UNIVERSITY OF CALIFORNIA

Los Angeles

Zoomorphic Design, Interchangeable Components, and Approximate Dissections: Three New Computational Tools for Open-Ended Geometric Design

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Noah Duncan

2017

ABSTRACT OF THE DISSERTATION

Zoomorphic Design, Interchangeable Components, and Approximate Dissections: Three New Computational Tools for Open-Ended Geometric Design

by

Noah Duncan
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2017
Professor Demetri Terzopoulos, Chair

This thesis introduces three new computational tools that employ geometric methods to solve open-ended design problems, an emerging trend in Computer Graphics modeling research.

First, we present a computational tool for the design of zoomorphic objects—man-made objects that possess the form or appearance of an animal. Given a man-made shape with desirable functional qualities and an organic shape with desirable animalistic qualities, our tool geometrically merges the two shapes, incorporating the salient features of the organic shape while preserving the functionality of the man-made shape.

Second, inspired by mix-and-match toys such as Mr. Potato Head, we introduce a computational tool for designing interchangeable components that can form different objects with a coherent appearance. Our tool works by deforming and partitioning a set of initially incompatible input models. A key challenge is the novel geometric problem of minimally deforming the models and partitioning them such that the resulting components connect smoothly.

Third, we present a computational tool for creating geometric dissections, a set of pieces that can be re-arranged to form two distinct shapes. Previous techniques for creating dissections are limited to forming very simple or geometrically ideal shapes, whereas ours supports complex naturalistic shapes.

We experiment with and evaluate the above three tools in a variety of applications.

The dissertation of Noah Duncan is approved.

Song-Chun Zhu

Stanley J Osher

Joseph M Teran

Demetri Terzopoulos, Committee Chair

University of California, Los Angeles

2017

Dedicated to my parents for encouraging me to take my own path in life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to deeply thank my advisor Demetri Terzopoulos for introducing me to the world of Computer Graphics and giving me an optimal amount of freedom. Demetri has always encouraged me to conduct research based on my own ideas, which was a wise policy that forced me to develop an independence crucial to long-term success in research. At the same time, Demetri was always willing to let me know when an idea was a dead-end or to help me refine a flawed idea into something viable. Demetri's considerable talent for writing is responsible for the brevity and clearness of my published papers, and he spent many hours improving this thesis. Finally, I thank Demetri for encouraging me to collaborate with other researchers, which resulted in several fruitful relationships.

I would like to thank my co-advisor Lap-Fai Yu. I met Lap-Fai when I was a first year PhD student and he was a highly accomplished fourth year UCLA student about to graduate. Despite the fact that I had no research accomplishments, Lap-Fai immediately began intensively mentoring me when I asked to collaborate with him, when he could have been using this time for his own work. Ever since, he has been an invaluable source of productive discussion, an inspiration for navigating the perilous world of academia, and a tireless critic who lets me know where my work falls short and how to improve it. His keen sense of design for figures has benefited all of my papers.

I would like to thank my co-advisor Sai-Kit Yeung. Sai-Kit graciously hosted me in his lab at SUTD in Singapore several times. Like Lap-Fai, Sai-Kit took a risk working with me when I was still a first year student. Our discussions have uncovered many creative ideas, and he has greatly developed my ability to tell a compelling story about an idea in a paper. Finally, he has always helped me squeeze out every ounce of productivity as the deadline grew near. Our submitted papers may have been rejected if not for his ability to prioritize what should be focused on and his insistence on clarity.

Finally, I would like to thank my fellow lab members, Tomer Weiss and Masaki Nakada, for fun discussion about our research and other topics, and for keeping things enjoyable even as deadlines loomed.

VITA

2012	B.S. (Computer Science), Harvey Mudd College.
2013–2017	Teaching Assistant, Computer Science Department, UCLA.
2015–2017	Research Assistant, Information Systems Technology and Design Pillar, Singapore University of Technology and Design (SUTD).

PUBLICATIONS

Duncan, N., Yu, L.-F., Yeung, S.-K., and Terzopoulos, D. Approximate Dissections. *ACM Transactions on Graphics (TOG)*, **36**(6), 2017, 182:1–13.

Duncan, N., Yu, L.-F., and Yeung, S.-K. Interchangeable components for hands-on assembly based modelling. *ACM Transactions on Graphics (TOG)*, **35**(6), 2016, 234:1–14.

Duncan, N., Yu, L.-F., Yeung, S.-K., and Terzopoulos, D. Zoomorphic design. *ACM Transactions on Graphics (TOG)*, **34**(4), 2015, 95:1–13.

Yu, L.-F., Duncan, N., and Yeung, S.-K. Fill and transfer: A simple physics-based approach for containability reasoning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, 711–719.

CHAPTER 1

Introduction

Open-ended design problems are design problems which are non-trivial to map to mathematical problems. They are ultimately defined in terms of human perception. Since it is unclear what is the "correct" formalization of these problems, some research communities have avoided them. However, they have significant practical value. Recently, the Computer Graphics research community has become increasingly interested in such problems, which range from computational interior design (Yu et al., 2011; Merrell et al., 2011), to the design of clothing (Umetani et al., 2011), accessories (Igarashi et al., 2012), puzzles (Zhou et al., 2014), mechanical toys (Zhu et al., 2012), and even cities (Aliaga et al., 2008; Vanegas et al., 2012). The development of computational tools for solving these and other open-ended design problems is an emerging trend in the field. These tools often involve some sort of optimization.

Geometric open-ended design problems are those that can be fully defined in terms of geometric shapes. The variables of the optimization usually specify how one or more objects are *deformed*. The unifying theme of this thesis is the use of geometrical methods to solve such problems. Specifically, this thesis introduces three novel open-ended design problems and proposes three geometric approaches to solve them. The common framework around these three problems is a functional constraint and an aesthetic objective. We want to preserve the appearance of the input object as much as possible, given the satisfaction of a functional constraint. Figure 1.1 shows how the works fit into this framework.

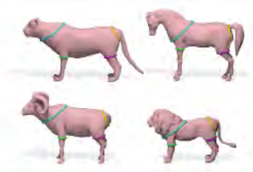
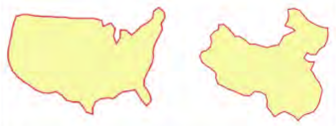
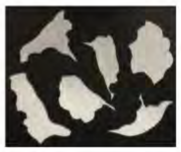
Aesthetic Object(s)	Output	Functional Constraint	Optimization Technique
		Manmade shape's functionality is preserved	Gradient-free coarse-to-fine search.
		Parts are interchangeable	Gradient-based continuous optimization.
		Pieces are connectable	Branch-and-bound search. A node is evaluated by solving a continuous optimization problem.

Figure 1.1: The three open-ended design problems covered in this thesis and how they fit in a common framework.

1.1 The Contributions of this Thesis

The main contributions of this thesis are threefold, as follows:

1. **Zoomorphic Design:** We introduce the first process to allow the efficient design of zoomorphic shapes. This work was published in ACM SIGGRAPH 2015 (Duncan et al., 2015). The key technical ingredient in the process is the volumetric design constraint (VDC), a general technique for ensuring that introducing new geometry to a man-made object does not interfere with its functionality.

   In particular, we introduce a computational tool for the design of zoomorphic objects, which are man-made objects that possess the form or appearance of an animal. Our tool works by combining two shapes: a man-made shape that represents the functional qualities our zoomorphic object should possess and an organic shape that represents its animalistic qualities. The key technical challenge in this work is merging the two shapes so that the salient features of the organic shape are prominent while preserving the functionality of the man-made shape.

2. **Interchangeable Components:** we introduce the first process to allow the efficient design of interchangeable components that connect to form coherent shapes. This work was published in ACM SIGGRAPH Asia 2016 (Duncan et al., 2016). Previous sets of such components were limited to forming a small range of shapes and required laborious manual design.

   In particular, for hands-on assembly-based modeling, we introduce a computational tool for the design of components which are interchangeable, but can connect to form objects with a smooth natural appearance. These components are inspired by Mix-and-match toys such as Mr. Potato Head. Our tool works by deforming and partitioning a set of input models, which are initially incompatible. The key challenge here is the novel geometric problem of deforming and partitioning the models such that the resulting components connect smoothly, while minimizing the extent of the deformation needed.

3. **Approximate Dissections:** We introduce the first process to allow the creation of geometric dissections between complex, naturalistic shapes. This work was published in ACM SIGGRAPH Asia 2017 (Duncan et al., 2017). Previous dissections were limited to simple, abstract shapes.

   In particular, we find a novel technique for generating geometric dissections. A geometric dissection is a set of pieces which can be assembled in different ways to form distinct shapes. A well-known example is the ancient Chinese tangram puzzle. Existing techniques for dissection design are limited to dissections between geometrically ideal or extremely simplified shapes. By relaxing the traditional dissection design problem so that the dissection pieces only need to reconstruct the target shapes approximately, we allow the creation of dissections between complex naturalistic shapes for the first time.

The three subsequent sections motivate the Zoomorphic Design, Interchangeable Components, and Approximate Dissections problems in greater detail.

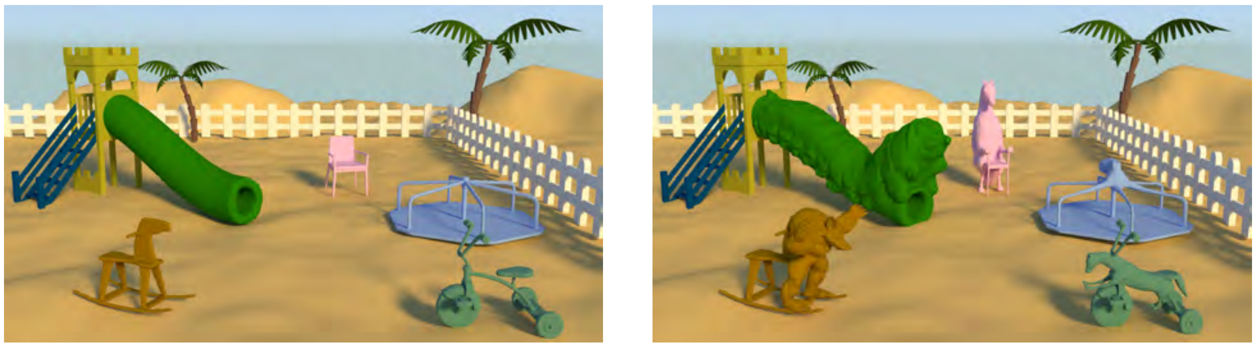

(a) Input (b) Result

Figure 1.2: A zoomorphic playground created by our Zoomorphic Design approach.

