta = [0, 2\pi)$ and $\phi = [-\pi/2, \pi/2]$. These coordinates are undefined when the normal vector is vertical, and discontinuous where the azimuth coordinate $\theta = 0$ or approaches 2π. The inverse map is the parameterized plane-space (we will call this the "plane-space" from here onward). The plane-space is well-defined everywhere, but many to one when the plane's normal is parallel to the polar axis ($\phi = -\pi/2$ or $\phi = \pi/2$). We discretize the space of all possible planes by partitioning the plane-space into a collection of bins. In our experiments, we sample θ and ϕ every 2 degrees, and d every 0.02 units. All models are centred at the origin and normalized to fit within a unit cube. Initially, all the bins are empty.

We populate the plane-space bins using candidate feature planes. Our framework handles a collection of geometric features $F := \{f_1, f_2, \dots\}$ extracted from an input mesh. Each feature $f_i \in F$ has an importance weight w_i, which is learned from the user study data set (see Section 4.3).

Suppose feature f_i is such that it can be defined by the single plane $\mathbf{n}_f^T \mathbf{p} - d_f = 0$ (some features are defined by sets of planes in fact, discussed later). Each coordinate in plane-space (θ_b, ϕ_b, d_b) maps to a corresponding plane $\mathbf{n}_b^T \mathbf{p} - d_b = 0$. Note that the mapping from a plane-space coordinate to plane is surjective, as at singularities ($\phi = -\pi/2, \pi/2$) multiple bins map to the same plane. We define the distance between feature f_i's plane and bin b's plane as

$$dist(f_i, b) := \arccos(\mathbf{n}_{f_i}^T \mathbf{n}_b) + \mid d_{f_i} - d_b \mid, \tag{4.1}$$

recalling that all models used in our experiments are normalized to fit within a centred unit cube, which is important to note as Equation 4.1 is not scale-invariant.

Discretization of a candidate plane to its corresponding bin in plane-space produces aliasing artifacts. Addressing this, we distribute the effect of a feature $f_i \in F$ to a local collection of bins within an influence radius r_i (set to 0.1 in our implementation). We define f_i's coverage of bin b as

$$c_i^b := 1 - dist(f_i, b)/r_i, \tag{4.2}$$

for those bins b where $dist(f_i, b) < r_i$, and set $c_i^b = 0$ otherwise. Note that since the model is normalized, we treat the contributions from angular and distance deviations equally. The final weight for each bin b is then taken as the accumulated contribution over all the features as $w_b := \sum_{f_i \in F}(w_i \cdot c_i^b)$. Figure 4.7-top shows the plane-space after initialization with features from the *human* model with the bins coloured according to their respective feature types.

Features can take different forms in the plane-space, based on whether they are one of three types: *point*, *capsule* or *spindle* (see Figure 4.7-top right). *Point* forms are used to represent feature types described by a unique plane, such as a symmetry plane. For other types of features, such as the global PCA planes, the plane's normal direction is important but the particular d value is not. To capture this acceptable variation in d (i.e. ensure intersection with mesh segments relevant to the feature), we use the *capsule* form in plane-space. To create this form, we use a slightly modified form of Equation 4.1, choosing a d_f value in the acceptable range closest to d_b to minimize the distance. Finally, feature types with a dominant axis such as segment PCA axis features and symmetry axis features we capture using the *spindle* form. For this form we allow variation in the plane's normal (varying d accordingly so the entire axis intersects the plane). We again rely on a modified form of Equation 4.1 where we choose a normal n_f that is both orthogonal to the axis and minimizes distance to bin b's plane. Our system can be extended to support new feature types with minimal effort by encapsulating them with one of these three forms.

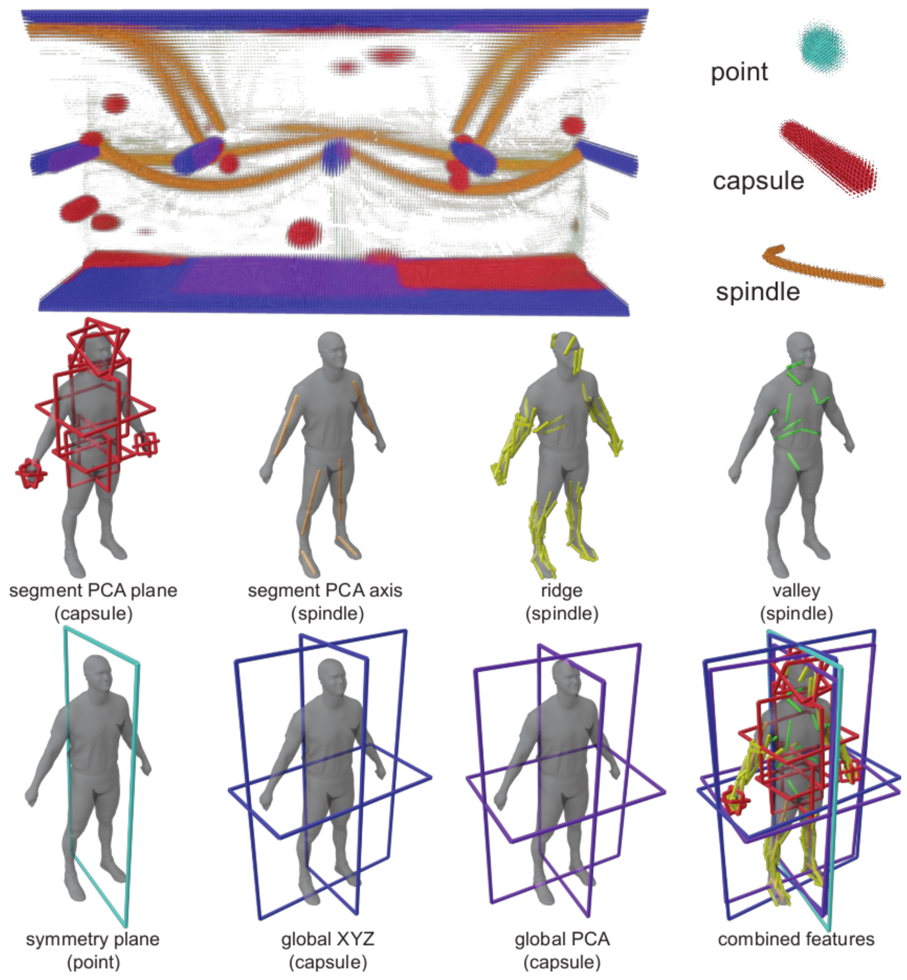


Figure 4.7: Plane-space (top) after initialization, populated with the set of features (bottom) detected for the human model, has three coverage forms: point, capsule and spindle. Learned weights scale the importance of each feature type, and we visualize this by scaling the size of each plane-space bin.

4.2.2 Covering the plane-space

Next, we select planes to cover the populated bins $\mathcal{C}$. We maintain the current set of selected bins in $\mathcal{P}$ starting with an empty set $\mathcal{P} \leftarrow \emptyset$. In each iteration, we choose the bin $b^* \in \mathcal{C}$ with maximum coverage value, provided w_{b^*} is above a threshold weight, set to 1 in our experiments. The chosen bin is added to the current set $\mathcal{P} \leftarrow \mathcal{P} \cup b^*$.

We then *adjust* the weights of the remaining bins to account for the selection of bin b^*. For each feature $f_j \in b^*$, we adjust the weights of each of the *other* bins containing f_j. Specifically, we identify bins $b_i \in \mathcal{C}$ that are touched by feature f_j, and for each we remove its contribution to weight w_{b_i}: $w_{b_i} \leftarrow (w_{b_i} - w_j \cdot c_j^{b_i})$.

4.2.3 The MPS problem is NP-complete

We show the minimum planar section (MPS) problem to be NP-complete in order to justify our choice of a greedy algorithm in our approach. Two steps are necessary in order to show that the MPS problem is NP-complete:

(i) show that MPS is NP-hard,

(ii) show that MPS is in NP.

MPS is NP-hard

We show that finding a minimum planar section (MPS) that covers a set of shape features is NP-hard by reducing the classic NP-complete minimum vertex cover (MVC) problem to MPS in polynomial time. Formally, given a graph $G = (V, E)$ a k-cover is a set of k vertices in V such that every edge in E has at least one incident vertex in the set. The MVC problem is to find a k-cover with the smallest k.

We construct an input shape to the MPS problem as a collection of $|E|$ disjoint cylinders. The shape features are thus the $|E|$ cylindrical axes (the heights and radii being irrelevant). We will construct every cylindrical axis C_e to correspond to an edge $e \in E$ such that:

Property 1: Axes C_e and C_f are co-planar *iff* e and f are adjacent.

These axes will be defined as the intersection lines of $|V|$ planes representing the vertices of the graph. We first define the plane normals such that:

Property 2: Any three normal vectors n_i, of planes $i \in 1..|V|$, are linearly independent.

The set of vectors $n_1..n_{|V|}$ arranged in a cone around the Z axis, where

$$n_i = \langle \cos(2\pi i/|V|), \sin(2\pi i/|V|), 1\rangle$$

satisfies this property. Thus any axis $C_{(i,j)} = n_i \times n_j$ is non-zero. Given distances to the origin d_i and d_j for planes i and j we can fix a point $p_{(i,j)}$ on $C_{(i,j)}$ as

$$p_{(i,j)} = \frac{[d_j(n_i \cdot n_j) - d_i]\, n_i + [d_i(n_i \cdot n_j) - d_j]\, n_j}{(n_i \cdot n_j)^2 - 1}.$$

Any point $p(t)$ on $C_{(i,j)}$ is thus $p(t) = p_{(i,j)} + (n_i \times n_j)t$. The intersection of axis $C_{(i,j)}$ and $C_{(j,k)}$ is now defined by $d_i = n_i \cdot (p_{(j,k)} + t(n_j \times n_k))$. We define $intersect(i, j, k)$ to be the parameter t along axis $C_{(j,k)}$ where it intersects $C_{(i,j)}$. Rearranging the first equation:

$$intersect(i, j, k) = \frac{d_i - n_i \cdot p_{(j,k)}}{n_i \cdot (n_j \times n_k)}.$$

Note that $n_i \cdot (n_j \times n_k) \neq 0$ by *Property 1.*

The distances to origin d_i for the planes $i \in 1..|V|$ is fixed such that:

> *Property 3:* The intersection axes $n_i \times n_j$ and $n_k \times n_l$ of any four distinct planes i, j, k, l do not intersect.

We perform this assignment iteratively through the planes i from $1..|V|$ setting distance d_i as follows:
Enumerate every permutation of planes j, k, l whose d values have been set (i.e. $j, k, l < i$). For each permutation j, k, l we compute the d for plane i where $C_{(i,j)}$ and $C_{(k,l)}$ would intersect. We add this value of $d = n_i \cdot p_{(j,k)} + \mathit{intersect}(l, j, k)\,[n_i \cdot (n_j \times n_k)]$ (from the first equation) to a set D_i of forbidden d values for plane i. We then set d_i to be any arbitrary value not in D_i.

Property 1 holds for the above construction: If axes C_e and C_f are adjacent at a vertex v_i, they lie in plane i and are thus co-planar. Conversely, any co-planar axes must either be parallel or intersect. By *Property 2* no two cylindrical axes are parallel and by *Property 3* two axes can only intersect if they are adjacent.

> *Lemma 1:* Any set of k planes that covers the cylindrical feature axes of this shape is equivalent to a k-cover of the graph.

Proof: Given a k-cover, if we pick the planes corresponding to the vertices in the cover, we will cover all feature axes corresponding to the graph edges. Given a set of k planes that cover the $|E|$ symmetry axes, we note by *Property 1* that for any plane to contain two or more axes, the edges corresponding to the axes must be adjacent and the plane must correspond to their common incident vertex. We add this vertex to our vertex cover. Finally, any plane that contains only one symmetry axis $C_{(i,j)}$ can be rotated around $C_{(i,j)}$ to match the vertex plane i or j and we can add either v_i or v_j to our vertex cover. As a consequence of *Lemma 1* and our $O(n^4)$ reduction, we have MVC $\leq_T$ MPS and thus MPS is NP-hard.

MPS is in NP

The MPS problem is combinatorial, and involves the selection of a subset of $|V|$ possible planes that cover $|E|$ shape features. It can be cast either as a decision problem or an optimization problem.

The MPS decision problem is the following: "can all shape features be covered using no more than k planar sections?" The output solution is either "yes" or "no".

The MPS optimization problem is the following: "what is the minimum number of planar sections required to cover all shape features?" The output solution is the minimum number of planar sections required. To find the minimum number, one can solve the MPS decision problem $|V|$ times, using the values $k = 1, \ldots, |V|$. The solution to the optimization problem is the minimum k where the solution to the decision problems was "yes".

For the MPS decision problem to be in NP, it is necessary that one can prove a solution in polynomial time (polynomial with respect to the size of the instance of the problem). A certificate is used to prove a solution. For MPS, the certificate is a set of planar sections, and two steps are required to verify the certificate:

(i) verify the set of planar sections has k or fewer planar sections ($O(|V|)$);

(ii) iterate through each of the E shape features, verifying each is covered by one of the planar sections in the certificate ($O(|V| \cdot |E|)$).

The verification of the certificate can thus be done in polynomial time. MPS is in NP and is therefore NP-complete.

4.3 Shape Features

We describe our user-study guided feature set selection, while noting that other features can easily be incorporated if they take one of our plane-space forms. We learn the relative feature weights using user study data to capture their relative perceptual importance.

4.3.1 Principal planes

The PCA axes of the shape and of the individual segments define the oriented bounding box of the shape and its segments, respectively. One or more of these planes captures the principal directions. Based on the relative strength of the principal directions (eigenvalues of the covariance matrix), we add features as follows: (i) for a model with a single dominant principal axis, we use a single spindle form for the axis, (ii) for a model with two dominant directions, we use a single plane feature (capsule) with normal pointing in the direction of the least significant eigenvector, otherwise (iii) we use three planes (capsules), each having a normal in the direction of each eigenvector. In the case of segment PCA planes that take capsule forms, we constrain the range of d value so that the segment will be intersected.

4.3.2 Global planes

Since the initial database models are mostly oriented, the global X, Y, Z coordinate axes help to ground the shape and fix its orientation with respect to the environment. Their representative planes, like PCA planes are represented using capsule forms and can thus vary in d value.

4.3.3 Ridge and valley curves

Ridge and valley curves represent extrema of surface curvature on the shape. They can be efficiently computed for detecting features at multiple scales. Most importantly, they are believed to capture geometrically and perceptually salient surface features. We detect ridge and valley curves using the algorithm in [106]. A greedy algorithm splits these curves into line segments, treating each as a spindle feature and giving each a weight based on segment length. Since each segment is a spindle feature, a coplanar group common to a larger ridge or valley feature (e.g. a ridge along the rim of a cup) will intersect in plane-space (with maximal coverage of these ridge segments). In our implementation we chose to use a strict threshold for detecting ridges and valleys, instead of weighing each by a measure of strength (e.g. average principal curvature along the curve). By filtering out many potential ridges and valleys whose average curvature is low, we avoid the processing associated with adding these relatively insignificant features to the plane-space.

4.3.4 Symmetry planes and axes

Symmetry is strongly connected to shape abstraction and perception, and often is persistent across variations in object collections relating to part hierarchies [137]. We consider global reflective and rotational symmetries in our framework. We detect global symmetries [101] for the entire shape. Reflective symmetries are captured well by a planar section defined by the symmetry plane itself. The symmetry is also visually clear in the shape contour of a planar section that is perpendicular to symmetry plane. We add symmetry planes to the plane-space as point forms, and symmetry axes as spindle forms.

4.3.5 Perpendicular supports

We observe from the user-study that it is desirable to capture *spindle* features like a PCA axis or global symmetry axis of a model (see Figure 4.7) using a pair of orthogonal planes, with the spindle being their intersection line.

If the maximal bin b^* covers a spindle feature with axis direction $\mathbf{a}$ and intersection point $\mathbf{p}$, we add a perpendicular support feature f_{perp}. Feature f_{perp} defines a plane feature that also intersects the axis, but in a direction perpendicular to the plane defined by b^*. The parameters for our supporting feature $\mathbf{n}_{perp}$ and d_{perp} are given by:

$$\mathbf{n}_{perp} = \mathbf{n}_{b^*} \times \mathbf{a}_f \quad \text{and} \quad d_{perp} = -\mathbf{p}^T \mathbf{n}_{perp}.$$

If d_{perp} is negative we flip the signs of both $\mathbf{n}_{perp}$ and d_{perp} to keep all d values non-negative. Perpendicular supports are point forms in plane-space. Note that each spindle feature is allowed to generate a maximum of one perpendicular support.

4.3.6 Symmetric supports

In order to retain planar symmetries of the model, if we select a plane that passes through a symmetric segment that is not almost perpendicular to the symmetry plane ($< 80°$ in our implementation), we add its reflected plane, if not already selected, as a point-form feature to plane-space to increase the likelihood that it will be selected.

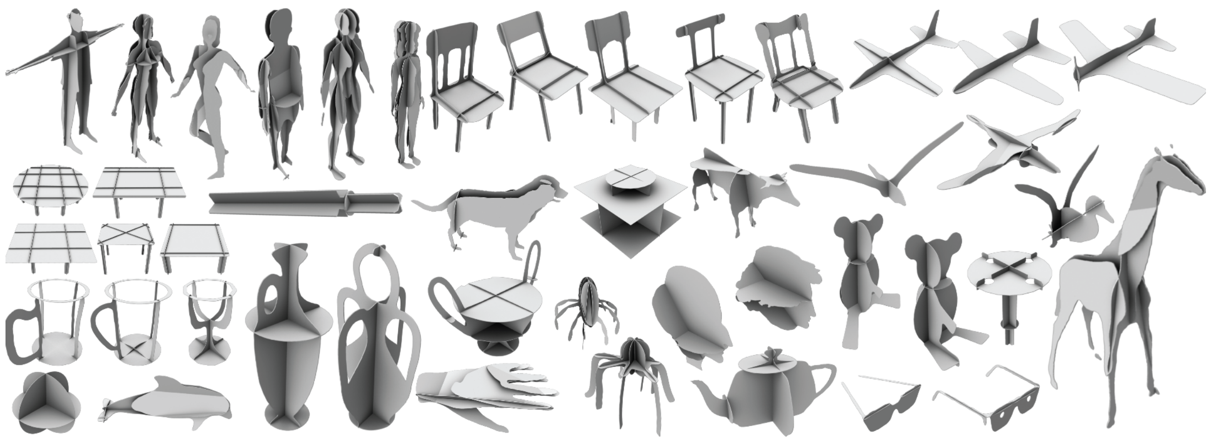


Figure 4.8: Algorithmic results on a variety of models from the PSB (see supplementary for full set). Note the consistent behaviour on cups, tables, and other models similar to those in the user-study (see Figure 4.3) as well as completely novel model examples like heads, vases and birds.

4.4 Learning Weights

In absence of a computational model for evaluating planar abstractions, we use the user study findings to learn relative weights for the various geometric features. We do this in two parts: (i) we obtain a representative set of planes called the *super proxy*, and (ii) we derive a large number of linear inequalities from this set, and apply a constrained optimization solver to learn the feature weights.

First, for each model, we form a set of planes called the super proxy using the planes chosen by the 18 users (see Section 3).

Next, we create a linear system that consists of inequalities obtained from the super proxy data. We assume that super proxy planes are selected by a process that mimics our algorithm, that is, planes are selected one by one in order of maximal feature coverage as long as the value of each plane is sufficiently high. For the super proxy, this importance ordering of planes is captured by their user agreement value (see Section 3). We derive two types of inequalities. The first type are for each plane in the super proxy – these are planes that our algorithm should select, so we want the feature weights to be such that the value of these planes (measured at the plane-space bin) exceeds the plane selection threshold. For the 10 feature types in our system, using the selection threshold of 1, super proxy plane inequalities take the form:

$$\sum_{i=1}^{10} w_i C_i \geq 1. \tag{4.3}$$

Note that these inequalities are created *in order* of plane selection, since plane selection influences subsequent inequalities (i.e. perpendicular and symmetric support features may be introduced). The left-hand side is the value of each super proxy plane, and is the sum of the total coverage C of each of the 10 types of features multiplied by the unknown weights. Each C_i is calculated by summing the coverage across each feature of type i in the model: $C_i = \sum_j c_j f_j$, where each j indexes a feature of type i, c_j is the coverage of feature j by the plane (Equation 4.2), and f_j is a feature-specific scalar (e.g. for ridges, f_j scales by the length of ridge j).

Generally, not every feature is captured by the super proxy. After processing each plane in the super proxy set and covering each of the mesh features in the plane space, there may be (and almost always are) a number of bins containing one or more features. These bins correspond to all planes that were not selected, and so their total value in terms of the features they cover must be below our threshold. For each of these bins, which map to a plane in model space, we have the second type of inequality:

$$\sum_{i=1}^{10} w_i C_i < 1. \tag{4.4}$$

We now choose weights for the features such that they satisfy all the inequalities across all models in the study. Our study consisted of 18 users and 19 models, and a total of 1630 planes were authored by all users across all models. Of these, 101 planes were selected as the representative super proxy set across all models. This set of planes formed 101 inequalities of the first type. Most of the 1630 user authored planes are captured by the super proxy, indicating consistent plane selection (see Figure 4.3). Our system contained 107,992 inequalities of the second type, one for every bin not selected by the super proxy planes for each model.

A simple solution can involve a least squares approach to best meet the inequality constraints. However, in our case we are interested in satisfying the inequality constraints, rather than forcing equality relations, e.g., we do not differentiate between relations once they cross the threshold value. The least squares approach resulted in many zeroed out coefficients as the inequalities were driven towards equality. Such an overfitting does not help as the set of weights do not generalize in producing useful results on new models, especially in the case where new models contain features that may be under-represented in the training set.

Simultaneously satisfying all of the inequalities is infeasible. We found numerous cases where planes in our super proxy did not have strong feature coverage. These planes corresponded to inequalities of the first type where the sum of total coverage $\sum_{i=1}^{10} C_i$ was low. Intuitively, as these planes do not cover features well, they are not very well suited for deriving feature weights. To address this, we progressively removed inequalities derived from the super proxy with low total coverage, until a solution was feasible. When $\sum_{i=1}^{10} C_i > 1.5$, the system becomes feasible and we use 74 inequalities from the super proxy set. For many planes in the super proxy, we observe a strong correlation between feature coverage and user agreement (see Figure 4.2). Interestingly, our algorithm uses the learned weights to consistently pick planes of high user agreement, while leaving out planes of low user agreement (see Figure 4.3, where the algorithmically-generated planes closely match the human-authored super proxy).

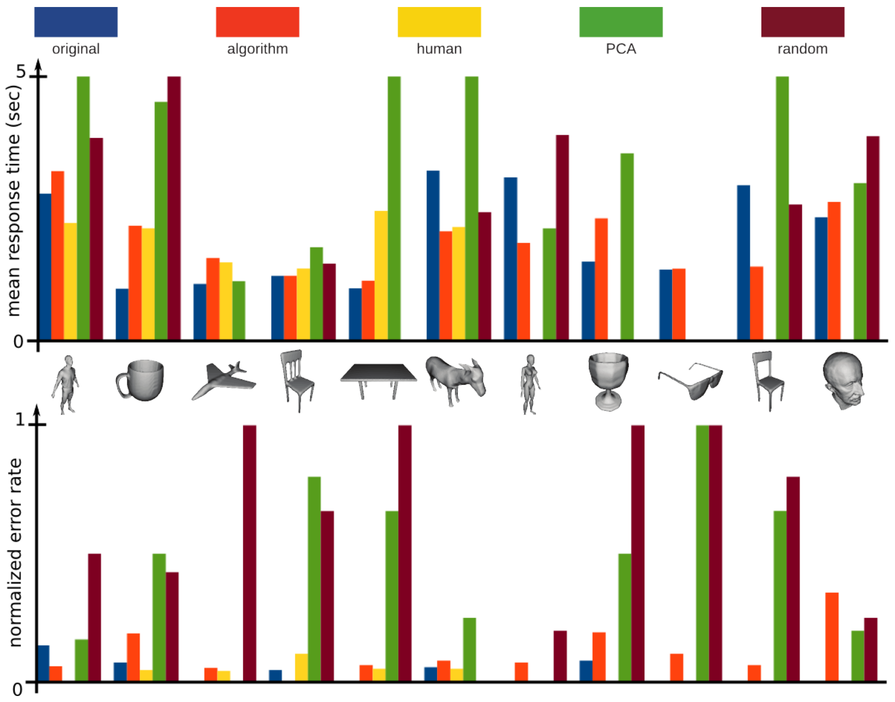


Figure 4.9: (Top) Average recognition times of objects and planar section proxies, for each of the representations. (Bottom) Normalized error rates in identifying objects across each representation.

To regularize the solution, we use an objective function for the solver that minimizes the difference of each weight from 1 in order to be well balanced across each type of feature, specifically, $\sum_{i=1}^{10}(w_i - 1)^2$. To determine the weights (see Figure 4.2) we used MATLAB's 'fmincon' function, a constrained optimization solver that relies upon the active set method [112].

4.5 Evaluation

In order to evaluate our algorithm, first we used the learned weights to recreate the planar slices for the 19 models from the training set (see Figure 4.3-bottom). Note the general correlation between the planes chosen by our algorithm and the level of user agreement for planes belonging to the super proxy.

In a more demanding test, we used the learned weights to create planar slices for all the models from the PSB. Figure 4.8 shows models similar to those in the user study dataset and is a validation of the learned weights, as well as showing visually compelling results of our algorithm on many new models (see supplementary material). As our goal is to produce perceptually sound planar slice abstractions, we performed another user study to judge the quality of the produced results, as described next.

Shape recognition user study

We conducted a user study involving 66 users, with ages ranging from 16–50, where each participant was shown one of 5 representations randomly selected from 11 models (6 from initial user study, 5 new without human annotation) that were shown in randomized order. Users were asked to identify the model, and their response time was recorded (up to 5 seconds). Each model representation was rendered from the same pre-selected viewpoint and was either: a mesh rendering, algorithm generated planar slices, human annotated planar slices, global PCA planar slices, or random planar slices. Our results for response times and recognition rates are summarized in Figure 4.9.

For the mesh rendering, the algorithm generated planar slices and human annotated planar slices, recognition rates were consistently high ($> 90\%$) and reaction times consistently low (less than 2 seconds), averaged over all 11 models and all users. This confirms two of our predictions: planar slices are a recognizable abstraction of common objects, and that our algorithm produces comparable results to human annotations. An interesting exception was the Max Planck model (model #317 in the PSB, rightmost model in Figure 4.9), where recognition rates were between 66–80% for all planar slice abstractions but 100% for the mesh rendering. In other occasional failure cases, users marked the 'donkey' as a 'pig' or 'sheep', or a 'vase' as an 'ashtray'. Absence of a scale and no prior information about what to expect partially explains such erroneous entries. The global PCA planar slice and random planar slice representations were notably much worse, with average recognition rates of 57% and 38%, and average reaction times exceeding 4 seconds (twice that of the algorithm generated and human annotated abstractions).

Persistence of abstraction under model perturbation

Persistence and resilience to shape resolution and perturbation are important properties of shape abstractions. In our framework, this resilience is largely handled by the algorithms that extract our set of geometric features and some degradation in the quality of their performance is expected. Figure 4.10 shows that our algorithm produces perceptually persistent proxies when applied to decimated or noisy versions of the original mesh. As expected, planes of lesser importance are perturbed or excluded, since these features are not dominant. Also, the creation of many spurious features (e.g., ridges emerging from noise applied to vertex positions) is unlikely to result in new planes, as their weight are scaled by their lengths.

Persistence of abstraction under model transformations

The metric used to measure the distance between two planes involves the addition of two terms of different units, one linear and one angular. Without treatment, if the input surfaces are scaled to different sizes, this could lead to different planar section representations being generated (the scaling causes the linear term of the distance metric to change, but not the angular term). In Figure 4.11, we demonstrate the effect of scale on the chosen planar sections when models are

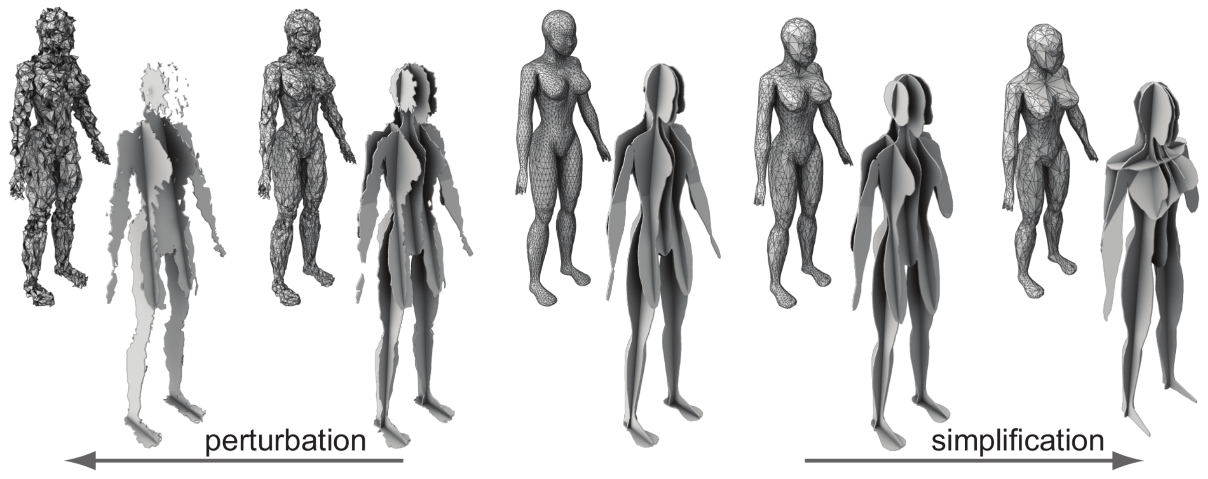


Figure 4.10: Persistence of model perturbation: model (centre) under increasing Gaussian noise moving left 0.005, 0.01 standard deviation and increasing decimation moving right 20%, 40%.

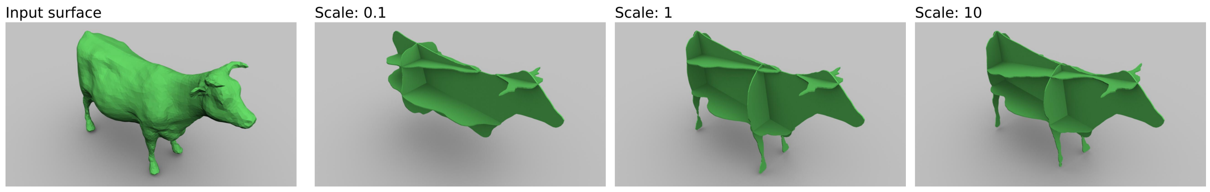


Figure 4.11: A models that is not scaled to the unit cube can potentially result in an unsuitable planar section representation. The model that is scaled to 0.1 has a representation that significantly differs from the other two scales.

not first normalized by scale. At scales 1 and 10, a plane is chosen for each pair of the cow's legs. However at scale 0.1, a single plane is deemed adequate ("close enough") to cover both pairs of legs, which is not an ideal result. To address this issue with scaling, models are always first normalized to fit within the unit cube (and translated so the centre lies at the origin). This ensures that the configuration of planes in the generated representation will be the same, regardless of scale or translation transformations.

But what about rotations? Recall that the parameterization for planes is non-homogeneous (e.g., when $\phi = -\pi/2$, all values of θ define the same plane). This non-homogeneity of the parameter space may potentially cause a bias in plane selection (i.e., the algorithm will prefer planes that have a vertical orientation, corresponding to the "spherical poles" in the parameterization). This could potentially result in the algorithm producing a different representation for a rotated model.

However, in our approach, this is not much of a concern. Features are not represented with a single bin, but with a volume in the plane parameter space that can contain many bins. The volume of a feature in plane parameter space near the "spherical poles" ($\phi = -\pi/2, \pi/2$) is much larger than a feature near the "equatorial line" ($\phi = 0$), where volume is evaluated by counting the number of bins within the volume. This difference in the shape of feature volumes

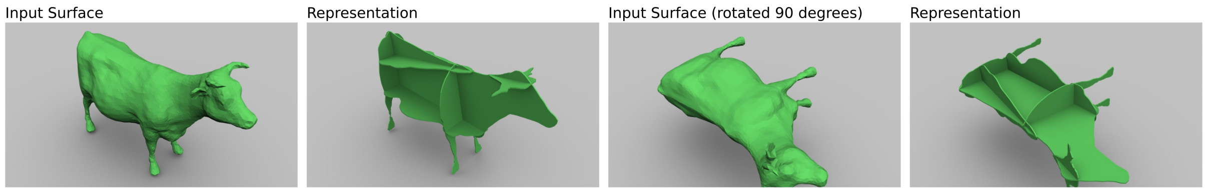


Figure 4.12: The planar section representations for a model with and without a 90° rotation applied have the same local configuration.

based on location in the parameter space compensates for its non-homogeneous nature.

Another reason why rotations are not a concern is that the features themselves are transformed with the model. As an example, we consider a model that has been rotated 90 degrees (see Figure 4.12). Since the model is rotated 90°, all but the global XYZ features will be rotated accordingly. Then, since the feature volumes are transformed in the plane parameter space, the bins corresponding to selected planes will differ. (Note that in order for our approach to be completely invariant to rotation, use of the global XYZ features or any feature that does not vary with rotation would need to be disabled. In practice however, there is a strong bias for models to be oriented along major axes, so this situation is rarely encountered.)

4.5.1 Planar-section aesthetics

Viewers generally found our algorithmic results in Figures 4.3 and 4.8 to be of high quality and recognizable as shape abstractions. Despite this, neither our result nor the human-derived super proxy is as magical as some of our motivational images (such as the examples in Figures 1.1 and 1.4 in Section 1.1). We believe the major reasons are: (i) an intangible artistic eye for composition that we did not set out to capture; (ii) our results echo the PSB, which was designed not for aesthetic reasons but to capture a range of shapes with consistent mesh topology, resolution and size; (iii) sections as seen in the motivational images are partial planar sections of the mesh, (iv) artistically created planar section contours sometimes deviate from the planar section to partially conform to the shape silhouette.

The application we designed for automatic planar section representations can also be used as an interactive tool enabling artists to compose their own shape abstractions (however we pursue this goal more rigorously in Chapter 5, where we consider the interactive creation of planar sections from scratch, as well as the interactive creation of planar sections from surfaces that are not ideally represented with planar sections).

For the basic interactive system we present here, we chose not to require our users to define partial planar sections to keep our user interface simple and the user study manageable. For fair comparison, our results in Figures 4.3 and 4.8 are shown as complete and unprocessed planar sections. We do, however, address the automatic computation of partial sections in three ways: (i) We note that segmentations in objects often indicate rigid parts that can articulate with respect to each other. We capture this property within our abstraction by cutting the planar section contours at segmentation boundaries, and capping them with tangent-continuous cubic Hermite splines into multiple overlapping but topologically disjoint planar sections. (ii) We only retain portions of the contours that pass through segments whose features are covered by the given plane, and cap them smoothly into closed contours. (iii) We observe that planar sections provide the strongest shape cue along the section contour where the surface normal of the shape is near perpendicular to the plane normal. Recalling the Frenet-Serret formulae and the Darboux frame introduced in Section 3.2, the shape cue is strongest when the section contour normal N and the surface normal N_S are collinear, resulting in zero geodesic curvature ($\kappa_g = 0$) and curvature and normal curvature being equal ($\kappa = \kappa_n$). We verified this by examining the distribution of the angle between the surface normal along the section contour and the plane normal for the super proxy planes (e.g., $> 70\%$ of curve length has angle $> 70°$). We can thus retain and cap portions of the section contour where the angle between the surface normal and the plane normal exceed a specified range. While it is straightforward to allow users to interactively edit parts of section contours to conform to model silhouettes, we leave the automatic inference of such hybrid planar sections to future work.

4.5.2 Optimizing selected planes

We propose three simple approaches to optimize the generated proxies (illustrated in Figure 4.13): (i) Plane refinement: The discretization of plane-space can cause selected planes to minimally deviate from the features they capture. A simple bounded linear search through plane-space in the vicinity of a selected plane to maximize section area or perimeter improves the visual quality of the planar section. In our implementation, this optimization is always performed. (ii) Segment coverage: Our algorithm does not guarantee that all segments get covered. A user may optionally enable algorithmic plane selection to continue beyond the threshold only for planes that section uncovered segments. (iii) Adaptive threshold: The user may wish to include a desirable plane below the default threshold that the algorithm did not include, or have a specific number of planes in mind. Tuning the threshold allows users to refine the planar proxies created by our algorithm to select more (or fewer) planes.

4.5.3 Performance

We created and released a publicly available OpenGL/Qt demo application (the code is also publicly available). The running time of our algorithm is dominated by the number of features detected for a given model as populating plane-space for each feature is expensive and scales linearly (typically 50-200 features were detected on models from the PSB). Our algorithm typically takes 5-10 seconds for most models on a 2.8GHz AMD Phenom II processor.

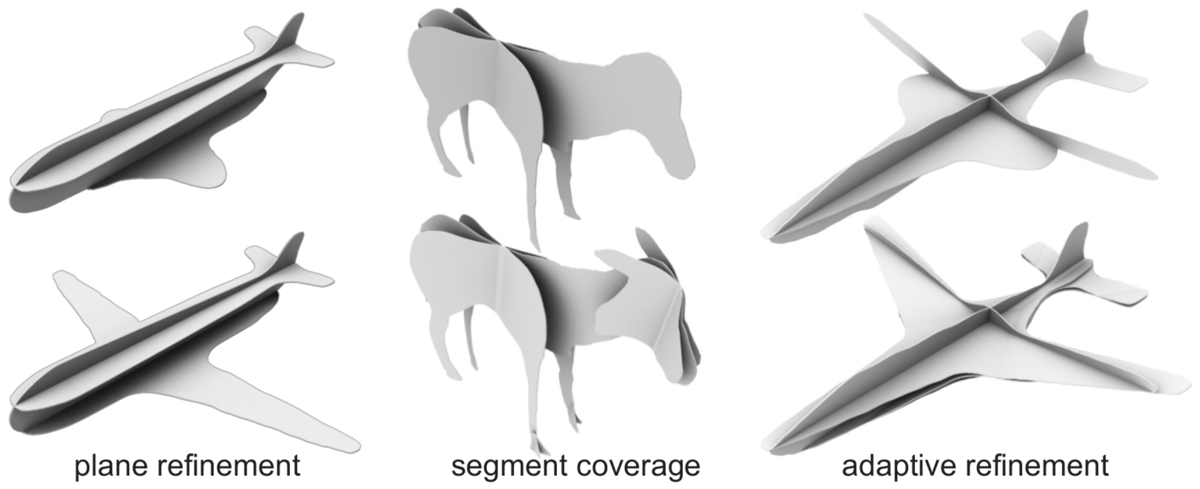


Figure 4.13: Planar sections (top) can be optionally optimized to improve abstraction quality (bottom). From left to right: locally maximizing section area, using additional planes for uncovered segments, reducing weight threshold to choose additional planes.

4.5.4 Limitations

While our approach produces reasonable results and is extendable, it has some fundamental limitations, which we illustrate in Figure 4.14. Our algorithm is only as good as the quality of features it is able to extract. Excessive and noisy features (the armadillo), or the absence of features (e.g., bust) can result in too many or too few planes being selected.

Articulated figures can be problematic since features do not line up well along planes, and produce disjoint abstractions (e.g., right arm of the running woman). In Chapter 5, we create a system to create planar sections interactively from an optional input surface, where planar sections can be traced along individual segments of the surface. This allows for the creation of suitable representations for even highly-articulated surfaces. Later, in Chapter 7, we propose the creation of articulating planar section representations as an application. We detail how the planar sections and joints can be automatically formed given an appropriate segmentation of the input surface.

Excessively curved structures like the chair with a curved backrest are not directly suitable for representation with a minimal set of planar sections. In Chapter 5, we will propose a solution to handle these "non-ideal" surfaces. The approach involves a local deformation of the surface that yields a planar section whose contour is very much like that of a silhouette boundary contour.

4.6 Conclusion

Inspired by the consistency of abstractions produced by human subjects during our initial user study, we developed a computational approach to create geometric feature-guided abstractions of man-made shapes using only a handful of planar sections. The formulation is flexible and

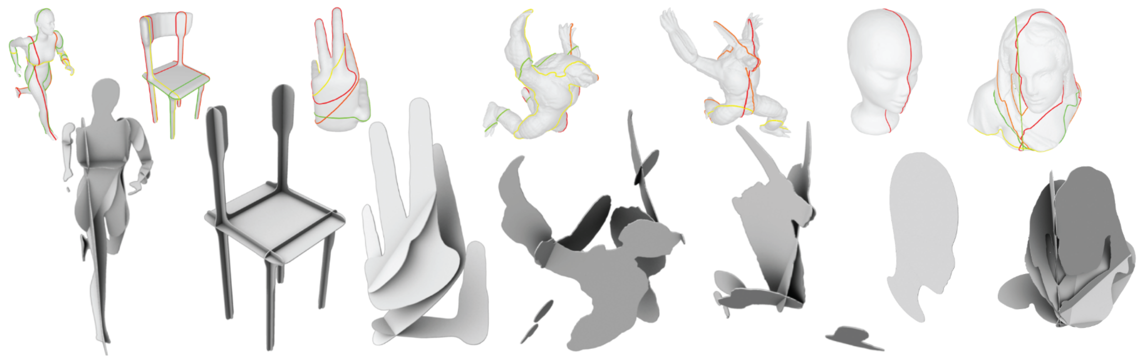


Figure 4.14: Limitations: Imperfect results for articulated poses, models with highly curved characteristics components, or models that are overly smooth lacking clear geometric features.

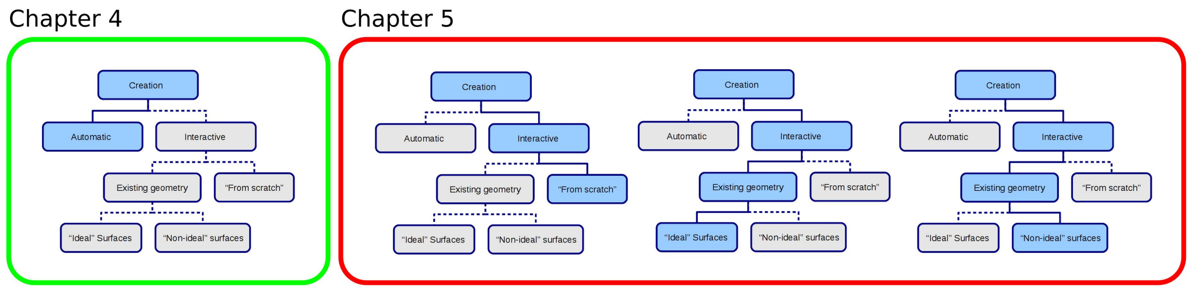


Figure 4.15: Four possible approaches to creating planar section representations, which are covered in Chapters 4 and 5.

directly incorporates a variety of shape features. In the absence of any suitable computational model of human perception, we learned the relative preference weights for various features using our user study data.

We tested our algorithm on a variety of input models, and presented robustness results with respect to noise and varying mesh resolution. Finally, in the course of a conducted survey, we observed that both recognition quality and efficiency of abstracted models are comparable to those of the source models.

In Figure 4.15, we show that the material in this chapter covers one possible path for creating planar section representations. In Chapter 5, we cover three cases in the interactive setting: creation of planar representations (i) from scratch, (ii) from ideal surfaces, and (iii) from non-ideal surfaces.

Chapter 5

Interactive Creation of Planar Section Representations

With the increasing popularity of 3D modelling software and digital manufacturing devices aimed at hobbyists, there has been a flurry of recent research that attempts to amalgamate the stages of design and manufacturing. While the creation of manufacturable planar sections by slicing existing 3D objects has received much attention, fluid interfaces for sketching these abstractions from scratch are relatively unexplored. Interfaces for creating such abstractions have application outside their direct outcome of 3D planar section sculptures, as these planar structures can form scaffolds for digital modelling [124] and armatures for instrumented animatronics [105, 169]. We address this space of blank screen modelling and present an interface tailored to the sketching of 3D planar section structures (see Figures 5.1 and 5.2).

We present a drawing system to author 3D planar section assemblies from scratch. To gain insight into how people imagine and draw planar sections we first authored a free-form drawing program where users manually manipulate a 3D view and 3D planes on which their sketches are projected (Section 5.1). We conclude that planar sections are best drawn in situ, but without foreshortening and orthogonal to intersecting planar sections. We thus present a novel 3D planar section drawing workflow where with a single fluid stroke a user specifies both a 3D plane in relation to existing planes, and draws a planar contour without foreshortening (Section 5.2). We also present a set of procedural operations specifically tailored to planar section modeling (Section 5.3).

We formulate a simplified physics model for planar section assemblies, enabling real-time visual feedback of model stress under gravity and other interactively-specified forces, and validate predictions with real-world fabrications (Section 5.4). Aimed at 3D fabrication, our system also provides real-time feedback to verify the model is stable, connected and can be assembled (Section 5.5). As evaluation, we report the results of 12 users who were all able to use the system effectively with minimal instruction to create a range of interesting examples (Section 5.6).

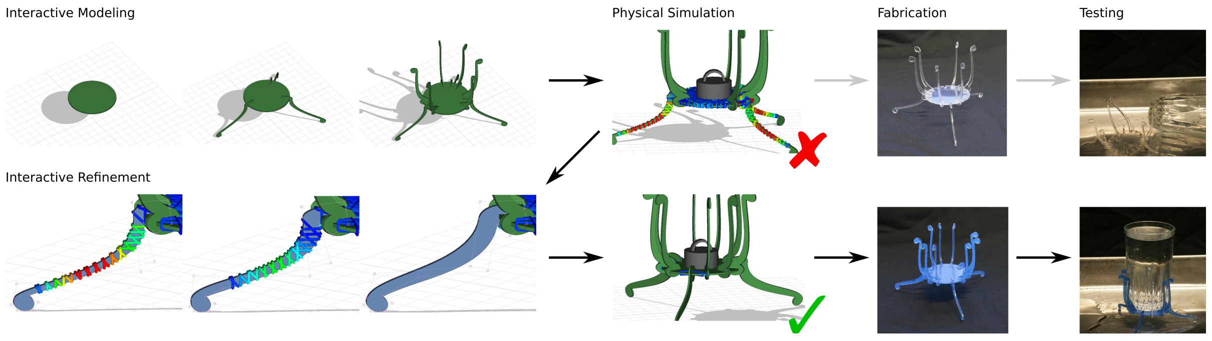


Figure 5.1: We model 3D planar sections interactively using a fluid single-stroke workflow. We also present a vocabulary of procedural planar section operations, to provide shape regularity and aid fabrication. In addition to geometric tests for assembly, connectedness and stability, our system simulates gravity and other interactively applied physical forces acting on the model, and visualizes in real-time lines of stress along planar sections where fracture is likely. Designers interactively refine the model using this physical feedback. When satisfied with a robust design, our system allows users to export 2D cutting paths, from which sections can be physically fabricated and assembled.

5.1 Pilot Study

We performed a pilot study to understand how users imagine and draw planar sections in 3D. We asked 6 users with varying degrees of artistic ability and 3D modelling experience to create planar section representations of both imagined objects and a set of reference 3D models (see Figure 5.3). For this task, we developed an interactive system allowing artists to freely manipulate a planar canvas to sketch planar 3D curves upon. Throughout their interaction, our system logged their strokes, plane manipulation, camera motion and other geometric measurements.

5.1.1 Initial prototype

We experimented with an initial design prototype, where from a static single view a user would sketch a new planar curve onto an existing planar curve network (see Figure 5.4), and our prototype would attempt to automatically compute the intended planar canvas and true intersection points of the new planar curve. To do this, our algorithm exhaustively evaluated combinations: it selects subsets of the *apparent* intersections (points where existing curves appear to intersect the new curve in the 2D view) testing them as *true* intersections (true curve-curve intersections in the 3D space). Three or more co-planar true intersections define the planar canvas for the curve. To evaluate each subset of true intersections, our prototype computed the sum of terms that captured the principles learned from the user study: that intersecting curves met orthogonally, and that the planar canvas should be perpendicular or parallel to existing planar curves. The true intersections (and thus the planar canvas) were selected such that the sum was minimized.

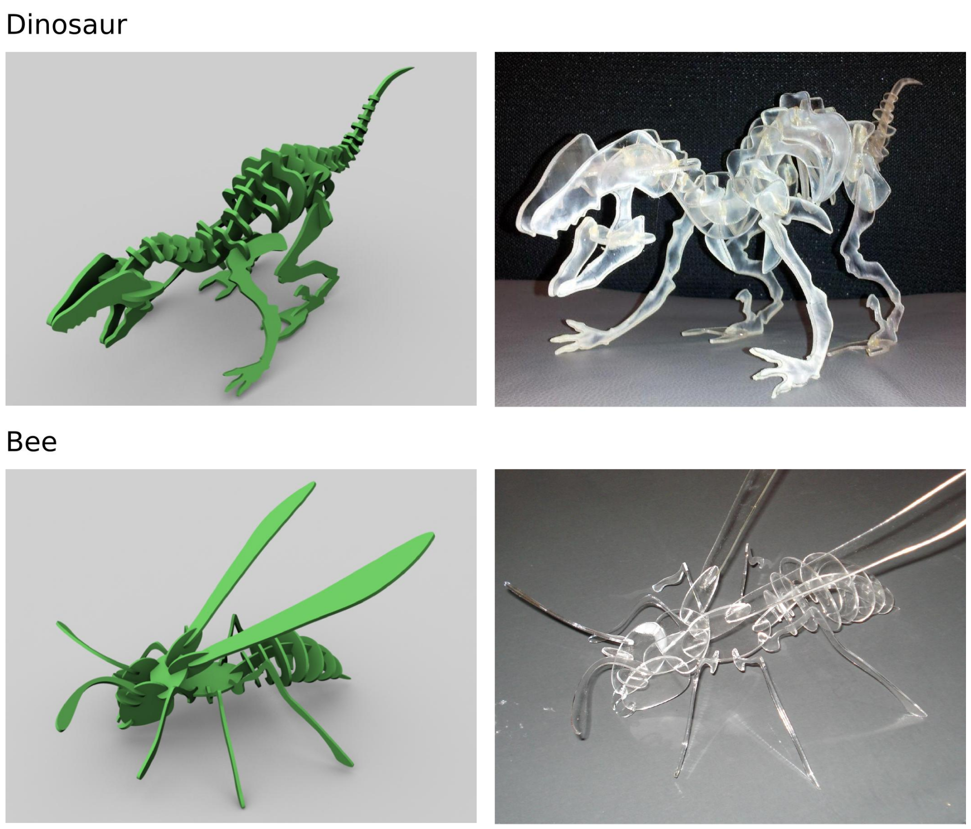


Figure 5.2: Two planar section representations created by artists using our system, fabricated using a laser cutter and acrylic sheets.

Though the approach at first seemed reasonable, use of this initial prototype presented many difficulties, which we broadly categorize:

Dense curve networks

For a dense curve network, the algorithm's choice becomes poorer in terms of user expectation, due to an increasing number of apparent intersections obfuscating the true intersections. Though true intersections could be specified manually to correct a poor choice, requiring the user to perform this task frequently makes the interaction tedious. Also, to test only all 3-combinations of apparent intersections (the minimum to define a planar canvas), our combinatorial algorithm has $O(n^3)$ time complexity, which does not scale well for complicated sketches. In addition, as planar section contours must all be visible so the user can draw a curve to intersect true intersections precisely, this also leads to confusion in terms of 3D perception of the curve network.

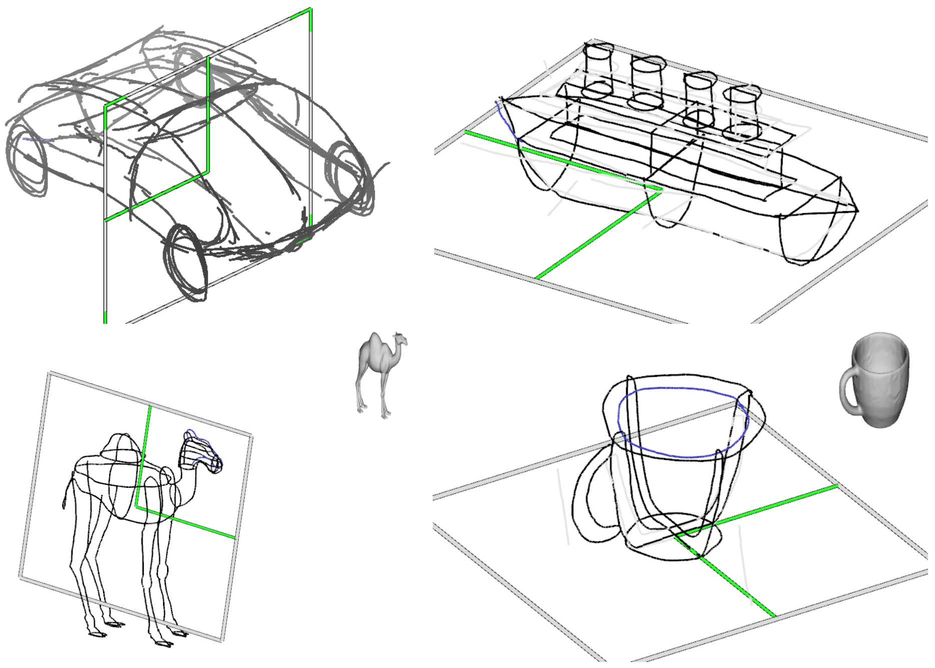


Figure 5.3: Planar curve representations created by artists during our pilot study. (Top row) Artist-created freeform objects. (Bottom row) Artist-created reference objects (objects shown inset at top right).

Non-frontoparallel views and foreshortening

With this system, beyond drawing the first plane, ideally the user could rotate to a stationary view from which multiple planar curves could be drawn with varying orientations, in order to draw effectively. However, effects of foreshortening make this task challenging since the curve that is produced from the planar canvas, even with the true intersections being correct, is often not what the user intended. These results are consistent with those presented by Schmidt et al. [123], where as the planar canvas becomes less frontoparallel the greater the deviation of the curve from user intention. We experimented with both perspective and orthogonal projection (in perspective the effect was especially pronounced), but as foreshortening applies in both instances, sketching accurately-constructed 3D curves was found difficult. This difficulty is compounded by planar canvas selection: the algorithm does not always choose well, and the sketched true intersections may not actually be co-planar.

Constraints on the planar canvas

Invariably, the user sketch will contain noise, yet our planar canvas selection process relies on a derivative measurement (comparison of the tangent directions at intersection points). This is not ideal, as the noise is especially problematic for its effect on derivative measurements. Often, even the true intersection points themselves cannot be drawn accurately (for example,

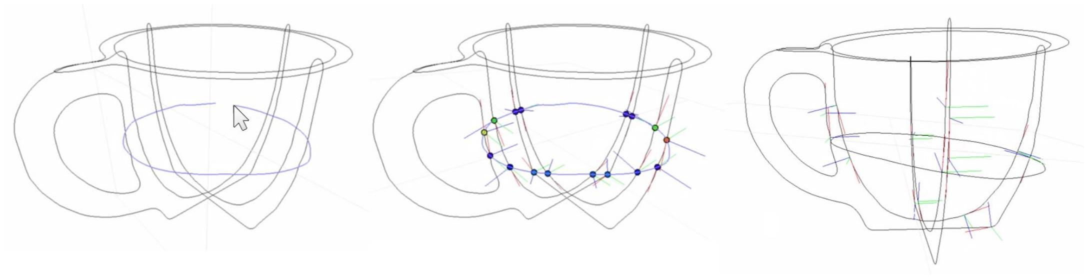


Figure 5.4: A design prototype where one could sketch a planar curve onto an existing planar curve network. (Left) Sketching the planar curve. (Centre) From all *apparent* intersections (shown blue), the system selects a subset of them to be *true* intersections (shown non-blue), which define a 3D planar canvas to project the sketched curve onto. (Right) An alternate view of the sketched curve, projected onto the chosen 3D planar canvas.

it is difficult to sketch a curve intersecting 4 co-planar points without a guide). Further, the constraint of only choosing candidate planes among apparent intersections can be highly restrictive, as is the constraint that the canvas plane exhibit global orthogonality or parallelism. These restrictions impede the creative process, especially during the earlier stages of design.

5.1.2 Principles learned

Plane orthogonality

We performed an analysis of the pilot study data, and examined the distribution of angular differences between each curve's plane normal with each of the other planes to find that canvas planes were predominantly either parallel or perpendicular to one another (see Figure 5.5 for the distribution). Intersecting planes almost always met orthogonally, thus one of our design principles is to enforce plane orthogonality.

Orthogonality is also enforced for manufacturing-related reasons. Laser cutters only make cuts in directions orthogonal to the plane normals, and for materials with substantial thickness, planes that do not meet up orthogonally have a *loose* connection due to a widened slit to accommodate the angle of the plane, and can freely pivot on the slit axis. Multi-axis CNC milling machines can cut in non-orthogonal directions creating tighter connections, but such devices are less accessible.

Frontoparallel views

Schmidt et al. [123] present a study where participants sketch ideal curves onto surfaces embedded in 3D. The results provide evidence that the angle between surface normal and view direction plays a significant role in sketching error, even for experts. In our own analysis, we found that sketching planar contours onto canvas planes that were not viewed frontoparallel consistently resulted in unpredictable contours that deviated significantly from one's intention, and that artists tended to sketch curves from a view at or near frontoparallel. Thus, another

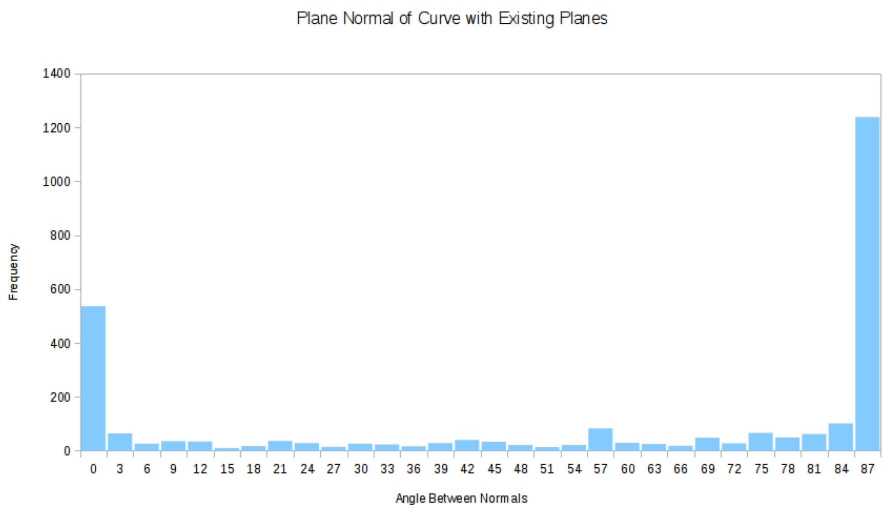


Figure 5.5: Normal angle distribution of each curve with other planes in our pilot study. Note the two spikes at 0 and 90 degrees, indicating that planes are mostly either parallel or perpendicular to one another.

principle guiding the design of our system is that frontoparallel views should be used during curve sketching.

Single-view 3D interface

We also experimented with a multi-view system such as [65] where users worked in disparate views to manipulate planes and contours. Artists overwhelmingly preferred a single 3D view for multiple reasons: a large single-view maximizes screen space allowing more detail to be seen; attention is not shifted between views; cursor motion is not shifted between views; and a single-view provides greater context relating the position of existing curves and planes with a new curve. Thus we create a single-view 3D design interface.

5.2 Creating Planar Sections

Our system uses a single-view and pen-based interactions. Users are able to create and save planar section representations and export them as OBJ (suitable for rendering) or SVG (suitable for fabrication) formats.

5.2.1 From scratch

Central to our approach is an interaction used to define a new planar section that consists of a single stroke and is illustrated in Figure 5.6. In the following we detail each step, and our treatment to arrive at fabricable models.

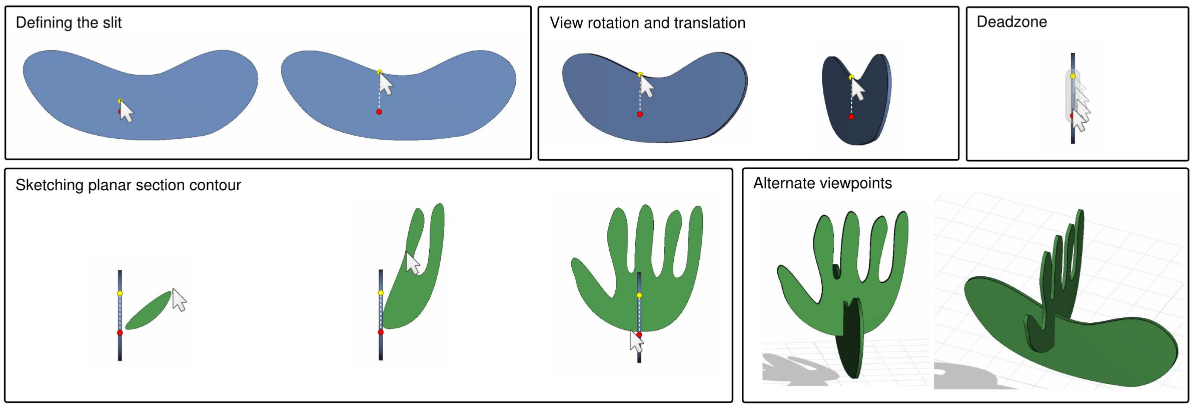

Defining the slit
View rotation and translation
Deadzone
Sketching planar section contour
Alternate viewpoints

Figure 5.6: The core interaction in our system: a single stroke to create planar sections. (Top row) The user defines the slit for the new planar section and the system rotates to a frontoparallel viewpoint, while keeping the slit endpoint beneath the cursor. (Bottom row) The user moves the cursor outside of the dead zone (shown top right) and sketches the planar section contour. (Bottom right) Alternate views of the newly-created planar section.

Defining the slit

The stroke first defines the *slit*, where two orthogonal planar sections join. The stroke starts on the interior of an existing planar section, defining one endpoint for the slit axis. The slit is defined when the cursor exits the planar section interior. A new canvas plane is defined with a normal orthogonal to the slit axis and existing planar section normal, and intersects the slit endpoints.

View rotation and translation

The system rotates the view so the new canvas plane is viewed frontoparallel. Importantly, rotation is constrained so the slit endpoint remains fixed beneath the cursor. This is ideal: first, the rotation is predictable – an unknown or arbitrary pivot point is not used; second, cursor movement controls view translation and is used to ensure sufficient space is available to draw the next planar section contour.

Dead zone

Artists felt constrained being forced to start the planar section contour at the slit endpoint. The *dead zone*, a capsule shaped region in the viewport around the slit, addresses this issue. The dead zone prevents the planar section contour from being defined until the cursor exits the region, which has a fixed radius relative to the dimensions of the viewport. A planar section contour can thus begin near either slit endpoint, or anywhere in between.

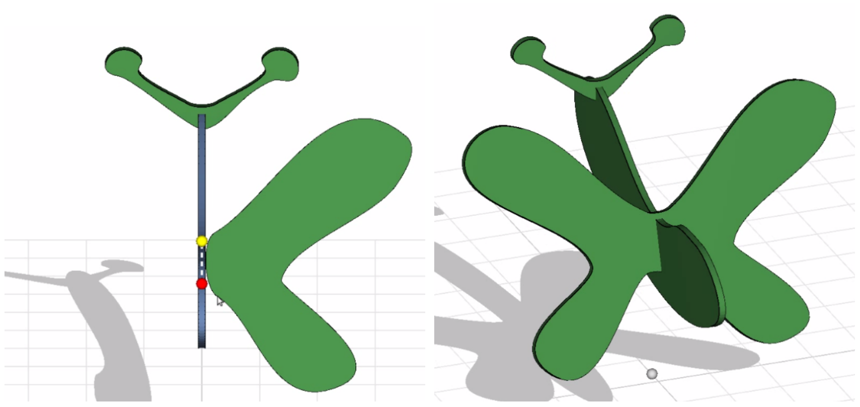


Figure 5.7: (Left) Planar section contour drawn principally on one side of the slit axis. (Right) Stroke is mirrored across the slit axis and smoothly joined.

Sketching the planar section contour

As the user sketches, our system fits a cubic Bézier spline, with a final Bézier segment to close the curve. When the stroke is completed, the planar section is created, completing the interaction. When there are no existing planar sections to start with, the contour is defined on a frontoparallel viewed plane that intersects the origin.

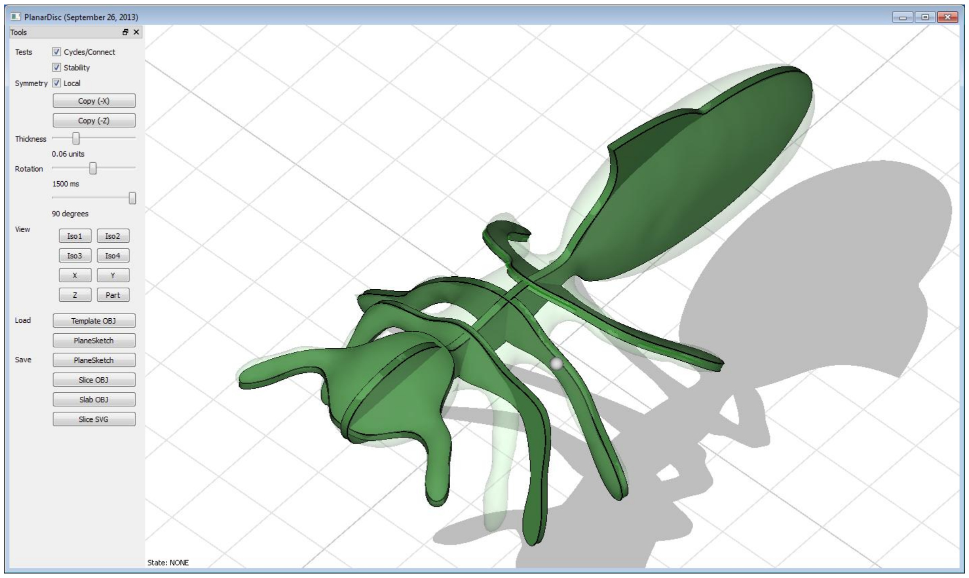
A stroke principally on one side of the slit axis indicates the user's intent to create a symmetric planar section. If the ratio of maximal distances of the stroke on either side of the axis is sufficiently high, the stroke is mirrored across the axis and smoothly joined (see Figure 5.7). We use the ratio of 6.0 in our implementation.

5.2.2 From ideal surfaces

The interaction to create planar sections from an existing surface is similar to the case from scratch, with some notable differences.

The surface

First, the user of the application loads a surface, which we assume to be watertight (though given a plane-surface intersection that results in one or more open contours, one could link them together using cubic Bézier segments, for example, to create a smoothly-joined closed contour). The surface is scaled so the diameter of its bounding box is a specific length (in our implementation, we use 10 units). The surface is translated in X and Z so that its centre is at the origin. Along the Y direction, our system translates the surface so that it rests upon the ground plane (i.e., the surface vertex with minimum Y value satisfies $Y = 0$). Note that one need not be concerned that the surface has proper contact points on the ground plane, as the created planar section contours are free to deviate from surface contours either during creation or can be adjusted at any point afterward to ensure proper contact with the ground plane.



Tests ✓ Cycles/Connect ✓ Stability
Symmetry ✓ Local
Thickness 0.06 units
Rotation 1500 ms
90 degrees
State: NONE

Figure 5.8: A screen shot taken of our interactive system, where an existing surface representing an ant was used to create a planar section representation. Note that in some regions that planar section contours deviate significantly from the surface (such as the front left leg, or protruding fin at the rear), as the artist has freedom to deviate from the surface template.

The surface is rendered transparently, as it serves as a guide for planar section creation (in our implementation, we use $\alpha = 0.25$ with back-face culling enabled and no writes to the depth buffer). This ensures that regions of planar sections created within the surface are clearly visible (surface visualization shown in Figure 5.8).

The canvas plane

As when drawing the first planar section contour from scratch, our system chooses a frontoparallel canvas plane that passes through the origin. For any initial view of the surface the user selects, planar contours resulting from an intersection between the canvas plane and surface are rendered in dark green; we call these *guide curves*.

When one or more planar sections exist, as with from-scratch creation the user must first define the slit for a new intersecting planar section. However with an existing surface, a key difference is that the guide curves are constantly updated as the slit (and thus canvas plane) are defined. The interactive visualization of the guide curves informs the user, allowing them to adjust the slit direction until they find the guide curve satisfactory. A frontoparallel view of the planar section before defining the slit results in the guide curve appearing as a line segment, which is often sufficient; however, one may wish to select a non-frontoparallel view in order to view the guide curve profile explicitly along two different directions.

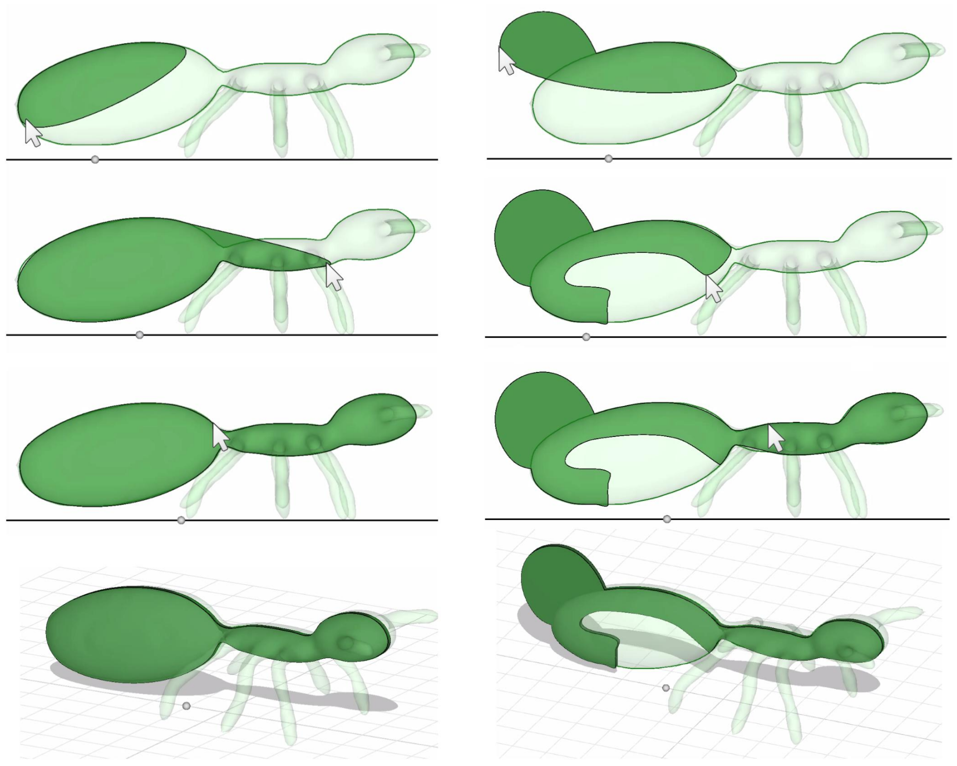


Figure 5.9: Snapping to a guide curve. (Left column) User sketches roughly along a guide curve to define a planar section contour that conforms precisely to a planar surface contour. (Right column) User's sketched curve deviates significantly from the guide curve. Our interactive system provides the flexibility to create planar section representations to precisely match or significantly deviate from the shape of an existing surface.

Creating planar section contours with guide curves

When a canvas plane has been defined for the user to sketch a planar section contour along, the guild curves are visible. For each point of the user's sketched stroke, if the point is within a distance δ of any guide curve, the point is translated (or *snapped*) to the nearest point on that guide curve. In our implementation, a fixed threshold of $\delta = 0.5$ is used by default. As an alternative, one might employ the peeling interface for curves presented by Igarashi et al. [64] in order to dynamically define this threshold, based on the distance from the curve the user deviates. The automatic snapping behaviour allows the user to create planar section contours that can either follow guide curves precisely or allows for deviation from them (see Figure 5.9).

There is an advantage to applying our automatic symmetry approach when working with an existing surface in the case where the surface has articulation on only one side of the slit axis. We can sketch the planar section contour for the non-articulated region, and a symmetric planar section contour will be created as if it came from a surface that was not articulated on both sides. This "automatic symmetrization" of a planar section can be useful for creating a planar section representation that is a symmetric variation of an asymmetric input surface (see

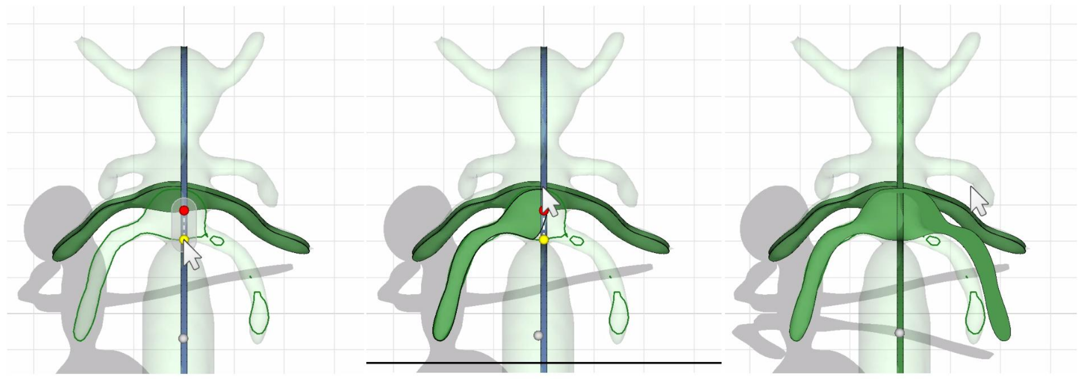


Figure 5.10: (Left) An articulated rear right leg on the surface of the ant results in a poor guide curve on the right side. (Centre) User sketches along the guide curve on the left side. (Right) The sketched stroke is mirrored across the axis, producing a symmetric planar section.

Figure 5.10 for an example).

5.2.3 From non-ideal surfaces

The interior volume bounded by a surface may not conform well to the shape of a plane. In such instances, rather than relying on a planar contour, we explored using the surface contour defined by points whose normals are orthogonal to the plane normal (see Figure 5.11). Such contours define the boundary of the *silhouette* when the surface viewed orthographically with a view direction parallel to the plane's normal. Once the non-planar silhouette contour is created, it can be projected onto the selected plane, flattening it and allowing the creation of a planar section. (Note that the silhouette contour is not well-defined for flat surface regions that are perpendicular to the plane's normal.)

This silhouette-based approach captures all segments of the surface. This is both advantageous and disadvantageous. The advantage is that no segment of the surface is missed. But the disadvantage is the reduced design freedom – one cannot selectively choose which segments to include and which not to, and the position of the plane is of no significance when creating the section. We therefore would like to introduce a sense of *locality*, such that only the regions of the surface near to the selected plane are used. This would make the selected plane's position meaningful, and provide greater design freedom.

Our solution is the *magnetic* cut: we deform the surface to pass through the plane when points come within a specified distance. The adjustable distance threshold defines a *field of influence*. Specifically, for surface points that come within the field, we translate the vertices along the plane's normal direction based on the surface normal direction at that point (see Figure 5.12). Surface points whose normal is parallel to the plane's normal will be translated to the boundary of the field; surface points with a normal orthogonal to the plane's normal will be translated *onto* the plane. In practice, this deformation is analogous to a magnetic force that pulls the surface across the plane wherever it enters the field of influence. Thus, the magnetic

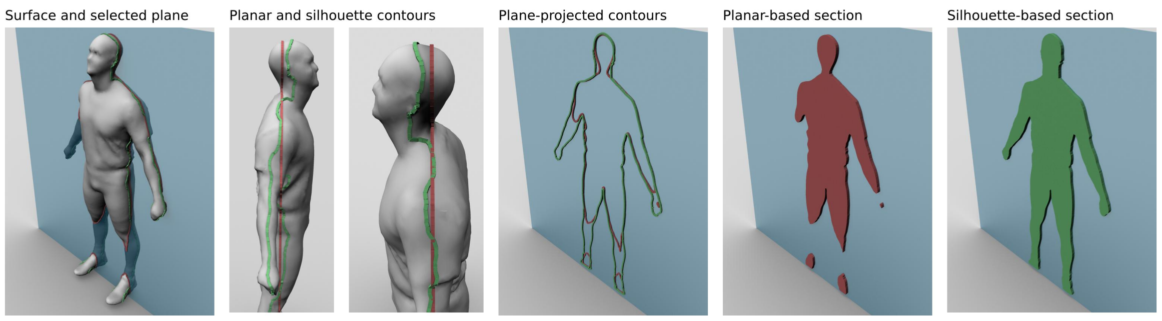


Figure 5.11: (Left and middle-left) Given the input surface and a selected plane, two contours are formed: the traditional planar section contour (shown red) and the non-planar silhouette contour (shown green). The silhouette contour is defined by all points on the surface where the surface normal and plane normal are orthogonal. (Middle) The silhouette contour is made planar by projecting it orthogonally onto the plane. (Middle-right and right) The traditional plane-based approach misses segments of the surface (the right arm, left hand, and calves) while the silhouette-based approach captures everything.

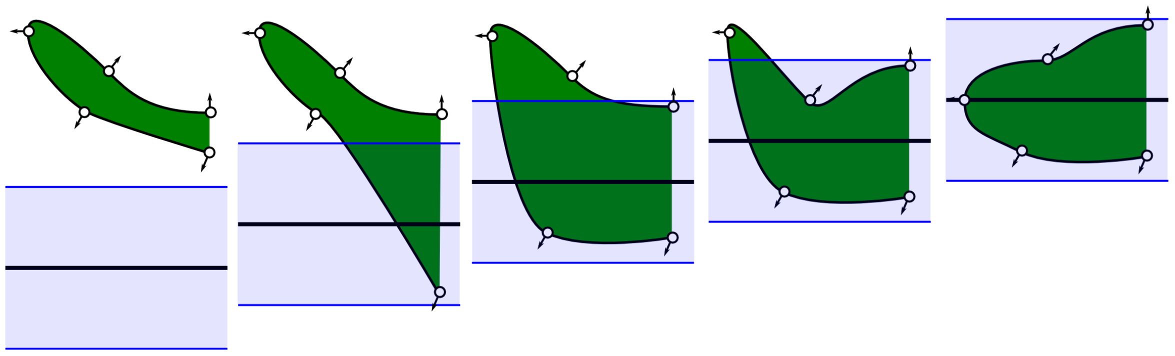


Figure 5.12: Visualization of how the magnetic cut deforms geometry. The plane is shown as a black line, and the field of influence is shown in blue. As the plane is translated upward, vertices of the surface (green) enter the field of influence, and are translated along the plane's normal direction to a new position based on their surface normal.

cut is an approach to creating a planar section that is geometrically equivalent to a silhouette, but also incorporates locality – the deformation is applied only where the surface falls within the field of influence.

We now define the transformation explicitly. Consider a surface point P with surface normal N_S within the bounds of the magnetic field. The bounds of the magnetic field reach a distance r from the plane, the plane is defined as introduced in Section 3.1 by a normal N and plane point P_0. The updated vertex position P' will be:

$$P' = P + N(N \cdot P_0 - N \cdot P) + N(N_S \cdot N)r. \tag{5.1}$$

The first two terms on the right hand side project P onto the plane along the normal direction N. The last term translates P within the field according to the dot product of the

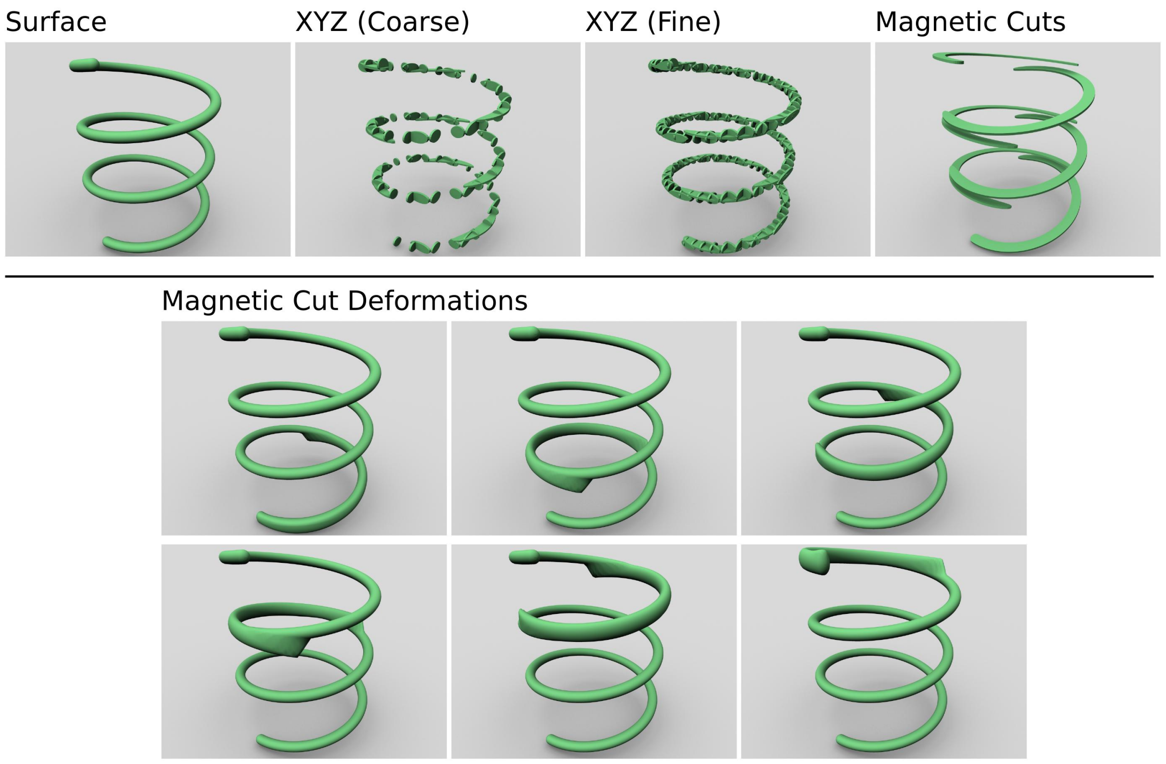


Figure 5.13: A surface representing a thin helix, whose non-zero curvature and torsion make planar section representations problematic. (Top) We show some planar section approaches such as using regularly-spaced XYZ sections require many cuts for an adequate representation, however using magnetic cuts (far right) only 6 planar sections were needed. (Bottom) We show the 6 surface deformations used for each of the planar sections of the magnetic cut representation. For the first two magnetic cuts, a field of influence large enough to contain the surface was used.

plane normal and surface normal.