1.2 Zoomorphic Design

For centuries, humanity has attempted to capture the marvels of nature in man-made objects. Such objects range from ancient pottery vessels, to modern day piggy banks, designer chairs, and even buildings (Figure 1.3). Man-made objects that have the form or appearance of an animal are called *zoomorphic*. Since the beginning of recorded history, artists have created zoomorphic objects by "applying animalistic-inspired qualities to non-animal related objects" (Coates et al., 2009).

Zoomorphic concepts are present in architecture (Aldersey-Williams, 2003), furniture (Coates et al., 2009), and product design (Bramston, 2008; Lidwell and Manacsa, 2011). Research suggests that children have a natural affinity for animals, which may explain the frequent presence of zoomorphism in children's toys (Lidwell, 2014). Figure 1.2 illustrates how zoomorphic design can create a more appealing children's playground.

We propose a novel computational approach to tackle the unique challenges involved in creating zoomorphic objects. Some zoomorphic designs mimic animals at only an abstract level, such as the design of the Milwaukee Art Museum which is inspired by the shape of a bird in flight (Figure 1.3(d)), while others include components that directly mimic the shapes of animal parts. Our approach focuses on the latter category.

Designing a zoomorphic object entails high-level tradeoffs, such as compromising between

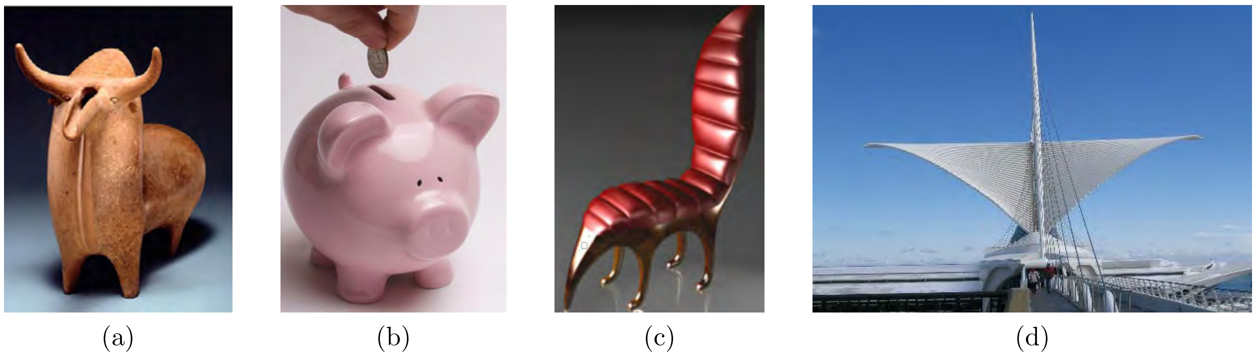


Figure 1.3: Historic and modern zoomorphic objects. (a) Bull-shaped vessel circa 1000 BC symbolizing fertility. (b) Piggy bank. (c) Anteater chair by artist Maximo Riera. (d) The Milwaukee Art Museum, designed in the shape of a bird in flight.

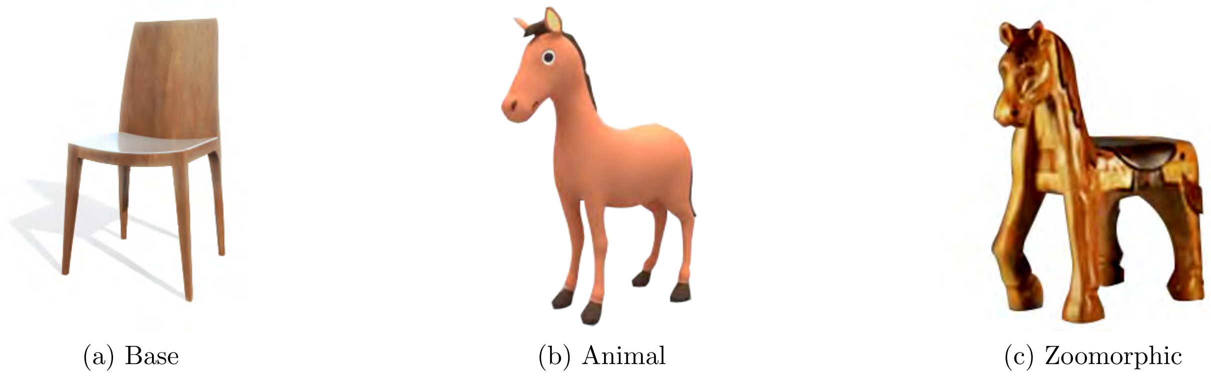


Figure 1.4: The three objects in our approach. Note: The objects shown were not created by our approach.

faithfulness to the animal form and retaining usefulness. For example, the chair in Figure 1.3(c) has proportions similar to an anteater, but the sitting area is narrow and the "snout" may interfere with a sitter's legs. High-level design goals correspond to low-level geometric operations, such as deformations of the animal-inspired and man-made components in the shape. The low-level operations can be tedious to execute manually. Therefore, our goal is to enable the user to direct the high-level design, while automating the low-level operations. We also hope to inspire the user by suggesting unusual yet viable designs that may not have been considered, such as the pink horse chair in Figure 1.2.

Our approach takes two surface meshes as input, which we call the "**base object**"and the "**animal object**". The base object represents the portions of the zoomorphic object that are

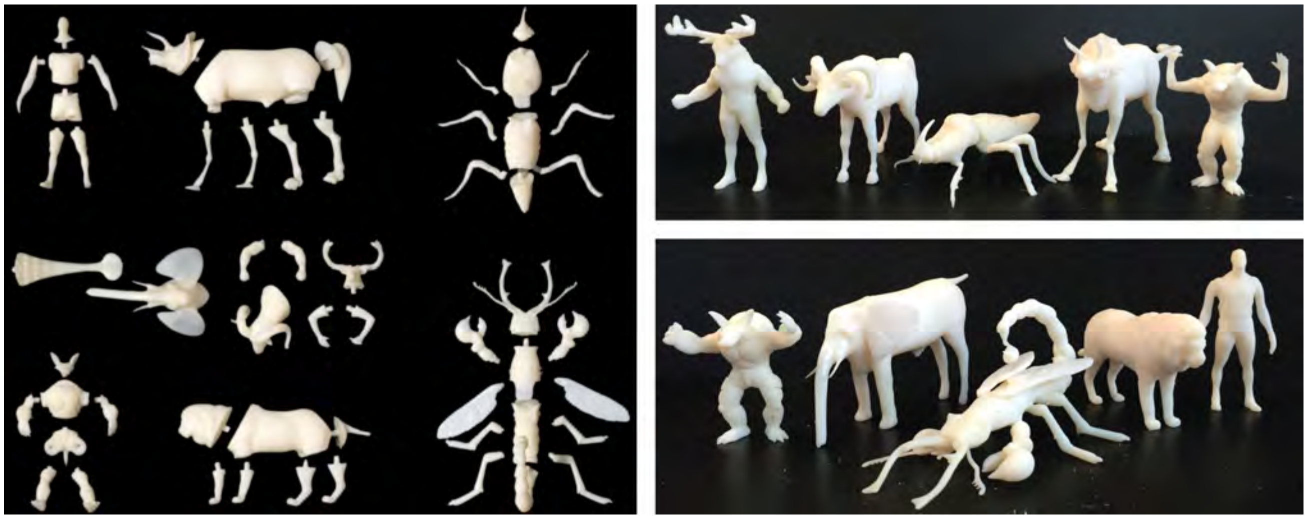


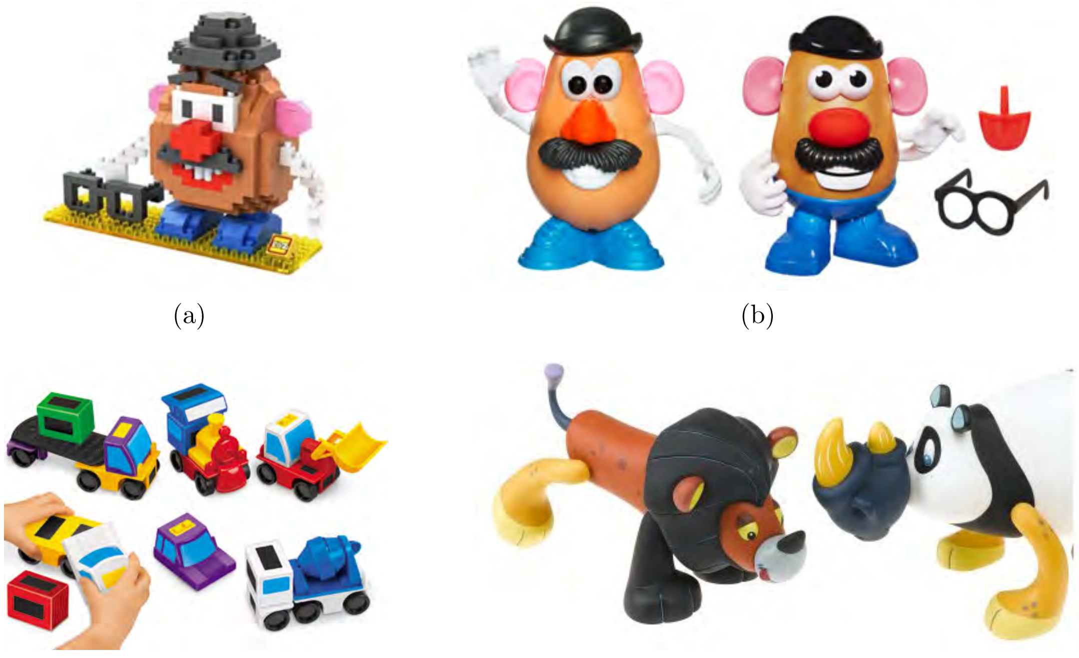
Figure 1.5: Our Interchangeable Components approach deforms and partitions a set of 3D models to fabricate a set of fully interchangeable components, which can be assembled into novel objects of coherent appearance.

not animal-related. The base object is generally man-made and represents the ‘functional category’ of the object we want to create. The animal object represents the portions of the zoomorphic object that are animal-related. For the zoomorphic object in Figure 1.4, the base object is an ordinary chair and the animal object is a horse. Our approach constructs a zoomorphic object by merging the two input objects.