Once the deformation has been made, a planar section is formed by the intersection of the plane and deformed surface. The deformation of the surface is ideal in the sense that the planar section contour will be locally maximal in terms of its surface area.

However, some surfaces are not amenable to planar section representations – for instance, a thin helix-shaped surface, whose medial axis has both non-zero curvature and torsion as shown in Figure 5.13. For such surfaces, a satisfactory representation consisting of a small number of purely planar primitives may not be possible – the “purely planar” doctrine fails. The goal of the magnetic cuts approach is to produce a representation that is reasonable, but there are obvious problems: first, the surface deformations involved may lead to planar sections that are geometrically distant from the original surface (see Figure 5.14). Secondly, there is a problem of connectedness between planar sections in the representation (one would have to introduce extra planar sections to link the sections manually, or rely on an algorithmic solution – a possibility for future work).

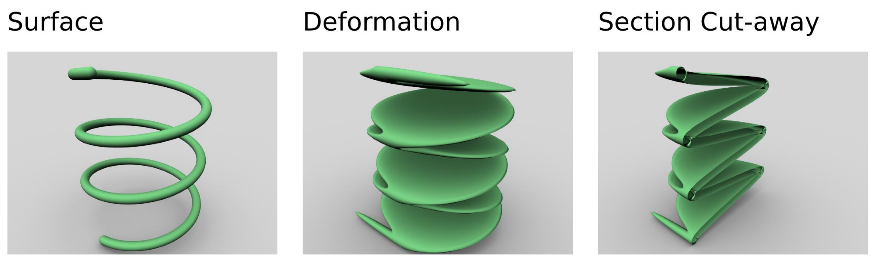


Figure 5.14: (Left) The surface, (middle) deformation for a magnetic cut, (right) using the magnetic cut's plane as a geometric clipping plane, in order to better observe the planar contour resulting from the surface deformation.

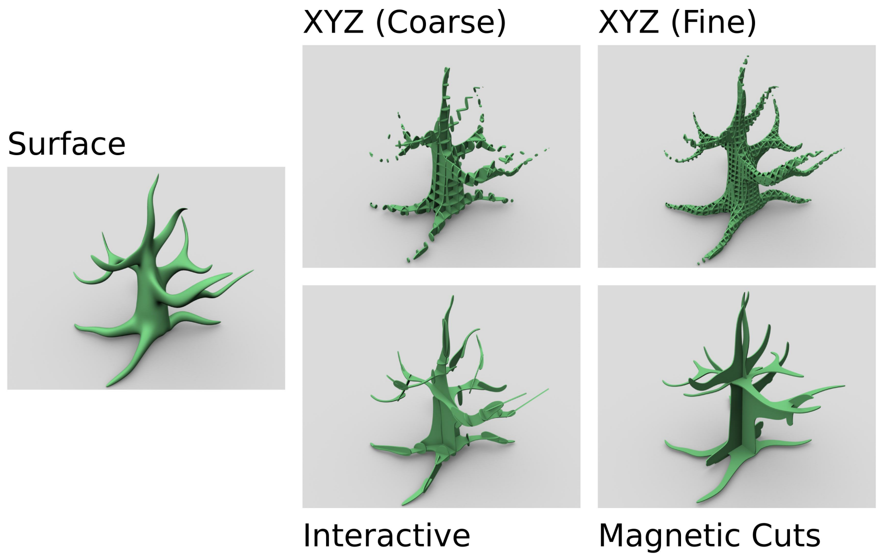


Figure 5.15: A surface with thin, curving branches is problematic for planar section representation. We show the surface, regularly-spaced XYZ sections, interactively-authored planar sections, and magnetic cuts.

As another example, we modelled a surface to represent an aging tree with numerous thin branches that bend in various directions, purposely made to be non-planar. We show a number of methods to create planar section representations for this surface in Figure 5.15. Many sections are required for the XYZ style to capture the surface, as a coarse-grained spacing does not adequately represent the branches as shown. Interactive creation of the planar sections along the highly-articulated branches is possible, but the process is somewhat tedious – the example shown took 15 minutes to complete. Using magnetic cuts, only 4 planar sections were required to represent all segments of the surface, and the example was created in less than a minute.

Note that a surface deformation is applied independently for each plane, as shown in Figure 5.16. The deformation is not a continuous one, and the planar sections that are created often do not fall within the interior volume defined by the surface as they do traditionally.

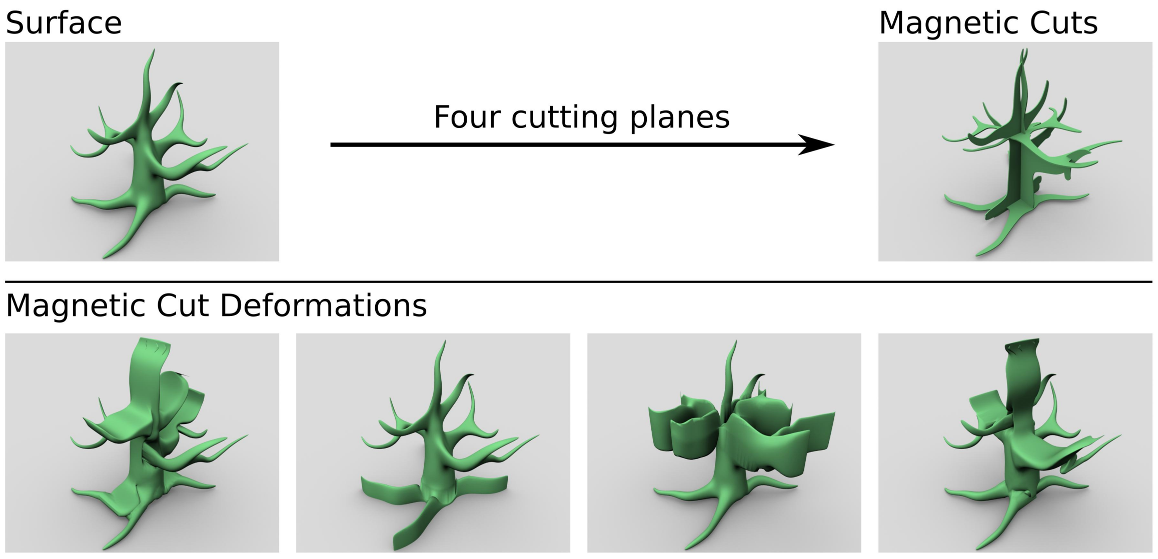


Figure 5.16: (Top left) The surface and (top right) planar section representation consisting of four magnetic cuts of the surface. (Bottom row) Each of the four deformations of the surface required to obtain the contours for each planar section.

Because of this, planar section representations composed of magnetic cuts can be thought of as stylized representations of the original surface.

5.3 Procedural Modelling

We are inspired by procedural modelling research [53, 84, 102], in defining a vocabulary of modeling operations that provide support for various inter-plane and contour shape regularities observed in artistic planar section structures.

5.3.1 Affine operations

We define a family of shape operations that replicate a *duplicated* planar section along a *root* planar section. The linear operation makes a linear arrangement of duplicates, along one side of the root planar section, preserving the normal direction of the duplicated planar section at a fixed offsets. The sections are also translated along the intersection axis, to account for the spatial variation along the root contour; the translation ensures that the line of intersection between duplicate and root is of equal length for all duplicates (see Figure 5.17).

To create variation for duplicated planar sections along the contour of the root planar section, our approach computes a *thickness measurement* with the root. The thickness at a position along the root is performed by casting a ray on the root plane that intersects the contour. The distance between the two points of intersection that surround the line of intersection between duplicated and root planar sections is used as the thickness value. The thickness value is used to scale the contour of the duplicated planar sections, making their size larger or smaller. When

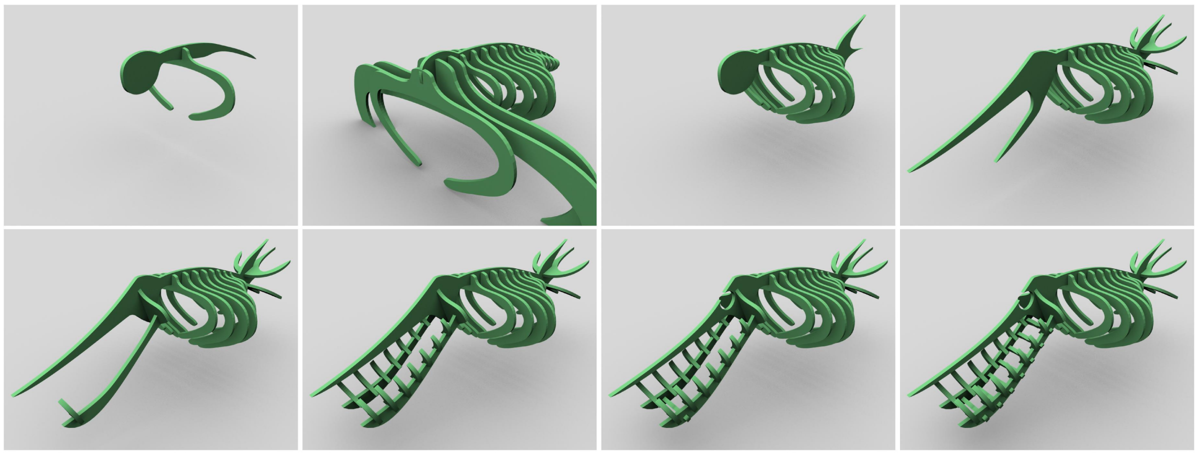


Figure 5.17: Linear operations are applied to create the rib cage, teeth and jawbone for a planar section representation of a fish skeleton. Duplicated planar section contours are scaled in both directions on the plane based on the cross-sectional width of the base planar section, creating variation along the rib cage and teeth. A few large planar sections not part of the rib cage are deleted manually (top centre images).

scaled, duplicated planar section contours are translated so that their line of intersection with the root is in proportion with the duplicated and root planar sections (sections are *smaller* as the root's planar section gets *thinner*, and *larger* when the root section gets *thicker*) (see Figures 5.17 and 5.18).

Sections can also be duplicated around a root section by rotating the sections using the tangent direction along the root. To keep the orientation of copies consistent for the duplicates, the relative angle θ between the duplicate's normal and root's boundary tangent is applied to all duplicates (see Figure 5.19).

5.3.2 Blend operation

The blend operation creates blended copies of two planar sections P_2 and P_3, along a third base planar section P_1. The details of this operation are composed of two parts: first, the blending of each 2D planar section boundary; second, the 3D placement of each blended planar section. We show some examples using the blend operation in Figure 5.20.

To first create the blended 2D boundary curve, a point-to-point correspondence between the control points of the two boundary curves of P_2 and P_3 is formed (we assume that P_2 and P_3 each have only one boundary curve). If the boundary curves of P_2 and P_3 do not have the same number of control points, the boundary curve with fewer points is subdivided into two halves at its longest segment. This process, of computing the longest segment and dividing into two, is repeated until the two curves have an equal number of control points. Then, to find the correspondence, the sum of ordered 2D distances between points of the curves are computed, trying each point offset and both forward and backward directions.

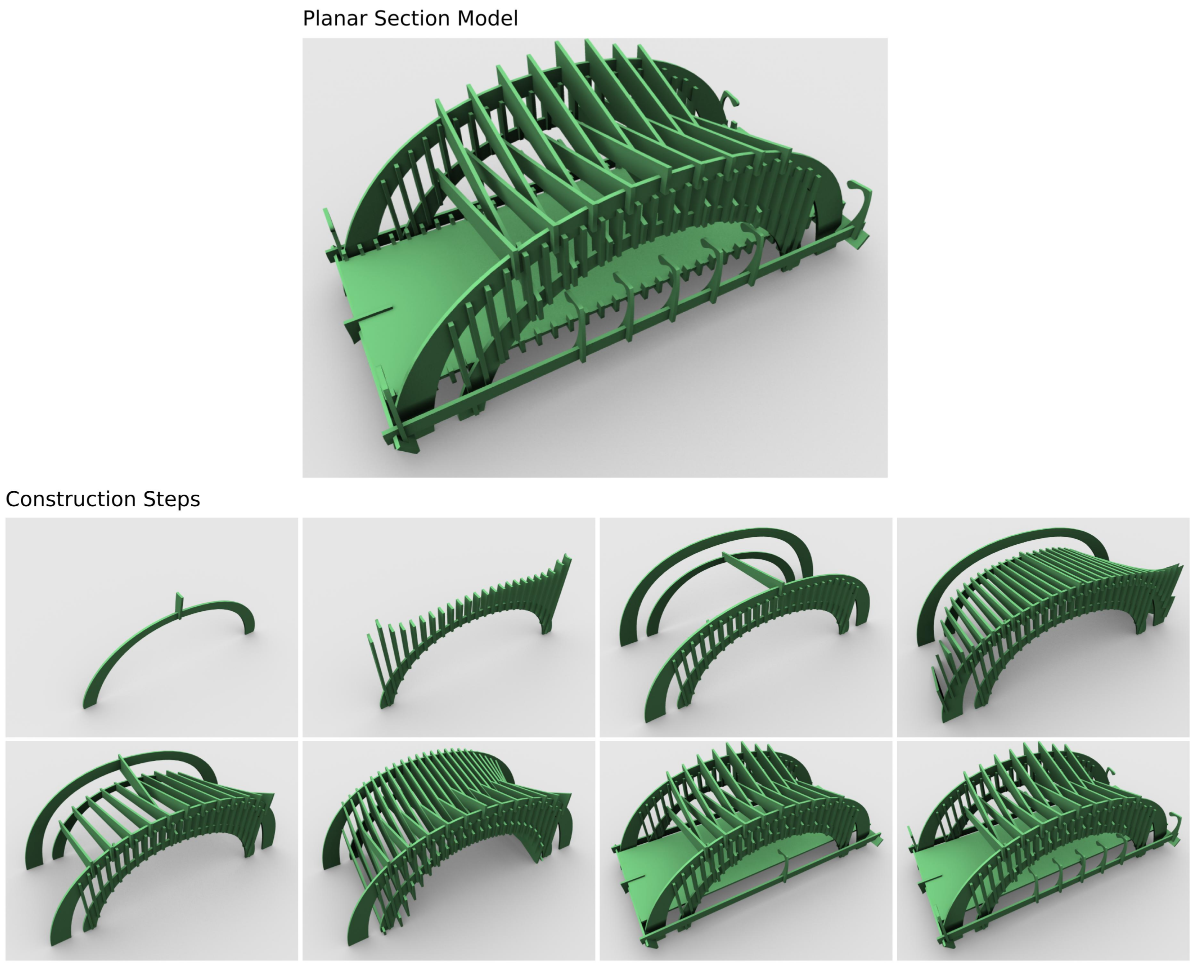


Figure 5.18: (Top) Planar section representation of a bridge, created using linear operations. (Bottom row) The steps required in the construction, which shows the numerous linear operations required: one to connect coplanar arches, two to connect lower to upper arches across the bridge, one to create lampposts along the bridge, and one to create a barrier at the sides of the road. For these linear operations, duplicated planar section contours are only scaled vertically, along the intersection axis.

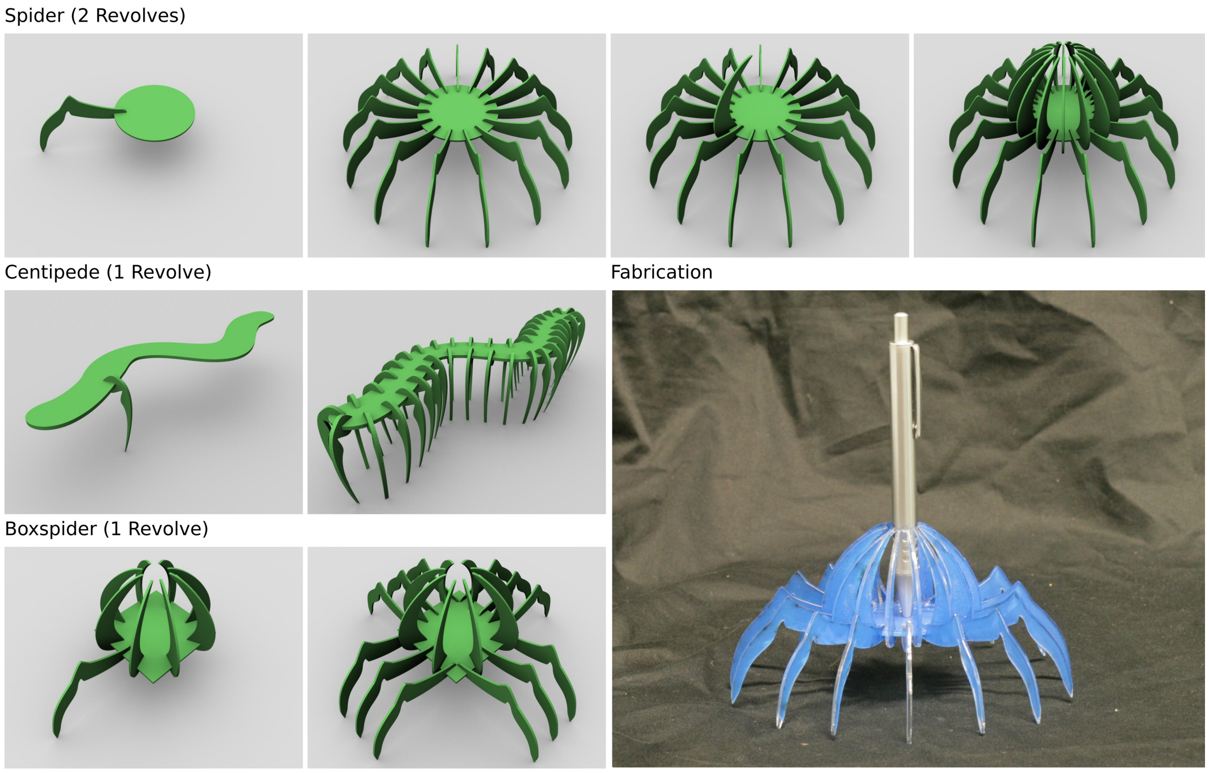


Figure 5.19: A number of insect-like examples, created by applying the revolve operation one or more times. (Bottom right) A fabricated spider holds a pen.

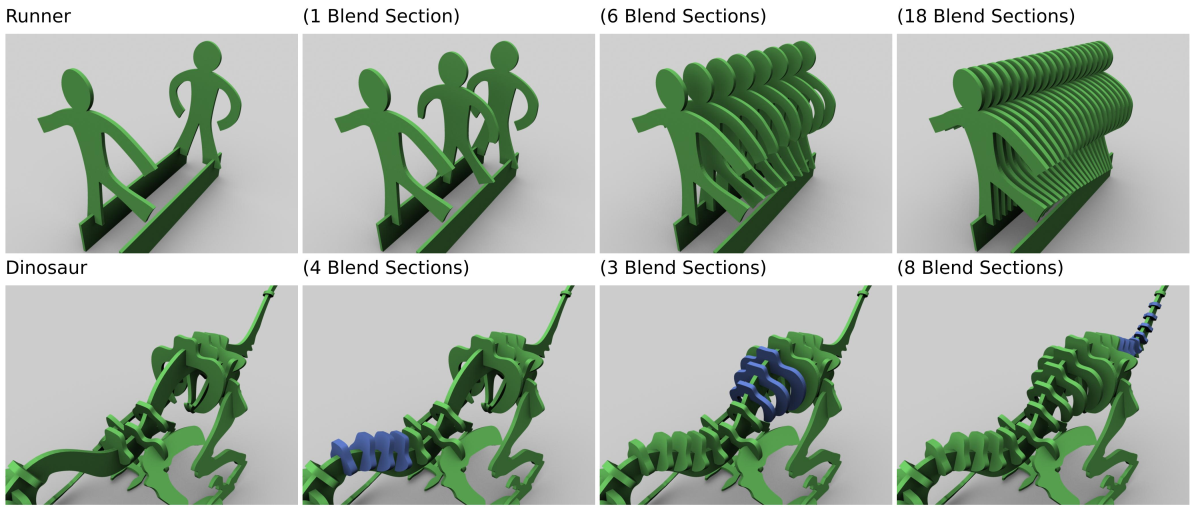


Figure 5.20: (Top row) Two planar sections show a figure in different poses. Intermediate sections are created using the blend operation, here we show 1, 6 and 18 blended sections. (Bottom row) We apply the blend operation three times to create vertebrae and ribs along intervals of the spine of a dinosaur (blended sections created are rendered in blue for emphasis).

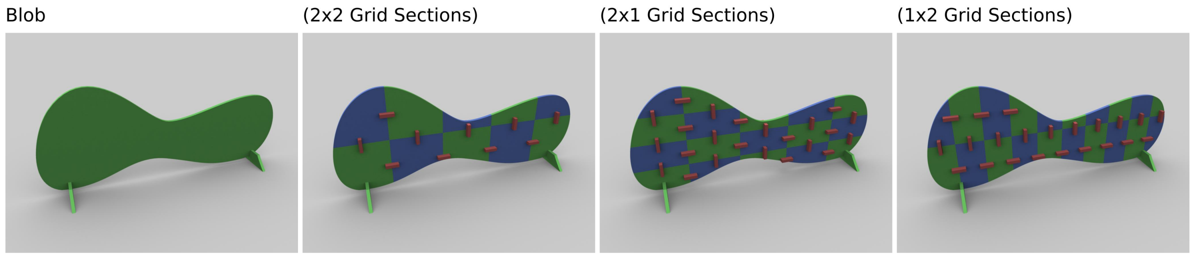


Figure 5.21: A planar section (left) is split into sections in a grid pattern (sections are rendered in blue and green to reveal the structure). Sections to connect adjacent grid sections (rendered in red) are automatically generated along edge midpoints.

If we let $\{p_0, \ldots, p_{n-1}\}$ and $\{q_0, \ldots, q_{n-1}\}$ be the equally-sized sets of 2D boundary curve points of P_2 and P_3, we compute an offset i that minimizes the sum of pairwise distances:

$$\min_{i \in \{0,\ldots,n-1\}} \sum_{j=0}^{n-1} \left|\left|p_{index(j+i)} - q_{index(j)}\right|\right|_2, \tag{5.2}$$

where $index(i) = i \mod n$. (Note that the set of boundary curve points of P_3 is also reversed, to test for an ordered correspondence in both directions.) The optimal correspondence is defined by an integer value for the offset and a Boolean value for the direction.

The intersection of P_2 and P_3 with P_1 define two points along a boundary curve of P_1 (we assume P_2 and P_3 each intersect one common boundary curve of P_1 at one point, or else the operation does not take place). Since P_1's boundary is a closed curve, there are two paths one could travel from P_2's intersection to P_3's intersection. We assume that when blending two sections P_2 and P_3, the shortest path (arc-length along P_1's boundary) should be selected. We use the shortest 3D curve segment of P_1's boundary connecting P_2 to P_3 to position the blended sections. The tangents and normals along the curve are used to define a coordinate frame for each blended planar section. Recall that since a cubic Bézier spline defines the boundary curve, the coordinate frames will be vary smoothly, except where a tangential discontinuity is explicitly defined by the user at a control point. Since P_2 and P_3 may not have a normal parallel to the tangent of P_1 where they intersect the boundary, the two rotation offsets θ_2 and θ_3 are first computed, and blended coordinate frames are additionally rotated about P_1's normal by an angle linearly interpolated between θ_2 and θ_3.

5.3.3 Grid operation

The maximum dimensions of each planar section may be restricted due to the size of materials used for fabrication. A laser cutter for instance is constrained to make sections only as large as the sheets of material or the rectangular bed the material rests in. The *grid* procedural modelling operation divides large sections into a collection of smaller sections that assemble together, in order to address this specific problem.

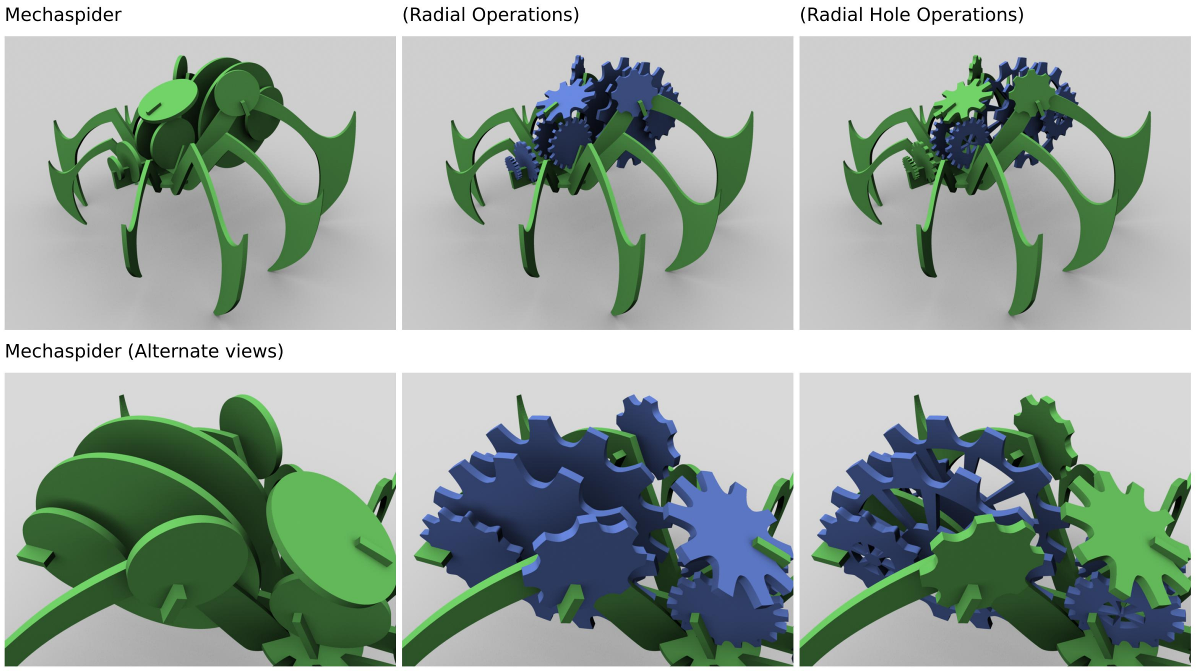


Figure 5.22: (Left) A spider model composed of some disc-shaped sections. (Middle) The radial operation is applied to create gears on the model. (Right) The radial hole operation is used to cut holes into some of the gears. (Sections the operations are applied to are rendered in blue.)

The operation is parameterized by the maximum width and height of each grid section, as well as the size of square-shaped sections used to connect the grid sections together along the midpoint of adjacent edges. In Figure 5.21, we show various settings for the grid section dimensions to make both rectangular and square grid patterns.

5.3.4 Radial contour operations

The *radial* operation modifies the shape of a selected planar section radially, allowing users to control the distance to a center point to make circles, stars or holes. The operations are parameterized by point and tangent control point distances from the center of the section, as well as the number of times the pattern should repeat, for example to create various gear shapes with varying holes and teeth in Figure 5.22.

5.3.5 Branching operation

We start off with n connected planar sections ($n \geq 2$). We define one of the planar sections to be the *root* within a hierarchy (a tree). The root planar section intersects k *child* planar sections, which are the roots of their own sub-trees. Recall that each child planar section interlocks with the root using a defined slit at the intersection. The branching operation replaces each of the k existing sub-trees with the current entire tree of n planar sections.

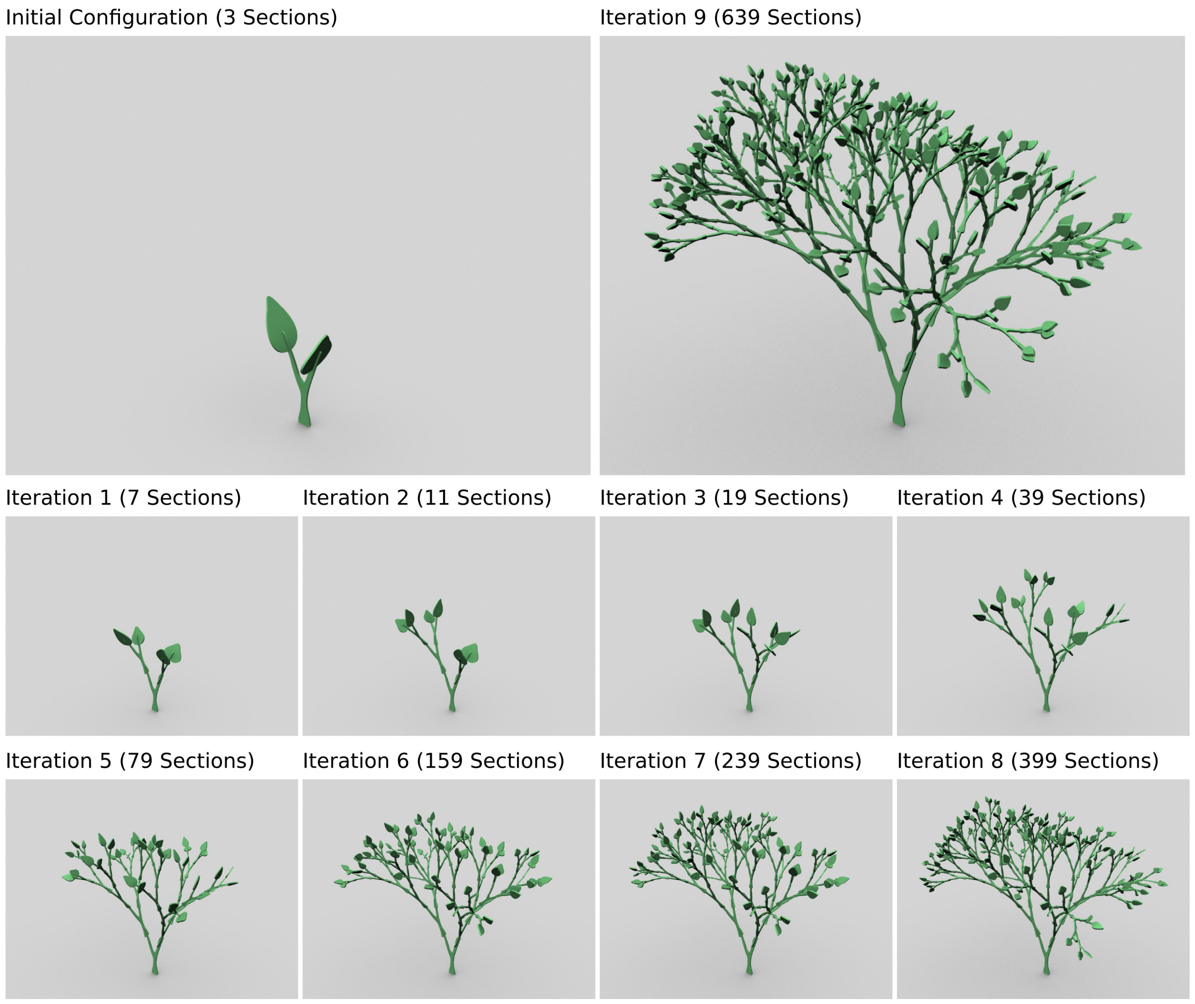

Initial Configuration (3 Sections)
Iteration 9 (639 Sections)
Iteration 1 (7 Sections)
Iteration 2 (11 Sections)
Iteration 3 (19 Sections)
Iteration 4 (39 Sections)
Iteration 5 (79 Sections)
Iteration 6 (159 Sections)
Iteration 7 (239 Sections)
Iteration 8 (399 Sections)

Figure 5.23: (Top left) The parent planar section (a branch in the shape of a "Y") and its two children (shaped as leaves) are all the required input for our procedural modelling approach, which produces a planar section representation of a tree (top right) after 9 iterations. (Bottom rows) Visualizing the progression of the representation after each iteration.

Note that the planar sections, and the definition of *slits* that define how to spatially interconnect them, are all that are needed for the branching operation to take place. However, we require one additional slit that describes how the root section (and thus the entire tree) connects when it is copied to replace each of the k sub-trees. The *root slit* is interactively specified by the user.

Applying the operation iteratively, the representation becomes progressively more complex. After each iteration, the resulting tree will have $nk + 1$ planar sections (without modifications). To represent more natural-looking objects, we can make modifications such as *reducing scale* of child planar sections after each iteration. Another modification is to *randomize* the outcome of the operation – for instance to copy over only some of the existing k sub-trees, at random. We demonstrate a planar section representation created using an operation with both of these modifications in Figure 5.23, where we are able to create a fabricable, natural-looking tree representation from minimal designer input.

5.4 Physical Simulation

The system performs physical simulation[1] in order to provide visual feedback to the user revealing the location of structural weaknesses. The feedback makes physically weak designs obvious, and the user is able to fix these weaknesses interactively during the shape modelling session. These structures are often not meant to be isolated, instead they are functional objects fabricated for everyday use. The user can model such intended use by interactively adding weights, which apply additional external forces within the simulation.

The physical simulation consists of two parts. The first is *inter-plate analysis*, and the second is *within-plate* analysis. For the inter-plate analysis, we assume the plates are undeformable and unbreakable rigid bodies, and compute the forces on the joints and contact points. Then, for the within-plate analysis, we compute the stress for each individual plate using the forces computed in the inter-plate analysis. (See Appendix A for the details behind the formulation and implementation of this approach.)

The approach allows for one to sample arbitrary cross-sections for a plate. We uniformly sample the plate using cross-sections along four unique directions within the plate: $(1, -1)$, $(1, 0)$, $(1, 1)$ and $(0, 1)$. A physical experiment revealed that the plates can fracture local to slits where sections join, so we ensure cross-section samples are selected that are local to each slit (see Figure 5.24).

Cross-section colors transition linearly through hues of green, yellow, and orange until the maximum stress the material can handle is reached and sections are colored in red (see Figure 5.26 (left column)). This effectively enforces a *factor of safety* of five when no cross-sections are displayed. The user is free to interactively vary each of three required physical measure-

[1]Note that the formulation and implementation of the physical simulation approach used is work by Nobuyuki Umetani that resulted from collaboration. Details of the approach are included in Appendix A.

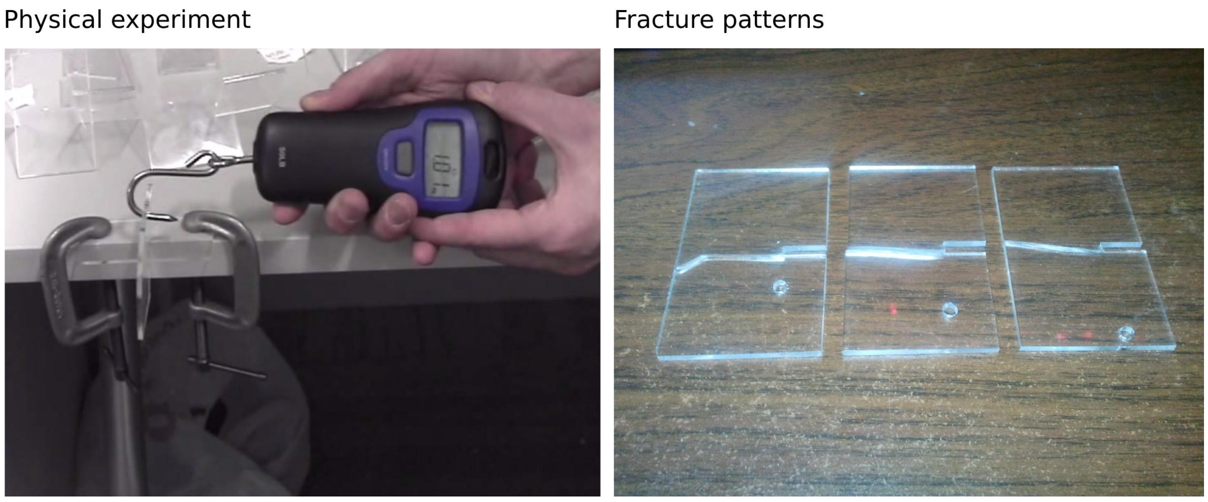


Figure 5.24: (Left) Our physical experiment where force was applied to planar section joints along three major axes to determine the amount required for fracture. (Right) Fractures occur in a consistent pattern, across planar sections.

ments: the maximum stress of the material (specified in megapascals), the density of the material (specified in kilograms per cubic metre), and the material thickness (specified in metres).

5.4.1 Verifying simulation predictions

To demonstrate our workflow and to verify that the physical simulation is able to predict fractures on fabricated objects, we created objects that serve a functional purpose by handling external weight. We tested fabrications to see if they would fracture or not, and whether the outcomes were consistent with the predictions of the physical simulation.

First, we created a cutlery holder inspired by da Vinci's "Vitruvian Man". The two forms captured by this figure were drawn on two separate planes 90° apart about the vertical axis within our system. The hands are instead shaped like the cutlery intended to be held by that arm. Our four-armed object is intended to hold (viewed clockwise from above) 6 forks (376g), tablespoons (476g), teaspoons (185g) and knives (405g).

Prior to fabrication, we use our system to verify that the model is structurally strong enough to handle the weight of the cutlery (see Figure 5.25 (top right)). Values for the material thickness, stress and density, as well as the masses that apply external force – meant to simulate the downward force of hanging cutlery – are all inputs provided to the system. Our system reveals cross-sections of planar sections where the stress is too great and where model is likely to fracture. Cross-sections where the model is likely to break are rendered in red. Our system colors cross-sections using a range of hues through red, orange, yellow, green and blue to warn the user of potential fracture regions (blue regions indicate that the stress along the cross-section exceeds 1/5 of the maximum stress the material can handle). In this example, there were no red cross-sections presented to be concerned with, and fabrication and testing revealed that the

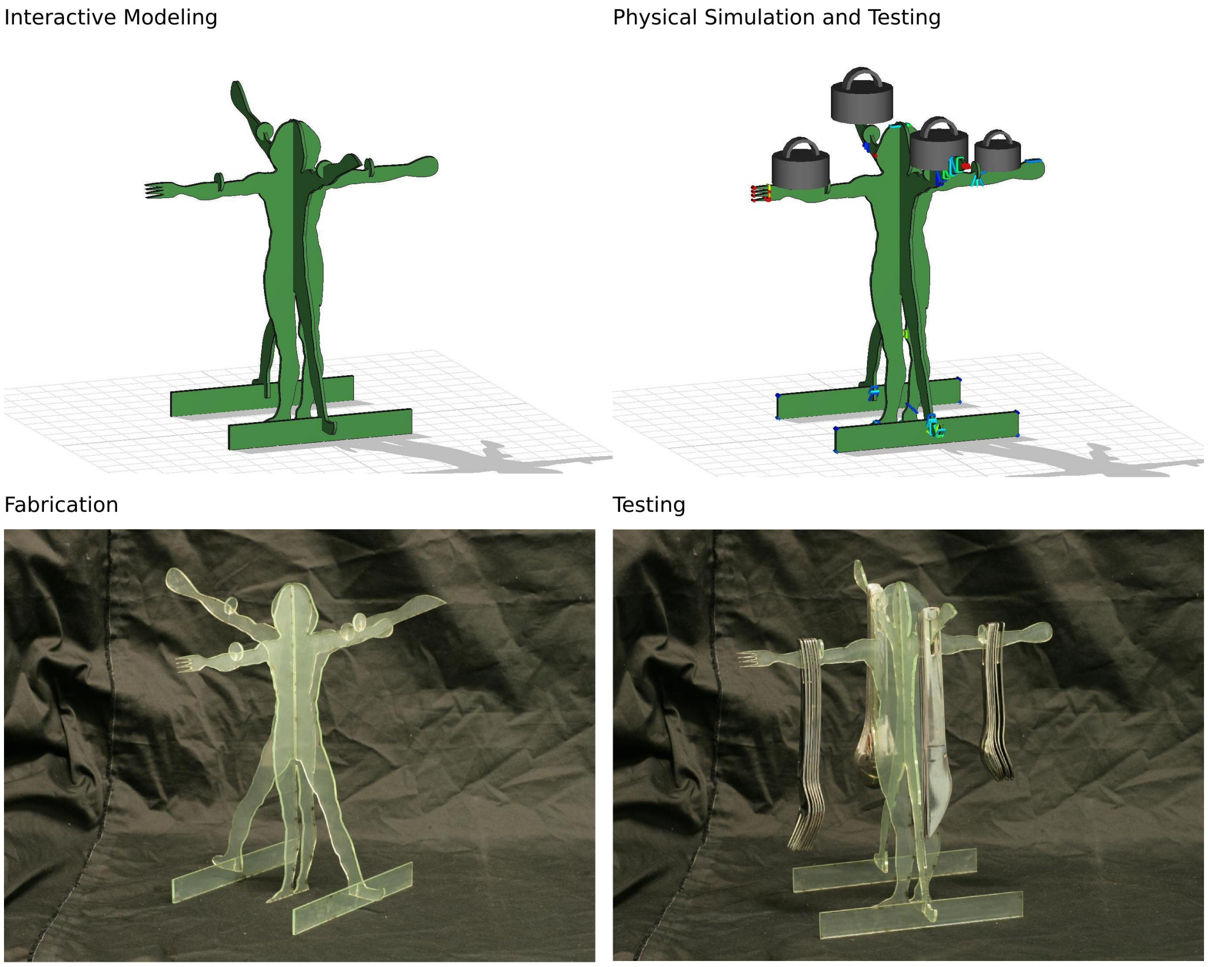


Figure 5.25: Planar sections representation of da Vinci's Vitruvian Man, repurposed to organize cutlery. Prior to fabrication using a 1/8" acrylic sheet, the model was tested by interactively attaching and manipulating weights equal to the cutlery to be held.

object was indeed strong enough for its intended purpose (see Figure 5.25 (bottom row)).

As shown in Figure 5.26, we created four variations of an object intended to hold a glass of water (275g, 350ml capacity). The first variation created had three thin legs, which fracture well before a glass can be filled. A variation with 4 thin legs, also fractures under the weight of a glass of water but shows improvement. A stronger variation with three thicker legs is able to withstand the weight of a full glass of water, but our system warns of a potential fracture where the legs and central disc meet. An even stronger 4 leg design also handles the weight of a glass of water, and our system reduces the warning of potential fracture where legs and central disc meet. This experiment verified that fabrications break when our system provides visual warning to the user, and fabrications do not break when our system provides little to no warning to the user.

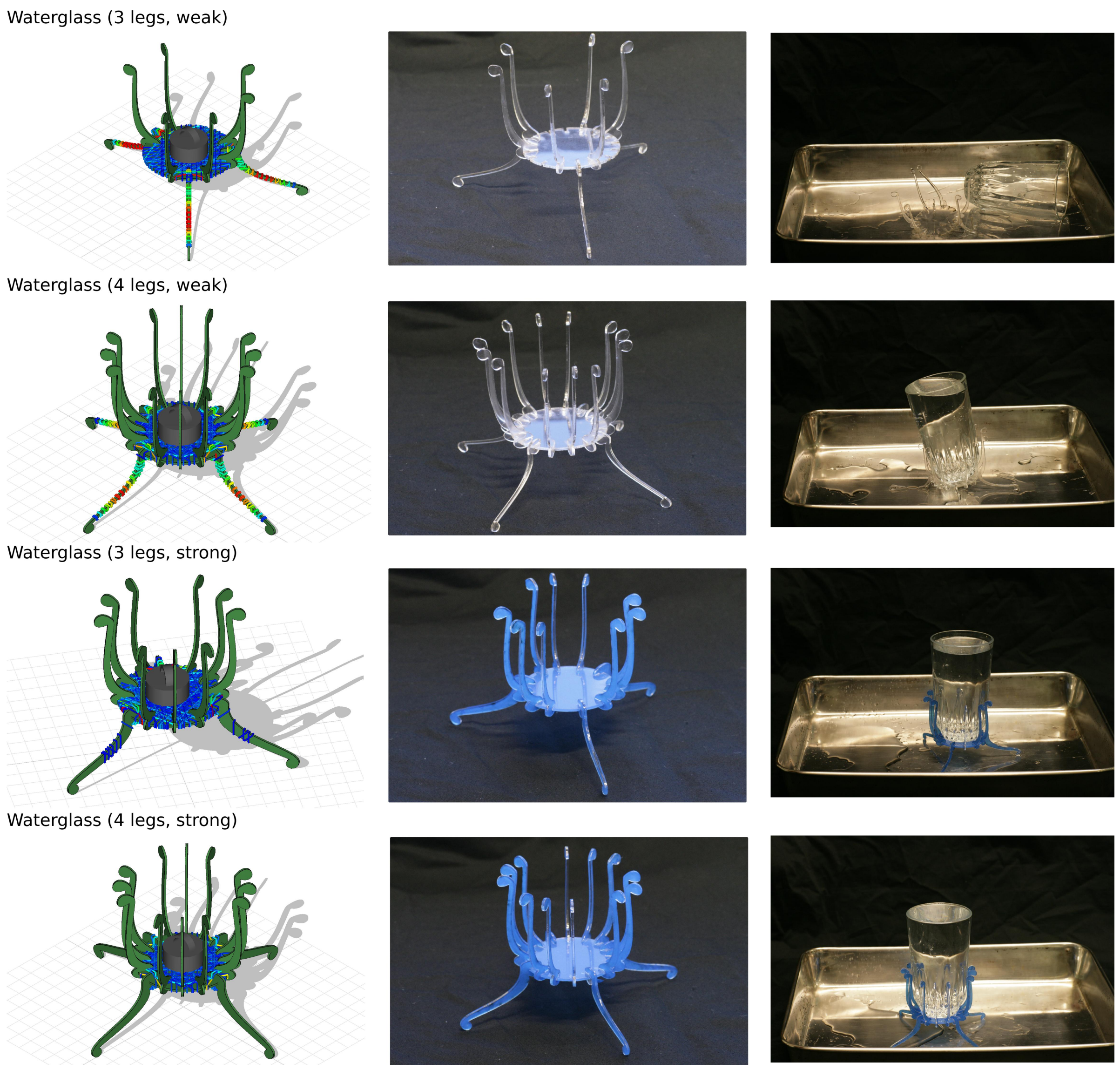

Waterglass (3 legs, weak)
Waterglass (4 legs, weak)
Waterglass (3 legs, strong)
Waterglass (4 legs, strong)

Figure 5.26: Four variations of an object intended to hold a glass of water. (First row) The first variation has three thin legs, which fracture before a glass can be filled. (Second row) A variation with 4 thin legs, which also fracture under the weight of a glass of water. (Third row) A stronger variation with three thicker legs is able to withstand the weight of a full glass of water, but our system warns of a potential fracture where the legs and central disc meet. (Fourth row) An even stronger 4 leg design also handles the weight of a glass of water, and our system reduces the warning of potential fracture where legs and central disc meet.

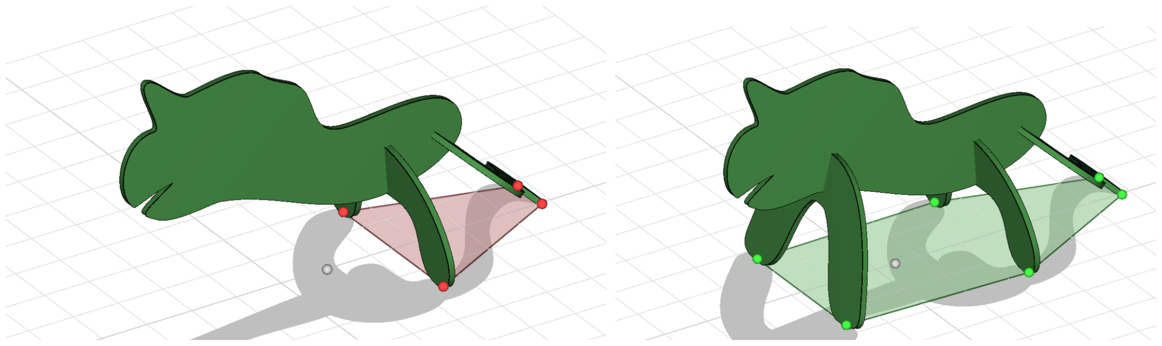


Figure 5.27: (Left) A model that will fall since the centre of mass does not fall within the contact region. (Right) Adding new ground contact points results in a geometrically stable model.

5.5 Fabrication

Our system is geared towards making planar section representations for fabrication. Representations can be exported in the SVG format, which can be used to drive a laser cutter. We detail some considerations for creating and exporting fabricable models with our system.

5.5.1 Feasibility tests

Geometric stability

Our system computes the centre of mass of the model and the convex hull of any contact points with the ground plane. If the projected centre of mass falls within the convex hull, the model is *geometrically stable* – it does not "fall over" when placed on a flat surface. Our system shows the convex hull of the extremal contact points and the centre of mass on the ground plane (see Figure 5.27).

Following any interaction to create new planar sections or modify the section contour, our system automatically translates the control points of the Bézier spline representing the contour so that planar sections do not pass through the ground plane. This ensures that once the model is manufactured it will have co-planar contact points. When the planar section contour is being created, our system renders a thick black line to denote the intersection of the current canvas plane with the ground plane (see Figure 5.28). This eases the process of ensuring that there are suitable contact points on the ground plane.

Physical assembly

Not all planar section configurations are feasible to physically assemble. Planar sections that form cycles with non-parallel intersections make physical assembly impossible. Two parallel slits are required since they divide the cycle into two parts, and these parts will slide in opposite directions along the parallel slit axis to separate the cycle. In fact, due to the orthogonality of intersecting planar sections in our system, cycles of size 6 or less that can be assembled must

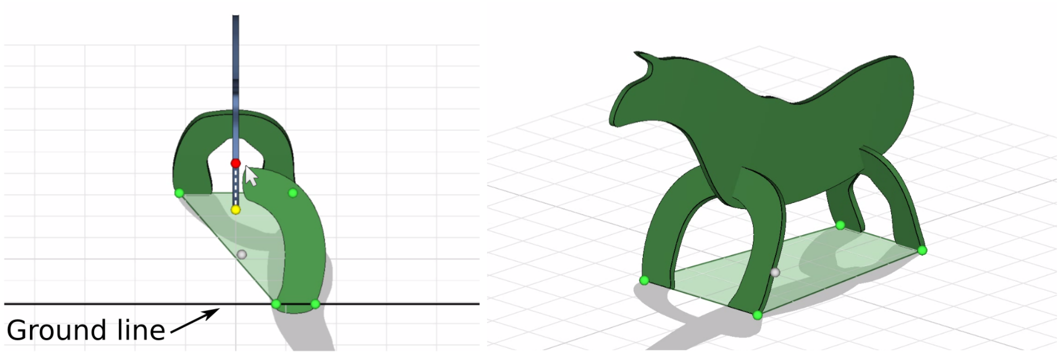


Figure 5.28: (Left) A ground line (horizontal and rendered black) reveals where the canvas plane intersects the ground plane, and extremal contact points for the planar section are also shown. (Right) After the stroke is drawn, contour control points are translated so that the planar section does not penetrate the ground plane.

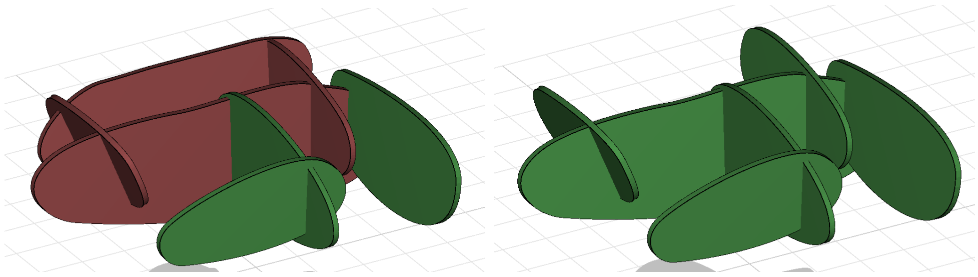


Figure 5.29: Our system tests for feasibility of physical assembly. (Left) A cycle of planar sections with non-parallel intersections (red) cannot be physically assembled. (Right) The user deletes a planar section to break the cycle, fixing the problem.

contain at least two planar sections that are parallel (see [127] for more detailed discussion). Our system detects and reveals cycles, allowing the user to resolve the problem (see Figure 5.29).

Connectedness

Planar sections that "float in space" cannot exist in the physical world, therefore we test the *connectedness* throughout interaction. This can be caused either by removal or modification of existing planar sections or removed throughout the modelling process, this may influence the connectivity of other planar sections in the model. It is possible that the user may assume that two planar sections intersect, when they actually do not. Finally, because of the dead zone, it is possible when creating a new planar section to sketch a contour that does not intersect any other planar sections, leading to an unconnected planar section being formed.

Planar sections are treated as vertices in a graph, and edges are defined by their intersections. From the planar section that has existed in the model the longest, our system attempts to compute a path in the graph to each other planar section. For those planar sections where a path cannot be found, these sections are not connected ("floating") and are rendered red to

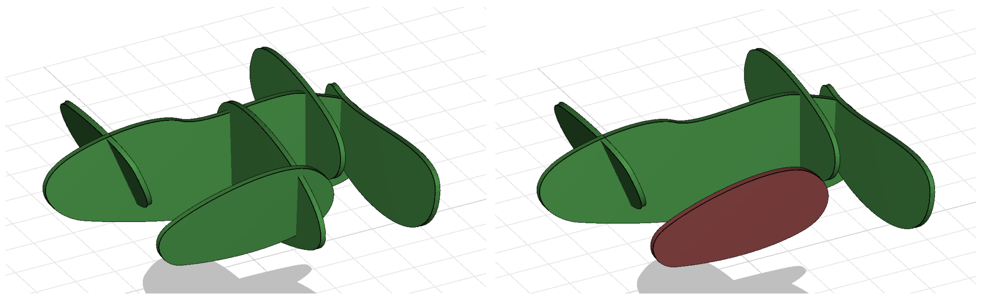


Figure 5.30: Our system tests for connectedness. (Left) A model where all planar sections are connected. (Right) The result of deleting a planar section, causing another (red) to lose its connection to the rest of the model.

bring them to the user's attention. During any interaction involving the editing of any planar section, our system computes and reveals unconnected planar sections (see Figure 5.30).

5.5.2 Slits and section correspondence

Our system automatically computes the slits to cut into each planar section. For a given pair of planar sections, the axis of intersection of the planes is calculated. Then, the points on the planar section contours that intersect the axis are computed and sorted in order along the axis. When two adjacent contour points with index k and $k+1$ along the intersection axis belong to different planar sections, an intersection condition is detected and slit rectangles are made between points p_{k+1} and p_{k+2}. To locally approximate how the two planar sections slide together, we compute and compare the distances $d_1 = ||p_k - p_{k+1}||_2$, and $d_2 = ||p_{k+2} - p_{k+3}||_2$. Depending on the direction, one slit endpoint will be defined by the midpoint between p_{k+1} and p_{k+2}, and the other endpoint will be either p_{k+1} or p_{k+2}.

The slits are shaped as rectangles that follow the intersection axis, with a width equal to the thickness of the planar sections as they are shown during interaction. The planar section thickness can be adjusted interactively (see Figure 5.31).

Planar section contours are assigned specific colours when written to the file. Slit colours matches the colour of the joining planar section, for correspondence (see Figure 5.31 (bottom row)). Fabrication output can thus be used as a guide for assembly.

5.6 Evaluation

Our evaluation was two-fold. We validated our physics model in Section 6.3 to verify both the prediction of fracture in fabricated models, as well as stability when subject to external loading (see Figures 5.1, 5.25, 5.26).

We also provided our system to 12 users (with varying computer graphics experience), with a 15 minute tutorial video and a one-page instruction sheet. We show some of their creations in Figures 5.32 and 5.33. Users also filled out an informal questionnaire, relating to modeling

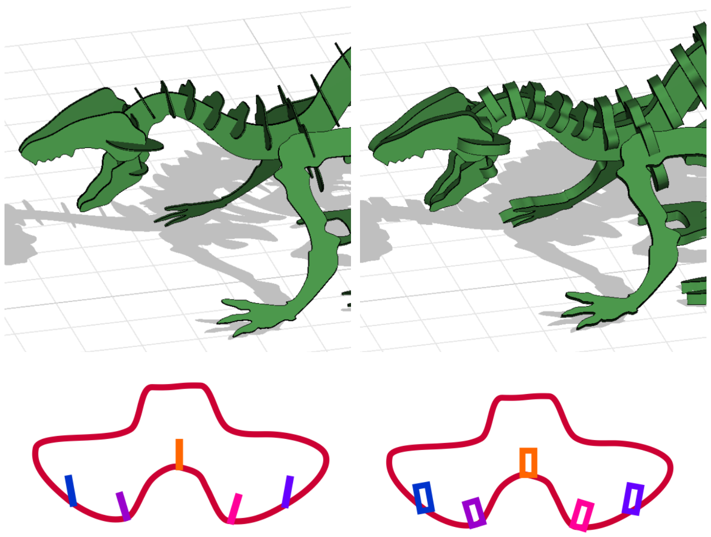


Figure 5.31: (Top row) Adjusting planar section thickness (left: 0.02, right: 0.2). (Bottom row) Vectorized 2D output for fabricating a planar section; the curves define where to cut the material. The thickness value should be set to match the material thickness, since it defines slit width.

times, experience, and specific features. Most users were able familiarize themselves with the system within 5 to 30 minutes after watching the video. Each user spent between 2 to 8 hours modeling shapes with the system, and created between 2 to 6 objects. Typical creation times for the models shown ranged between 15 to 30 minutes. The *single stroke* interface was considered by far the most "fun" and useful feature of the interface barring one user who said he would have preferred a split-view, engineering drawing based precision workflow. The *procedural operations* were easily understood and heavily used by all users. About half the users found the *physical feedback* very meaningful. The remainder (who were not fabricating the models) ignored it.

Global symmetry

While artists found the automatic local symmetry performed by our system to be a useful feature (in addition to the ability to duplicate planar sections across specific planes), our system does not currently enforce global symmetry, requiring the artist to repeat specific tasks manually to enforce symmetry. For instance, if an artist makes an edit to the contour of a planar section on one side of a symmetry plane, the system does not automatically make the corresponding edit to the other side to enforce global symmetry. While ultimately this does not impede the creation process, it makes specific editing operations more time-consuming than they could be.

Frontoparallel views and loss of context

Motivated by observations of our pilot study data, we made a decision early in the design of our interactive system that the view would rotate about the defined slit axis so the plane is

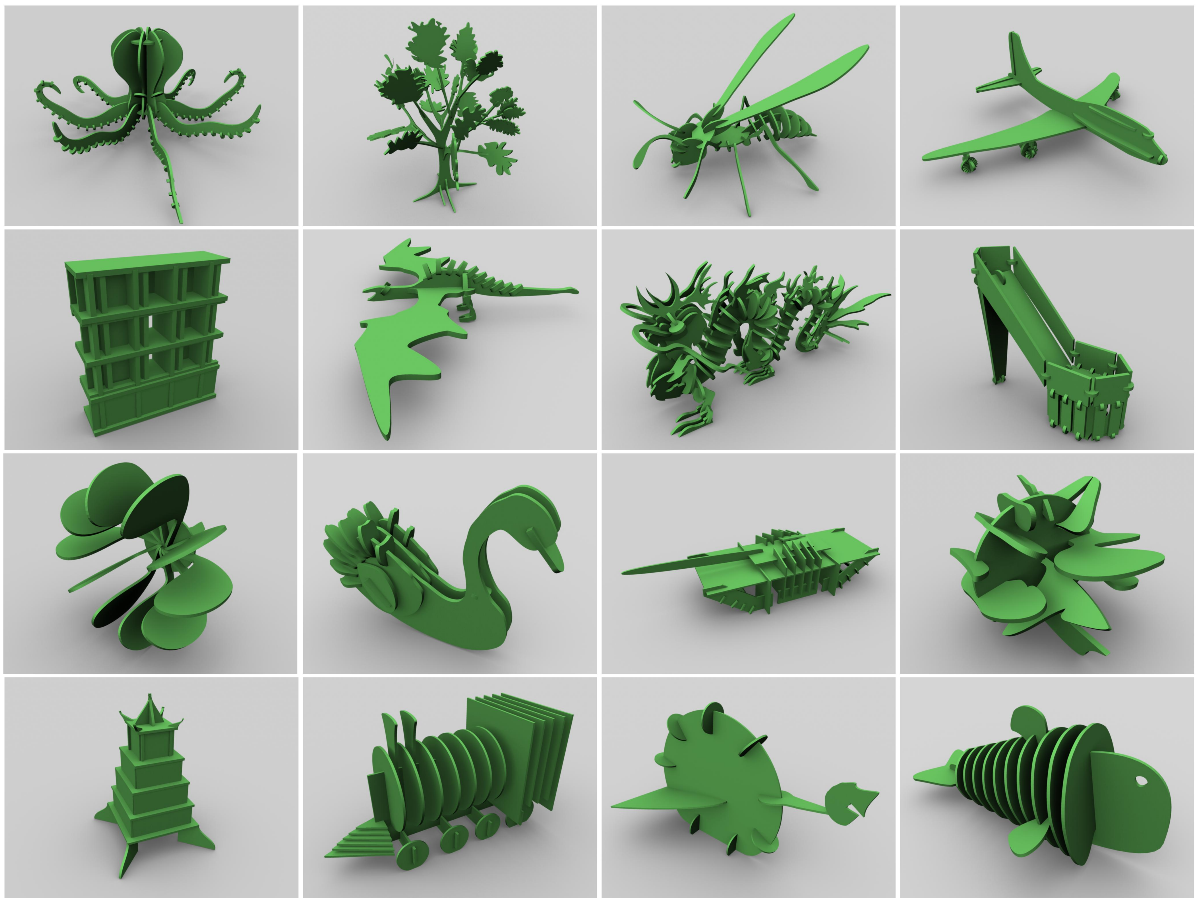


Figure 5.32: A gallery of examples created using our system by 12 unique participants of our evaluation study.

oriented frontoparallel. Determining the viewpoint is simple as there are only two possibilities - the view direction follows the canvas plane normal or its negation. We always select the view that is *downward facing*, since objects in reality are most frequently viewed from above, and since planar section representations are to be fabricated and thus viewed in general the same way, using downward-facing views during the interactive creation process makes the most sense.

One artist felt there was a loss of context resulting from the frontoparallel view, despite our decision to render the slit and its endpoints, since the planar section the slit was placed upon appears as a line (a thin rectangle) at the rotated view direction. The artist suggested that having the ability to specify an angle for the rotation – for instance to select a view direction 15 or 30 degrees from the axis of the canvas plane normal – may be beneficial to help preserve context while creating the new planar section contour.

We were able to quickly implement this additional feature. One important change that was required was to reconsider how the rotated view direction is selected. We define θ as the desired angle between the new view direction and the axis of the new canvas plane's normal; previously, θ was always $0°$. For θ such that $0° < \theta < 90°$, and where the view direction remains perpen-

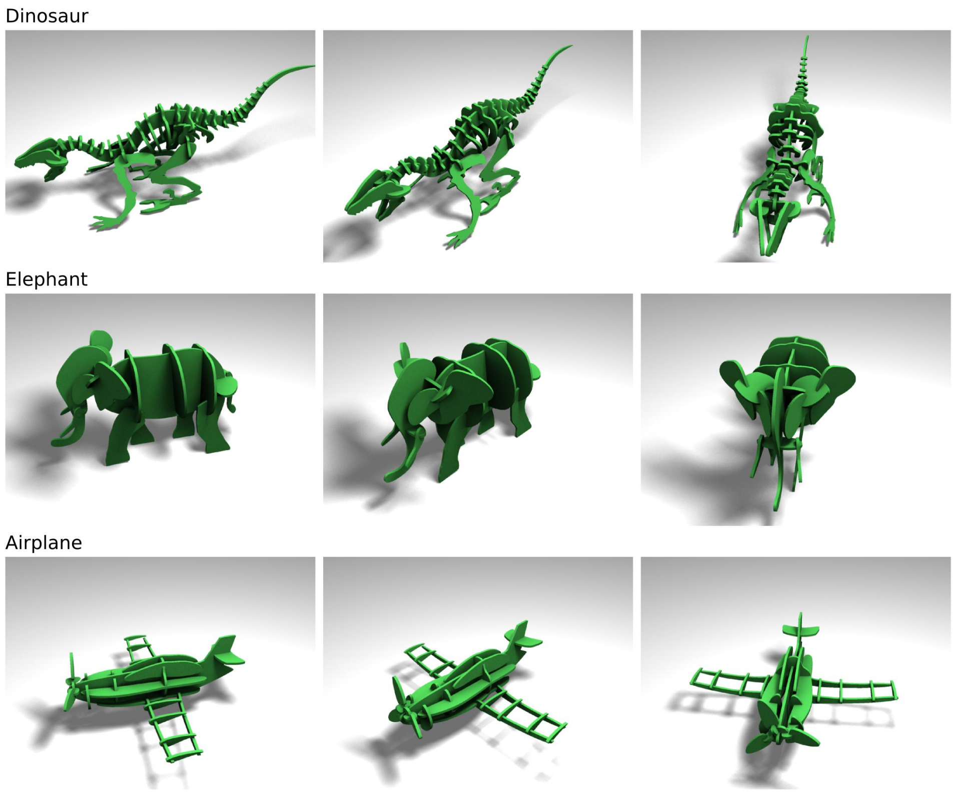


Figure 5.33: Planar section representations of *dinosaur* (top row), *elephant* (centre row) and *airplane* (bottom row), interactively created from scratch using our system.

dicular to the slit axis, there are exactly *four* possible directions that satisfy the constraints. We can narrow this down to two since only two are associated with desirable downward-facing directions (technically, a perfectly vertical slit would result in all four directions being neither downward or upward-facing). Of these directions that remain, our system chooses the one that results in minimal rotation from the current view direction.

We put our updated system into the hands of the artist, who experimented with different settings for θ. The artist found that $\theta = 25°$ was a reasonable compromise, and expressed the following:

> "25 is just enough so that I can get a pretty good understanding of where the shape will fit with respect to the rest of the model and still draw the shape as desired... Anything over 25 degrees, the foreshortening starts to become too much and it decreases my ability to draw the shape accurately. Under 25 and I start to lose 3D understanding. I find it interesting that even 5 degrees is quite a lot better than 0. I think people would generally want it somewhere between 0 and 45 degrees. As you get higher than 45 it becomes difficult to draw the shape correctly."

This particular artist is experienced and is familiar with 3D modelling software, and we conclude that experienced artists may benefit from this functionality since they are better able to compensate for the foreshortening resulting from a slanted canvas plane. But as noted by the artist, users of any level of experience may benefit from the improved 3D context resulting from even a very conservative choice for θ, such as 5°.

5.7 Conclusion

We considered the cases of both a blank screen modelling metaphor, as well as use of reference 3D geometry for the purpose of interactive creation of planar section representations. Concerning reference geometry, a more complete system could include the ability to import multiple surfaces and freely place the parts in space. This would allow for a process of selectively mixing and matching when aiming for a design that incorporates contours from multiple reference geometries. Another interesting direction for future work involves using a set of template curves as reference geometry that act as French curves, and automatically fitting parts of the template curves to sketched planar contours using an existing approach [98].

Suggestive interfaces are becoming increasingly popular [124, 156], which offer a number of possible suggestions for resolving domain-specific issues that emerge throughout the process of interactive design. For our system, it would be interesting future work to incorporate a suggestive interface to aid in the resolution of failure of the various feasibility tests that our system presently performs: stability, physical assembly and connectedness, as well as the structurally weak areas identified by the physical simulation.

With the advent of consumer-grade 3D printing technology, devices to create curved primitives are becoming more accessible. In the future, it would be interesting to create non-planar representations composed of interlocking curved surfaces.

Chapter 5 concludes our examination of the four cases by which planar section representations can be created (see Figure 5.34). Next, in Chapter 6, we revisit our algorithmically-created planar section representations in order to explore human perception of the abstracted surfaces.

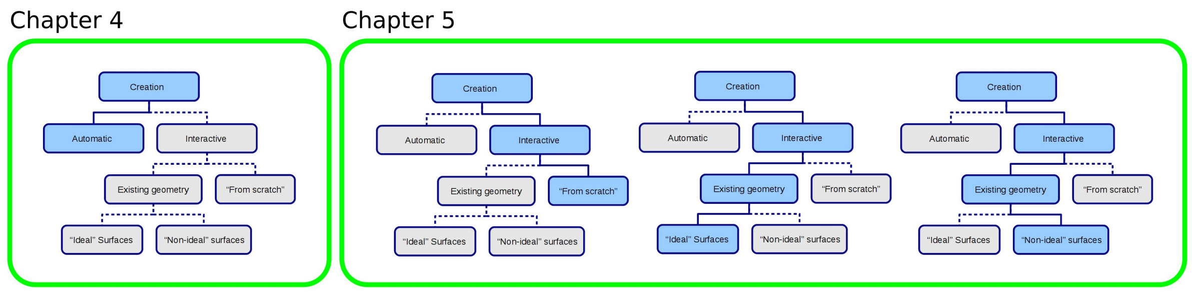


Figure 5.34: The four possible paths to creating planar section representations that have now been covered in Chapters 4 and 5.

Chapter 6

Surface Perception of Planar Section Representations

Shape information conveyed and perceived by viewers of a 3D shape abstraction is an important measure of its efficacy as a shape proxy for both art and engineering applications. Various algorithms have been proposed to produce such abstractions, evaluated either using geometric measures or visually by a few individuals. While such evaluations may be sufficient for judging recognition tasks, little is known about the accuracy of perception of the *abstracted surface*. Since the end-goal of most abstractions is in fact to be a perceptual stand-in for the original shape, a principled study of perception of 3D shape abstractions remains an ambitious, unavoidable, and worthy goal.

In this chapter, we perform a large crowd-sourced study involving approximately 70k samples to evaluate how well users can orient gauges on planar abstractions of commonly occurring models (Section 6.2). We test four styles of planar abstractions against ground truth surface representations, and analyze the data to discover a wide variety of correlations between task error and measurements relating to surface-specific properties such as curvature, local thickness and medial axis distance, and abstraction-specific properties (Section 6.3). We use these discovered correlations to create linear models to predict error in surface understanding at a given point, for both surface representations and planar abstractions (Section 6.4). Our predictive models reveal the geometric causes most responsible for error, and we demonstrate their potential use to build upon existing planar abstraction techniques in order to improve perception of the abstracted surface.

Summary of important results

Before moving forward and describing all the details of the studies performed and our analysis, we spend a moment to briefly summarize at a higher level the most important results:

We initially explored the creation of various experimental designs: we considered measuring position, orientation or a combination as a measure for task error. In the end, we settled on a wide-scale study and analysis of the orientation data only. We detail some approaches we attempted, and the difficulties we encountered. We include these initial approaches for completeness.

Some important results from the main study: presenting a participant with a planar section representation in the XYZ abstraction style (where planar sections are placed regularly along the X, Y and Z directions) resulted in approximately the same task error as presenting the ground-truth surface itself, which was surprising. (The mean task error for the surface condition, which establishes a baseline for task error, was 20 degrees.)

The other planar section abstraction styles we tested (MPS, crdbrd) resulted in significantly higher error. The radial abstraction style produced the greatest error. However, the XYZ abstraction style uses many more planar sections in its representation (mean of 40.6 sections) than MPS (mean of 5.0 sections) or crdbrd (mean of 11.3 sections). Consider if task error was normalized – for instance multiplied by the number of planes used, or the total surface area of planes used. MPS or crdbrd abstraction styles would produce far less "normalized task error" than the XYZ style.

The principal curvature κ_1 is the most significant geometric predictor for task error, for almost every condition. Other curvature measurements (mean, Gaussian) are also significant predictors for task error. For the abstractions, we found that the measurement of "abstraction angle difference", which we detail later, had a strong correlation with task error in some cases, and was statistically significant in all cases. Our analysis included correlation tests for 10 geometric measurements in total, and though there were some correlations that were significant in a statistical sense for every time of measurement we tried, these correlations were not nearly as strong.

The predictive models were an improvement over base error estimates in all conditions. For instance, for the MPS abstraction style, predictive models provide nearly a 50% improvement over the base error estimate, which we believe to be a great result. For other abstraction styles, such as XYZ, radial or for the surface itself, the improvements were not as great, but still notable (improvements ranging between 10-30%).

Finally, the last important result was that we incorporated a geometric measurement used in the MPS predictive model (specifically the abstraction angle distance) into our existing MPS algorithm to generate new "MPS-modified" planar section representations. We ran a second smaller crowd-sourced study, and found significant improvements in task error using the MPS-modified representations instead of the standard MPS – for one model in particular the improvement was a reduction in task error of over 50%.

This work explores the geometric sources responsible for error in the perception of surface orientation for planar section representations, and demonstrates that linear predictive models trained from the study data can predict task error with a marked improvement over base error estimates. Finally, we demonstrate that these geometric insights can in fact be utilized to optimize existing algorithms in order to improve perception of the abstracted surfaces.

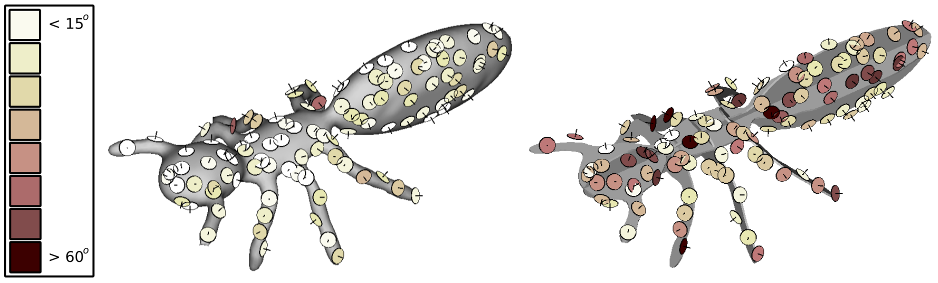


Figure 6.1: We conduct a large crowd-sourced user-study to evaluate human perception of models represented by their shaded surfaces (left) and planar abstractions (right) using an established orientation task. Participants manipulate gauges scattered across the visible surface of models to define perceived normals at each point. Overlaid are error visualizations for a fixed view rotation task for the model *Ant* (error legend at left). The complete set of error visualizations are shown in Section 6.3.

6.1 Background

Perception of surfaces has been studied before: In a seminal effort, Koenderink et al. [81] study how well subjects adjust a local gauge figure to perceptually *fit* a photographed surface. Cole et al. [30] recently adapt the work to perform a large crowd-sourced study to evaluate accuracy of orientation estimates as perceived from line drawings. In our context, since planar section representations are intended to be viewed in 3D, either once manufactured, or virtually via view manipulation, we need to investigate further. Specifically, we design an experiment to evaluate perceptual accuracy for orientation tasks, accounting for the perceptual impact of 3D viewpoint changes.

In light of a recent flurry of research in creating 3D planar abstractions [54, 55, 88, 99, 126, 127, 159], we present the first comparative study on the surface perception of 3D shape abstractions. Our work is inspired by the research of Cole et al. [30], which provided a much needed perceptual basis and validation for a body of research on 2D line drawings of 3D objects. Line drawings of 3D shape however, are largely view-dependent and their perceived surfaces are appropriately studied using static 2D images [30]. We instead explore the perception of view-independent planar 3D abstractions, designed for 3D viewing.

We acknowledge a corpus of research in shape understanding of line drawings and other artistic illustrations of shape [29, 30, 111, 119, 160] and attempt to provide similar answers to the surface perception of view-independent 3D planar section representations.

Planar abstractions ranging from pop-up sculptures [88], radial [159], orthogonal [126], or axis-aligned *crdbrd* [54] planar sections have been proposed, with a focus on 3D manufacturing. We validate this body of research by providing a comparative perceptual evaluation of radial, orthogonal (XYZ), and axis-aligned (*crdbrd*) abstractions.

In Chapter 4, we discovered that most 3D objects could be abstracted by humans using a small number of planar sections and presented an algorithm to construct such a set of *minimal planar sections* (MPS). We also provided evidence that abstractions were as recognizable as the original objects. In this chapter, we instead study the comparative fine-grained 3D surface perception of the MPS, XYZ, radial, and *crdbrd* abstractions.

6.2 User Study

Our goal is to study the effectiveness of various planar abstractions for representing 3D shapes. Broadly, we investigate the following questions:

(i) how do planar abstractions compare with the original 3D models in terms of human perception of the underlying shape;

(ii) how well can humans orient normals on the surface of the abstracted shape;

(iii) how consistently are the same planar abstractions perceived by different humans;

(iv) how do geometric features relate to the error distribution across planar abstractions?

We performed several user studies to answer the above questions. Designing the user study comprised of the following stages: choosing among candidate planar abstractions, designing specific tasks for the subjects to perform, gathering inputs from a large pool of subjects, and detecting and removing outliers before analyzing the data.

6.2.1 Candidate planar abstractions

In related user studies, Cole et al. [30] compare accuracy of perceived orientations based on human and machine generated line drawings from fixed viewpoints, while Secord et al. [129] develop a perceptual model for viewpoint preference for 3D models based on the view preferences of a large number of subjects. We also test human ability to orient normals on the perceived surface, both for shaded surfaces and their corresponding planar abstractions. What further separates our work from previous work is that we also test the condition of allowing view manipulation for the orientation task, since the planar abstractions are 3D objects themselves and are meant to be viewed from a number of different directions.

As ground truth, we used 3D surface models shaded with Lambertian reflectance to test human ability to perceive the surface. For comparison, we also selected four different planar abstractions (see Figure 6.2): (i) minimal planar slices (MPS) [99]; (ii) regular XYZ-aligned planar slices; (iii) radially arranged planar slices with a manually selected centroid and the model-up vector as the axis direction; and (iv) crdbrd [54]. In each case, we selected a suitable number of planes such that each abstraction was compact, yet of sufficient subjective quality to represent the original surface. The range of styles of planar abstraction in our study captures the large variation of the number of planes used. For example, taking the mean over all 6 models used in our studies, MPS used 5.0 slices, crdbrd used 11.3 slices, radial used 18.0 slices, and XYZ used 40.6 slices.

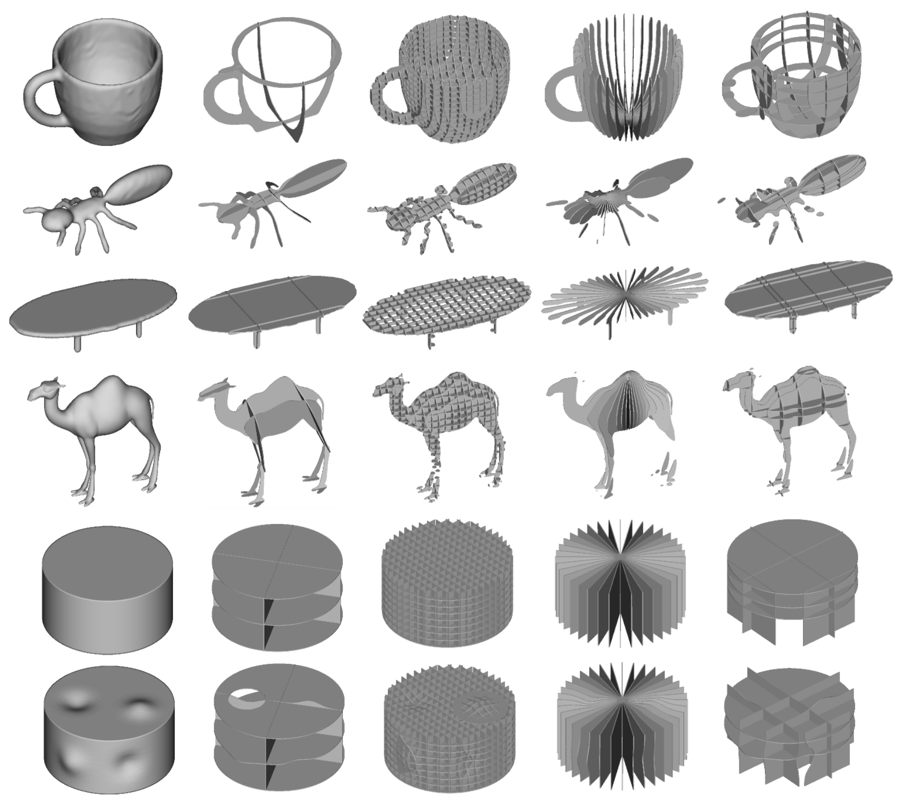
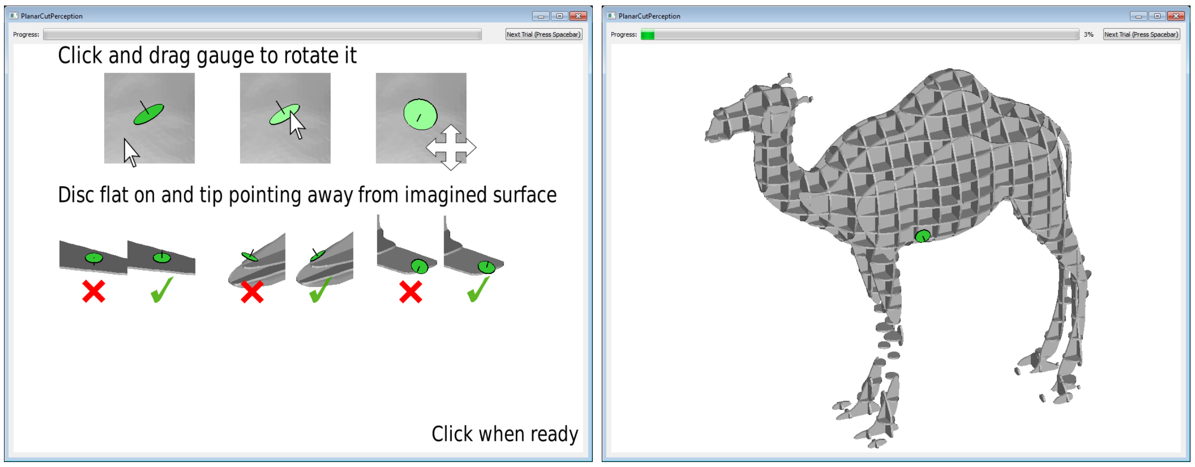


Figure 6.2: The models and various representations used in our study. From top to bottom: *Cup*, *Ant*, *Table*, *Camel*, *Cylinder*, *Bumpy*. From left to right: surface, MPS (minimal planar sections), XYZ (axis-aligned slices), radial (regular radial slices) and crdbrd. Models *Cup*, *Ant*, *Table*, *Camel* from the Watertight Track of SHREC 2007 [50] and used with permission.