1.3 Interchangeable Components

In the typical process of shape creation, a shape is constructed virtually on a computer, and then fabricated into the real world. Once fabricated, the shape’s geometry is fixed. Computer Graphics research has made great strides in allowing non-experts to create shapes through this process.

In this problem, we focus on an alternate shape creation process in which a set of components is fabricated that is capable of being assembled into a *range* of possible shapes. The advantage of this process is that the shape’s geometry is easy to *reconfigure*. This property is useful when a different shape is desired at a different time and for a physical exploration



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

Figure 1.6: Mr. Potato Head constructed from (a) Legos and (b) Interchangeable Components. The Legos induce an unnatural pixelated appearance. (c), (d) Commercially available interchangeable components for vehicles and animals. Note the simple geometries and clear boundaries between components.

of possible shapes. Furthermore, the set of possible shapes may be much larger than the set of components. For example, the set of components shown in Figure 1.5 can construct over 50,000 different humanoid figures. Directly fabricating this set of shapes would be prohibitively expensive.

Two real world examples of this process are construction toys such as Lego Bricks and Mix-and-Match toys such as Mr. Potato Head. These systems vary in the range of shapes they can construct, the ease of reconfiguring to a different shape and how coherent the shapes' appearance is. Lego Bricks are flexible enough to construct almost any shape, but are tedious to reconfigure and produce shapes with a distinctive blocky appearance as shown in Figure 1.6(a). Mix-and-Match toys use a set of interchangeable components to construct a much narrower range of shapes, but are easier to reconfigure and produce shapes with a smoother appearance. However, the geometry constructed by these toys is usually extremely simple and possesses an abstract look, which makes the component boundaries perceptible,

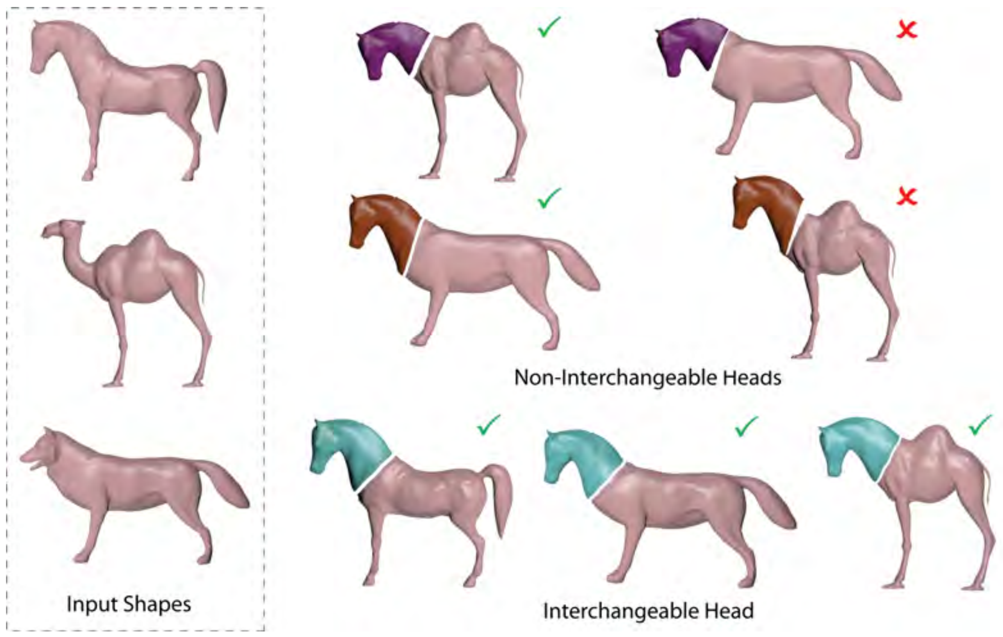


Figure 1.7: Without considering interchangeability, the horse's head can fit with either the camel's body or the wolf's body seamlessly, but not both. Considering interchangeability, the horse's head can fit with both the wolf's body and the camel's body seamlessly.

as shown in Figure 1.6(b)–(d).

We introduce a computational approach for designing interchangeable components which construct complex, diverse geometry and connect so that the visual impact of the junctions between them is minimal. Designing such components by hand would be very difficult. For example, in the humanoid components shown in the teaser, twenty-five pairwise compatibility constraints must be considered to ensure that any head can connect to any body. Fulfilling these constraints while preserving the appearance of the shapes is challenging with traditional modeling tools. Figure 1.7 illustrates how several constraints must be satisfied simultaneously to produce interchangeable parts.

Our approach takes a set of compatibly segmented models as input. Guided by the segmentations, it deforms and partitions the models into physically interchangeable components. Because of their interchangeability, the components can construct a wide range of novel shapes not seen in the input models.

At the essence of our approach is a novel geometric problem: Given a set of models, output

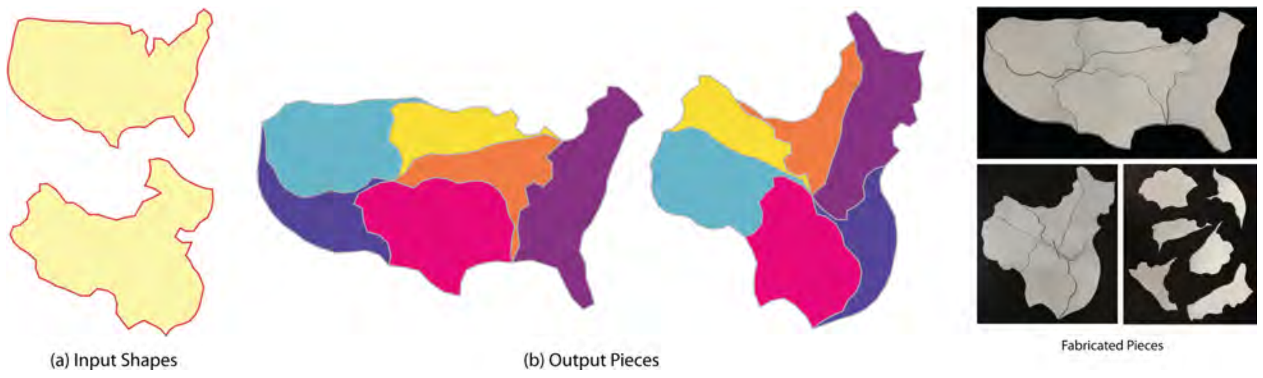


Figure 1.8: Given two input shapes (a), our Approximate Dissections approach generates a small number of pieces that may be arranged to form close approximations of the shapes. Here, we generate six pieces that can form the outlines of either the continental United States or China, demonstrating that the countries have roughly equal area (at the same scale). Note that in (b) both shapes are composed of the same set of pieces.

a set of components, such that the connecting boundaries of compatible components are identical (up to rigid transformation), and the deviation of the components from their original geometry is minimized.

Our solution proceeds in two steps. First, to determine the component boundaries, we apply a novel optimization which evolves a set of closed contours on surfaces such that their geometric similarity is maximized. Second, we deform the meshes so that the interchangeability constraint is met. Our deformation scheme distributes the distortion evenly over the meshes, and allows the user to interact with the optimizer to find a deformation that preserves semantic attributes.

1.4 Approximate Dissections

Geometric dissections are a popular type of puzzle and mathematical tool. A geometric dissection between two shapes is a partition of one shape into pieces, such that the pieces can be rearranged through rigid motion to form the other shape. Dissections have been known since ancient times. For example, Plato described a dissection between two equally sized squares and one larger one (Frederickson, 2002). Perhaps their first known appearance

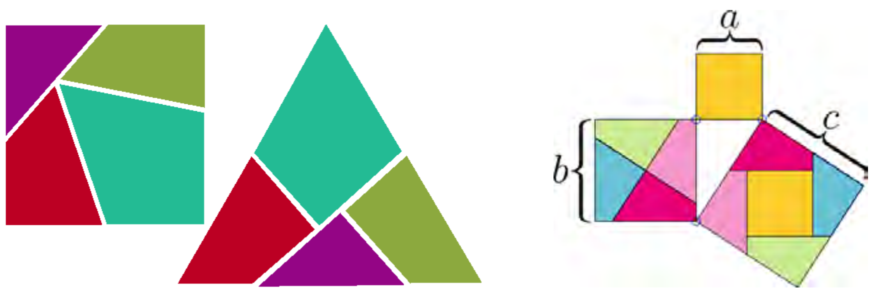


Figure 1.9: (a) A classic four piece dissection between a square and a triangle. (b) A dissection between a square and a pair of smaller squares, which illustrates the Pythagorean theorem.

is on a Babylonian tablet from 1800 BC, which shows the Pythagorean theorem for the special case of a right isosceles triangle. Figure 1.9 shows a classic dissection and one that illustrates the classic theorem.[1]

Dissections fascinate us because of the counter-intuitive property that a suitable set of pieces can transform between two distinctive shapes. This striking property means that dissections are popular as recreational puzzles. The transformation property is most impressive when the number of pieces used in the puzzle is minimized. Therefore designers of dissection puzzles usually try to minimize the number of pieces.