Planar abstractions in general have been observed [99, 54] to represent some classes of 3D models better than others. For example, man-made objects being regular and comprising of large planar faces are better abstracted with planar slices as compared to a highly articulated organic shape, such as a spider. We can expect such variability within the genre of planar abstractions itself, such as XYZ slices being better suited to box-like objects than radial slices. We thus chose a range of models of varying geometric complexity to check the representative qualities of the different planar abstractions.

6.2.2 Methodology

Both tasks in the study involve orienting normals at sampled surface points, perceived from a presented representation (a shaded surface model or planar abstraction). We evaluated two tasks: orientation with a fixed viewpoint, and orientation while allowing the viewpoint to rotate around what is shown.


Click and drag gauge to rotate it

Disc flat on and tip pointing away from imagined surface

Click when ready

Figure 6.3: (Left) An instruction screen shown initially for every run, with information specific to the experiment condition for the participant, shown is the fixed view MPS condition. (Right) The study interface shows a representation at centre, here the XYZ style planar abstraction of model *Camel*. For the *fixed view* task the viewpoint is stationary for each model, but for the *rotated view* task the user can rotate the viewpoint around the model by left-dragging.

User interface

Participants were first shown a webpage that contained a link to download the study application, as well as instructions that are applicable to all conditions (task and representation). These first instructions explain the basic goals of the participant: how to rotate the gauges, how to proceed to the next trial, and how to submit the generated study data. This also lets the participant decide if this is a study they are interested in.

Running the downloaded study application, participants first enter their unique ID provided by the AMT framework. We map this unique ID to the study conditions for the participant, and additional instructions are displayed specific to these conditions. For a participant given the viewpoint rotation task, instructions on how to rotate the view are provided. All participants are shown examples of bad and good gauge orientations, but whether a surface or planar abstraction is shown in the examples is based on condition. For the conditions where planar abstractions are shown, the instructions state that while performing the task, they should be visualizing or imagining the surface being represented. Figure 6.3 shows the typical interface to our studies along with sample instruction images shown to the subjects (see also supplementary material).

Gauges

Similar to Cole et al. [30], we show participants a series of gauges (i.e. an oriented disc at given points) in random order. Participants are asked to rotate each gauge to the correct (perceived) orientation before moving on to the next gauge. Previous investigations of perception of pictures and sketches [81, 30] use Halton sequences to define candidate gauge positions in screen space.

Like these studies, we use Halton sequences to define 100 quasi-random gauge positions for each of the 6 models in the study. In addition, we hand author 5 additional gauge positions, accounting for regions where the spatial sampling is sparse, or if an interesting surface feature lacks gauge coverage (such as a bump). Thus each model in the study has a total of 105 unique positions where gauges are placed and ground-truth surface normals are known.

The gauges are drawn as small circular discs, with a single line segment orthogonal to the disc indicating the normal. Gauges are coloured green to make them clearly visible. We use a fixed gauge size of 1.8% of the diameter of the models (see Figure 6.3 (right)). Occlusion between the gauge and model would unintentionally provide hints about orientation, so to avoid this gauges are effectively drawn "on top" of the rendering of the presented model representation.

View and viewpoint rotation

For each model, we manually pick non-accidental default views based on the principles from Secord et al. [129]. For rendering, the gauges and model share the same orthographic projection.

Specific to the rotating view task, when a new model is presented users observe a 2 second animation where the viewpoint spins 90 degrees about the up vector and comes to rest at the pre-defined viewpoint. Importantly, when the user initiates a viewpoint rotation, gauges are not rendered at any time throughout the manipulation. The reason behind this is that viewpoint rotation should only aid in the spatial understanding of the presented representation and not the orientation task itself. If the gauge were to remain visible as a floating 3D object as the view rotates, information is being provided about a 3D point on a surface that is not explicitly presented by a surface abstraction (i.e., we are revealing more perceptual information than what is represented by the abstraction). By hiding the gauges during viewpoint rotation, we prevent any extra information relevant to the orientation task from being unintentionally provided. When the user ends the viewpoint rotation, a 0.75 second animation rotates the view back to the pre-defined viewpoint rather than immediately resetting it.

We also experimented with two other candidate tasks in a pilot study: a gauge *positioning* and *combined* task (both rotating and positioning). For the former, participants would left-drag a gauge to translate it along the fixed axis it was oriented. To do this effectively, a viewpoint rotation is often necessary, to view the gauge at angle distant from the translation axis. For conditions where the surface is shown, the positioning task becomes trivial – simply choose a view where the surface point is a silhouette and place the gauge on it. Even for the abstraction conditions, the direction the gauge is oriented in the position task reveals information about a local normal that is not explicitly represented by the abstraction. The idea of the combined task aimed to overcome these problems by not revealing any information explicitly represented, but in our pilot experiment we discovered the combined task to be too complex to explain to participants, and that it was difficult for them to produce meaningful results.

Shading and lighting

Models are rendered with Lambertian reflectance for shading, with the light coming from behind the camera at infinity. There are several reasons for this: we choose shaded models since that is how 3D abstractions would appear if manufactured; we do not add lines or contours to the shading so as not to confuse with the percept produced by line drawings that are an abstraction in themselves; we choose Lambertian shading since the effect of specular highlights on surface perception is unclear [41]; cast shadows are avoided since they can interfere with surface perception [109]; and while O'Shea et al. [109] suggest that an elevation of $20 - 30°$ above the viewpoint for best perceptual results, we observed little visual difference between that and light from behind the view, and further expect users to manipulate the view under some conditions, where the light direction will move with viewpoint. If the light direction is kept stationary, surface points not facing the pre-defined viewpoint would be uniformly dark, leading to less-complete shape understanding. In addition, surface normals at points near silhouette edges are more difficult to perceive because shading is uniformly dark there. Since the purpose of a viewpoint rotation condition is to explore effects of improved 3D shape understanding, our choice of a light direction that changes with viewpoint supports this. (Equivalently, one can also think of the viewpoint as fixed with the light direction fixed from behind at infinity; only the presented representation rotates.)

Study trials

In each trial the participant manipulates one of the gauges. We randomly set the initial orientation of the gauge to a point on the front-facing hemisphere. We selected six models in total, four from the Princeton Shape Benchmark (*Cup*, *Ant*, *Table* and *Camel*) that are a mix of man-made and organic shapes, and two manually created shapes (*Cylinder* and *Bumpy*). For each of the models presented in a random sequence, the participant performs 10 trials (5 are duplicates) totalling 60 trials in each study run. The participant is allowed to run the study up to a maximum of 10 times, but always subject to the same condition (task and shape representation) based on their unique ID, in order to avoid learning effects.

At the end of a study, the participant is informed if the performance was acceptable and the results can be submitted, or the performance was unacceptable and the results were discarded (see Section 6.2.4).

Tasks

In our study, participants were assigned one of the following two tasks:

i) Fixed view task. The user prescribes orientations by rotating gauges on fixed positions on shaded models or their abstractions, viewed from pre-specified viewpoints. The participant can left-drag gauges in order to rotate them. This task is performed with static viewpoints to keep it simple and connect our study to an established task protocol [30].

ii) Rotated view task. Each presented model is first rotated 90 degrees in an animation lasting 2 seconds. At any time, the participant can then rotate the viewpoint by left-dragging anywhere on the screen, but off the gauge. Otherwise, this task is the same as the fixed view task.

Error measures

Gauge positions on the surface are determined by inverting the projection from each screen-space position, casting a ray outward and using the closest intersection point on the surface. The measured error for a gauge is the angle between the gauge normal and the true surface normal at that point, always expressed in degrees. Initial gauge orientations are set randomly to point within the front-facing hemisphere.

6.2.3 Data collection

We deployed our studies to a large number of users using the Amazon Mechanical Turk (AMT) framework (see also [30, 129]). While this allowed us to efficiently gather large number of inputs from an uncontrolled pool of participants, we had to detect and discard outlier samples generated by insincere users (see Section 6.2.4).

The scale of each model has been normalized to fit within a unit-diameter sphere. For each model, we randomly choose a sequence of 5 gauge positions from the 105 prepared. Each of the 5 gauges is presented twice, but each has its own randomized initial configuration. All gauge positions are taken at surface positions that are both front-facing and unoccluded relative to the pre-defined viewpoint for the model.

In order to avoid cross-condition learning effects, participants are required to enter a unique ID provided by AMT that is used to define the experiment conditions (task and representation presented). Participants are encouraged to run the study up to a maximum of 10 times, and conditions remain constant throughout multiple runs (e.g., a participant selected to be shown a planar abstraction as the model representation would never later see the surface representation). If a participant opts to perform multiple runs of the study, we keep track of the gauges they have previously been assigned, so that they are presented new gauge positions each run. We manually verify that the ID entered by each participant into the study application matches the ID of the data submission on AMT.

Our complete study consists of 2 tasks: fixed view and rotating view. It also consists of 5 model representations: the surface and four styles of planar abstraction (MPS, XYZ, radial and crdbrd). In a single run of the study, a participant manipulates 60 gauges in total, 10 for each of 6 models. 178 unique participants performed a total of 1161 runs of the study, fairly evenly covering the 10 combinations of task and model representation (each condition having between 84 and 152 runs). We gathered a total of approximately 70,000 gauge samples that were used in the subsequent analysis. For each run of the study a participant was awarded \$0.35.

6.2.4 Data verification

Participants coming from the AMT may only be interested in maximizing money earned in the shortest time and with the least effort. On the other hand, for 8 of our 10 conditions participants were required to imagine or fill in details of a surface only from a planar abstraction, leading to errors despite best intentions. It is such errors we are most interested in. Hence, we design a filter that tolerates errors arising from (honest) intent.

We adapt a solution similar to those proposed in other crowd-sourced user studies [30], namely having the participants repeat the same task. Specifically, participants set 10 gauges for each model, but there are in fact 5 *gauge pairs*, where gauges in a pair share a unique position. Each gauge within a pair is set with a different initial gauge orientation. For each pair, we evaluate if the two gauge settings roughly agree (i.e., their orientations are similar). Since gauges are initialized with random orientations, participants who are trying in earnest and have a specific mental "goal" at a specific gauge position will tend to perform consistently, but not if they are carelessly manipulating the gauges.

A participant's work is deemed unacceptable if (i) less than 70% of gauge pairs are such that the gauges in each pair are within 30 degrees of each other; or (ii) if the standard deviation of the angle between pairs is less than 5 degrees. The second condition detects insincere participants who do not perform the task they have been instructed (e.g., they align all gauges in the view direction to pass the first condition, but the inputs are meaningless) (see also [30]).

On task comprehension

One might point out that when a participant is presented with only the planar abstraction, from their perspective the hidden surface is *more abstract* than the planar abstraction. We maintain that our planar abstractions are more abstract than surface representations, despite presentation of only the planar abstraction. Objectively, surface representations reveal object shape explicitly, while planar abstractions allow room for interpretation of shape. Subjectively, our reasoning is that surface representations are of greater geometric similarity to how most objects in reality present themselves, which all participants will be acquainted with.

One might also point out that when a participant is presented with only the planar abstraction, this may produce difficulty with task comprehension – the participant may align gauges to the planar sections directly rather than to an imagined surface. Further, one may consider the observation that a participant's estimated normals matching nearest planar section normals is evidence that they misunderstand the task. However, how does one reliably tell the difference between (i) the participant's intent of orienting the gauge directly onto a planar section, or (ii) the participant fully comprehends the task yet orients onto an (incorrectly) imagined surface (that happens to align with the nearest planar section)? Because of this fundamental difficulty, we did not remove or filter participant data where we may suspect the task was not fully understood. In the design of our study, we were careful in our validation of participant data to allow inclusion of any sources of error, as long as results were consistent. If we did not do so, we would potentially introduce bias and arrive at incorrect conclusions. The best we can do is to strive to make the instructions clear so that participants understand the task. On both AMT and in our study application, each time the program is run, the initially presented instructions state (for the planar abstraction conditions) that gauges should be oriented on the imagined surface. Visual examples where gauges are slanted/tilted away from the nearest planar section normal are also provided (see Figure 6.3).

6.2.5 Pilot study and surface position tasks

In a pilot study, we aimed to design task protocols that went beyond the measurement of surface normals as in previous work. We considered two other tasks: (i) a surface *position* task, and (ii) a *combined* task where both surface position and normal were specified simultaneously by a gauge within a trial.

As both of these tasks involve using the gauge to define a perceived surface position, allowing unconstrained view manipulation (where the view does not "snap back" to a default position) is a requirement. The reasoning is that as surface normals become parallel to the view direction, the ability to discern where the gauge's position in space and how much translation occurs during manipulation becomes challenging. In addition, one must observe the gauge from multiple points of view in order to establish a sense of position (any single view the position along the depth axis is ambiguous). Finally, since the initial gauge positions are selected randomly and the assumption they are constrained to (possibly abstracted) surfaces no longer applies, unconstrained view manipulation is necessary in order to resolve position.

We describe in more detail and discuss specific challenges relating to each of the two position tasks.

Position task

The participant can change view by left-dragging anywhere on the screen, but off the gauge. Left-dragging on the gauge itself allows movement along an axis defined by the gauge's position and direction (set to the normal direction). Note that view manipulation is essential for this task since the translation effect becomes increasingly ambiguous as the view direction becomes aligned to the gauge direction.

Unlike tasks involving only the surface normal, where the gauge position unambiguously defines the intended surface normal for that point, initializing the gauge with a randomized position can create ambiguity about which true surface point is being used. Hence, we limit the maximum distance perturbation for the gauge from the true surface point (we used a limit of 10% of the model diameter).

The position task becomes trivial when the surface representation is shown, as one can choose a view direction orthogonal to the surface normal and position the gauge to lie on the silhouette. For participants who were meticulous, we would expect task error to be approximately zero. This makes a comparative analysis between surface and abstract representations not meaningful.

Combined task

An issue with the position task is that it arguably provides viewers with surface orientation information for the point of interest, which is not explicitly present in abstractions. This motivated the design of a more complex *position and orientation* task that does not reveal any information about the surface, other than what is visually provided by the planar abstractions.

This is a 6 degree of freedom task along with the ability to manipulate viewpoint. In our initial tests, we found the task to be overly complex for most users. Instead, we allowed pilot study participants to change the view (left-drag off gauge), gauge orientation (left-drag on gauge), or gauge position (right-drag on gauge) one at a time.

While measuring error for the position task is simple, for the combined task it is not because gauges are unconstrained (within a sphere of radius 10% of the model diameter). Note that we first randomize the position, and then the orientation. Oppositely ordered, one could simply perform a single translation to return the gauge to the initial surface position. Also, since participants are unaware of which surface position was used to initialize the gauge, we cannot assume they will have the goal in mind to return the gauge to a specific position; the gauge may instead be placed anywhere within a local neighbourhood, we use a sphere with bounding radius of 10% to constrain translation to a local region of the model during interaction.

To evaluate error for a gauge (gauge position g_p and normal g_n), we find the nearest surface position s_p (the surface normal at s_p we denote s_n). The gauge error is then a weighted combination of errors of position and orientation:

$$error = \frac{1}{0.02} \mid\mid s_p - g_p \mid\mid_2 + \frac{3}{\pi} \cos^{-1}(s_n \cdot g_n). \tag{6.1}$$

Note that as the units for position and orientation differ, we propose weights that serve to normalize these two quantities prior to their addition. We selected weights for each such that when the *error* exceeds 1, the gauge would subjectively be classified as “maximally incorrect”. For the position term, a distance between s_p and g_p of 0.02 units maximizes the *error* quantity (note that models in the study are normalized to be of unit diameter, and also note that the position of each gauge is constrained to be within a ball of 0.1 units). For the orientation term, a weight is selected such that the angular distances beyond $\pi/3$ radians (or $60°$) are maximal values for *error*.

Pilot study results

Four of the models, *Camel*, *Ant*, *Table* and *Cup* from our main study appeared in our pilot study, but two (*Human* and *Airplane*) did not. An additional representation, *Variational Shape Approximation* (VSA) [28], also appeared in our pilot study. Though VSA representations are similar to planar abstractions in that both can be made to consist of piecewise planar sections,

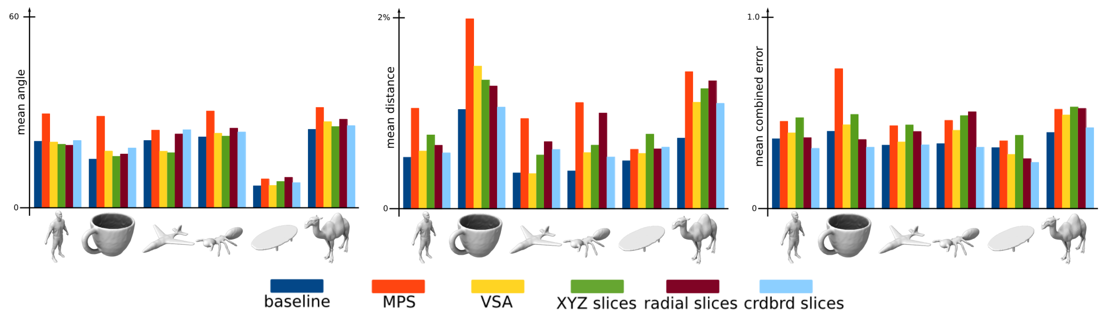


Figure 6.4: A visualization of mean task error for each model and condition in our pilot study. (Left) the *orient* task (units in degrees); (middle) the *position* task (units expressed as percentage of model diameter); (right) the *combined* task (units computed using Equation 6.1).

VSA examples geometrically are similar to the ground truth surfaces they represent. Because of this redundancy with surfaces, we removed the VSA representation from our main study.

Since the tasks we designed in the pilot study are each measured in a different unit, it is not reasonable to perform a direct comparison of performance across tasks. However, comparisons between models (while keeping task and representation fixed), or comparisons between representations (keeping task and model fixed) are both reasonable to make. Figure 6.4 visualizes mean task error for each task, model and representation used in our pilot study.

As expected, the surface representation almost always produced the lowest error. However this was not always true, consider the results for the model Table in the combined task where some abstractions caused less error than the surface representation. This result indicates that perhaps objects of highly planar structure can be represented by planar abstractions at least as well as surface representations. We observed that almost universally, the MPS representation produced the greatest error, even across all tasks. For the position task, the relative error of MPS compared to the (baseline) surface representation was largest – our reasoning for this observation is that the MPS abstraction, consisting of the fewest planar sections, has the highest average surface-to-abstraction distance of all planar abstractions. We believe surface perception is made more difficult when the positions being tested are geometrically distant from the abstraction. (A follow-up sub-study, using modified MPS abstractions that minimized the surface-to-abstraction distance, revealed substantially reduced error providing evidence in support of this claim.)

6.3 Analysis of Results

We first perform some general analysis of the data: we detect additional outliers, consider the bas-relief ambiguity, and compute mean error for each model and condition. A visualization of the entire dataset following our initial treatment is shown in Figure 6.5.

We consider measurement of task error only at gauges positioned on flat surface regions in order to consider a baseline for task performance. We also detail where we discovered correlations for a number of non-geometric measurements including: user and gauge persistence, gauge consistency, trial duration and number of camera rotations. (Note that all correlations we present were tested for statistical significance: $p < 0.05$.)

We then consider a number of geometric measurements one can evaluate at each gauge position, and discover in many cases correlations between the measurements and task error. Many of these measurements are based on the geometry of surfaces: curvature, local thickness, medial axis distance, centroid distance and view-normal angle difference. Two others are unique to planar abstractions, these are: abstraction distance, and abstraction angle difference.

We detail the meaning and how we computed all of these measurements in the following subsections. The results of correlation tests between all of these measurements and task error are presented in Table 6.3.

6.3.1 Initial analysis

Outliers

Despite taking measures to verify that participants are well-meaning through demonstrating consistent performance, we still encountered outliers. Looking at mean error per participant, we discovered one that had a mean error exceeding 120 degrees taken over 10 runs of the study. Such participants fall well outside the normal range, but are few. Our method to detect these outliers is: for each participant, we compare their mean error with the group of other participants with the same experiment condition (task and model representation). If the participant's mean error falls outside of three standard deviations of the group mean, the participant is deemed an outlier and their data is discarded. Using this method we classified 4 participants out of 182 as outliers, leaving the results of 178 unique participants to use in our analysis.

Error

Table 6.1 shows the mean task error for each combination of model and condition in our study. Comparing the performance between tasks, we observed a difference in mean error within about 3° in most cases. The exception to this observation was the radial representation, where the ability to rotate the viewpoint seemed to result in a significant improvement of about 13°. However for all other representations, showing and allowing viewpoint rotation does not seem to significantly reduce average error; we find this result to be of interest.

Comparing the performance between representations by averaging tasks, as expected we observed the surface representation produces the lowest error. Among the planar abstractions, the best style was XYZ. This result was expected given that this abstraction style uses the most planar sections of any abstraction. However, the worst style was radial (despite using 18 planar sections in each abstraction, more than both the MPS and crdbrd styles).

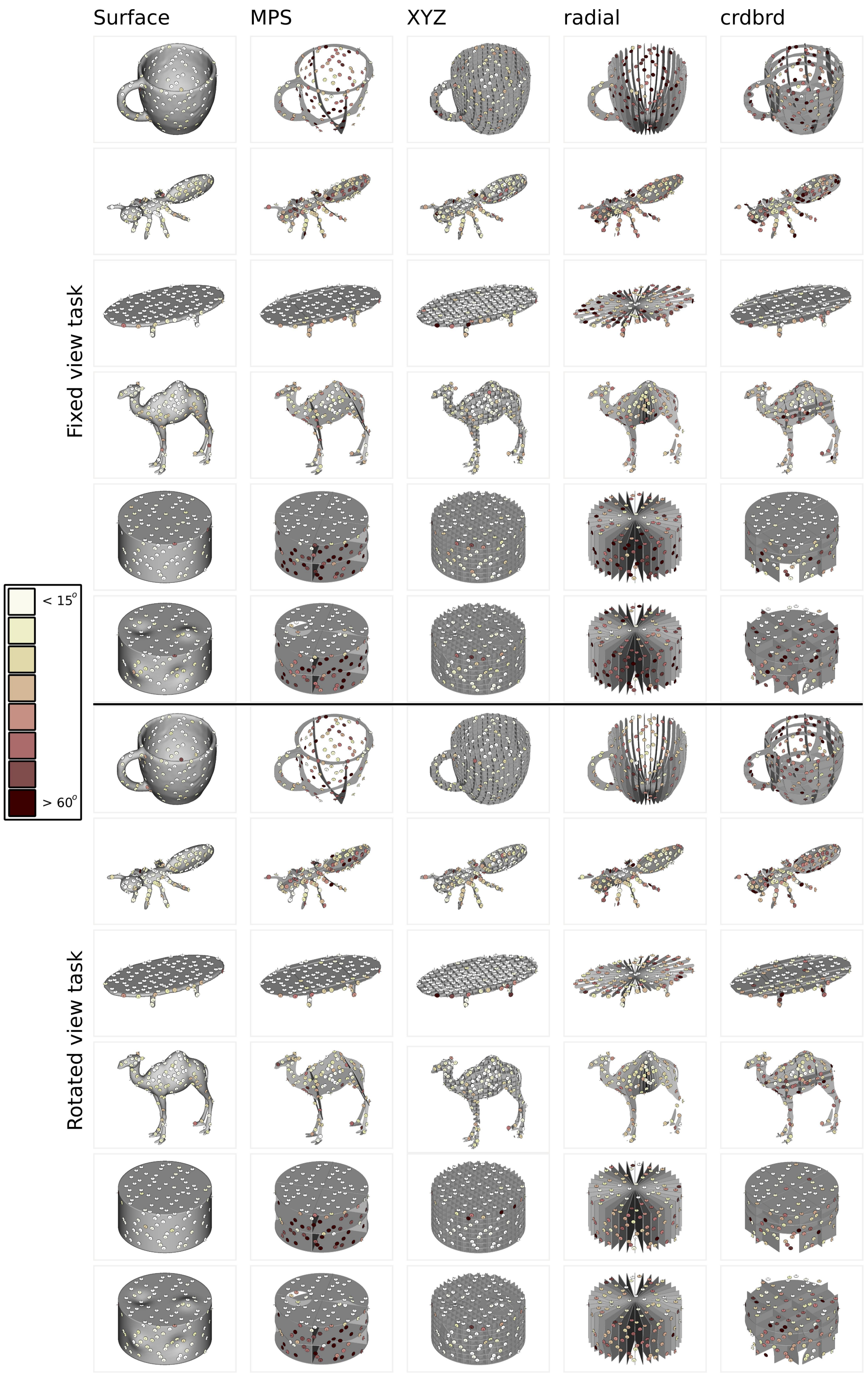


Figure 6.5: Error visualizations for all models and representations for the (top) fixed view and (bottom) rotated view tasks.

Table 6.1: Mean errors for the fixed view task (white rows) and rotated view task (grey rows). Units are in degrees.

	surface	MPS	XYZ	radial	crdbrd
Cup	20.7	50.4	31.5	62.7	54.7
	22.5	46.7	23.9	44.4	55.2
Ant	24.6	40.8	32.5	44.9	48.0
	24.5	39.9	30.8	38.0	47.9
Table	13.2	14.8	17.4	49.9	16.8
	13.1	15.0	13.4	38.5	26.0
Camel	26.1	35.9	28.5	42.2	40.9
	25.5	34.3	25.7	34.6	44.9
Cylinder	17.7	36.8	21.1	58.3	25.7
	16.2	35.2	15.6	41.5	31.4
Bumpy	19.2	38.3	20.8	59.2	42.0
	18.8	35.5	16.3	42.3	40.9
Average	20.2	36.2	24.3	52.8	38.0
	20.1	34.4	21.0	39.9	41.0

On bas-relief ambiguity

Bas-relief ambiguity [9] exists when a surface shaded with Lambertian reflectance is viewed under orthographic projection. The generalized bas-relief (GBR) transformation, consisting of a scaling and shearing along the depth axis, captures the family of plausible surfaces that all produce the exact same rendering (note that a corresponding change in surface albedo is required for the shading to appear unchanged following the transformation).

A surface point $\mathbf{p} = (x, y, f(x, y))$ becomes $\bar{\mathbf{p}} = (x, y, \lambda f(x, y) + \mu x + \nu y)$ under transform parameters λ, μ, ν (see Figure 6.6). In addition to stretching and shearing along the depth axis, the transformation changes the surface normals. For each normal $\mathbf{n}$ (transformed so that x and y components follow the right and up directions in the image plane, and the z component follows the depth axis) the GBR transformed normal $\bar{\mathbf{n}}$ is such that

$$\frac{1}{\lambda}\begin{bmatrix} 1 & 0 & -\mu \\ 0 & 1 & -\nu \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} \bar{n}_x \\ \bar{n}_y \\ \bar{n}_z \end{bmatrix}. \tag{6.2}$$

In previous work by Cole et al. [30], given a participant's gauge orientations for a specific view, the parameters of the GBR transform are optimized so that the true surface normals at the gauge positions most closely match the participant's samples. We also experimented with this, implementing an algorithm to compute optimal GBR transform parameters via gradient descent for each of 6 independent sets of 10 participant samples. We discovered that participant error is aggressively reduced, the mean error for 10 samples decreasing typically by at least 5°.

The bas-relief ambiguity only applies to participants with a fixed view condition, as viewpoint rotation will reveal surface variation along the pre-defined viewpoint's depth axis. Exam-

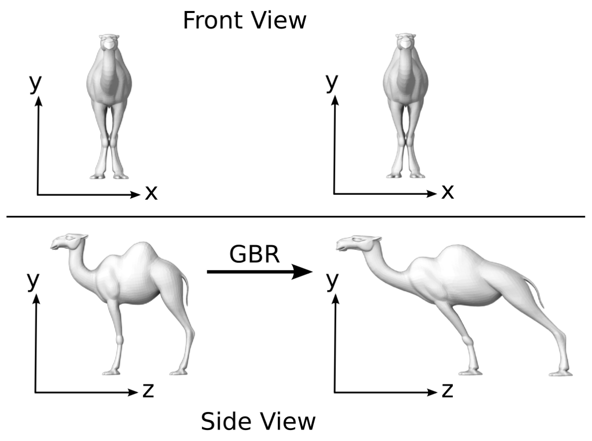


Figure 6.6: The user's static view (top row) is orthographic and remains unchanged following the GBR (Generalized Bas-Relief) transformation shown from an alternate side view (bottom row). Parameters for the transformation shown are $\lambda = 1.25$, $\mu = 0$, $\nu = -0.5$.

ining the mean error between conditions (see Table 6.1), we observe roughly consistent performance (excluding radial) to within a few degrees. For example, for the surface representation, mean errors were 20.2° and 20.1° for the fixed and rotated view conditions.

Based on this observation, we decided not to filter our data to account for the ambiguity present in the fixed view conditions. If we had done this, mean errors for fixed view conditions would be significantly lower than the equivalent rotated view condition. This would not be meaningful, as the fundamental difference between these two conditions is that viewpoint rotation provides more geometric information to participants; this should only serve to reduce error or have no effect, but should certainly not cause a significant increase.

Error at flat surface regions

Two models in our study, *Table* and *Cylinder* have large flat regions that can be used to establish a sense of baseline error for task performance. Since the geometry of flat regions are easiest to perceive, we can speculate that the error observed may be due largely to the participant's inability to perform the task accurately (that is, they may perceive the correct normal, but have limited ability in manipulating the gauge to an intended orientation). We include these findings because they may be of interest, especially for comparing error with all other gauges used in the study.

The results are shown in Table 6.2. As expected, the mean error for flat regions alone is generally consistently lower. From these findings, we believe that the baseline error to be somewhere between 10 and 13°, excluding the radial representation and an interesting finding where the rotated view crdbrd condition resulted in 8.7° more error than the fixed view crdbrd condition. We attribute the increased error of the rotated view crdbrd task to a difference in ability between the groups of participants selected for the conditions (17 participants for the

Table 6.2: Mean error taken only for gauge positions at flat surface regions, for the fixed view task (white rows) and rotated view task (grey rows). Units are in degrees.

	surface	MPS	XYZ	radial	crdbrd
Table	10.6	10.9	12.7	48.8	12.5
	9.2	10.5	8.3	37.1	22.2
Cylinder	15.5	10.6	14.4	48.2	11.6
	13.3	9.9	7.9	34.9	19.1
Average	13.1	10.7	13.5	48.5	12.0
	11.2	10.2	8.1	36.0	20.7

Table 6.3: A table showing Pearson's correlation coefficient r for correlations between task error and a variety of other measurements. Rows show correlations for a specific condition, columns show correlations for a specific measurement with error. Red entries indicate any correlation that was very low ($|r| < 0.1$) or not statistically significant ($p \geq 0.05$). The three groups of measurements from left to right are: general, surface-specific and abstraction-specific.

Condition	user persistence	gauge persistence	gauge consistency	trial duration	number of view rotations	absolute κ_1	absolute κ_2	Gaussian curvature	mean curvature	local thickness	medial axis distance	centroid distance	view-norm angle difference	abstraction distance	abstraction angle difference
Fixed surface	0.95	0.17	0.51	🟥	N/A	0.51	0.42	0.44	0.49	🟥	-0.13	🟥	-0.17	N/A	N/A
MPS	🟥	0.31	0.51	🟥	N/A	0.42	0.26	0.19	0.42	0.49	-0.20	0.23	🟥	0.39	0.72
XYZ	0.47	0.21	0.59	🟥	N/A	0.54	0.42	0.46	0.53	🟥	-0.24	0.11	-0.14	0.13	-0.26
radial	🟥	0.25	🟥	0.15	N/A	🟥	-0.14	-0.16	🟥	0.39	-0.10	0.28	0.21	0.20	0.30
crdbrd	0.52	0.33	0.49	🟥	N/A	0.46	0.41	0.33	0.46	0.14	-0.16	0.15	🟥	0.17	0.43
Rotated surface	0.57	0.25	0.59	🟥	🟥	0.60	0.50	0.53	0.57	🟥	-0.21	🟥	-0.19	N/A	N/A
MPS	🟥	0.31	0.50	0.17	🟥	0.40	0.25	0.18	0.39	0.46	-0.20	0.21	-0.13	0.37	0.72
XYZ	0.63	0.22	0.57	🟥	🟥	0.64	0.50	0.56	0.62	🟥	-0.28	🟥	-0.22	🟥	-0.35
radial	🟥	0.18	0.45	🟥	🟥	0.12	🟥	🟥	0.11	0.22	-0.16	0.21	0.15	0.19	0.17
crdbrd	🟥	0.15	0.42	🟥	🟥	0.43	0.37	0.31	0.42	🟥	-0.20	0.10	🟥	🟥	0.43

🟩 ≥ 0.5 🟨 ≥ 0.3 🟧 ≥ 0.1 🟥 ≥ 0.0 N/A not applicable

fixed view crdbrd condition, 18 participants for the rotated view crdbrd condition).

On comparing abstractions

It is worthwhile to note that a direct comparison between the errors resulting from different styles of planar abstraction may not be meaningful as two abstractions may vary in other important ways, and may be created with any other requirement in mind than task error minimization. For instance, some styles such as XYZ use many planar sections, while others such as MPS and crdbrd are more frugal. Another consideration may be the surface area (or amount of material, in a physical sense) used. Some abstractions have a configuration that may or may not be *realizable* – meaning the abstraction is physically assemblable by creating slots and sliding them together [54, 55]. Thus a direct error-based comparison between one abstraction style and another to determine which is "better" may not be a fair one as it ignores properties such as these, which may be of importance.

6.3.2 Participant-specific correlations

Persistence

Participants are given co-located gauge pairs to set as part of our approach to validation. The measurement of the difference in angle between the two samples for the gauge pair we call the *persistence* measurement. Mean persistence can be computed for a participant, or for a gauge. In general, we expect persistence measurements to be below 30° since the main criteria for accepting a participant's results is that persistence is below 30° at least 70% of the time. Despite this, persistence measurements sometimes do exceed 30° and the variation in persistence may lead to some interesting correlations.

Testing for correlation between persistence and error averaged over each participant, we discovered some strong correlations of statistical significance for the surface and XYZ conditions, and also the fixed view crdbrd condition. We also tested correlation between mean persistence and error for gauges, but found correlations were generally small. Correlation results for both measurements are shown in Table 6.3.

Consistency

We explore the notion of *consistency* of performance at a gauge position. To do this, we measure the standard deviation of task error for samples taken at the gauge position. We tested for correlation between consistency and error and discovered that generally, there was strong correlation. Since consistency and error share a clearly evident linear dependence in most cases, for any given gauge position where mean error of the samples is low, the samples will be consistent, and vice versa. For the surface, MPS and XYZ abstraction styles, consistency and error are strongly correlated for both tasks.

Trial duration

In our study, we measured the duration in milliseconds it takes for the participant to complete the orientation task for each presented gauge. We tested for correlation between mean trial duration at the gauge and mean error at the gauge in order to determine if there was any relationship. When testing for correlation between mean trial duration and mean error at a gauge position, we discarded those samples where the trial duration exceeded 15 seconds – in our data there were even rare trials where the trial duration exceeded five minutes (likely a participant taking a break mid-run).

We predicted that an inverse correlation between trial duration and mean error might exist. Upon testing we were surprised to find that generally, there were no strong correlations between trial duration and error (see Table 6.3).

Number of view rotations

Only applicable to the rotated view task, we measured the total number of view rotations each participant performed and tested for correlation with their mean error. We discovered no correlations of statistical significance for any of the five applicable conditions. We also tested normalizing the number of view rotations by the number of runs of the study the participant completed, in order to account for the variation, but still discovered no correlations of statistical significance (see Table 6.3).

6.3.3 Surface-specific correlations

For the following correlations, we aim to maximize the coefficient of determination R^2. To do this, for each measurement we considered the class of *power transformations* that include logarithm, square root, reciprocal, etc., as instances. Specifically, we found that curvature-related measurements had correlation maximized when taking the *fourth-root*, and the local thickness measurement had maximum correlation when *squared*.

Curvature

We tested for correlation between curvature at a gauge position on the surface and the mean error at the gauge position. To evaluate curvature at surface positions, we used the method of Rusinkiewicz [118]. We consider four different measurements: absolute principal curvatures κ_1 and κ_2, Gaussian curvature and mean curvature. As expected, we found strong correlations with error across many conditions, the correlation coefficients can be seen in Table 6.3.

Local thickness

We were curious about a relationship existing between a notion of *thickness* at a surface point and task error there. To estimate local thickness, we implemented a ray collision test: we emit a ray from the gauge position in a direction opposite to the surface normal, intersecting it with the surface, and use the nearest intersection point whose distance is non-zero. We use the distance to the nearest intersection point as our measure of local thickness. Testing for correlation between local thickness and mean error, we discovered positive correlations for the MPS and radial styles of abstraction for both fixed and rotated view tasks, as shown in Table 6.3.

Medial axis distance

To compute an approximate medial axis for each model, we implemented the approach of Palágyi [110]. To compute our medial axis distance measurement for a gauge position, we use the smallest distance to a point on the medial axis. We tested for correlation between the medial axis distance measurement and the mean gauge error. We expected there may be an inverse correlation between them and for all conditions, we discovered a small inverse correlation. These results are shown in Table 6.3.

Centroid distance

The centroid is computed for each model, and we evaluate the distance between the centroid and each surface point. Checking correlation results in Table 6.3, we found statistically significant small positive correlations for the majority of conditions – suggesting that error increases at points further from the centroid of the surface.

View-normal angle difference

We measured the angle between the view direction and the (back-facing) surface normal at each position. As shown in Table 6.3, we found statistically significant small negative correlations. These correlations indicate the tendency for error to increase as the surface normal becomes parallel with the view direction. The exceptions to this general observation are the conditions with the radial representation – since the radial abstraction style has the most "geometric coverage" at the centre of models and less where the surface's silhouette will be from the given view, the view-normal angle difference is high.

6.3.4 Abstraction-specific correlations

Abstraction distance

For each gauge position, we computed the distance to the nearest point on the planar abstraction. We call this measurement the *abstraction distance*. Styles of planar abstraction that use the fewest planes in their model representation (such as MPS or crdbrd) we expect to exhibit the highest variance for this measurement. We tested for correlation between abstraction distance and mean gauge error for the conditions where abstractions are used. We discovered that for the MPS style abstraction that the abstraction distance most correlated with error, with a few other small correlations as shown in Table 6.3.

Abstraction angle difference

As with abstraction distance, our method to compute the abstraction angle difference measurement first involves finding the nearest point on the abstraction to the gauge position. The nearest point lies on a planar section that has a constant surface normal, we call this the *abstraction normal*. To compute the *abstraction angle difference* measurement, we compute the angular difference between the abstraction normal and the true surface normal at the gauge position. We tested for correlation between abstraction angle difference and mean error, and discovered significant correlations of varying strength for all 8 conditions. Of these, the correlations for the MPS style were especially strong and medium strength correlations were also found for the crdbrd conditions. Conversely, for XYZ and radial abstractions, because abstraction angle differences are often very high (distributions have a mean closer to 90 degrees and smaller variance), correlations with error are not strong. The correlation coefficients are shown in the last column of Table 6.3.

6.4 Predicting Error

We perform *supervised learning*, where geometric measurements or *predictors* are used to predict a dependent outcome variable or *response*. We use the study data for *learning*, meaning we create a predictive model from it. In our study, the predictors are the measurements that can be made at surface points (such as curvature or medial axis distance), and the response is the task error measured at that point.

6.4.1 Linear models

Given the variety of strong linear correlations discovered in the data and presented in Section 6.3, we chose to use a linear predictive model. Linear models tend to be more stable than nearest neighbour models, but are more biased. We use the least squares error criterion to measure model fitness.

Expressed as a function $f : \mathbb{R}^p \to \mathbb{R}$, linear models take the following form:

$$f(\mathbf{x}) = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j h_j(\mathbf{x}),$$

where p is the number of predictor variables, $\mathbf{x}$ is a p-vector of input predictor measurements, $h_j : \mathbb{R}^p \to \mathbb{R}$ is predictor j's transformation function and $\hat{\boldsymbol{\beta}}$ is a $p+1$-vector of linear parameters for the model ($\hat{\beta}_0$ is the *intercept* or *bias* term).

6.4.2 Improving linear model fit

For a linear model to be a reasonable choice, the residuals should exhibit homoscedasticity (constant variance) and normality (residuals follow a normal distribution). To test this, we fit a least-squares-optimal linear model to the predictors and response, and examined the residuals as shown in Figure 6.7 (first column, red) using histogram and normal probability plots. Note that the distribution in the histogram exhibits skewness and does not appear normal. In the normal probability plot, deviation of the data points from the line indicates non-normality of the residuals.

To correct this, we use the Box-Cox transformation. Figure 6.8 (left) shows the log-likelihood as a function of λ, which is the power parameter used to transform the responses. The Box-Cox method assumes that the residuals are most likely to be normally distributed for the choice of λ that minimizes variance of residuals. We choose the value of λ that maximizes the log-likelihood, within the 95% confidence interval as shown. The optimal value of λ varies with the data obtained for each condition, but for many conditions a value of 0.25 fell within the 95% confidence interval. The choice of 0.25 corresponds to a fourth-root transformation of the response (task error). Table 6.4 details the choice of λ for each condition.

Figure 6.7 (second column, green) shows the distribution of residuals to be roughly normal and the mean is brought closer to zero. The normal probability plot also reveals the residuals

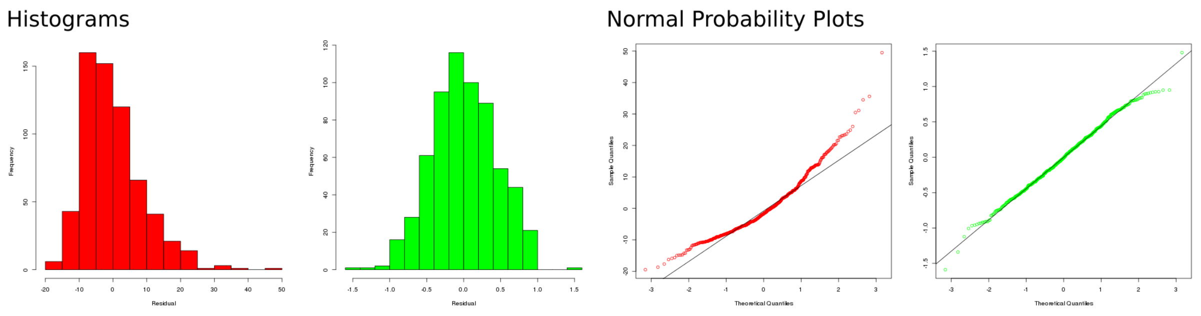


Figure 6.7: Histograms and normal probability plots for the residuals of curvature (red) and the transformed curvature (green) for the fixed view surface condition. Note that the distribution of residuals is more normal (an assumption for a linear model using least squares fit). The normal probability plots compare the quantiles (percentage of residuals below a given value) to those quantiles of a theoretical distribution – this reveals that the distribution of transformed residuals is more normalized.

to closely match a normal distribution, except at the extremes of the tails which is acceptable.

6.4.3 Regularization

We perform *regularization* – determining which predictors significantly contribute to the predictive model and eliminating those that do not. Regularization prevents over-fitting and improves generalization by simplifying the model.

We use the LASSO method [151] to select predictors. The LASSO method works by penalizing the $\mathcal{L}_1$-norm of $\hat{\boldsymbol{\beta}}$. A parameter λ can be used to adjust the amount of penalization, traditionally a value of λ that is within one standard error of the minimal mean squared error is selected – this choice is known to be *parsimonious* and builds the model with as few predictors as possible, while keeping the mean squared error of the fit low. We visualize a choice of λ in Figure 6.8 (right).

6.4.4 Validation

We perform k-fold cross-validation using $k = 10$, which is known as a good compromise between bias and variance (e.g., leave-one-out cross-validation, where $k = n$, exhibits low bias but can have high variance). The n input samples are split up into 10 groups or *folds* of approximately equal size. We train the model with data from 9 folds and use the last one for testing. This process is repeated 10 times, so that each fold gets used for testing once. We are careful to re-train the model each time using only the folds selected for training.

The *estimated prediction errors* for the model are evaluated as the average of the mean absolute errors over all 10 folds. We can compare this value to the *base error*, which is the error resulting from using the mean of the responses in each training set as the predictive model. The ratio between estimated prediction and base errors reveals how much the predictors collectively

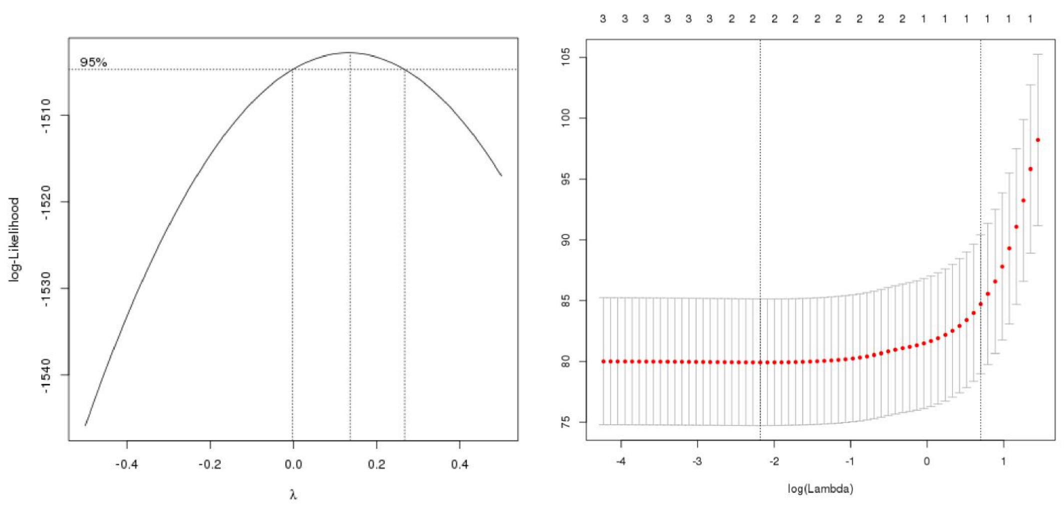


Figure 6.8: (Left) Box-Cox transformation reveals the best power parameter λ to use to linearize the data and normalize residuals, here we show the transformation for the fixed view surface condition. The dashed lines indicate the 95% confidence interval where optimal choices for λ lie, in this case $\lambda \in [0, 0.25]$ is ideal. (Right) A plot showing mean squared error (MSE) as it relates to the log of LASSO penalization parameter λ. Choosing $\lambda = 0.112$ (left dashed line) minimizes error, but uses two predictors for the model. Choosing $\lambda = 2.016$ (right dashed line) results in higher penalization, and only one predictor is used in the model. The largest λ is selected where MSE is within one standard error of the minimum.

improve each linear model. Table 6.4 shows cross-validated error statistics for models fit to each condition.

We used the *R* statistical computing software [115] and the *glmnet* package [47] to perform the regularization and validation steps.

6.4.5 Models and performance

The purpose of our models is to predict the value of the response: the task error. Given a surface representation, the 8 relevant predictors are: curvature (κ_1, κ_2, Gaussian and mean), local thickness, medial axis distance, centroid distance and view-normal angle difference. Given a planar abstraction based on a (hidden) surface representation, there are two additional predictors: abstraction distance and abstraction angle difference. Note that all of our predictors capture some geometric property of the representation at each surface point – we do not incorporate non-geometric measurements from the first group of columns in Table 6.3 (e.g., trial duration, measures of persistence, etc.).

Predictor selection

Table 6.4 shows all $\hat{\boldsymbol{\beta}}$ parameters for our linear models built for each of the 10 conditions. Regularization causes numerous predictors to be excluded from the model in many cases – the corresponding $\hat{\boldsymbol{\beta}}$ value of the predictor is set to zero – and this is indicated with an "."

Table 6.4: A table revealing parameters for each condition's linear model, and cross-validated error measurements. The linear model parameters $\hat{\beta}_0$ to $\hat{\beta}_{10}$ correspond to: (0) intercept, (1) fourth-root absolute principal curvature κ_1, (2) fourth-root absolute principal curvature κ_2, (3) fourth-root absolute Gaussian curvature, (4) fourth-root absolute mean curvature, (5) medial axis distance, (6) squared local thickness, (7) centroid distance, (8) view direction-normal angle difference, (9) abstraction distance and (10) abstraction angle difference. Cross-validated errors include mean absolute predicted error (MAPE) and mean absolute base error (MABE). The final column expresses the percentage improvement resulting from model predictors, using the ratio of MAPE to MABE.

Condition	$\hat{\beta}_0$	$\hat{\beta}_1$	$\hat{\beta}_2$	$\hat{\beta}_3$	$\hat{\beta}_4$	$\hat{\beta}_5$	$\hat{\beta}_6$	$\hat{\beta}_7$	$\hat{\beta}_8$	$\hat{\beta}_9$	$\hat{\beta}_{10}$	MAPE	MABE	Improved
Fixed surface	13.8	4.69	.	.	.	.	.	.	.	–	–	6.43	7.64	16%
MPS	9.07	4.75	.	.	.	.	40.9	.	0.02	126	0.27	10.2	19.5	47%
XYZ	16.0	6.79	.	.	.	.	.	.	.	.	.	9.08	11.0	18%
radial	27.0	.	.	.	.	-37.3	29.5	14.3	0.12	194	0.15	11.6	14.3	19%
crdbrd	9.75	.	4.53	.	7.83	.	12.0	.	0.02	209	0.19	14.1	19.2	27%
Rotated surface	11.9	5.69	.	0.26	.	.	.	.	.	–	–	5.81	7.74	25%
MPS	10.25	3.59	.	.	.	.	33.3	.	.	111	0.29	10.5	19.0	45%
XYZ	10.7	7.73	.	0.40	.	.	.	.	.	.	-0.01	7.36	10.7	32%
radial	19.5	1.91	.	.	.	-44.3	10.6	6.36	0.10	177	0.11	9.98	11.3	12%
crdbrd	9.23	7.90	8.08	-3.01	.	-9.44	2.72	.	0.14	139	0.18	12.8	16.6	23%

entry. (Note that although ideal, $\hat{\boldsymbol{\beta}}$ values cannot be used for relative comparison of predictor importance, even if predictor scales are standardized. However, the inclusion of the predictor in the model does ensure it to be of some *significance* in characterizing task error for the condition, which leads to some insights.)

The absolute principal curvature κ_1 was the most frequently selected predictor across the conditions. Our intuition was that high curvature – that is, high local variation of the surface normal – would result in higher task error. The presence of the κ_1 predictor in most of the predictive models generated is evidence in support of that claim.

In contrast, the principal curvature κ_2 predictor was not of nearly as much use in our predictive models. We expected κ_2 to be of greater significance – consider the error visualizations presented in Figure 6.9. In these examples, both locally flat and cylindrical ($\kappa_2 = 0$) surface regions generally have low error, but the convex and concave ($\kappa_2 \neq 0$) surface regions do not. Therefore, one may expect κ_2 would be an ideal predictor since it varies with error. However, since $\kappa_2 \neq 0$ only when $\kappa_1 \neq 0$, and κ_1 correlates well with error when curvature is especially high, κ_1 is the more useful predictor.

A few other predictors we selected also did not find much use. For instance, the centroid distance predictor was only of use in the radial abstraction style, where error appears to positively correlate with distance from a central axis formed at the intersection of the planar sections (see Figure 6.10). The mean curvature predictor was hardly utilized – since we have predictors for each individual principal curvature, there is little advantage to be had in combining them through addition to form a new predictor, as linear models do this already.

The abstraction-specific predictors, distance and angle difference, found use in all predictive models for abstractions except for the XYZ style. Since the XYZ style results in a "dense" geo-

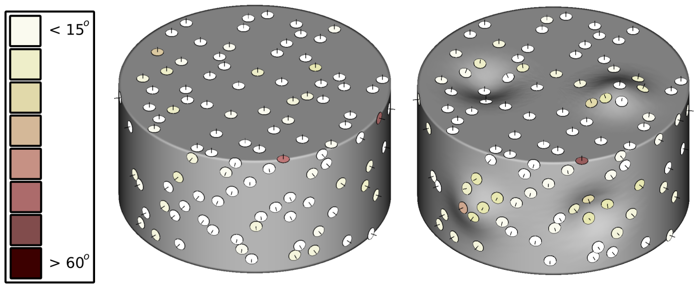


Figure 6.9: A visualization of task error for the *Cylinder* and *Bumpy* models and fixed view surface condition. For these surfaces, error is generally low for locally flat and locally cylindrical regions, but not for locally convex or concave regions.

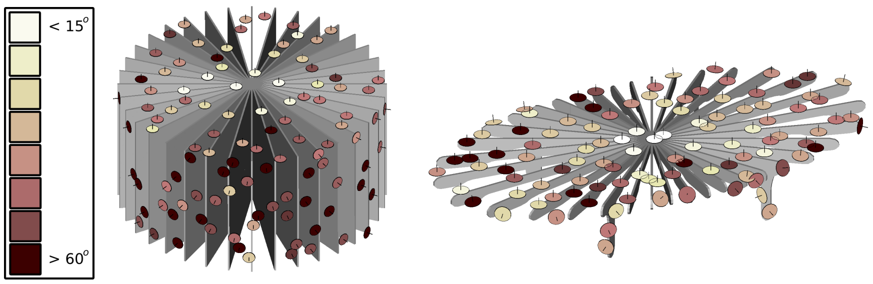


Figure 6.10: A visualization of task error for the *Cylinder* and *Table* models and fixed view radial condition. Note the positive correlation between error and distance from the central axis formed by this abstraction style.

metric abstraction where surface-to-abstraction distances are relatively very small and abstraction angle differences are generally 90 degrees, these predictors will have little if any correlation with error as a result. We believe these predictors are increasingly useful if the surface measurements can increase in variability, which happens naturally as the number of planar sections in the abstraction decreases.

Performance

We shift our perspective from predictors to the individual models, and examine their estimated prediction errors relative to base error. Appendix Figure 6.11 shows scatter plots for the predicted error vs observed error for all conditions in our study, and the final columns of Table 6.4 show mean absolute predicted error and mean absolute base error for each model. The final column expresses the percentage improvement over base error – a result of the predictors selected.

The least improvement in error was 12%, by the model for the rotated view radial condition. This result is to be expected, as *none* of the predictors we selected had a medium or higher correlation ($|r| > 0.3$) for this condition. Note that while there is a 19% improvement for the fixed view radial condition, the mean errors are significantly higher as well and provide greater room for improvement. In the scatter plot for the rotated view radial condition in Figure 6.11, the distribution of points is very narrow horizontally, indicating that our predictive model performs comparably to simply using mean error as the prediction.

The greatest improvements in error were 47% and 45% for the fixed and rotated view MPS conditions, which are substantial. On average, we estimate our predictive models will be approximately 10° from the true error. Considering the variation of error in data collected for the MPS abstraction, and that the base error is approximately 19°, our models make relatively good predictions and capture the geometric sources of error. Since the MPS abstraction style also had the highest base errors at approximately 19°, there was the greatest potential for improvement.

6.4.6 Improving abstractions

We ran a separate sub-study to exemplify the potential of our predictive models to improve surface perception by modifying the *process* of planar abstraction creation. Without much difficulty, we were able to incorporate the positively-valued abstraction distance predictor of the MPS predictive models into the MPS algorithm of McCrae et al. [99], so planes that minimize the average abstraction distance would be considered in the selection process (see Figure 6.12). Running a second smaller crowd-sourced study on AMT for these three models, we discovered significant reductions in mean task error: *Cup* by 52%, *Camel* by 18% and *Ant* by 14%.

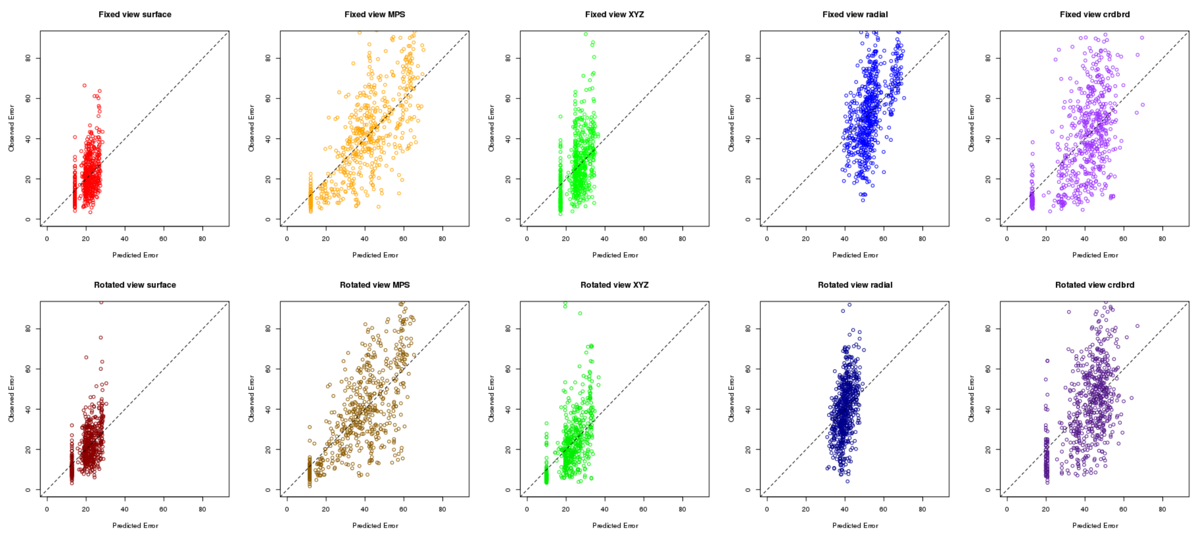


Figure 6.11: Scatter plots for each condition comparing predicted and observed task error. Each point represents an observation at a gauge position, the distance of the point from the dashed diagonal line indicates error of the linear model in prediction. The vertical distribution of points indicates the task error, while the horizontal distribution of points reveals the significance of the predictors used in the model – high horizontal variation also tends to correspond with increased improvement over base error rate. These scatter plots reveal that predictors yield the greatest improvement over base error for the MPS models, as the point distributions both tend most toward the diagonal line and exhibit high horizontal variation. For comparison, although surface model point distributions are even closer to the diagonal line and this indicates lower estimated prediction error, the lack of horizontal variation reveals improvements over base error are not as great.

6.5 Conclusion

We have conducted a large-scale study involving gauge orientation to define perceived surface normals, which expands upon previous methods by incorporating conditions where viewpoint rotation is encouraged. We considered a set of geometric measurements – many exhibited strong linear correlation with task error – and used these as predictors in linear models. The chosen parameters to the linear models themselves provide insight into the geometric causes of task error, and the models can be used as an automatic means to evaluate perception at a given point (for either surfaces, or for planar abstractions). Finally, we demonstrate in a second sub-study that the predictive models can be used to augment existing planar abstraction creation algorithms in order to improve perception of the abstracted surface.

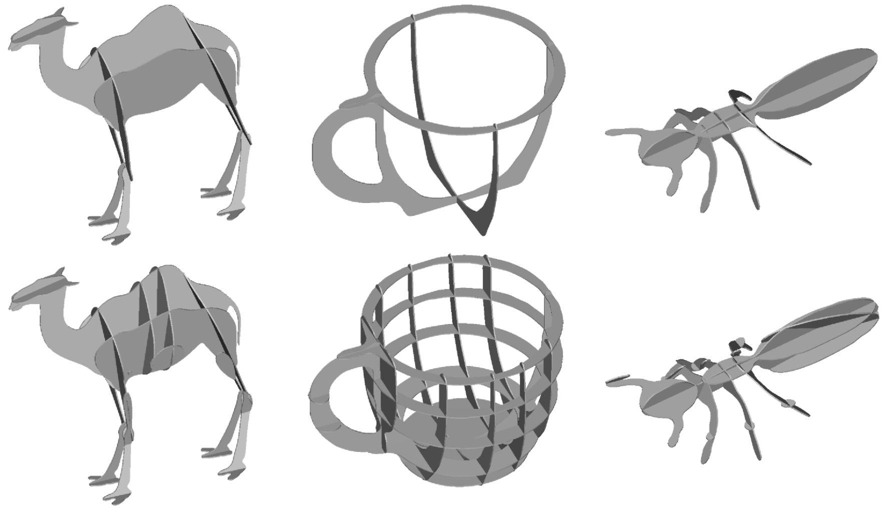


Figure 6.12: (Top) The MPS style abstraction of models *Camel*, *Cup* and *Ant*. (Bottom) Our "modified MPS" abstractions, which additionally seek to minimize the abstraction distance predictor.

Chapter 7

Applications

In Chapters 4 and 5, we explored the automatic and interactive creation of planar sections, and in Chapter 6 we explored the perception of abstracted surfaces. In this chapter, we consider many different applications for planar section representations that span both art and science.

It can be informally stated that planar section representations perform a form of *dimensionality reduction* – a subset or projection of 3D geometry onto (piecewise) 2D planes embedded in 3D. This observation alone can motivate the use of planar section representations in many potential applications. For some applications presented here, such as our "scanimations" (see Section 7.4), the set of input surfaces can be thought of as a single 4-dimensional geometric form, from which we derived a piecewise 2-dimensional form to be used as its representation.

7.1 Artistic Composition

Given a complicated input surface like the example shown in Figure 7.1, the planar section representation techniques presented can be applied to the individual elements that together constitute a more complex object. For the automobile example shown, we applied the automatic planar section creation algorithm detailed in Chapter 4 to each segment of the surface (wheels, steering wheel, body panels, etc.) in order to create the final results shown.

The space of artistic compositions is fully addressed through a combination of the approaches presented in Chapters 4 and 5. In Chapter 4, the automatic approach presented can be used to create planar sections from individual parts of a larger model. In 5, we show a variety of interactive workflows, which give an artist the freedom to manually select the planar sections from an existing template surface as well as the ability to deviate from the template surface if necessary. Finally, artistic compositions may require not just planar sections from a template surface but also planar sections that are created from scratch, and our interactive creation system was designed specifically for this purpose.

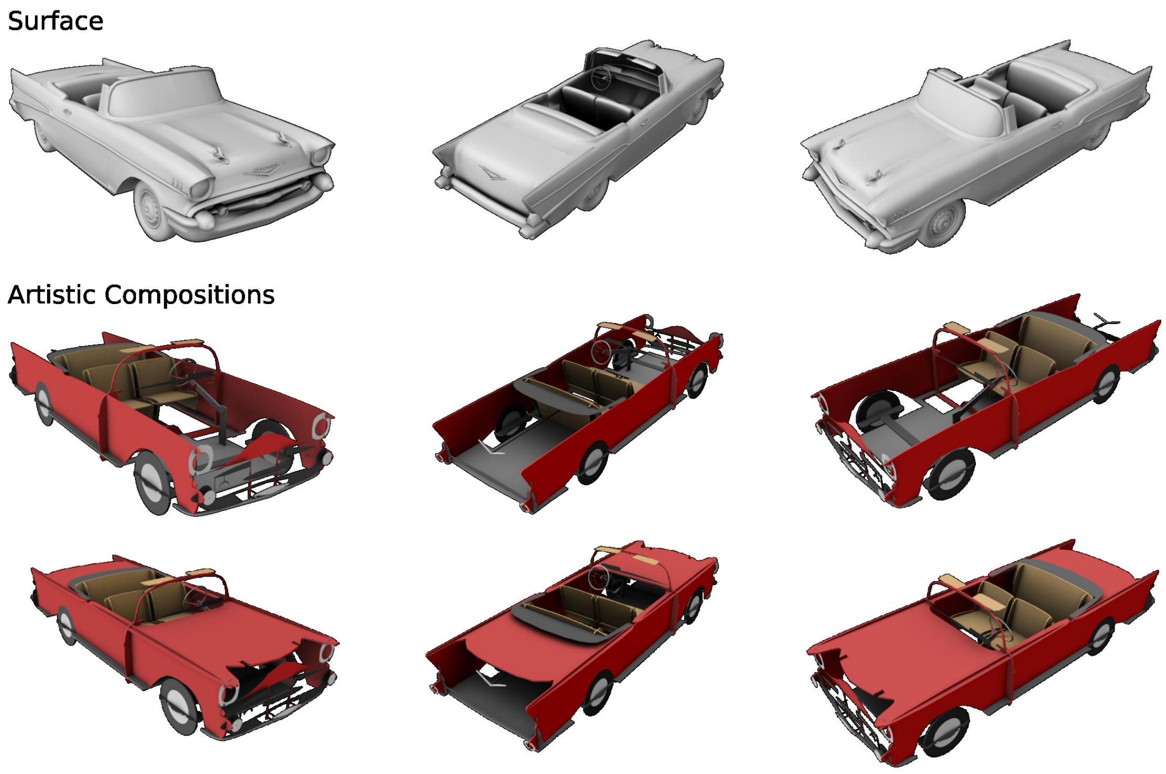


Figure 7.1: Artistic composition of a surface representing an automobile.

7.2 Surface Reconstruction

We consider applications involving computing a surface given only a planar section representation. Work by Liu et al. [90] details an algorithm to reconstruct surfaces from non-parallel curve networks, which is of great use. We discovered that this implementation worked well for some planar section representations we had already created (see Figure 7.2, top and middle rows).

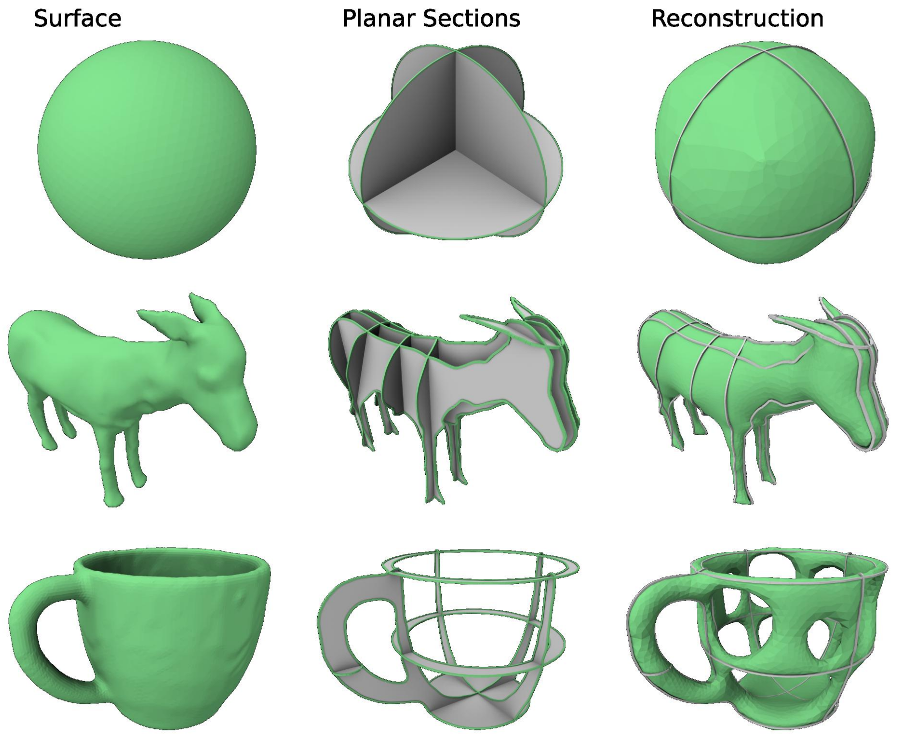


Figure 7.2: Surfaces representing a sphere, donkey and teacup (left) had the MPS algorithm applied to them to create planar section representations (centre), whose contours (rendered green) were used as input to a surface reconstruction algorithm whose output is shown (right). Note the difference in genus of the original teacup surface and its reconstruction.

For any surface with non-zero genus, for each hole a planar section needs to *capture* the hole (that is, the planar section must have a hole in it as well). However, even if this has been performed, existing methods [90] need further work as each chordless cycle of the planar contour network should have its interior completely filled by a surface patch (see Figure 7.2, bottom row). Doing so will ensure that the genus of the reconstructed surface is equal to that of the original surface. Still, we have shown the potential for existing methods to reconstruct the original 3D shape.

There are numerous applications possible given the ability to construct surfaces from planar section contours alone:

(i) *Surface compression.* Only storing the 2D planar section contour information reduces the amount of data needed to describe the surface, making for a more *compact* geometric representation.

(ii) *Surface simplification and selective removal.* The surfaces generated only capture detail that occurs at the planar section contours, any other surface detail is lost. Given a surface with an undesirable region or detail, if it is not captured by a planar section, the reconstructed surface will not contain it either.

(iii) *Surface modelling using planar sections.* One can use the system presented in Chapter 5 to interactively create planar section representations from scratch, and then apply the surface reconstruction algorithm of [90] on the planar sections to obtain a modelled surface. This effectively lets one use planar sections as a design primitive for the rapid modelling of surfaces.

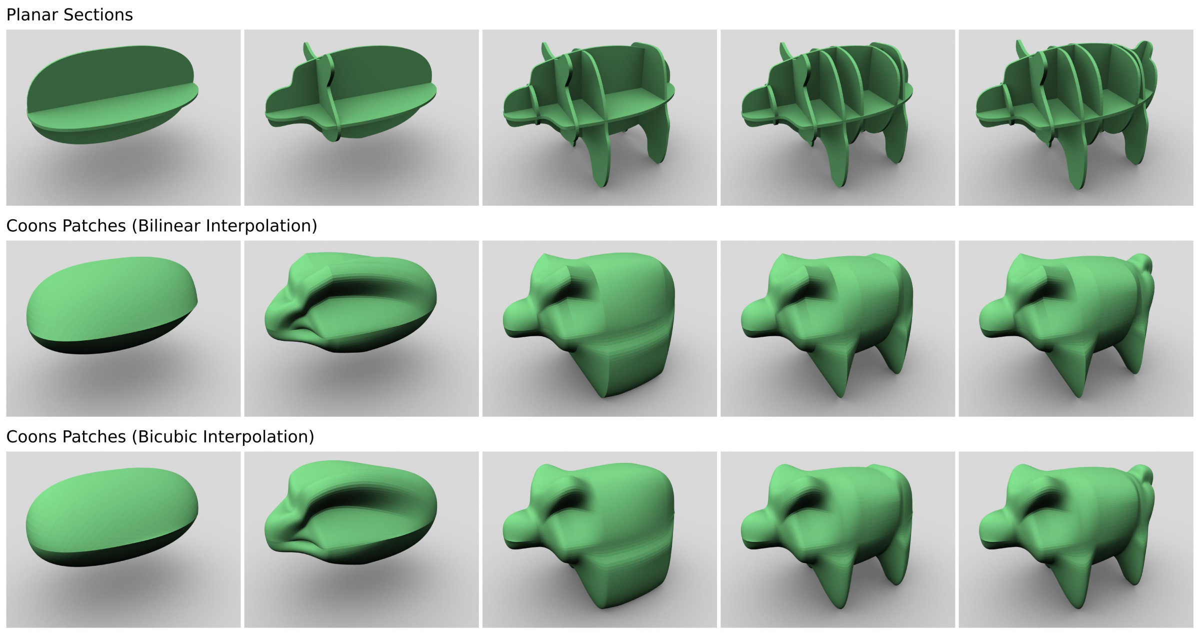

Planar Sections
Coons Patches (Bilinear Interpolation)
Coons Patches (Bicubic Interpolation)

Figure 7.3: Planar section contours form a boundary curve network. Using our interactive planar section system to create the planar sections from scratch, our system creates Coons patches for each planar section contour cycle. We show the surfaces resulting from either bilinear or bicubic interpolation.

To further demonstrate the use of planar sections for *surface modelling*, we implemented our own approach to generate surfaces (see Figure 7.3). Our surfacing algorithm performs the following steps:

(i) generate a graph for all the connected planar section contours, where contour intersections are vertices and contours are edges;

(ii) compute cycles for this graph (up to a maximum length);

(iii) as each contour rests on a plane, we keep only those cycles whose union of plane half-spaces has positive volume (these are cycles that should filled in);

(iv) for all cycles not consisting of exactly 4 contours, split or join contours as necessary;

(v) re-parameterize each of the 4 contours over the unit interval;

(vi) fit (bicubically blended) Coons patches to each of the re-parameterized 4-cycles.

Our surfacing approach can be combined with procedural modelling operations that make regular planar sections. We show an example of this in Figure 7.4.

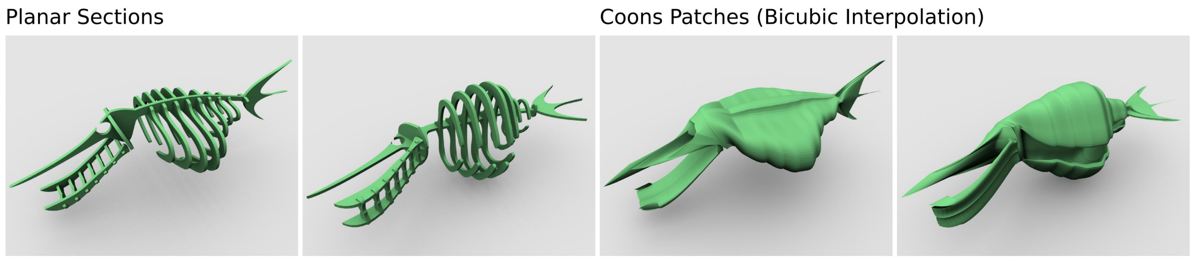


Figure 7.4: (Left) Two views of a planar section representation, which was created using linear procedural modelling operations for the rib cage and jawbone. (Right) A surface composed of Coons patches created from the planar sections.

7.3 Paper Statues and Puppets

Paper statues and puppets are appealing figure abstractions that can be physically constructed from planar section representations. A paper statue is made by physically printing, cutting and assembling the 2D contours of a planar section representation. Both the automatic and interactive systems presented automatically define the intersection slit between two planar sections for each section. The slits for two intersecting sections are made such that they are each cut from the opposite side, so they can be interlocked (see examples in Figure 7.5). Planar sections and slits are colour-coded to aid assembly.

Without any additional work, both our automatic and interactive approaches can be used to make real sculptures using any material that can be cut into the shape of planar sections. For instance, creations can be made from paper, cardboard, wood, or plastic. We show a collection of fabricated examples in Figure 7.5.

An interesting direction to consider are creations that do not have a static pose, such as *paper puppets*. Given an input surface that has been segmented to define the locations for joints, this information can be used to automatically create pivoting joints in locations such as elbows and knees to allow for articulation, as shown in Figure 7.6. Since the plane-surface segment intersection will create open curves, the open contours for each segment are smoothly closed together using a cubic Hermite curve, which preserves tangential continuity along the contour

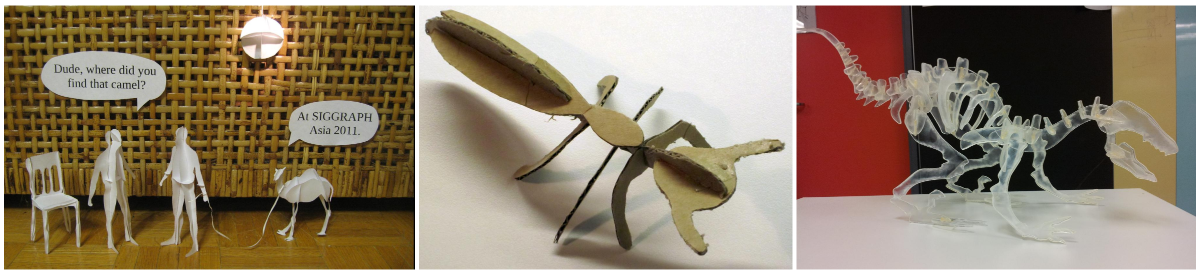


Figure 7.5: (Left) A scene consisting of a collection of objects made from paper. (Centre) A cardboard representation of an ant. (Right) An acrylic representation of a dinosaur.

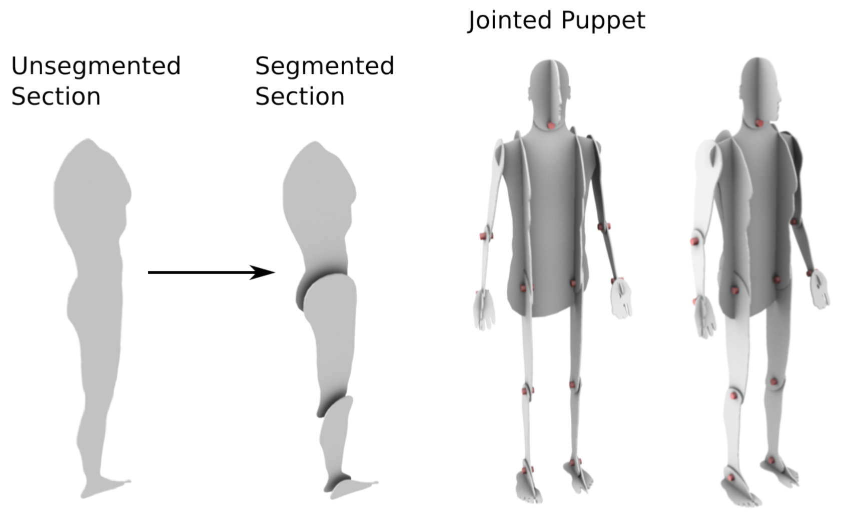


Figure 7.6: (Left) If an input surface is segmented, planar sections are created independently for each segment. (Right) Example paper puppet made using this approach. The pins that are used to connect planar sections at the joints are shown in red.

(see the segmented sections in Figure 7.6, left). These contours are physically pinned in the middle of their overlap, creating a hinge joint.

7.4 Scanimations (or "space-multiplexed surfaces")

We are given a set of input surfaces in an ordered sequence that represent a surface's changing shape over time (the set of surfaces define an animation). With the set of surfaces, we can create a single planar section representation that captures some geometry of each surface in the sequence, combining it all into a single, stationary object. With an appropriate viewing apparatus, the representation can be made to appear to animate (see Figure 7.7).

Regularly-spaced slices in a linear arrangement along a single axis is one way to create the scanimation. If there are n surfaces in the sequence, and we would like to view m slices of each, then the scanimation – that is, a planar section representation for the entire surface set – will consist of exactly nm planar sections. A section thickness of $\frac{1}{nm}$ is chosen, so that neighbouring sections touch and the size of each section is maximized.