In mathematics research there has been significant work in finding minimal dissections for analytical shapes such as circles, triangles and regular polygons.[2] The Wallace-Bolyai-Gerwien theorem (Gardner, 1985) describes a procedure to create a dissection between any two polygons of equal area, but the number of pieces it uses may be much larger than the minimal number needed. The computational task of determining whether a K-piece dissection exists between two polygons has been shown to be NP-hard by Bosboom et al. (2015). Manurangsi et al. (2016) showed that the task is similarly hard to solve approximately.

[1]More recent mathematical results include the proof that any two polygons of equal area have a dissection between them (and that two polyhedra of equal volume do not, in general, have a dissection between them).

[2]Cohn (1975) investigated the minimum number of pieces needed for a triangle-square dissection. Kranakis et al. (2000) gave an asymptotic result on the minimum number of pieces needed for a dissection between a regular m-gon and an n-gon.

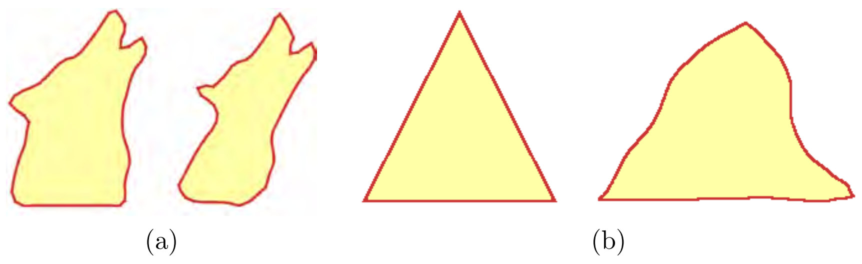


Figure 1.10: (a) The distorted dog head is still perceived as a dog head. (b) The distorted triangle is no longer perceived as a triangle.

A practical tool for designing dissections between arbitrary shapes with a minimal number of pieces is desirable. However, the complexity arguments cited above make this an intractable task and even if such a tool did exist, the minimum number of pieces might still be extremely large for many shapes.

We introduce a practical technique for dissection design that largely avoids these issues (Figure 1.8). Our technique is based on the observation that, for a large class of shapes, it is acceptable for a dissection to *approximate* rather than exactly construct the shapes. These are the shapes of complex real-world objects whose geometric specification is fuzzy, such as the dog's head in Figure 1.10(a). Our technique is not intended for use on abstract shapes with an exact geometric specification, because human perception is sensitive to distortions in these shapes (see, e.g., Figure 1.10(b)). Based on this observation, we propose a modified dissection problem in which the input shapes impose soft rather than hard constraints. This relaxation of the problem lets us develop an algorithm that generates dissections that differ qualitatively from traditional ones.

Our core technical contribution in this work is the introduction of the approximate dissection problem and a practical technique for solving it. To our knowledge this is the first general technique for dissections between naturalistic shapes. As an extension to our method, we develop a graphical user interface for refining dissections that suggests edits to the user and visualizes how altering one part of the dissection affects the remainder.

1.5 Dissertation Overview

The remainder of this dissertation is organized as follows: Chapter 2 surveys prior work published in the literature that is relevant to the research reported in this dissertation. Chapter 3 develops the technical details of our zoomorphic design technique, Chapter 4 develops the technical details of our interchangeable components technique, and Chapter 5 develops the technical details of our approximate dissections technique. Chapter 6 presents our experiments with each of these three techniques and reports on the results obtained. Chapter 7 concludes the thesis and presents promising avenues for future work. Appendices A, B, and C present supplemental material associated with each of our three techniques.

CHAPTER 2

Related Work

In this chapter, we give a brief overview of research related to each of the three problems that we address in this thesis.

2.1 Zoomorphic Design

To our knowledge, no prior work in computer graphics has proposed or developed a computational approach to designing zoomorphic objects. However, our work is related to existing research on 3D shape modeling, optimization and analysis, mesh composition, and computational design.

3D Shape Modeling. Many methods have been developed to automatically or semi-automatically create novel 3D objects. Igarashi et al. (1999) introduce a sketching interface for designing 3D freeform models. Schmidt and Singh (2010b) and Takayama et al. (2011) developed interactive tools for transferring geometry and surface details between models. Funkhouser et al. (2004), Kraevoy et al. (2007) and Jain et al. (2012) introduce approaches for generating novel 3D models by combining the components of existing models. Chaudhuri and Koltun (2010); Chaudhuri et al. (2011) develop tools to automatically suggest components that could be attached to an existing model based on the model's shape or semantic attributes. Our work is similar in that we suggest animal objects to be added to base objects. However, their approach does not optimize for design factors in the final shape or take measures to ensure that the design restrictions of the original object are satisfied. Kalogerakis et al. (2012) introduce a probabilistic model for synthesizing plausible man-made objects in a category by combining components of existing objects in the same category. Since we combine shapes from different categories, our criteria for plausibility differ.

3D Shape Optimization. Several works alter the geometry of an existing object in order to optimize for certain criteria, such as stackability (Li et al., 2012), stability (Prévost et al., 2013), spinnability (Bächer et al., 2014), and aerodynamic characteristics (Umetani et al., 2014). Zheng et al. (2016) optimize man-made shapes to fit a given humanoid figure better. We optimize two objects jointly for how well they can be combined to create a zoomorphic object.

3D Shape Analysis. We focus on a few highly related works in the rich body of literature on 3D shape analysis. Laga et al. (2013) find semantic correspondences between shapes and identify functional regions on a shape by using a graph representation of the shape and a graph kernel score to identify shape regions with similar contexts. We use similar technical ingredients for a different purpose. Our graph kernel score quickly identifies shapes that are likely to result in good optimization. Zhang et al. (2008) conduct a search for a partial correspondence between two shapes, using a deformation energy associated with each correspondence to identify the best correspondence. In our correspondence search, the ultimate goal is not to identify a correspondence, but to perform a coarse exploration of a highly complex energy landscape as the first stage of our optimization process. Shapira et al. (2010) detect analogous parts between objects that may belong to different categories using a hierarchical segmentation based on the shape-diameter function. Our correspondence search may be regarded as a different way of finding analogous parts, with considerations particular to our problem. Our work contributes to structure-aware shape processing (Mitra et al., 2013) by introducing a general approach for ensuring that the addition of new geometry to an object does not violate the object's design restrictions.

Computational Design. Our work is a novel instance of a recent stream of research that addresses highly open-ended design problems by computer that have traditionally been the domain of artists and designers. Such problems range from computational interior design

(Yu et al., 2011; Merrell et al., 2011), to the design of clothes (Umetani et al., 2011), accessories (Igarashi et al., 2012), puzzles (Zhou et al., 2014), mechanical toys (Zhu et al., 2012), and even cities (Aliaga et al., 2008; Vanegas et al., 2012). We tackle the new problem of computationally designing zoomorphic objects.

2.2 Interchangeable Components

Mix-and-Match Toys possess interchangeable components that allow the user to change their appearance. These toys are often designed to form shapes with an abstract look which emphasizes the fact that they are assembled from components. In contrast, our approach aims to make shapes with a coherent appearance. The enduring popularity of these toys in the digital era demonstrates the appeal of physically creating new shapes from a collection of components. Indeed, research in developmental psychology finds that hands-on toys are an effective way for children to learn spatial reasoning and express their creativity (Bond, 2014; Golinkoff et al., 2004). Researcher Roberta Golinkoff advises parents to "look for [toys] that children can take apart and remake or reassemble into something different, which builds their imagination." To our knowledge, we are the first to introduce specialized software for the design of such toys.

Assembly-Based Modeling. Our work can be thought of as a physical realization of Assembly-based Modeling, a popular modeling paradigm in which new shapes are constructed by connecting components from existing shapes. Funkhouser et al. (2004) introduced the concept of Assembly-based Modeling. In their work, the shapes to extract components from are found by querying a database based on shape similarity to an existing shape. The user then interactively extracts the components through intelligent scissoring. In a work by Kraevoy et al. (2007) the extraction and composition of components was fully automated. Chaudhuri and Koltun (2010); Chaudhuri et al. (2011) introduced techniques for automatically suggesting components to be added to an existing shape, using the shape's geometric or semantic attributes. Jain et al. (2012) used assembly-based modeling and analysis of shape contacts to generate plausible blends between two existing shapes. Kalogerakis et al. (2012) introduced a fully automated method which used assembly-based modeling and a probabilistic model of component compatibility to synthesize plausible novel shapes from a database of existing shapes. In the virtual setting of these works, there is no need to enforce component interchangeability since a unique deformation can be computed whenever two components are connected. However, interchangeability is highly desirable in our physical setting, because it allows a small number of components to construct a large number of shapes. Hence these works focus on very different problems from ours.

Partitioning Shapes for Fabrication. Luo et al. (2012) proposed an approach to automatically partition a shape into components using a binary space partitioning tree, in order to maximize 3D printing efficiency. Hu et al. (2014) partitioned into pyramidal components. Chen et al. (2015) and Yao et al. (2015) also optimize for the component packing. Our work partitions shapes into fabricable components as well, but we determine the partition based on completely different criteria.

Shape Optimization for Fabrication. Several works optimize the geometry of an existing shape so that it possesses a desirable physical property when fabricated. The various properties examined include stability (Prévost et al., 2013), spinnability (Bächer et al., 2014), and aerodynamics (Umetani et al., 2014). These works solve physical problems, whereas our work deals with the geometric problem of generating interchangeable components.

Fabrication-aware Design. Several works introduced methods which assist the user in creating 3D designs suitable for fabrication. Umetani et al. (2012) introduced an interactive furniture design system that provided suggestions to help the user achieve a stable and durable design. Lau et al. (2011) proposed a method to convert non-fabricable furniture models to fabricable ones by parsing the models with a grammar and automatically adding connectors and hinges. Schulz et al. (2014) introduced a data-driven system in which parametrized components can be attached together to create designs suitable for fabrication. Koo et al. (2014) described a system that automatically creates a fabricable shape with mechanical parts that possess functional relationships specified by the user. In these works, the process of exploring the shape design space takes place in the virtual realm, whereas our work brings it into the physical world.