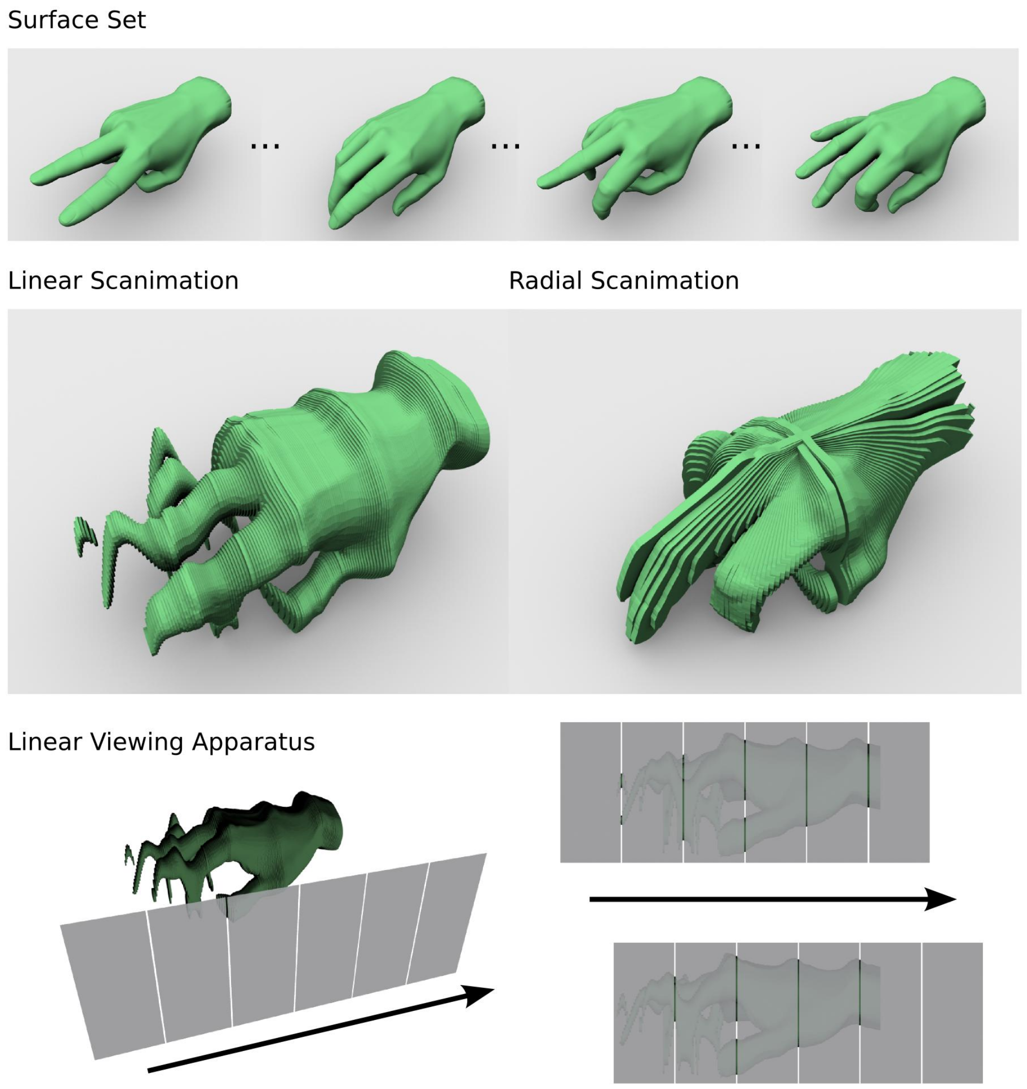


Figure 7.7: (Top) A set of surfaces, used as input. (Middle) Generated scanimations, using planar sections in linear and radial arrangements. (Bottom) A viewing apparatus for linearly-arranged scanimations consists of a plane with periodic slits, which when peered through and translated creates the animation effect. The arrows indicate the direction of translation of the apparatus.

Along the slicing axis, assume that the surfaces have been scaled to be within the interval of 0 and 1. To determine the set of m cutting plane positions for each surface $i \in \{0, \ldots, n-1\}$, we use cutting plane positions at each $j \in \{0, \ldots, m-1\}$ according to $\frac{nj+i}{nm}$. Intuitively, we are interleaving the planar sections from n surfaces together in space, taking m planar sections from each that are spaced exactly $\frac{1}{m}$ units apart, centre-line to centre-line. The value of m is selected so that the density of the sampling of each surface is suitable. Note that to do this for a radial arrangement, the approach is the same except we define an angle of rotation for each plane: $360° \left(\frac{nj+i}{nm}\right)$. The choice of n and m has a significant effect on the quality of the experience – if n and m are both large, the slits will be thinner and harder to see; if m is small, the sampling will be too sparse and the visual quality poor; and if n is small, the animation is short. For these reasons, there is always a trade-off in selecting values of n and m.

The appropriate viewing apparatus for a linear scanimation is a planar window with slits parallel to the sections (see Figure 7.7, bottom row). The window is moved along the slicing axis, and only the sides of sections representing the surface at a specific point in time will be visible. Since the sections are linearly arranged along a common axis, the ideal view of and through the apparatus is orthographic – if the slits were relatively thick, this can aid in collimating the light coming through the window.

While we considered radial arrangements, these are not as practical for a number of reasons. The apparatus for viewing could be stationary, while the object spins upon an axis inside it, however an apparatus with numerous slits requires a solution so that the direction of light travelling through each slit converges along the central axis. This would require some form of lens to bend the light. Secondly, since all planar sections intersect along a common axis, the lowest and highest points amongst all planar sections will always be visible regardless of angle. This can potentially severely limit the observed variation that occurs. For these reasons, we believe radial scanimations are not as practical as linear ones, but it was still interesting to explore the idea.

7.5 3D Shape Annotation

3D shape annotation is an important problem in scientific visualization and education. While it is easy to project 2D strokes onto the underlying 3D shape in a given view, often annotation occurs in the space proximal to the 3D shape to indicate dimensions or to avoid obscuring the region being annotated. In such scenarios [124], the annotation has no 3D representation and becomes meaningless when the current view is changed. An invisible planar section proxy, however, provides a natural set of 3D planes that annotations can be projected onto based on the alignment of the plane to the current view, and the proximity of the projected annotation to the section contour (see Figure 7.8). While one example in the figure illustrates the 3D-ness of the annotations with excessive foreshortening, in practice past an oblique angle of 30°, text annotations gradually fade or rotate about symmetric axes to improve readability.

The planar section representations of a model can be used as a hidden set of *canvas planes* that can be sketched upon, in order to annotate the model. A key benefit is that the user need not be concerned with the task of spatial placement of the planes – they can be derived automatically using our approach, and this process can happen behind the scenes.

Given a collection of canvas planes that annotations can be sketched upon, the view position and directions are used to determine which of these planes at any moment to use as the *active* canvas plane. Our approach is to select the plane that is most frontoparallel given the view

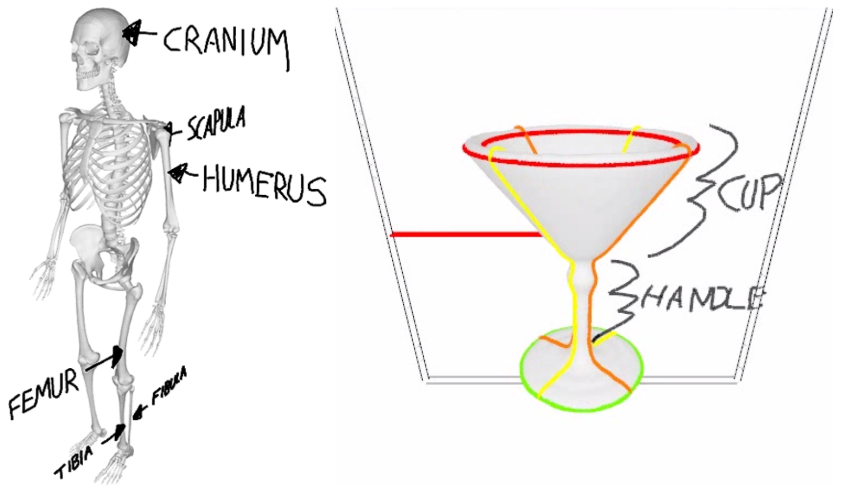


Figure 7.8: Two examples of annotations made onto planar canvases, which are derived from planar section representations of the surfaces shown.

direction, and if there are numerous planes equally frontoparallel, the plane whose distance is shortest with the view position is used. If the object contains an axis of symmetry, we can use this to improve the experience by having canvas planes rotate automatically with change in viewpoint, as shown in Figure 7.9 (top and middle rows).

To avoid the issue of too many annotations being displayed at once, which can be unappealing because they clutter or obscure other important details in the scene, we again use an approach based on the current view direction. The intuition behind our approach is that canvas planes that are not (mostly) front-facing should have their annotations hidden, or made less visible (see Figure 7.9, bottom row). Given θ, the angle between the canvas plane's normal axis and the view direction, we determine α (the transparency value) to draw the annotations. We choose α according to the following: for $0^\circ \leq \theta \leq 30^\circ$, $\alpha = 1.0$; for $30^\circ < \theta < 60^\circ$, α is linearly interpolated between 1.0 and 0.0; and finally for $60^\circ \leq \theta \leq 90^\circ$, $\alpha = 0.0$.

7.6 View-Independent "Poly-postors"

Planar section representations are also invaluable as *poly-postors* for the lightweight rendering of crowds [75] and other geometrically dense scenes. A texture is created for the front and back sides of each plane by rendering the model orthographically for each side. Our automatic planar section algorithm can be adapted to create effective poly-postors by selecting planes below the default threshold until the difference between the silhouette of the poly-postor and the input surface is deemed acceptable. Or as shown in the example (see Figure 7.10), we used our interactive system to quickly create a planar section representation of the surface.

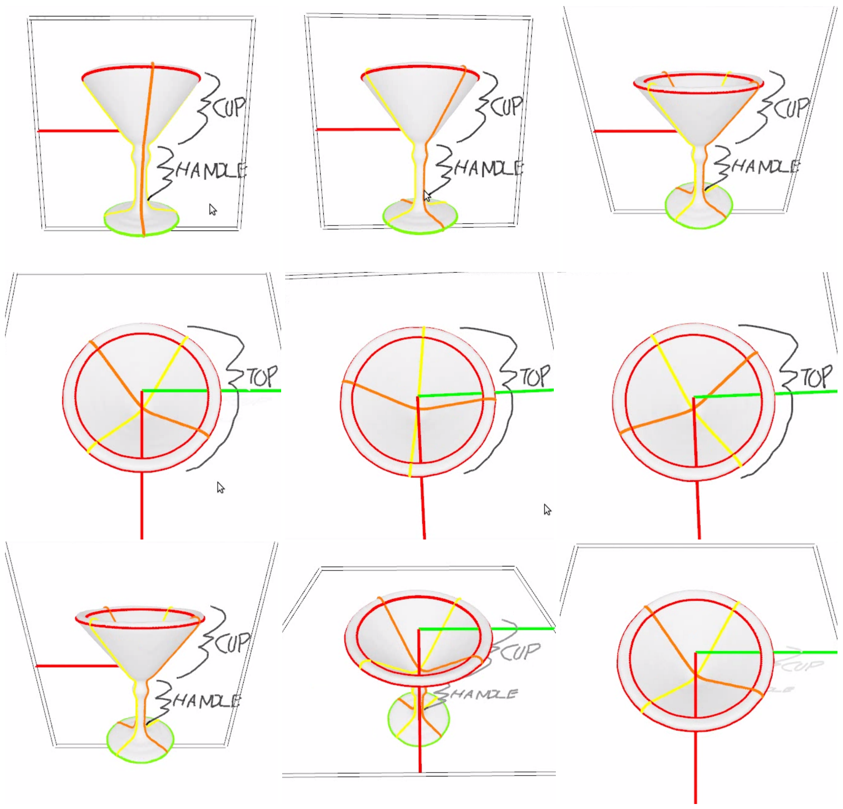


Figure 7.9: (Top row) Annotations on a plane where the model has a symmetry axis rotate with the viewer. (Middle row) An annotation on the top of the model also rotates accordingly. (Bottom row) Annotations are gradually hidden as the view direction and canvas plane normal become perpendicular.

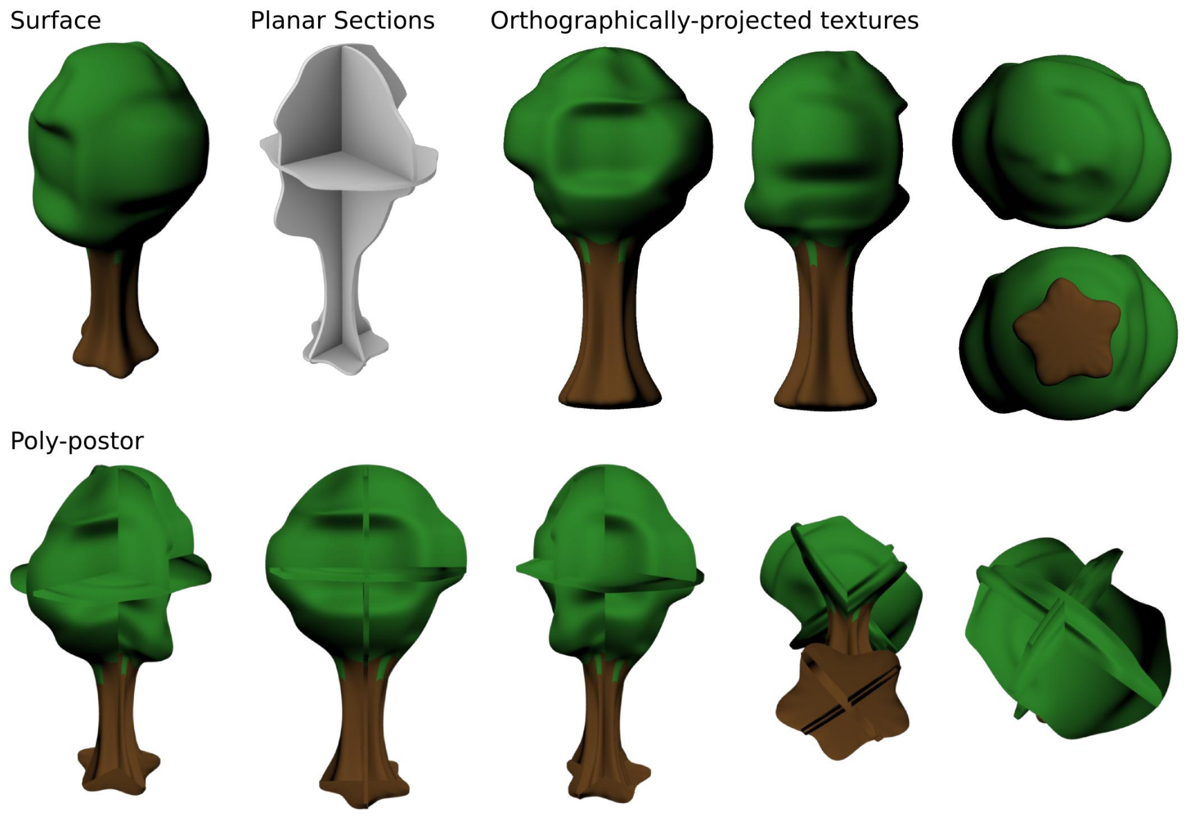


Figure 7.10: (Top row) Input surface, planar section representation, and a collection of orthogonally-projected textures obtained along each planar section normal direction. (Bottom row) Collection of views of the created poly-postor – a textured planar section representation.

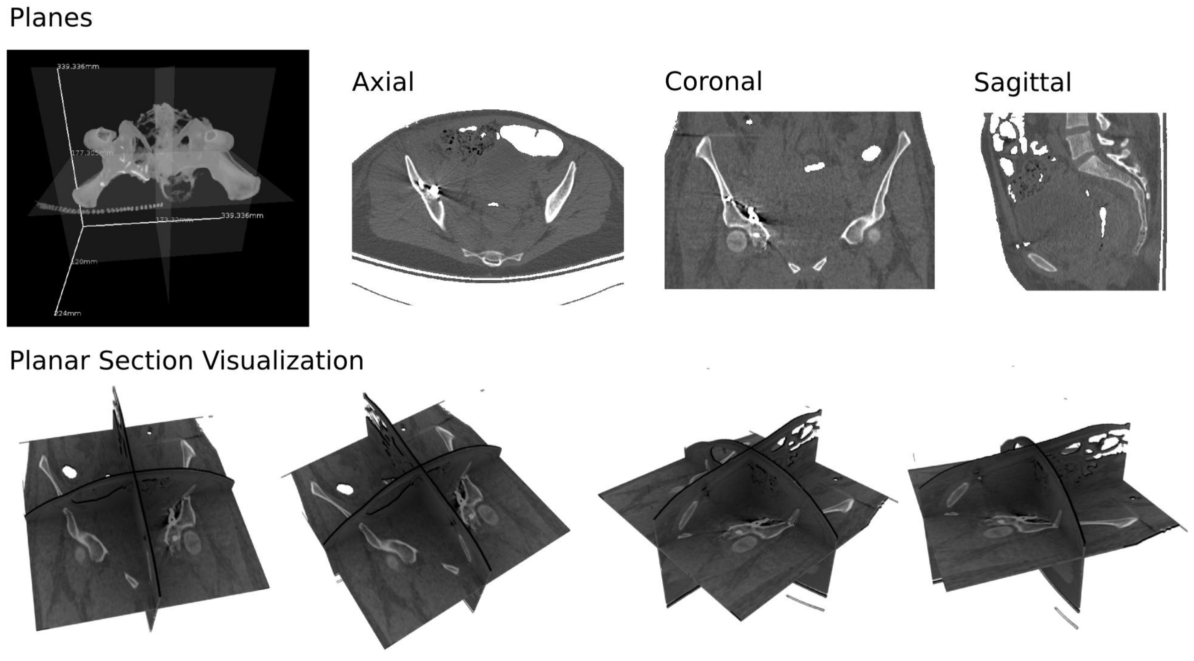


Figure 7.11: (Top row) A volumetric dataset, which consists of a CT scan of a pelvis. Three planes of interest have been specified along patient-aligned directions within an interactive application we created that loads DICOM-formatted data. (Bottom row) A textured planar section representation is created, which visualizes the data captured by the specified planes.

7.7 Volumetric Data Visualization

Due to their dimensionality, volumetric datasets are often large and special approaches may be required in order to do real-time rendering. Planar section representations can be used as a lightweight representation for real-time rendering. Similar to our poly-postor application, to represent a volumetric dataset we create a number of textured planar sections, but rather than coming from an orthographically-projected surface, values are sampled at voxel positions that intersect the plane directly. An example is shown in Figure 7.11.

Determining where to place planes is another important issue – broadly, the planes should cover all *features* within the dataset. However, the definition of and approach to determining the "features" will be specific to a given domain (e.g., a CT scan of a fractured bone), and is beyond the focus of this work. By representing the features – however they are detected – as a point, spindle or capsule in the plane parameter space, we can apply our plane selection algorithm to automatically compute the planes that cover the set of features. Once these planes have been determined, we sample along them to create 2D textures from the 3D dataset. Ideally, when sampling textures one would only do so for those voxels within proximity of a feature. This would reduce the amount of data presented to the user, and bring more attention to the data nearest the features (in Figure 7.11, we sampled only from voxels whose Hounsfield value (radiodensity) was above a specific threshold).

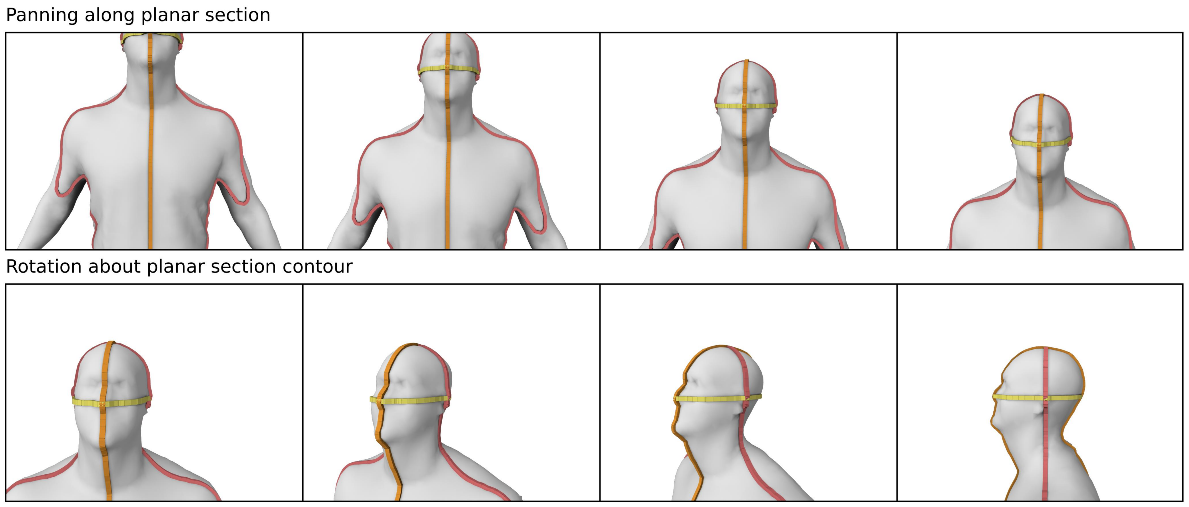

Panning along planar section
Rotation about planar section contour

Figure 7.12: View navigation using planar sections. (Top row) Moving the mouse performs a panning operation, where the view direction is parallel to the active planar section's surface normal. (Bottom row) Panning to the extent of the active planar section causes a controlled rotation of the viewpoint about a point on the planar section contour (red), which is kept centred in the viewport.

7.8 View Manipulation

The HoverCam approach to 3D navigation [76] and a derivative approach we later implemented [97] employs a controlled panning interaction where the viewpoint (the "camera") remains at a fixed distance from all surface geometry in the scene. Such a 3D navigation technique is effective when combined with traditional 2D input devices, and is easy to learn for a 3D software novice.

Our approach is inspired by the HoverCam technique, but instead of using the surface geometry, we instead use the planar section representation to constrain the camera (note that the surface and not the planar section representation is visible to the user throughout interaction). The interaction could be done in one of two ways:

(i) use the planar section contours to define a path network across the surface;

(ii) use the planar sections themselves as planar surfaces that the camera is bound to and can translate along.

The advantage of using the planar section contours and locking the camera onto them, is that rotation about the axis defined by the 3D contour is possible, whereas in the case of the HoverCam technique such a rotation is not possible about some curve on a surface – the view direction is generally always parallel to the surface normal at the centre of the viewport. Allowing rotation of the view direction would be of particular benefit for exploring a local surface feature that intersects the planar section contour.

An advantage to using the planar sections themselves as panning surfaces (see Figure 7.12), is that the surface normals for these planes will often correspond to ideal view directions for the surface. Thus given an input surface, in our approach the planar sections define a family of suggested views to aid navigation. When the camera is bound to a specific planar section and the camera reaches the extent of the planar section through panning, our system rotates to view the other side of the planar section about the extremal point of the planar section contour, while keeping its position centred in the viewport, as shown in Figure 7.12. Note that if during the rotation, a point on another planar section becomes within the set distance of the viewpoint, the view direction is rotated to face this new closest point at a rate proportionate to the rate of input and the camera becomes “fixed” to that planar section (this is quite similar to how the HoverCam technique maintains a fixed distance from all geometry).

7.9 Discussion

We have presented numerous practical applications for planar section representations. This list is not an exhaustive one, and we have indeed considered other applications. We present and discuss many of these ideas and extensions as future work in Chapter 8. We are confident that the reader may be able to come up with further applications that we did not foresee, where alternative shape representations are useful.

Chapter 8

Conclusion

In this chapter, we first discuss some limitations of our approaches and a number of directions for future work. We conclude the chapter with a summary of the thesis content and contributions.

8.1 Limitations

We categorize and discuss a number of limitations of the various approaches presented in this work. We discuss some limitations relating to the choice of a plane as the geometric primitive, followed by some limitations unique to our perception work.

Planar primitives and torsion

One of the main weaknesses of choosing a planar primitive is the inability to handle surfaces representing non-planar volumes or volumes with non-planar surface boundaries – generally thin helices and other coiled shapes, in which there is both non-zero curvature and torsion, are a worst case scenario. For this case, we propose the magnetic cut, which performs a local deformation of the surface near the plane, in order to bring the surface *through* the plane – essentially "planarizing" it. We believe that the resulting representations are reasonable, even though they are no longer completely faithful to the shape of the original surface. Other than deformation of the surface, the only remaining solution is to represent it with a non-planar primitive.

Geometric feature dependency

The quality of results of our automatic algorithmic approach is entirely dependent upon the collection of surface features obtained. Some surfaces are particularly smooth and featureless, and for example fall beneath the minimum pre-tuned thresholds we have in place to detect ridges and valleys. In the absence of detected features the plane-space is unpopulated, and the algorithm halts immediately since there are no points in this space beyond the threshold for selection. Having per-feature parameters adapt when no features are detected would be a welcome feature, but this is something we did not implement.

Articulations and global feature alignment

Highly-articulated surfaces where features do not line up on a common plane is another problem. Our algorithm parameters – weights that define the importance of each type of feature – are learned from models where the features generally have a structure such that they do line up on common planes, and the parameters are set accordingly. In the presence of a highly-articulated feature, this choice of weights may not be appropriate. To deal with such cases, our algorithm also has a single global parameter that defines when to halt – for surfaces where the algorithm is not choosing enough planes, one can simply lower this minimum threshold for plane selection. While lowering the selection threshold is not a complete solution, it can help to ensure a minimum number of planes are selected.

Perception evaluation using normal measurements

In our surface perception study, we employed an existing gauge figure setting task, where participants specified perceived surface normals. While the surface percept itself can be constructed with enough sampled normals (assuming the given normals define a consistent percept), a study where we could acquire perceived surface *positions* would have been beneficial. However, in the context of studying planar section representations, we found the design of such a study difficult for numerous reasons, but broadly: (i) it is imperative that we do not reveal any knowledge of the ground truth surface, and (ii) we must design a task that is simple enough to perform for users of Mechanical Turk (or whatever crowd-sourcing framework is used). We refer the reader to Section 6.2.5 that details our difficulty in designing study tasks that measured solely surface position as well as surface position and normal.

Error introduced by gauge figure task

The surface normal truly perceived by a participant and the normal that they supply to us by performing the gauge figure task will differ. In an ideal world, we devise a task where the perceived normal could be brought from the mind to the machine precisely, but this is not the case. Aside from issues in the physical world, such as the participant's coordination, or the capabilities of their input and display hardware, there are potential sources of error caused purely by the design of the gauge figure task itself that could be considered.

Gauges take up a small space in the viewport. The small size may make the exact orientation of the gauge difficult to perceive – consider the effect of doubling the size of the gauge, one may more easily perceive the intended orientation. Further, the image of the gauge is a raster graphics image, a discrete sampling of the actual gauge shape, one could consider that there are a range of angles where the image of the gauge appears the same.

The aspect ratio of the gauge disc is related to the cosine of the angle between the gauge and view directions. Thus for a gauge that is viewed frontoparallel, and where the angle is changed by a fixed amount, there will be less variation in its appearance compared to when it is highly slanted (edge-on). A related issue is that for a highly-slanted gauge, adjustment is principally along 1 dimension (tilt) rather than 2 (tilt and slant) – this change in dimensionality may be a cause of non-uniformity in error across gauges with varying slant.

8.2 Future Work

We consider a wide range of different ideas and directions one might proceed in to extend or improve upon the work presented in the thesis.

Considering fabrication constraints in our algorithmic approach

Our algorithmic approach does not incorporate any feasibility constraints for fabrication – our focus in this original work was to develop planar section representations as a shape abstraction. For instance, planar section representations created with our automatic method have planar sections that intersect non-orthogonally, and form cycles without parallel intersections. That said, our algorithm is highly extensible and could be modified to incorporate fabrication constraints – for instance, by assigning a value of zero to all plane-space positions where addition of the plane makes fabrication impossible. We did not do this in the original work, because this was not the original goal. We leave the addition of fabrication constraints to our automatic approach as future work.

Creating a richer library of procedural modelling operations

In our interactive system, we considered two procedural modelling operations (branching and linear section arrangements). These operations save the designer substantial time, and also produce a professional-looking result since sections are placed with in a regularized pattern. One could come up with many more operations (e.g, a radial arrangement that regularly duplicates sections along an entire contour), but we leave this as future work.

Creating a new surface perception task

In our main surface perception study, we only considered a gauge figure task that provides us with a perceived normal. It would be interesting to produce other tasks specifically to evaluate the abstracted surface percept, and for properties other than surface normal (such as surface position along a curve). We experimented with a position task, and a combined task where both position and orientation are specified, but we encountered some inherent difficulties with these alternate tasks. We leave the successful design of new tasks and their use in new studies as future work.

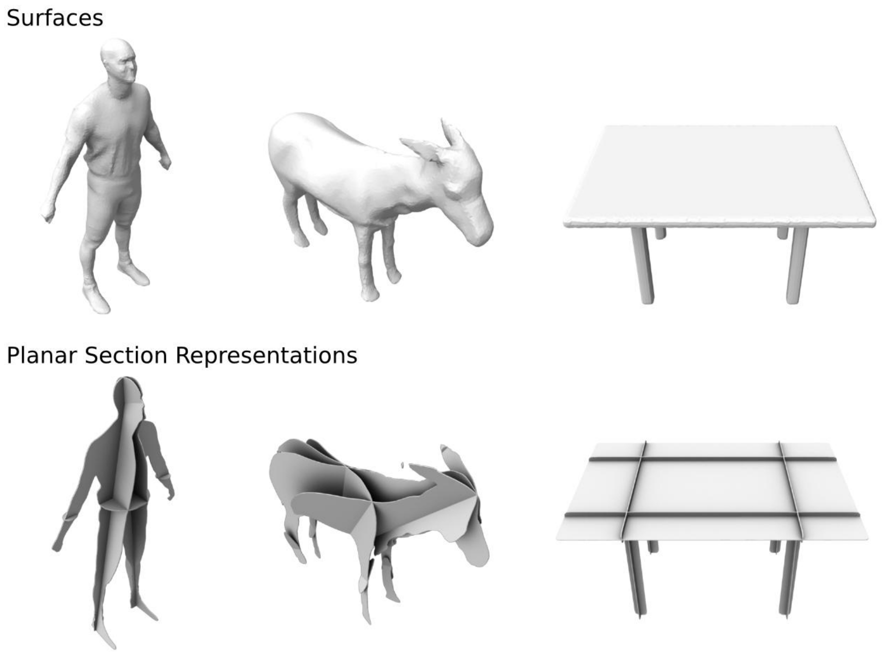


Figure 8.1: (Top row) Input surfaces and (bottom row) planar section representations of a *human*, *donkey*, and *table*. The planar sections could potentially be used as a means to describe shape.

Shape Descriptor

Planar section representations capture an *essence* of a shape. Shape descriptors aim to provide high-level descriptions of objects that are suitable for the purpose of object classification. Planar section representations themselves can be thought of as a shape descriptor because they also capture the high-level structure of a shape, as well as object features. To perform shape analysis using a planar section representation, of particular importance are:

(i) the global structure of planes used to define each section;

(ii) the shape of the curve for each planar section contour.

To illustrate this idea, we consider an example set of surfaces representing a *human*, *donkey*, and *table* (see Figure 8.1). Examining the global structure of the planes for each representation, the human (a biped) can be differentiated from the donkey and table. Examining the planar contours themselves, the table consists of contours that consist of straight, angular lines that vary sharply, whereas the planar contours of the donkey consist mostly of smoother lines without sharp variation; this observation allows for differentiation between the donkey and the table. Potentially, planar section representations could be used as a shape descriptor to aid in object classification. We leave further development of this idea as future work.

Developable sections as primitives

In this work, we have focused on the use of planar primitives for shape representation, where the curvature is everywhere zero. We consider as future work the use of developable surface primitives, since they are also amenable to fabrication.

As we have shown, planar sections can be physically assembled by cutting straight, rectangular slits into each pair so that they interlock. The shape of the slits was due to the fact that the principal curvatures of the planar section surfaces are everywhere zero; for any two non-identical planes that intersect, the intersection is a line. However in the case of developable surfaces, only the Gaussian curvature must be zero. Since slits are defined using the curve of intersection of two developable surfaces, they are not necessarily straight lines.

We consider six geometric configurations to explore cases where developable surfaces can and cannot be assembled, using the established technique of cutting slits and sliding sections together. Recall that slits must be defined along the curves of intersection of two sections in their assembled configuration. Each of the following cases is visualized in Figure 8.2:

(i) one planar section and one developable section, whose slits are defined by straight lines;

(ii) two developable sections, whose slits are defined by straight lines;

(iii) one planar section and one developable section, whose slits are defined by circular arcs;

(iv) two developable sections, whose slits are defined by helices;

(v) one planar section and one developable section, whose slits are defined by elliptical arcs;

(vi) two developable sections, whose slits are defined by elliptical arcs.

In the first case (i), the planar section intersects the developable section along a line where the curvature is zero. In the second case (ii), non-zero curvature only occurs along directions that are orthogonal to the slit directions. In both cases, curvature orthogonal to the slit direction does not pose a problem for assembly.

In the third case (iii), the slit is defined as a circular arc on the planar section, with a curvature that matches the maximum principal curvature of the developable section. It is easy to realize that such a configuration is assemblable: keeping one of the two sections fixed, once the slit openings are aligned, the other can be rotated into the assembled position. Note the importance of *constant curvature* of the slit, which ensures that the two sections do not collide during any instance of the assembly motion.

In the fourth case (iv), the slit is defined as a helix that lies within a cylinder. Slits defined along the helix thus have *constant curvature*. This configuration is also assemblable: keeping the cylinder fixed and with the helicoid (with some positive thickness) positioned above, gradually lower the helicoid while rotating it about the cylindrical axis until the shapes are fully interlocked.

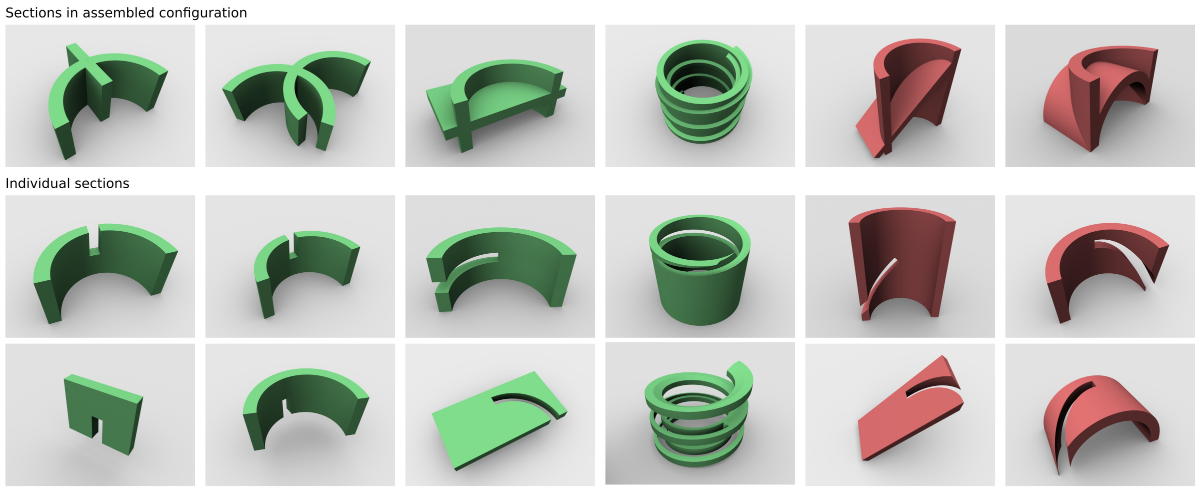


Figure 8.2: (Top) We show developable sections used in both assemblable and non-assemblable configurations. Those coloured green can be assembled, those coloured red cannot. (Bottom) We show each section of the assembly so that the slits that would be cut into each section are made visible.

In the fifth case (v), we consider the intersection of the cylinder with a plane along a direction which is not orthogonal to a principal curvature direction, which yields an ellipse (where the major and minor axis diameters are not equal). Since the slits cut into the planar section and developable section for assembly must then be elliptical arcs, they have *varying curvature*. However, in order for a section to slide along a slit for assembly, the slit must have *constant curvature*. Thus, this configuration cannot be assembled.

In the sixth and final case (vi), let the equation for these surfaces be $x^2 + y^2 = 1$ and $x^2 + z^2 = 1$. Their intersection is given by $y^2 = z^2$, which forms two closed planar curves C_1 and C_2 (see Figure 8.3). For instance, C_1 intersects the four points: $(0, 1, 1)$, $(1, 0, 0)$, $(0, -1, -1)$ and $(-1, 0, 0)$. Both C_1 and C_2 are ellipses with a major axis diameter of $\sqrt{8} \approxeq 2.83$ and minor axis diameter of 2. Since a slit is defined along a segment of either C_1 or C_2, the slits have *varying curvature*. However, in order for a section to slide along a slit for assembly, the slit must have *constant curvature*. This configuration also cannot be assembled.

Suggestive interface

Beyond the computation and visualization of mechanical stresses, our existing interface detailed in Chapter 5 could be improved by making it *suggestive* – that is, compute and suggest a set of potential geometric improvements for the representation in order to better cope with mechanical stresses. For instance, if a thin region of a planar section is likely to fracture if the representation is fabricated, the interface could offer a range of suggestions for improvement that the user could select (e.g., widening the narrow region by modifying the planar section contour, or adding an additional planar section to act as a brace).

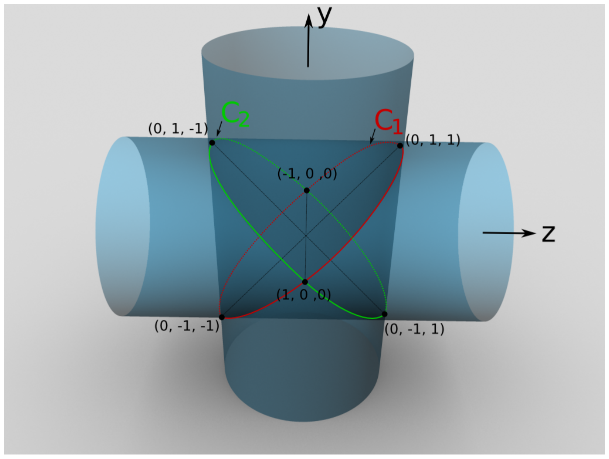


Figure 8.3: A visualization of the intersection of two cylinders given by equations $x^2 + y^2 = 1$ and $x^2 + z^2 = 1$, yielding planar curves C_1 and C_2. Both C_1 and C_2 are ellipses with varying curvature.

Constructive solid geometry and feature-based operations

Constructive solid geometry (CSG) is a technique that allows the intuitive modelling of solids. An elementary set of Boolean operations are defined (union, difference and intersection) and are used to compose a shape from a set of geometric primitives (e.g., spheres, cylinders, cuboids).

We consider the analogue of constructive solid geometry as it relates to planar sections. In our case, the geometric primitives are the individual planar sections. Treating each planar section is a solid with non-zero thickness, we can immediately apply the union, difference and intersection operations as they work traditionally. In fact, the difference operation can be used exactly in this way to model the slits that are cut into each solid planar section – slide one of the planar sections half-way off the other along the slit direction, and then apply the difference operation twice, reversing the order of the parameters.

We now consider other potential operations that may be defined, which make specific sense in the context of planar section representations. For instance, what might it mean to *add* two planar section representations together? We may for example rely on upon two slits being defined that determine how the two representations come together, which was our approach for our procedural modelling operations in Chapter 7. Considering a drag-and-drop style interaction (like that of MeshMixer [125]), where the slits to join the representations are defined automatically, may be an interesting point for future work since planar section representations could be "added" using a method that is easy to learn and use.

What about the notion of *multiplying* two planar section representations together? An idea for a *product* operation might be to replace *each* planar section of the first representation with the *entire set* of planar sections of a second representation. This would need to be done in a way that preserved the connectivity of the first representation. Thus for two representations with m and n planar sections respectively, the "product" of these representations would have

mn planar sections. Note that defined this way, the product operator is not symmetric.

As planar section representations act as perceptual stand-ins for 3D shape, we consider the notion of *perceptual CSG operations*. Each planar section is perceptually meaningful because it captures some subset of the original shape's features. We may then consider operations such as union, intersect and difference that operate on sets of the shape features, rather than the planar section representations themselves. As a concrete example, consider two planar section representations of the same shape, but that capture different sets of features of the shape $\mathcal{F}_1$ and $\mathcal{F}_2$. An intersection operation between these two representations would yield a new representation consisting of only those planar sections that capture shape features common to both $\mathcal{F}_1$ and $\mathcal{F}_2$ – that is, those features $\mathcal{F}$ where $\mathcal{F} = \mathcal{F}_1 \cap \mathcal{F}_2$. Analogous operations could be defined for difference ($\mathcal{F} = \mathcal{F}_1 \setminus \mathcal{F}_2$) and union ($\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$).

Articulation

In this work, concerning movable or *articulating* planar section representations, we only considered joints where two parallel planar sections were joined using a pin. The pin provides an axis of rotation to allow one degree of freedom for articulation. It would be interesting to extend the system by considering the broader selection of joints, hinges or other means to fasten planar sections together, where some manner of articulation is possible. One could additionally perform physical simulation so that when external forces are applied to a planar section, an articulated pose in response to those forces is computed. With such a system, one could potentially create robotic or animatronic devices. Deriving from a surface the planar section representation with the set of joints already in place would be another interesting direction for future work.

8.3 Summary

Overall, the dissertation demonstrates an effective workflow for studying scientifically ill-posed problems inspired by artistic practice: we perform free-form user studies to first show that the problem is at least well-understood, and there are consistent responses by humans. We make insights based on these consistent responses that allow the problem to be posed mathematically. We propose a solution, and finally perform a follow-up study to provide evidence that our solution is valid. We employed this exact workflow in each of Chapters 4, 5 and 6.

In Chapter 4, we developed a computational approach to create geometric feature-guided abstractions of man-made shapes using only a handful of planar sections. The formulation is flexible and directly incorporates a variety of shape features. In the absence of any suitable computational model of human perception, we learned the relative preference weights for various features using our user study data. We tested our algorithm on a variety of input models, and presented robustness results with respect to noise and varying mesh resolution. Finally, in the course of a conducted survey, we observed that both recognition quality and efficiency of abstracted models are comparable to those of the source models.

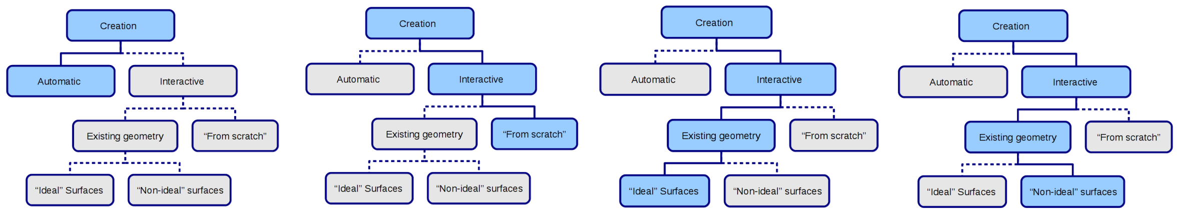


Figure 8.4: Four possible approaches to creating planar section representations, each of which we addressed in this work.

In Chapter 5, we explored the remaining path by which one can create planar section representations – interactive creation. We performed an initial user study where participants created a number of 3D planar curve drawings of freeform and reference objects. Through analysis, we arrived at a set of design choices that would guide our implementation of an interactive system catering specifically to the task of creating fabricable planar section representations. We introduce a novel interaction technique to create sections. We also adapt this technique to the creation of planar sections using an input surface as a template. We also consider the case of a non-ideal surface template, and propose the use of magnetic cuts that perform a local deformation of the surface near the cutting plane. To streamline fabrication, tests for stability, physical assembly, and connectedness are performed. In addition, a physical simulation approach is integrated that computes stress along regularly-sampled cross-sections of models, to ensure fabrications can withstand real-world forces. Our evaluation was two-fold: we verified that physical simulation predictions were consistent with real-world experiments, and provided our system to 12 users who with minimal instruction quickly learned to use the system to create interesting models. Using a laser cutter and acrylic sheets, we fabricated numerous examples, some of which also serve as functional objects.

In Chapter 6, we presented a large crowd-sourced study to evaluate and compare different planar section creation algorithms in terms of perception, particularly in the context of a gauge figure task where orientation defined a perceived surface normal. The study revealed numerous correlations of the error distributions to various geometric properties, with the original surface and between the original surface and abstraction. Based on the findings, we also propose ideas to improve the current MPS abstraction, and as first results, use another user study to verify the improvement in perception error on the newly abstracted test models.

In Chapter 7, we proposed many possible uses for our planar section representations, spanning areas such as: lightweight rendering, data visualization, shape compression and remeshing, 3D annotation, statues and jointed puppets, guided view manipulation, and animation.

8.4 Thesis

The work in this dissertation proves each claim made in the thesis (Section 1.2) is true. Recall these three claims made in the thesis regarding planar section representations:

(i) they are effective abstractions of shape,

(ii) they can be easily constructed interactively or algorithmically,

(iii) and they are applicable to a variety of domains in art and science.

To address claim (i), we refer to the results obtained in Chapter 4. There, we discovered that a minimal number of planar sections can be used to create a *recognizable* abstraction. We further validated this claim with a second user study aimed at evaluating how recognizable were objects represented by various abstraction styles, finding that both human-authored and algorithmically-created abstractions were as recognizable as the input surfaces themselves. Though this is sufficient to demonstrate that planar section representations are effective shape abstractions, in Chapter 6 we went a step further exploring how well humans perceived the fine-grained details of an abstracted surface (in particular, perception of local orientation). We discovered that some abstraction styles (such as the XYZ style) were perceived as comparable to the ground truth surface indicating some abstractions are effective for representing even such fine-grained surface detail. In addition, we discovered geometric sources of perceptual error, and demonstrated the use of this knowledge toward improving the perception of the MPS abstraction style.

To address claim (ii), we proposed two systems in Chapters 4 and 5. The system in Chapter 4 creates planar sections algorithmically within seconds, requiring no user input. The interactive system presented in Chapter 5 allows the construction of planar section models in a number of ways: from scratch, oversketching using existing images, and using existing template surfaces (even allowing for local deformations) – all within a unified workflow. Tests to ensure fabricability are well-integrated in this system, and models can be easily exported into a format directly suitable for fabrication. It is our belief that both of these systems, for their respective purposes, make planar section representation creation easy (and also fun).

Prior to our work, planar section representations have found application in architecture, decorative art, functional furnishings, and scientific visualization, due to their geometric simplicity, ease and low cost of manufacture. To address claim (iii), we refer not only to these prior examples but the numerous applications we have suggested, implemented and experimented with, in Chapter 7. The applications we presented span a wide range of fields, from the scientific (lightweight volumetric data rendering, 3D annotation, surface compression, simplification and modelling) to the artistic (statues, puppets, compositions). In addition, applications such as the scanimations, 3D annotation, and our approach to view manipulation we believe to be novel. The wide range of applications demonstrated, by no means an exhaustive list, provide ample proof of the applicability of planar section representations to various problems in art and science.

This work comprises a thorough exploration of various aspects of planar section representations of 3D shape: automatic and interactive creation, perception and application. We have provided both qualitative and quantitative perceptual validation that planar section representations are indeed effective proxies of 3D shape, which can be easily constructed, and are useful in both artistic and scientific settings.

Appendix A

Physical Simulation Approach

Note: the formulation, implementation and description in this section is the work of Nobuyuki Umetani, with whom I collaborated on the interactive planar section modelling work presented in Chapter 5. Concerning the physical simulation portion, my own work was in the physical experiments conducted, and implementation of interface components such as stress visualizations and the interactive placement of weights. The details of the physical simulation approach are included here for the sake of completion.

The physical simulation consists of two parts. The first is *inter-plate analysis*, and the second is *within-plate analysis*. For the inter-plate analysis, we assume the plates are undeformable and unbreakable rigid bodies, and compute the forces on the joints and contact points. Then, for the within-plate analysis, we compute the stress for each individual plate using the forces computed in the inter-plate analysis.

A.1 Inter-plate Analysis

In the inter-plate analysis, we assume that each plate is an unbreakable rigid body and we compute a balance of the forces acting on these plates. A plate is subject to forces from external loads, joints, contact points that touch the ground, and internal gravitational force. We use a technique that is described in [156] to analyze static equilibrium of these forces. Here, we only we briefly overview this technique to describe our overall framework, please refer to the original paper for more details.

We allow each rigid plate to translate and rotate when the gravitational and external forces are applied. A point $\mathbf{p} \in \mathbb{R}^3$ on the rigid plate is moved as $\hat{\mathbf{p}}(\mathbf{R}, \mathbf{u}) = \mathbf{R}(\mathbf{p} - \mathbf{p}_{cg}) + \mathbf{p}_{cg} + \mathbf{u}$, where $\mathbf{p}_{cg} \in \mathbb{R}^3$ is the center of gravity of the plate, $\mathbf{u} \in \mathbb{R}$ is the translation and $\mathbf{R} \in SO(3)$ is the rotation matrix.

The translation and rotation for each plate is obtained by energy minimization. The total energy of the rigid plates is computed as the sum of potential energy as

$$E = -\sum_{i \in P} m\mathbf{g} \cdot \mathbf{u}^i - \sum_{i \in Q} (\hat{\mathbf{q}}^{ex} - \mathbf{q}^{ex}) \cdot \mathbf{f}^{ex}, \tag{A.1}$$

where P is the set of plates, Q is the set of loading forces $\hat{\mathbf{q}}^{ex}$ and $\mathbf{q}^{ex}$ is the loading point location before and after the plates' movement, and $\mathbf{f}^{ex}$ is the loading force vector. The joint forces are obtained as constraint forces. At each joint, we apply a joint constraint that prohibits the relative displacement of each plate and relative rotation of the plates. We use a joint constraint that assumes a joint point that is located at the end point of the slit. Let two rigid plates A and B be connected at a joint point $\mathbf{p}_J$. The relative position differences of joint positions computed from plates A and B after movements can be written as:

$$\begin{aligned} C_1 &= \{\mathbf{R}_A(\mathbf{p}_J - \mathbf{p}_{cg}) + \mathbf{p}_{cg} + \mathbf{u}_A\} - \\ &\quad \{\mathbf{R}_B(\mathbf{p}_J - \mathbf{p}_{cg}) + \mathbf{p}_{cg} + \mathbf{u}_B\}. \end{aligned} \tag{A.2}$$

The constraint $C_1 = 0$ means that plates A and B are connected at the joint point. Furthermore, the rotations of A and B should be the same. The amount plate A is rotated relative to plate B can be written as:

$$C_2 = \mathrm{vect}(\mathbf{R}_A \mathbf{R}_B^T), \tag{A.3}$$

where vect$(\cdot)$ denotes the axis-angle vector of a rotation matrix. The constraint $C_2 = 0$ gives the same rotation of plates A and B. The contact points are described by a constraint such that the penetration distance against the ground plane is zero. We enforce the joint's connection using penalty-based constraints. We minimize the total energy with these constraints using Newton's method, and obtain the constraint forces.

A.2 Within-plate Analysis

After the joint force and contact forces are computed, we analyze the stress within the plate based on the computed forces. We use cross-sectional structural analysis [157] to quickly estimate the stress within a plate. The cross-sectional structural analysis provides an estimation of stress at the weak locations in a manner sufficiently accurate for interactive modeling. Given a cross-section in the plate, we compute the bending moment acting on the cross-section. We compute the stress in the cross-section using beam theory.

From the inter-plate analysis, we computed forces at the joint points and contact points. We call these points altogether *force points* $\mathbf{q}^i$, $i \in K$, where K is a set of force points on a plate. Each force point has a linear force $\mathbf{f}$ and torque $\boldsymbol{\tau}$. The contact points do not transfer torque, so the torque at contact points is zero. From the linear force equilibrium, the sum of the linear forces becomes zero: $m\mathbf{g} + \sum_{i \in K} \mathbf{f}^i = 0$ where m is the mass of the plate. From the

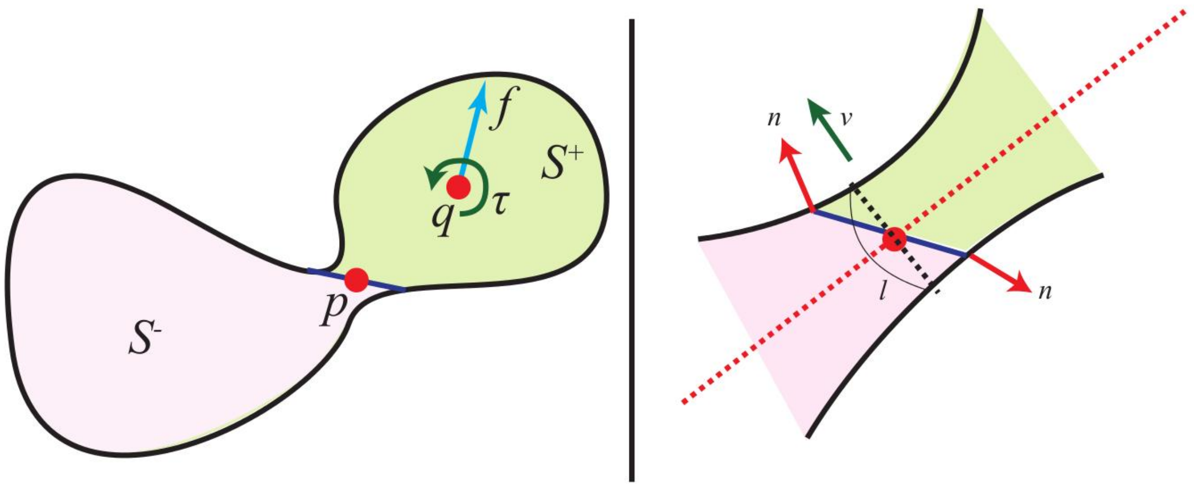


Figure A.1: Visualization of our cross-sectional structural analysis, based on beam theory.

moment equilibrium, the moment at the given point in the plate $\mathbf{p}$ becomes zero:

$$m\mathbf{g} \times (\mathbf{p}_{cg} - \mathbf{p}) + \sum_{i \in K} \mathbf{f}^i \times (\mathbf{q}^i - \mathbf{p}) + \boldsymbol{\tau}^i = 0. \tag{A.4}$$

Assuming the plate does not form a loop, the plate is divided into two loops S^+ and S^- by the cross-section (see Figure A.1 (left)). We take the center of the cross-section $\mathbf{p}$ and compute the bending moment acting on it. Let K^+ be the set of force points belonging to S^+. The rotational moment that S^+ produces along the cross-section can be written as:

$$\boldsymbol{\tau}^+(\mathbf{p}) = m^+\mathbf{g} \times (\mathbf{p}^+_{cg} - \mathbf{p}) + \sum_{i \in K^+} \mathbf{f}^i \times (\mathbf{q}^i - \mathbf{p}) + \boldsymbol{\tau}^i, \tag{A.5}$$

where m^+ is the mass of S^+ and $\mathbf{p}^+_{cg}$ is the center of gravity for the loop. The torque $\boldsymbol{\tau}^-$ on the other side is computed in the same manner for the other loop. Note that the sum of the two moments $\boldsymbol{\tau}^+$ and $\boldsymbol{\tau}^-$ becomes zero since it results in Equation (A.4).

There are stresses that occur in response to the bending moments. To compute the stress along a cross-section from its moment, we adopt beam theory [152]. Since the fracturing region is usually slender, we can approximate it as the breaking of a beam. In beam theory, we consider a neutral axis that is the center line of the slender region. We compute an approximate medial axis that intersects the center of the cross-section $\mathbf{p}$, and has a tangent whose angle with each of the two normals at the ends of the cross-section are equal (see Figure A.1 (right)).

We project the cross-section perpendicular to the medial axis, so that it is perpendicular to the neutral axis. Let l be the length of the projected cross-section, and $\mathbf{v} \in \mathbb{R}^3$ be the tangent direction. The 3D cross-section shape is rectangular with dimensions l and t, where t is the thickness of the material.

In beam theory, the ratio between bending moment and maximum stress along the cross-section is called the *section modulus*. The section modulus for a bending plate about $\mathbf{v}$ is $Z_v = \frac{1}{6}t^2 l$. Hence, the maximum stress for this bending direction is $\sigma_v = |\boldsymbol{\tau}^+ \cdot \mathbf{v}|/Z_v$. We also consider bending around the normal direction $\mathbf{n}$ of the plate. The section modulus around this axis is $Z_n = \frac{1}{6}l^2 t$. Hence, the maximum stress for this bending direction is $\sigma_n = |\boldsymbol{\tau}^+ \cdot \mathbf{n}|/Z_n$. If σ_v or σ_n exceeds the maximum stress of the material, the structure may break at this cross-section, so it is presented to the user as a weak cross section. In our implementation, we display and colour cross-sections (starting in blue and shifting to red) that exceed 1/5 the maximum stress the material can handle.

Bibliography

[1] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22:181–193, 2006.

[2] Autodesk. Autodesk 123d make, 2013.

[3] N. I. Badler, J. O'Rourke, and H. Toltzis. A spherical representation of a human body for visualizing movement. *Proc. IEEE*, 67:1397–1403, October 1979.

[4] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. Ilovesketch: as-natural-as-possible sketching system for creating 3d curve models. In *In Proc. of User interface software and technology*, pages 151–160, 2008.

[5] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. Everybodylovessketch: 3d sketching for a broader audience. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 59–68, New York, NY, USA, 2009. ACM.

[6] J. A. Bærentzen, M. K. Misztal, and K. WełNicka. Converting skeletal structures to quad dominant meshes. *Comput. Graph.*, 36(5):555–561, August 2012.

[7] Michael Banks and Elaine Cohen. Real time spline curves from interactively sketched data. *SIGGRAPH Comput. Graph.*, 24(2):99–107, February 1990.

[8] Thomas Baudel. A mark-based interaction paradigm for free-hand drawing. In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology*, UIST '94, pages 185–192, New York, NY, USA, 1994. ACM.

[9] Peter N. Belhumeur, David J. Kriegman, and Alan L. Yuille. The bas-relief ambiguity. *Int. J. Comput. Vision*, 35(1):33–44, 1999.

[10] Marshall Bern and David Eppstein. Mesh generation and optimal triangulation, 1992.

[11] Mikhail Bessmeltsev, Caoyu Wang, Alla Sheffer, and Karan Singh. Design-driven quadrangulation of closed 3d curves. *Transactions on Graphics (Proc. SIGGRAPH ASIA 2012)*, 31(5), 2012.

[12] James F. Blinn. A generalization of algebraic surface drawing. *ACM Trans. Graph.*, 1(3):235–256, July 1982.

[13] Jules Bloomenthal and C. Bajaj. *Introduction to Implicit Surfaces.* Morgan Kaufmann series in computer graphics and geometric modeling. Morgan Kaufmann Publishers, Incorporated, 1997.

[14] Harry Blum. A Transformation for Extracting New Descriptors of Shape. *Models for the Perception of Speech and Visual Form*, pages 362–380, 1967.

[15] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy. *Polygon Mesh Processing.* Ak Peters Series. Taylor & Francis, 2010.

[16] David Bourguignon, Marie-Paule Cani, and George Drettakis. Drawing for Illustration and Annotation in 3D. In A. Chalmers and T.-M. Rhyne, editors, *EUROGRAPHICS*, volume 20 of *EUROGRAPHICS Conference Proceedings*, pages C114–C122, Manchester, Royaume-Uni, 2001. EUROGRAPHICS, Blackwell Publishers.

[17] Gareth Bradshaw and Carol O'Sullivan. Adaptive medial-axis approximation for sphere-tree construction. *ACM Trans. Graph.*, 23(1):1–26, January 2004.

[18] Michael Burns, Janek Klawe, Szymon Rusinkiewicz, Adam Finkelstein, and Doug DeCarlo. Line drawings from volume data. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 24(3):512–518, August 2005.

[19] Franck Caniard and Roland W. Fleming. Distortion in 3d shape estimation with changes in illumination. In *Proc. Symp. Applied perception in graphics and visualization*, pages 99–105, 2007.

[20] E. Catmull and J. Clark. Seminal graphics. chapter Recursively generated B-spline surfaces on arbitrary topological meshes, pages 183–188. ACM, New York, NY, USA, 1998.

[21] Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 3-sweep: Extracting editable objects from a single photo. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2013)*, 32(6), 2013.

[22] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3D mesh segmentation. In *ACM SIGGRAPH*, pages 1–12, 2009.

[23] Xuejin Chen, Sing Bing Kang, Ying-Qing Xu, Julie Dorsey, and Heung-Yeung Shum. Sketching reality: Realistic interpretation of architectural designs. *ACM Trans. Graph.*, 27(2):11:1–11:15, May 2008.

[24] Joseph Jacob Cherlin, Faramarz Samavati, Mario Costa Sousa, and Joaquim A. Jorge. Sketch-based modeling with few strokes. In *Proceedings of the 21st Spring Conference on Computer Graphics*, SCCG '05, pages 137–145, New York, NY, USA, 2005. ACM.

[25] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics*, 22:37–54, 1997.

[26] Kenneth L. Clarkson, Richard Cole, and Robert E. Tarjan. Randomized parallel algorithms for trapezoidal diagrams. In *Proceedings of the seventh annual symposium on Computational geometry*, SCG '91, pages 152–161, New York, NY, USA, 1991. ACM.

[27] Jonathan M. Cohen, Lee Markosian, Robert C. Zeleznik, John F. Hughes, and Ronen Barzel. An interface for sketching 3d curves. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, I3D '99, pages 17–21, New York, NY, USA, 1999. ACM.

[28] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. In *Proc. SIGGRAPH*, pages 905–914, 2004.

[29] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. Where do people draw lines? *Commun. ACM*, 55(1), 2012.

[30] Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. How well do line drawings depict shape? In *TOG SIGGRAPH*, pages 28:1–28:9, 2009.

[31] S. A. Coons. Surfaces for computer-aided design of space forms. Technical report, Cambridge, MA, USA, 1967.

[32] F. de Goes, S. Goldenstein, M. Desbrun, and L. Velho. Exoskeleton: Curve network abstraction for 3d shapes. *Computer and Graphics*, 35(1):112–121, 2011.

[33] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 22(3):848–855, July 2003.

[34] Doug DeCarlo and Anthony Santella. Stylization and abstraction of photographs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 769–776, New York, NY, USA, 2002. ACM.

[35] Xavier Décoret, Frédo Durand, François X. Sillion, and Julie Dorsey. Billboard clouds for extreme model simplification. *ACM TOG*, 22(3):689–696, 2003.

[36] Tamal K. Dey and Samrat Goswami. Tight cocone: a water-tight surface reconstructor. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, SM '03, pages 127–134, New York, NY, USA, 2003. ACM.

[37] Daniel Doo. A subdivision algorithm for smoothing down irregularly shaped polyhedrons. In *Proced. Int'l Conf. Ineractive Techniques in Computer Aided Design*, pages 157–165. IEEE Computer Soc., 1978.

[38] Julie Dorsey, Songhua Xu, Gabe Smedresman, Holly Rushmeier, and Leonard McMillan. The mental canvas: A tool for conceptual architectural design and analysis. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, PG '07, pages 201–210, Washington, DC, USA, 2007. IEEE Computer Society.

[39] Betty Edwards. *The New Drawing on the Right Side of the Brain.* Tarcher, 2002.

[40] Koos Eissen and Roselien Steur. *Sketching: Drawing Techniques for Product Designers.* Bis Publishers, 2008.

[41] Arthur Faisman and Michael S. Langer. Perception of qualitative shape from diffuse and specular reflections. In *Proc. of the ACM Symp. on Applied Perception*, pages 123–123, 2012.

[42] G.E. Farin and D. Hansford. *The Essentials of Cagd.* Ak Peters Series. Taylor & Francis, 2000.

[43] Gerald Farin. *Curves and surfaces for CAGD: a practical guide.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2002.

[44] Gerald E. Farin. *NURBS for Curve and Surface Design.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1991.

[45] M. Fatih Demirci, Ali Shokoufandeh, and Sven J. Dickinson. Skeletal shape abstraction from examples. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):944–952, May 2009.

[46] J Feldman and Manish Singh. Information along contours and object boundaries. In *Psych. Review*, volume 112, pages 243–252, 2005.

[47] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[48] Joachim Giesen, Balint Miklos, Mark Pauly, and Camille Wormser. The scale axis transform. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, SCG '09, pages 106–115, New York, NY, USA, 2009. ACM.

[49] Yotam Gingold, Takeo Igarashi, and Denis Zorin. Structured annotations for 2d-to-3d modeling. *ACM Trans. Graph.*, 28(5):148:1–148:9, December 2009.

[50] Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. *SHREC: SHape REtrieval Contest: Watertight models track*, 2007.

[51] Aleksey Golovinskiy and Thomas Funkhouser. Randomized cuts for 3D mesh analysis. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA)*, 27(5), December 2008.

[52] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 171–180, New York, NY, USA, 1996. ACM.

[53] George W. Hart. Modular kirigami. *Journal of Mathematics and the Arts*, 2007.

[54] Kristian Hildebrand, Bernd Bickel, and Marc Alexa. crdbrd : Shape Fabrication by Sliding Planar Slices. *CGF Eurographics*, 31(2), 2012.

[55] Kristian Hildebrand, Bernd Bickel, and Marc Alexa. Orthogonal slicing for additive manufacturing. *Computers & Graphics*, 2013.

[56] Michael Holroyd, Ilya Baran, Jason Lawrence, and Wojciech Matusik. Computing and fabricating multilayer models. *ACM Trans. Graph.*, 30(6):187:1–187:8, December 2011.

[57] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, pages 71–78, New York, NY, USA, 1992. ACM.

[58] Alexander Hornung and Leif Kobbelt. Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 41–50, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[59] Philip M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. Graph.*, 15(3):179–210, July 1996.

[60] Takeo Igarashi and John F. Hughes. A suggestive interface for 3d drawing. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 173–181, New York, NY, USA, 2001. ACM.

[61] Takeo Igarashi, Sachiko Kawachiya, Hidehiko Tanaka, and Satoshi Matsuoka. Pegasus: a drawing system for rapid geometric design. In *CHI 98 Cconference Summary on Human Factors in Computing Systems*, CHI '98, pages 24–25, New York, NY, USA, 1998. ACM.

[62] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. Interactive beautification: a technique for rapid geometric design. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, UIST '97, pages 105–114, New York, NY, USA, 1997. ACM.

[63] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 409–416, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[64] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 1134–1141, New York, NY, USA, 2005. ACM.

[65] Gabe Johnson, Mark Gross, Ellen Yi-Luen Do, and Jason Hong. Sketch it, make it: sketching precise drawings for laser cutting. In *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts*, CHI EA '12, pages 1079–1082, New York, NY, USA, 2012. ACM.

[66] Gabe Johnson, Mark D Gross, and Ellen Yi-Luen Do. Flow selection: a time-based selection and operation technique for sketching tools. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '06, pages 83–86, New York, NY, USA, 2006. ACM.

[67] Tilke Judd, Frédo Durand, and Edward H. Adelson. Apparent ridges for line drawing. *ACM Trans. Graph.*, 26(3):19, 2007.

[68] Dan Julius, Vladislav Kraevoy, and Alla Sheffer. D-charts: Quasi-developable mesh segmentation. In *In Computer Graphics Forum, Proc. of Eurographics*, pages 581–590, 2005.

[69] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. Learning 3d mesh segmentation and labeling. In *ACM SIGGRAPH*, pages 102:1–102:12, 2010.

[70] Evangelos Kalogerakis, Derek Nowrouzezahrai, Patricio Simari, James McCrae, Aaron Hertzmann, and Karan Singh. Data-driven curvature for real-time line drawing of dynamic scene. *ACM Transactions on Graphics*, 28(1), 2009.

[71] Levent Burak Kara and Kenji Shimada. Construction and modification of 3d geometry using a sketch-based interface. In *Proceedings of the Third Eurographics Conference on Sketch-Based Interfaces and Modeling*, SBM'06, pages 59–66, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[72] O. Karpenko, J. F. Hughes, and R. Raskar. Free-form sketching with variational implicit surfaces. *Computer Graphics Forum*, 21(3):585–594, 2002.

[73] Olga A. Karpenko and John F. Hughes. Smoothsketch: 3d free-form shapes from complex sketches. *ACM Trans. Graph.*, 25(3):589–598, July 2006.

[74] Sagi Katz, George Leifman, and Ayellet Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):649–658, September 2005.

[75] Ladislav Kavan, Simon Dobbyn, Steven Collins, Jiří Žára, and Carol O'Sullivan. Polypostors: 2d polygonal impostors for 3d crowds. In *Proc. I3D*, pages 149–155, 2008.

[76] Azam Khan, Ben Komalo, Jos Stam, George Fitzmaurice, and Gordon Kurtenbach. Hovercam: interactive 3d navigation for proximal object inspection. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 73–80, New York, NY, USA, 2005. ACM.

[77] M. Kilian, S. Flöry, Z. Chen, N. J. Mitra, A. Sheffer, and H. Pottmann. Curved folding. *ACM SIGGRAPH*, 27(3):#75, 1–9, 2008.

[78] James T. Klosowski, Martin Held, Joseph S. B. Mitchell, Henry Sowizral, and Karel Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, January 1998.

[79] David C. Knill. Perception of surface contours and surface shape: from computation to psychophysics. *Journal of Optical Society of America*, 9(9):1449–1464, 1992.

[80] Jan J. Koenderink. *Solid shape.* MIT Press, Cambridge, MA, USA, 1990.

[81] Jan J. Koenderink, Andrea J. Van Doorn, and Astrid M. L. Kappers. Surface perception in pictures. *Perception and Psychophysics*, pages 487–496, 1992.

[82] Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04, pages 11–21, New York, NY, USA, 2004. ACM.

[83] Vladislav Krayevoy and Alla Sheffer. Variational, meaningful shape decomposition. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[84] Lars Krecklau and Leif Kobbelt. Smi 2012: Full interactive modeling by procedural high-level primitives. *Comput. Graph.*, 36(5):376–386, August 2012.

[85] Yu kun Lai, Shi min Hu, Ralph R. Martin, and Paul L. Rosin. Fast mesh segmentation using random walks. In *In ACM Symposium on Solid and Physical Modeling*, 2008.

[86] Eric Larsen, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. Fast proximity queries with swept sphere volumes. Technical report, 1999.

[87] Manfred Lau, Akira Ohgawara, Jun Mitani, and Takeo Igarashi. Converting 3d furniture models to fabricatable parts and connectors. *ACM Trans. Graph.*, 30(4):85:1–85:6, July 2011.

[88] Xian-Ying Li, Tao Ju, Yan Gu, and Shi-Min Hu. A geometric study of v-style pop-ups: theories and algorithms. In *TOG SIGGRAPH*, pages 98:1–98:10, 2011.

[89] Yangyan Li, Xiaokun Wu, Yiorgos Chrysanthou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics*, 30(4):52:1–52:12, 2011.

[90] Lu Liu, C. Bajaj, Joseph Deasy, Daniel A. Low, and Tao Ju. Surface reconstruction from non-parallel curve networks. *CGF EUROGRAPHICS*, 27(2):155–163, 2008.

[91] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.*, 25(3):681–689, July 2006.

[92] S. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137, September 1982.

[93] Kui-Yip Lo, Chi-Wing Fu, and Hongwei Li. 3d polyomino puzzle. In *ACM SIGGRAPH Asia 2009 papers*, SIGGRAPH Asia '09, pages 157:1–157:8, New York, NY, USA, 2009. ACM.

[94] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 163–169, New York, NY, USA, 1987. ACM.

[95] Linjie Luo, Ilya Baran, Szymon Rusinkiewicz, and Wojciech Matusik. Chopper: Partitioning models into 3D-printable parts. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 31(6), December 2012.

[96] M. Masry, D. Kang, and H. Lipson. A freehand sketching interface for progressive construction of 3d objects. *Comput. Graph.*, 29(4):563–575, August 2005.

[97] James McCrae, Igor Mordatch, Michael Glueck, and Azam Khan. Multiscale 3d navigation. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 7–14, New York, NY, USA, 2009. ACM.

[98] James McCrae and Karan Singh. Neatening sketched strokes using piecewise french curves. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '11, pages 141–148, New York, NY, USA, 2011. ACM.

[99] James McCrae, Karan Singh, and Niloy J. Mitra. Slices: a shape-proxy based on planar sections. *TOG SIGGRAPH Asia*, 30(6):168:1–168:12, 2011.

[100] Ravish Mehra, Qingnan Zhou, Jeremy Long, Alla Sheffer, Amy Gooch, and Niloy J. Mitra. Abstraction of man-made shapes. *TOG SIGGRAPH Asia*, 28(5):137:1–137:10, 2009.

[101] Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. In *TOG SIGGRAPH*, pages 560–568, 2006.

[102] Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer, and Luc Van Gool. Procedural modeling of buildings. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 614–623, New York, NY, USA, 2006. ACM.

[103] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Fibermesh: designing freeform surfaces with 3d curves. *ACM SIGGRAPH*, 26(3):41, 2007.

[104] D. Norman, J. T. Todd, and Flip Phillips. The perception of surface orientation from multiple sources of optical information. *Perception and Psychophysics*, pages 629–636, 1995.

[105] Jun-Ho Oh, David Hanson, Won-Sup Kim, Young Han, Jung-Yup Kim, and Ill-Woo Park. Design of android type humanoid robot albert hubo. In *IROS*, pages 1428–1433, 2006.

[106] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. In *SIGGRAPH*, pages 609–612, 2004.

[107] Luke Olsen, Faramarz F. Samavati, Mario Costa Sousa, and Joaquim A. Jorge. Technical section: Sketch-based modeling: A survey. *Comput. Graph.*, 33(1):85–103, February 2009.

[108] Joseph O'Rourke and Norman Badler. Decomposition of three-dimensional objects into spheres. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(3):295–305, 1979.

[109] James P. O'Shea, Martin S. Banks, and Maneesh Agrawala. The assumed light direction for perceiving shape from shading. In *Proc. on Applied perception in graphics and visualization*, pages 135–142, 2008.

[110] Kálmán Palágyi. A 3-subiteration 3d thinning algorithm for extracting medial surfaces. *Pattern Recogn. Lett.*, 23(6):663–675, 2002.

[111] Flip Phillips, James T. Todd, Jan J. Koenderink, and Astrid M.L. Kappers. Perceptual representation of visible surfaces. *Perception and Psychophysics*, 65:747–762, 2003.

[112] M.J.D. Powell. *A Fast Algorithm for Nonlinearly Constrained Optimization Calculations*, volume 630. Springer Verlag, 1978.

[113] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time hatching. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 581–, New York, NY, USA, 2001. ACM.

[114] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. Make It Stand: Balancing shapes for 3D fabrication. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 32(4):81:1–81:10, 2013.

[115] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[116] V. Ramachandran. Perceived shape from shading. *Scientific American*, pages 76–83, 1988.

[117] Kenneth Rose, Alla Sheffer, Jamie Wither, Marie-Paule Cani, and Boris Thibert. Developable surfaces from arbitrary sketched boundaries. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 163–172, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[118] Szymon Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Symp. on 3DDPVT*, 2004.

[119] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-d shapes. *Proc. SIGGRAPH*, 24(4):197–206, 1990.

[120] Hanan Samet and Markku Tamminen. Bintrees, csg trees, and time. *SIGGRAPH Comput. Graph.*, 19(3):121–130, July 1985.

[121] Greg Saul, Manfred Lau, Jun Mitani, and Takeo Igarashi. Sketchchair: an all-in-one chair design system for end users. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, TEI '11, pages 73–80, New York, NY, USA, 2011. ACM.

[122] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge. Shapeshop: sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[123] Ryan Schmidt, Azam Khan, Gord Kurtenbach, and Karan Singh. On expert performance in 3d curve-drawing tasks. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '09, pages 133–140, New York, NY, USA, 2009. ACM.

[124] Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. Analytic drawing of 3d scaffolds. In *ACM SIGGRAPH Asia*, pages 149:1–149:10, 2009.

[125] Ryan Schmidt and Karan Singh. meshmixer: an interface for rapid mesh composition. In *ACM SIGGRAPH 2010 Talks*, SIGGRAPH '10, pages 6:1–6:1, New York, NY, USA, 2010. ACM.

[126] Yuliy Schwartzburg and Mark Pauly. Design and optimization of orthogonally intersecting planar surfaces. In *Proc. of the Design Modelling Symposium*, 2011.

[127] Yuliy Schwartzburg and Mark Pauly. Fabrication-aware design with intersecting planar pieces. *Computer Graphics Forum*, 32(2pt3):317–326, 2013.

[128] Adrian Secord. Weighted Voronoi stippling. In *Proceedings of the second international symposium on Non-photorealistic animation and rendering*, pages 37–43. ACM Press, 2002.

[129] Adrian Secord, Jingwan Lu, Adam Finkelstein, Manish Singh, and Andrew Nealen. Perceptual models of viewpoint preference. *ACM TOG*, 30(5), 2011.

[130] Raimund Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1(1):51–64, July 1991.

[131] Carlo H. Séquin. Prototyping dissection puzzles with layered manufacturing. *Fabrication and Sculpture Track, Shape Modeling International Conference*, 2012.

[132] Cloud Shao, Adrien Bousseau, Alla Sheffer, and Karan Singh. Crossshade: shading concept sketches using cross-section curves. *TOG SIGGRAPH*, 31(4):45:1–45:11, 2012.

[133] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249–259–259, April 2008.

[134] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[135] Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22:1–3, 2001.

[136] Shymon Shlafman, Ayellet Tal, and Sagi Katz. Metamorphosis of polyhedral surfaces using decomposition. In *Computer Graphics Forum*, pages 219–228, 2002.

[137] Patricio Simari, Evangelos Kalogerakis, and Karan Singh. Folding meshes: hierarchical mesh segmentation based on planar symmetry. In *Proc. SGP*, 2006.

[138] Patricio D. Simari and Karan Singh. Extraction and remeshing of ellipsoidal representations from mesh data. In *Proceedings of Graphics Interface 2005*, GI '05, pages 161–168, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[139] Karan Singh. Interactive curve design using digital french curves. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, I3D '99, pages 23–30, New York, NY, USA, 1999. ACM.

[140] John M. Snyder. *Generative Modeling for Computer Graphics and CAD: Symbolic Shape Design Using Interval Analysis*. Academic Press Professional, Inc., San Diego, CA, USA, 1992.

[141] Jos Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 395–404, New York, NY, USA, 1998. ACM.

[142] K. A. Stevens. The visual interpretation of surface contours. *Artificial Intelligence*, 17:47 – 73, 1981.

[143] Ivan E. Sutherland. Sketchpad: A man-machine graphical communication system. In *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference*, AFIPS '63 (Spring), pages 329–346, New York, NY, USA, 1963. ACM.

[144] Graeme Sweet and Colin Ware. View direction, surface orientation and texture orientation for perception of surface shape. In *Proc. of Graphics Interface 2004*, pages 97–106, 2004.

[145] Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. Curve skeleton extraction from incomplete point cloud. *ACM Trans. Graph.*, 28(3):71:1–71:9, July 2009.

[146] Roger Tam and Wolfgang Heidrich. Shape simplification based on the medial axis transform. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 63–, Washington, DC, USA, 2003. IEEE Computer Society.

[147] Robert E. Tarjan and Christopher J. Van Wyk. An o(n log log n)-time algorithm for triangulating a simple polygon, 1988.

[148] Haruyuki Tatsumi, Eiji Takaoki, Koichi Omura, and Hisao Fujita. A new method for three-dimensional reconstruction from serial sections by computer graphics using metaballs: reconstruction of hepatoskeletal system formed by ito cells in the cod liver. *Comput. Biomed. Res.*, 23(1):37–45, January 1990.

[149] Yannick Thiel, Karan Singh, and Ravin Balakrishnan. Elasticurves: Exploiting stroke dynamics and inertia for the real-time neatening of sketched 2d curves. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 383–392, New York, NY, USA, 2011. ACM.

[150] Jean-Marc Thiery, Emilie Guy, and Tamy Boubekeur. Sphere-meshes: Shape approximation using spherical quadric error metrics. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia 2013)*, 2013.

[151] Robert Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58(1):267–288, 1996.

[152] Stephen Timoshenko. *Theory of Elasticity.* Mcgraw-Hill College, 3 edition, 6 1970.

[153] J Todd and F Reichel. Visual perception of smoothly curved surfaces from double-projected contour patterns. In *J. of Exp. Psych.: Human Percep.and Performance*, volume 16, pages 665–674, 1990.

[154] Steve Tsang, Ravin Balakrishnan, Karan Singh, and Abhishek Ranjan. A suggestive interface for image guided 3d sketching. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 591–598, New York, NY, USA, 2004. ACM.

[155] Greg Turk and James F. O'Brien. Shape transformation using variational implicit functions. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[156] Nobuyuki Umetani, Takeo Igarashi, and Niloy J. Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012)*, 31(4), 2012.

[157] Nobuyuki Umetani and Ryan Schmidt. Cross-sectional structural analysis for 3d printing optimization. In *SIGGRAPH Asia 2013 Technical Briefs*, SA '13, pages 5:1–5:4, New York, NY, USA, 2013. ACM.

[158] Gino van den Bergen. Efficient collision detection of complex deformable models using aabb trees. *J. Graph. Tools*, 2(4):1–13, January 1998.

[159] Karl D.D. Willis, Juncong Lin, Jun Mitani, and Takeo Igarashi. Spatial sketch: bridging between movement and fabrication. In *In Proc. Tangible, embedded, and embodied interaction*, pages 5–12, 2010.

[160] Holger Winnemöller, David Feng, Bruce Gooch, and Satoru Suzuki. Using npr to evaluate perceptual shape cues in dynamic environments. In *Proc. NPAR*, pages 85–92, 2007.

[161] Brian Wyvill, Andrew Guy, and Eric Galin. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2):149–158, 1999.

[162] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, August 1986.

[163] Shiqing Xin, Chi-Fu Lai, Chi-Wing Fu, Tien-Tsin Wong, Ying He, and Daniel Cohen-Or. Making burr puzzles from 3d models. *ACM Trans. Graph.*, 30(4):97:1–97:8, July 2011.

[164] Hitoshi Yamauchi, Stefan Gumhold, Rhaleb Zayer, and Hans-Peter Seidel. Mesh segmentation driven by gaussian curvature. *The Visual Computer*, 21(8-10):659–668, 2005.

[165] M.E. Yumer and L.B. Kara. Co-abstraction of shape collections. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2012)*, 31:166:1–166:11, 2012.

[166] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: an interface for sketching 3d scenes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 163–170, New York, NY, USA, 1996. ACM.

[167] Juyong Zhang, Jianmin Zheng, Chunlin Wu, and Jianfei Cai. Variational mesh decomposition. *ACM TOG*, 31(3):21:1–21:14, 2012.

[168] Youyi Zheng, Xiang Chen, Ming-Ming Cheng, Kun Zhou, Shi-Min Hu, and Niloy J. Mitra. Interactive images: Cuboid proxies for smart image manipulation. *ACM Transactions on Graphics*, 31(4):99:1–99:11, 2012.

[169] Kening Zhu and Shengdong Zhao. Autogami: a low-cost rapid prototyping toolkit for automated movable paper craft. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 661–670, New York, NY, USA, 2013. ACM.