2.3 Approximate Dissections

The last 50 years have seen a surge of recreational interest in finding minimal-piece dissections between certain abstract figures, which are often regular polygons. These results are only for specific instances of the dissection problem, but some heuristic techniques for finding solutions have been identified. Frederickson (2003) did the seminal work in this area. The shapes used in these dissections are much simpler than those featured in our work on approximate dissections.

2D Shape-Guided Synthesis. Our work belongs to a family of graphics research in which a 2D shape guides the synthesis of some object, such as ASCII art (Xu et al., 2010b), mazes (Xu and Kaplan, 2007), Escher tiles (Kaplan and Salesin, 2000), calligrams (Zou et al., 2016), and connect-the-dot puzzles (Löffler et al., 2014). In our work, we are guided by two shapes and the object is a set of pieces that can approximate both shapes.

Computational Dissection Design. There has been a modest amount of research in computational techniques for solving the dissection problem and some similar problems. Zhou et al. (2012) introduced an algorithm for finding the minimum number of pieces for an *exact* dissection between shapes. Their method uses a voxel grid to represent the input shapes and a stochastic search strategy. In theory, a sufficient number of voxels could accurately capture the complex organic shapes targeted by our approach, but the results shown in their paper are limited to coarse voxel grids and simple shapes. Our continuous solution representation allows for more complex shapes. Zhou et al. (2014) proposed an algorithm that partitions a 3D shape into approximately cubic pieces and connects them with hinges so that it can be folded into a cube. Their problem statement is related to the dissection problem in that they try to partition an object into pieces so that it can transform to another shape. Unlike our approach they make no effort to minimize the number of pieces, instead focusing on finding a viable hinge connectivity. Huang et al. (2016) proposed a method that, given two shapes, partitions the first into pieces that can be connected through hinges to transform into a shape that roughly approximates the second. Unlike the former work, they do not require physical feasibility. They determine the partition and hinge connectivity of the first shape through a user-provided skeleton, which limits the generality of their solutions. Their work targets significantly coarser approximations than ours. Kwan et al. (2016) recently proposed a novel shape descriptor that can be used to solve the 2D collage problem. In this problem, the goal is to tightly pack shapes from a given library so that they approximate a larger shape. Their problem is similar to ours in that they approximate a shape with a set of pieces, but different in that the pieces are fixed and only need to form a single shape. Concurrent research by Song et al. (2017) explored the design of furniture that can reconfigured to form a different type of furniture, which is highly related to the dissection problem.

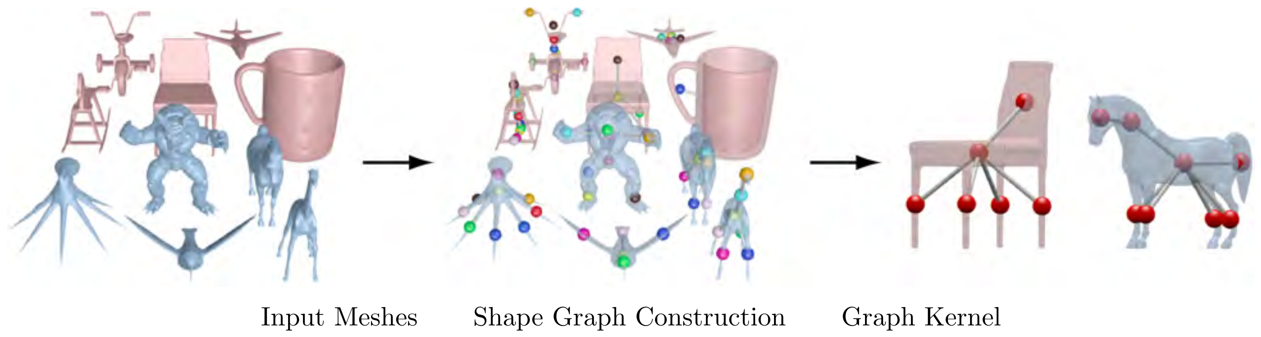
CHAPTER 3

Zoomorphic Design

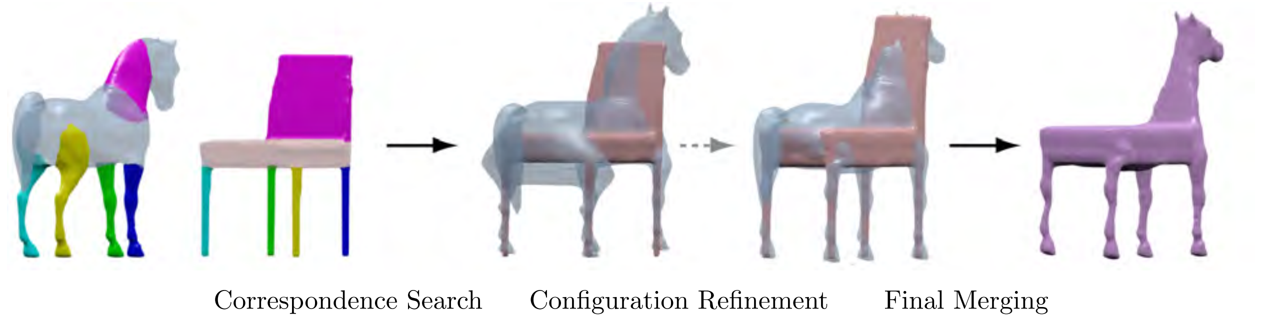
Figure 3.1 shows the main components and workflow of our approach to zoomorphic design. The first step is deciding what objects to use for the base object and animal object (Figure 3.1(a)). Our method efficiently identifies desirable pairings of a base object and animal object from a database, using a graph-kernel based method. It then merges the two objects by deforming, repositioning, and removing unwanted geometry from them (Figure 3.1(b)). We formulate this process as an optimization of several important design factors that include the prominence of the visually-salient regions of the animal object, the degree of distortion of the base objects and animal objects, and the smoothness of the transition between the base object and animal objects. We enable the user to adjust the weighting given to each factor, which provides high-level control over the resulting design. The process is guided by a novel technique, called the *Volumetric Design Restriction* (VDR), which ensures that the design restrictions of the base object are satisfied in the zoomorphic design.

3.1 Preprocessing

Input Data. The input objects take the form of triangular meshes. The input objects are divided into two categories, the animal objects and the base objects. We assume that the input objects are oriented upright and require a segmentation of all objects into semantically meaningful parts. For objects that have several similar objects in the database that have already been segmented, the segmentation can be transferred automatically using the method of (Kalogerakis et al., 2010). Nothing in our approach precludes a hierarchical segmentation,



(a) *Candidate objects Suggestion*



(b) *Zoomorphic object Creation*

Figure 3.1: Overview of our Zoomorphic Design approach.

but we used a simple segmentation to produce all the results shown.

Annotation. Each base object is annotated with the volumetric design restriction labels. These labels are used to describe design restrictions of the base object which should be preserved in the zoomorphic object (Section 3.3.1). This is done either manually or automatically using the method described in (Kalogerakis et al., 2010). Each animal object is annotated with the visual salience labels, which need not be precise, so the annotation can be done quickly. These labels are used to identify regions of the animal object which should be visible in the zoomorphic object (Section 3.3.3). The manual annotations need only be obtained once per input object rather than once per synthesis operation, so they are not overly time consuming.

3.2 Candidate Objects Suggestion

Given a database containing all the input objects, pairs of base objects and animal objects with high similarity scores are suggested as input candidates to create zoomorphic objects (Section 3.4). Our method performs this operation automatically using a graph kernel technique in which the input objects are represented as graphs.

3.2.1 Shape Graphs and Graph Kernels

For each object, which has already been segmented as described in Section 3.1, we construct a shape graph to capture the structural relationship between its segments. The similarity between different shapes can then be efficiently computed by comparing their respective shape graphs with a graph kernel.

The shape graph is constructed as follows: Each segment corresponds to a node. Each adjacency between segments corresponds to an edge. Each segment has some geometric attributes, which characterize the segment, such as the part scale or centricity. The attributes of a segment are stored in its corresponding node. Appendix A presents the details of all the attributes used.

Graph kernels are a general tool for measuring the similarity between two graphs (Kashima et al., 2004; Shawe-Taylor and Cristianini, 2004). We use graph walk kernels to compare the similarity between every base object and animal object pair, as illustrated by the example in Figure 3.2. A graph walk kernel evaluates the similarity of all pairs of p-walks on the shape graphs ($p = 3$ in our experiments), where a p-walk traverses $p + 1$ nodes and p edges. To evaluate the similarity between two walks, we employ a node kernel and an edge kernel to compute the similarity between the corresponding nodes and edges of the walks. Specifically, a node kernel takes two nodes as input and computes a similarity score using the attributes stored at the nodes. Analogously, an edge kernel takes two edges as input and computes a similarity score. A graph walk kernel can be evaluated efficiently by dynamic programming

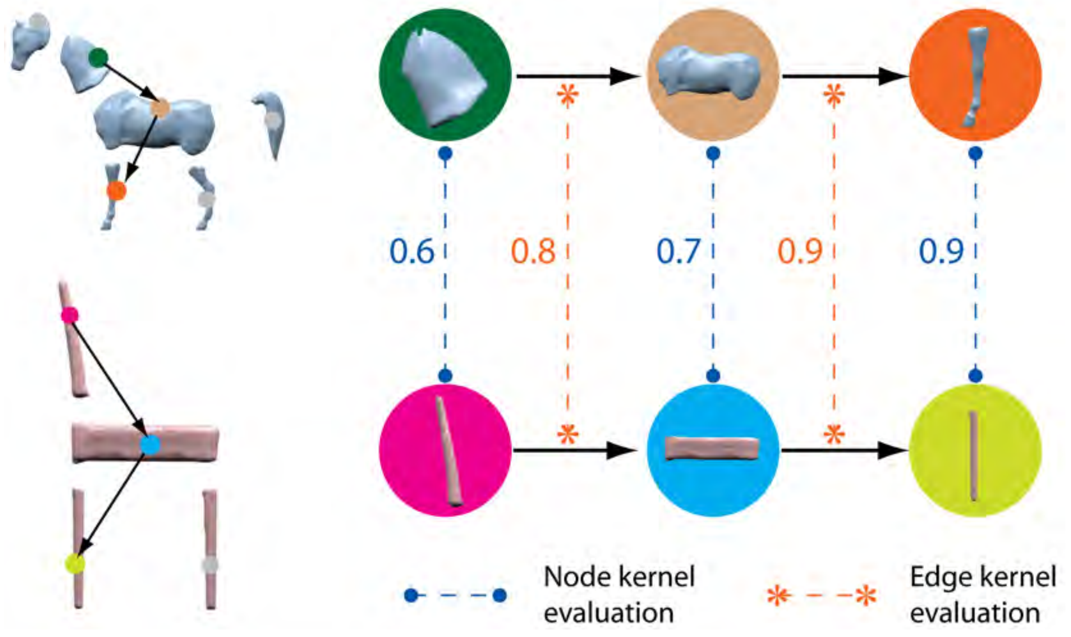


Figure 3.2: Example of a p-walk ($p = 2$) over the shape graphs of a horse and a chair. The node kernel and edge kernel evaluate the similarity between the corresponding nodes and edges. A larger number indicates a higher similarity.

(Shawe-Taylor and Cristianini, 2004).

We note that the graph kernel is conservative—it identifies some, but not all of the pairings that will result in a desirable zoomorphic object. In our approach, the animal object may deform itself considerably, which the graph kernel does not consider. Therefore, the user is free to ignore the graph kernel's suggestions and select pairings with low similarity scores. In our results showcase (Figure 6.1) each base object shown had a high similarity score (within ten percent of the highest score) with its paired animal object, with the exception of the go-kart.

3.3 Problem Formulation

Assuming that a base object and animal object pair has been selected, let us denote the base object as $\mathcal{M}_B$ and the animal object as $\mathcal{M}_A$. We first describe two important concepts used throughout our method—the "volumetric design restriction" and the "configuration

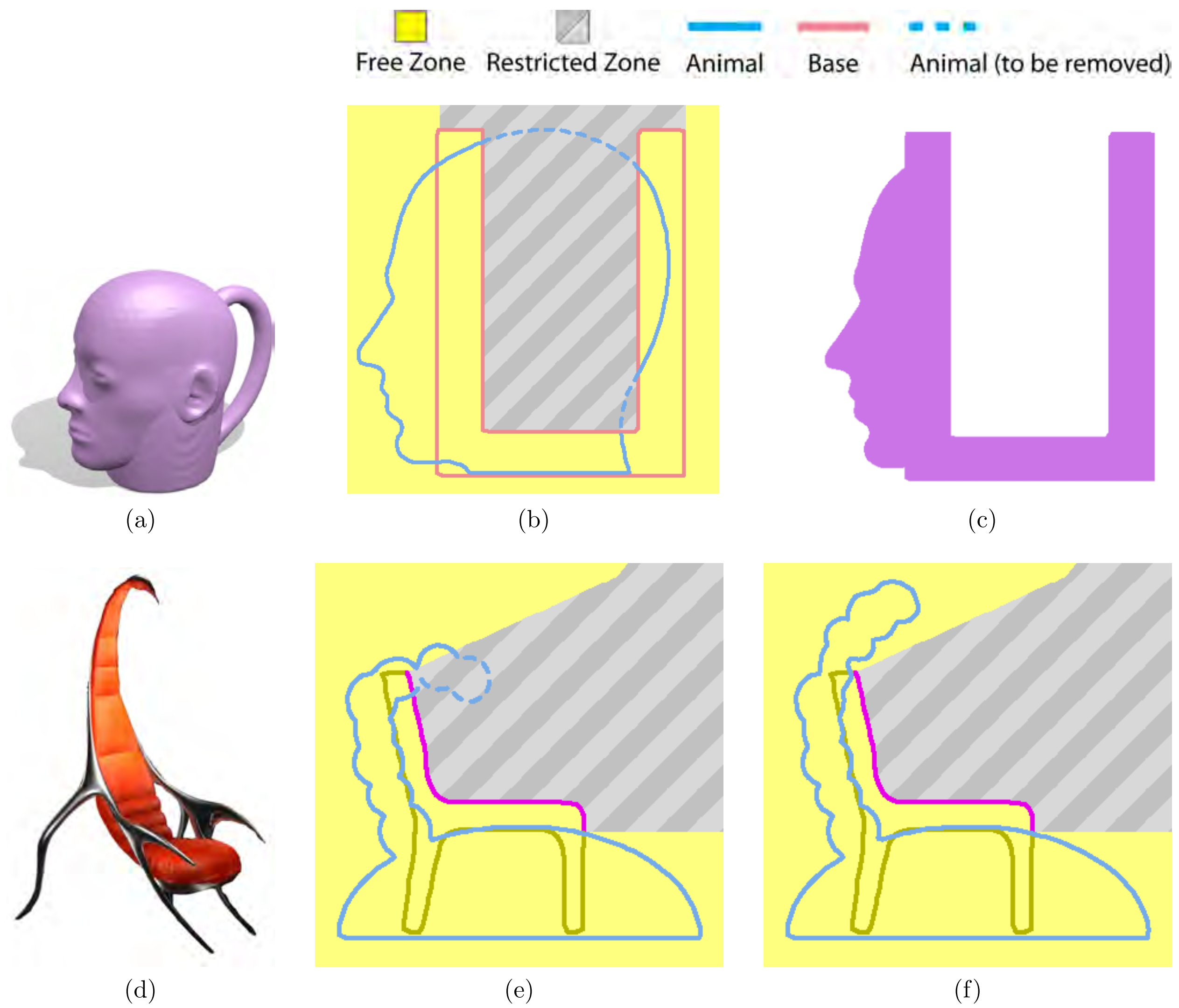


Figure 3.3: Examples of volumetric design restrictions. (a) A simple merge between the face and mug destroys the liquid-containing ability of the resulting zoomorphic object. (b) Objects under the volumetric design restriction. (c) The resulting zoomorphic object, which is still a container. (d) What poses for the insect's tail allow a person to sit on the chair? (e) The tail intrudes into a *restricted* zone and interferes with sitting. (f) A minor adjustment to the tail pose removes it from the zone and allows people to sit.

energy".

3.3.1 Volumetric Design Restriction

The base object, for example, a chair or a mug, usually possesses certain geometric features or structures that are crucial to its design and correspond to high level properties such as sittability or containability. For example, consider the face mug in Figure 3.3(a). The naive addition of the face conflicts with the restriction that the mug base object must hold liquid. For a more complicated example, consider the insect chair in Figure 3.3(d). In what poses does the insect's tail conflict with the design restriction that a person must be able to sit in the chair? The animal object can merge with the base object in a variety of ways, so it is important that the merge preserves these crucial properties on the base object, which we call "design restrictions". The problem of preserving certain qualities of a man-made object under geometric modification is uniquely challenging in our setting, because of the many ways the animal object can interfere with these qualities.

The volumetric design restriction (VDR) is a novel concept, which uses a labeling of the base object surface to specify the volume of space in which the presence of geometry from the animal object will violate the design restrictions of the base object (Figure 3.3(b),(e)). We call this volume of space the *restricted* zone, and the remaining volume the *free* zone. In creating a zoomorphic object, geometry from the animal object can lie in the *free* zone, but not in the *restricted* zone.

Labeling the base object to specify the *restricted* and *free* zones has two advantages over specifying the zones directly. First, the labeling means that as the base object deforms, the zones deform accordingly, which is necessary since our optimization procedure deforms the base object. Note that the zones are related to the geometry of the base object. For example, a container needs to preserve a zone for it to contain water. If the container widens, then this zone should also become wider. Second, once a labeling has been specified for several base objects in a category, we can train a classifier to transfer the labeling to other objects in that category (Kalogerakis et al., 2012). By default, the labels are generated manually. To assist users in the manual labeling task, we developed a basic user interface that displays the changes in the zones interactively as the user modifies the labeling.

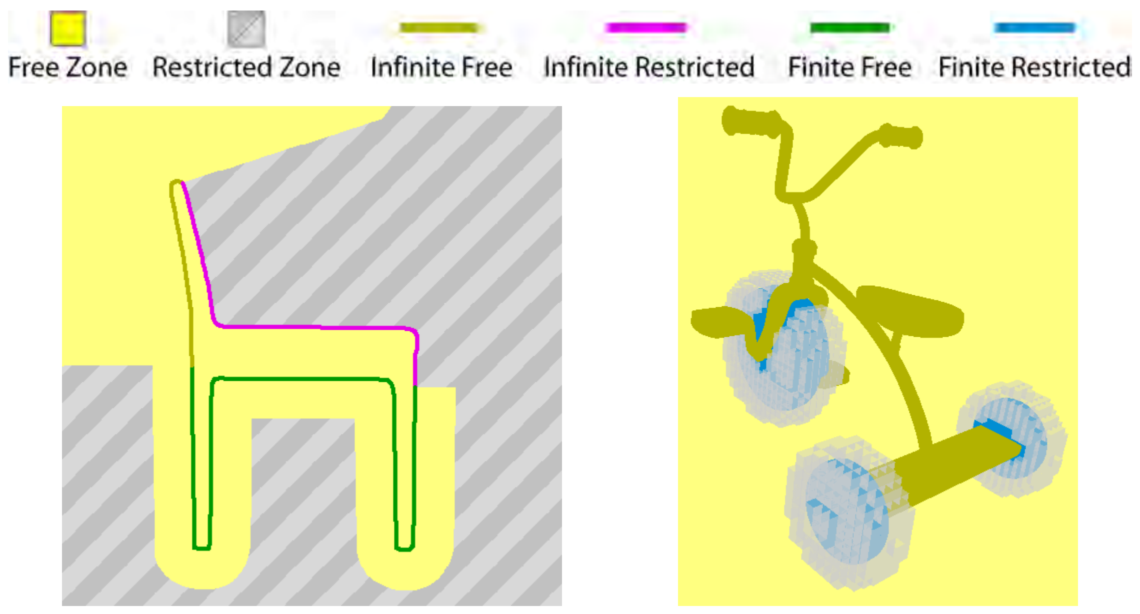
***Free* and *Restricted* Zones.** In our formulation, the label assigned to a face f on a surface mesh determines how the space around f is partitioned into *free* and *restricted* zones. We motivate our formal definition of this partitioning by showing four different cases—*Infinite Free*, *Infinite Restricted*, *Finite Free*, *Finite Restricted*, that arise in our problem of how to preserve a design restriction when adding new geometry to an object. Each case corresponds to a different way of partitioning the space.

Infinite Free: Consider adding geometry to the back of the chair in Figure 3.4(a). Here we do not place any restrictions on how much animal object geometry can be added (assuming we ignore the mass of the added geometry). Therefore, *all* the space around the surface is *free*.

Infinite Restricted: Consider the need to preserve a sitting region on the chair. Here, we must preserve the flatness of the surface so that sitting is comfortable and preserve the empty space around the surface so that a human can occupy it. These requirements mean that *all* the space around the surface is *restricted*.

Finite Free: Consider adding geometry to the legs of the chair. In contrast to the *Infinite Restricted* case, we do not care about preserving the flatness of the leg surface. In fact, adding geometry from the animal object would enhance the leg's appearance. However, we still want the leg to be roughly cylindrical in shape; e.g., we do not want animal object geometry that juts far out from the leg. In this case, space within a *finite* distance from the leg is *free* and space beyond that distance is *restricted*.

Finite Restricted: In Figure 3.4(b), consider the need for the wheels on the tricycle to spin freely and to have circular symmetry. We cannot allow any contact of the wheel with the animal object, but in contrast to the *Infinite Restricted* case, there is no need to preserve an empty space for a human to occupy. Here, the partition is the opposite of the *Finite Free*



(a) (b)

Figure 3.4: (a) Volumetric design restriction of a chair. Three segments (purple, olive, & green) have different outward zones corresponding to their label types (if all the *free* zones were filled with material, the resulting object would still be a chair). (b) Volumetric design restriction of a tricycle. The wheel segments (in blue) are painted with labels of the Finite Restricted case, which protect them from being covered by any animal object geometry (we illustrate only labels of the Finite Restricted case in this example).

case—a finite space around the surface is *restricted*, anything further out is *free*.

Note that for all cases, the space inside the base object is *free*, because if a space is already occupied by one object it doesn't matter if it is also occupied by another.

Let each face $f_i \in \mathcal{M}_{\text{base}}$ be assigned a label $l_i = (\alpha_i, \beta_i)$, where $\alpha_i \in \{1, -1\}$ denotes a zone type (*free* or *restricted*) and $\beta_i \in \mathbb{R}^+$ denotes a distance threshold from face f_i. Now consider an arbitrary point $\mathbf{p}$ in the space. Let $\hat{f}$ denote the closest face on $\mathcal{M}_{\text{base}}$ to $\mathbf{p}$. To determine the zone $r(\mathbf{p})$ assigned to $\mathbf{p}$, we compute the signed distance γ from $\mathcal{M}_{\text{base}}$ to $\mathbf{p}$. Suppose $\hat{f}$ has label $\hat{l} = (\hat{\alpha}, \hat{\beta})$. Then,

$$r(\mathbf{p}) = \begin{cases} 1 & \text{if } \gamma \leq 0, \\ \hat{\alpha} & \text{if } \gamma > 0 \text{ and } \gamma < \hat{\beta}, \\ -\hat{\alpha} & \text{if } \gamma > 0 \text{ and } \gamma \geq \hat{\beta}, \end{cases} \tag{3.1}$$

where $r(\mathbf{p}) = -1$ refers to the *restricted* zone and $r(\mathbf{p}) = 1$ refers to the *free* zone. The formula means that point $\mathbf{p}$ belongs to the *free* zone if it is inside the object; it belongs to zone type $\hat{\alpha}$ if it is outside the object and within the distance threshold $\hat{\beta}$ from $\hat{f}$; otherwise it belongs to the opposite zone type $-\hat{\alpha}$. Each of the four cases mentioned above corresponds to a labeling:

	$\beta < +\infty$	$\beta = +\infty$
$\alpha = -1$	*Finite Restricted*	*Infinite Restricted*
$\alpha = 1$	*Finite Free*	*Infinite Free*

Functionality. We briefly clarify the relationship of the VDR to functionality. We believe that the volumetric design restriction can preserve some types of functionality such as containability, graspability, or sittability. In general, it can preserve functionalities which are apparent from visually inspecting an object. In (Zheng et al., 2013), another work that deals with preserving functionality in man-made objects, this type of functionality was called "functional plausibility". However, the VDR cannot deal with more complex functionalities like stability, structural strength or aerodynamics which in general, cannot be determined from visual inspection alone. Our user study (Section 6.1.6) showed that the VDR has a major impact on whether our approach generates zoomorphic objects that are plausible examples of the category of man-made object they were derived from. Since plausibility is highly related to functionality for man-made objects this result supports our statement about functionality.

Examples. The volumetric design restriction can be applied to resolve the previously mentioned issues raised in the creation of zoomorphic objects. Figure 3.3(c) shows the face mug example. The *restricted* zone removes the geometry of the face that prevents the cup from holding water. Figure 3.3(f) shows the insect chair example. The *restricted* zone signals the animal object to alter its pose to preserve the chair's sittability.

The VDR has the additional advantage that it can be naturally integrated into our optimization framework. We discuss this in Section 3.4.

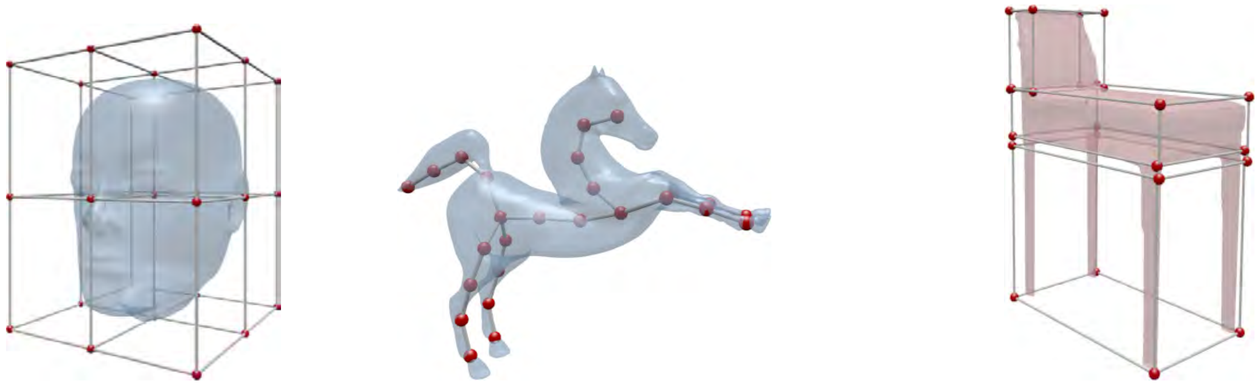

(a) FFD (b) LBS (c) Cuboids

Figure 3.5: Different deformation models. (a), (b) Animal object deformation models. FFD is used to deform a face (a) and LBS is used to deform a horse (b). (c) Base object deformation model. Each cuboid encloses a group of segments that are constrained to share the same transformation.

3.3.2 Deformation Models and Configuration

Unique to our problem is that a zoomorphic object is composed of a base object and animal object, which are generally an organic object and a man-made object. We allow both the animal object and base object to deform during the optimization process. The animal object and base object use different deformation models that are well-suited for organic and man-made objects, respectively.

Animal object Deformation Model. In general, different animal objects require different deformation models (Figure 3.5(a),(b)). Currently our approach supports two models—Linear Blending Skinning (LBS) and Free-form Deformation (FFD), and it may be extended to support others as well. Animal objects which use LBS are generally creatures with clearly defined limbs, such as squids or horses. In our model, the control parameters specify only the translations of the LBS handles. The remaining degrees of freedom are found by the method in (Jacobson et al., 2012). Their method also allows us to specify only a subset of the translations and find the rest automatically. LBS requires that the input mesh comes with a skeleton and weights. These can be found manually or automatically with (Baran and Popović, 2007) and (Tagliasacchi et al., 2012). Animal objects which use FFD are non-

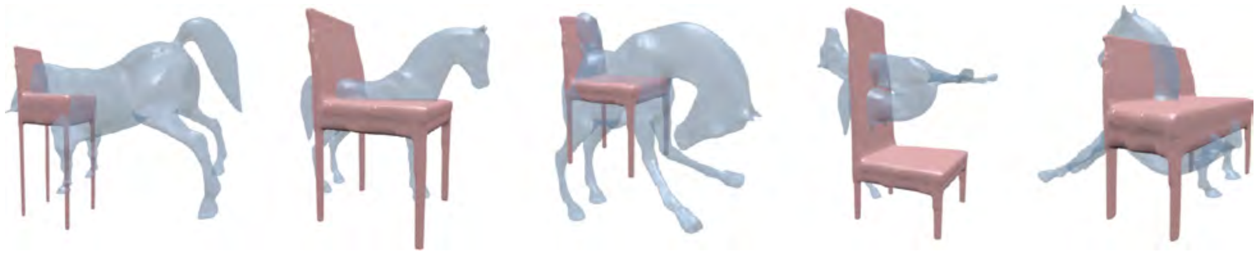


Figure 3.6: Different configurations. Each configuration ϕ encodes how the animal object $\mathcal{M}_A$ and base object $\mathcal{M}_B$ deform and position themselves to create a zoomorphic object.

articulated objects, such as faces, which are not well-described by a skeleton. We enclose the model in a FFD cubic lattice whose density can be specified by the user. Generally a 1×1 or 2×2 lattice offers enough control.

Base object Deformation Model. We use a model in which each segment in the base object $\mathcal{M}_B$ can be transformed by a scaling and translation (Figure 3.5(c)). The translations are constrained to preserve segment adjacencies. Groups of segments can be constrained to share the same transformation, which is useful for preserving symmetry and functionality, such as ensuring that the legs of a chair have equal length. Scales in different directions can be constrained to be equal, which is useful for ensuring that wheels remain circular. This deformation model is very simple, yet it provides a sufficient amount of freedom for a wide range of man-made objects.

Configuration. We define $\phi = (\phi_a, \phi_b)$, where ϕ_a and ϕ_b are the vectors of configuration parameters of the animal object and base object, respectively, to be the "configuration", which encodes how the animal object $\mathcal{M}_A$ and base object $\mathcal{M}_B$ deform and position themselves to create a zoomorphic object. The meaning of the parameters depend on the chosen deformation models. Figure 3.6 shows example configurations for a horse and chair.

3.3.3 Configuration Energy

We define a configuration energy to measure the desirability of the zoomorphic object resulting from a given configuration. A configuration that results in a desirable zoomorphic object should have a low energy. We identify desirable configurations by minimizing the configuration energy:

$$E(\phi, \mathbf{w}) = w^{\mathrm{a}}_{\mathrm{df}} E^{\mathrm{a}}_{\mathrm{df}}(\phi_a) + w^{\mathrm{b}}_{\mathrm{df}} E^{\mathrm{b}}_{\mathrm{df}}(\phi_b) + w_{\mathrm{r}} E_{\mathrm{r}}(\phi) + w_{\mathrm{vs}} E_{\mathrm{vs}}(\phi) + w_{\mathrm{g}} E_{\mathrm{g}}(\phi), \tag{3.2}$$

where $\mathbf{w} = [w^{\mathrm{a}}_{\mathrm{df}}, w^{\mathrm{b}}_{\mathrm{df}}, w_{\mathrm{r}}, w_{\mathrm{vs}}, w_{\mathrm{g}}]^T$ is a vector of weights. For fully automatic operation, setting all the weights to 1.0 generally produces a reasonable result. However, allowing the user to adjust these weights can lead to interesting changes in the designed zoomorphic object (see Section 6.1). We discuss the individual energy terms next:

Animal object Deformation. We penalize deformation of the animal object $\mathcal{M}_A$ by defining

$$E^{\mathrm{a}}_{\mathrm{df}}(\phi_a) = \frac{D(\phi_a)}{D_m} + C(\phi_a), \tag{3.3}$$

where $D(\phi_a)$ is the mesh deformation energy defined in (Sorkine and Alexa, 2007a) and D_m is a normalization term found by taking the median of the energies encountered during the correspondence search (see Section 3.4.1). The term $C(\phi_a)$ returns $+\infty$ if the deformation is so high that it is invalid, and 0 otherwise. For animal objects deformed using LBS, we define $C(\phi_a)$ in terms of the handle positions. Specifically, if any skeleton bone is stretched by more than a threshold, or if the angle between a pair of bones differs from the rest angle by more than a threshold, the deformation is invalid. For animal objects deformed with FFD, $C(\phi_a) = 0$.

Base object Deformation. This term penalizes non-uniform scaling of the base object. We formulate $E^{\mathrm{b}}_{\mathrm{df}}(\phi_b)$ as the sum of squared differences of all pairs of segment scales of the base object segments. Let $s_{i,u}$ be the segment scale of base object segment i with respect to axis $u \in \{x, y, z\}$. Then,

$$E^{\mathrm{b}}_{\mathrm{df}}(\phi_b) = \frac{1}{K^{\mathrm{b}}_{\mathrm{df}}} \sum_{i<j} \sum_{u,v} (s_{i,u} - s_{j,v})^2, \tag{3.4}$$

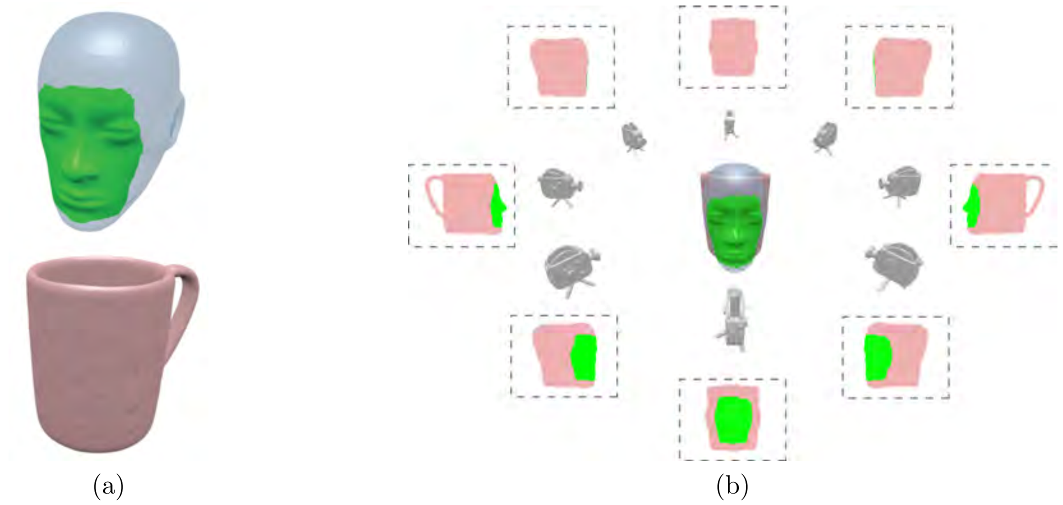


Figure 3.7: Computing the visual salience. (a) Input mug and face with visual salience annotations (green). (b) Images captured by 8 cameras looking at the objects from different viewpoints.

where $K_{\mathrm{df}}^{\mathrm{b}}$ is a normalization constant equal to the number of terms in the sum.

Registration. This term encourages the animal object and base object to align with each other. We compute the term over a set of uniformly sampled vertices $\mathcal{V}_{\mathrm{r}}$ from the animal object $\mathcal{M}_A$. Given a vertex $\mathbf{v} \in \mathcal{V}_{\mathrm{r}}$, define $\mathbf{n}_{\mathbf{v}}$ as the vertex normal and $d(\mathbf{v})$ as the distance function from $\mathbf{v}$ to the base object $\mathcal{M}_B$, and denoting the gradient of the distance from $\mathbf{v}$ to the base object $\mathcal{M}_B$ as $(\nabla d)_{\mathbf{v}}$,

$$E_{\mathrm{r}}(\phi) = 2 - \frac{1}{|\mathcal{V}_{\mathrm{r}}|}\sum_{\mathbf{v}} \exp(-\frac{d(\mathbf{v})^2}{{\sigma_{\mathbf{v}}}^2}) - \frac{1}{|\mathcal{V}_{\mathrm{r}}|}\sum_{\mathbf{v}} \exp(-\frac{\arccos(\mathbf{n}_{\mathbf{v}} \cdot (\nabla d)_{\mathbf{v}})^2}{{\sigma_{\mathbf{n}}}^2}), \tag{3.5}$$

where we set $\sigma_{\mathbf{v}}$ equal to 1/4 the diagonal of the bounding box enclosing both objects and $\sigma_{\mathbf{n}} = \frac{\pi}{4}$.

Visual Salience. This term encourages the appearance of visually salient regions from the animal object in the zoomorphic object. We assume the animal object $\mathcal{M}_A$ has had its visually salient regions labeled. The labeling can be provided automatically with existing methods (Lee et al., 2005) or manually by the user. Manual labeling offers greater control and need not be very precise. We penalize configurations that occlude or remove the visually salient regions of $\mathcal{M}_A$. We evaluate the degree of occlusion and removal by rendering the base object and animal objects deformed by configuration ϕ across a set of camera views Ω and measuring the area-weighted proportion of salient faces visible in each view. We do not render the parts of $\mathcal{M}_A$ that exist in *restricted* zones, as those parts of $\mathcal{M}_A$ will be absent in the synthesized zoomorphic object. Let $\mathcal{V}_{\text{vs}}$ be the set of visually salient faces in the animal object and let $\mathcal{V}_{\text{vs}}^{\omega} \subseteq \mathcal{V}_{\text{vs}}$ be the set of visually salient faces visible from camera view $\omega \in \Omega$. Our visual salience term measures the proportion of the visually salient regions which are visible:

$$E_{\text{vs}}(\phi) = -\frac{1}{\mathcal{A}(\mathcal{V}_{\text{vs}})\,|\Omega|}\sum_{\omega\in\Omega}\mathcal{A}(\mathcal{V}_{\text{vs}}^{\omega}), \tag{3.6}$$

where $\mathcal{A}()$ computes the total area of all the faces. Figure 3.7 depicts how we place the cameras. We uniformly arrange eight cameras in a circular-disc manner on a horizontal plane level to the objects, with the objects situated at the center. The cameras are at the minimum distance from the objects that allow them to see the entirety of the objects. Each visually salient face is rendered in a different color to detect if it is visible. Thus, the cameras capture eight images which are used to evaluate $E_{\text{vs}}(\phi)$.

Gash. In creating the zoomorphic object, any geometry from the animal object that intrudes into the *restricted* zones needs to be removed. This removal will create "gashes" on the animal object at the boundary between the *restricted* and *free* zones. On one side of the boundary, the animal object is preserved, while on the other side it is removed. Figure 3.8(a)–(b) shows a 2D example of a gash created on a horse when merging with a chair, and the resulting zoomorphic objects in 3D. In general, these gashes are aesthetically undesirable. The issue is resolved if the gashes occur inside the base object, because this conceals them from view. Therefore, our gash term penalizes only the visible gashes (Figure 3.8(b)). We explicitly define when a gash is visible in our gash energy formulation. In 3D, a gash is a surface. This surface extends into the interior of the animal object. To calculate the area of this surface, we would need a volumetric representation of the animal object, which would be costly to work with. Instead, we identify the intersection of the gash surface with the animal object's

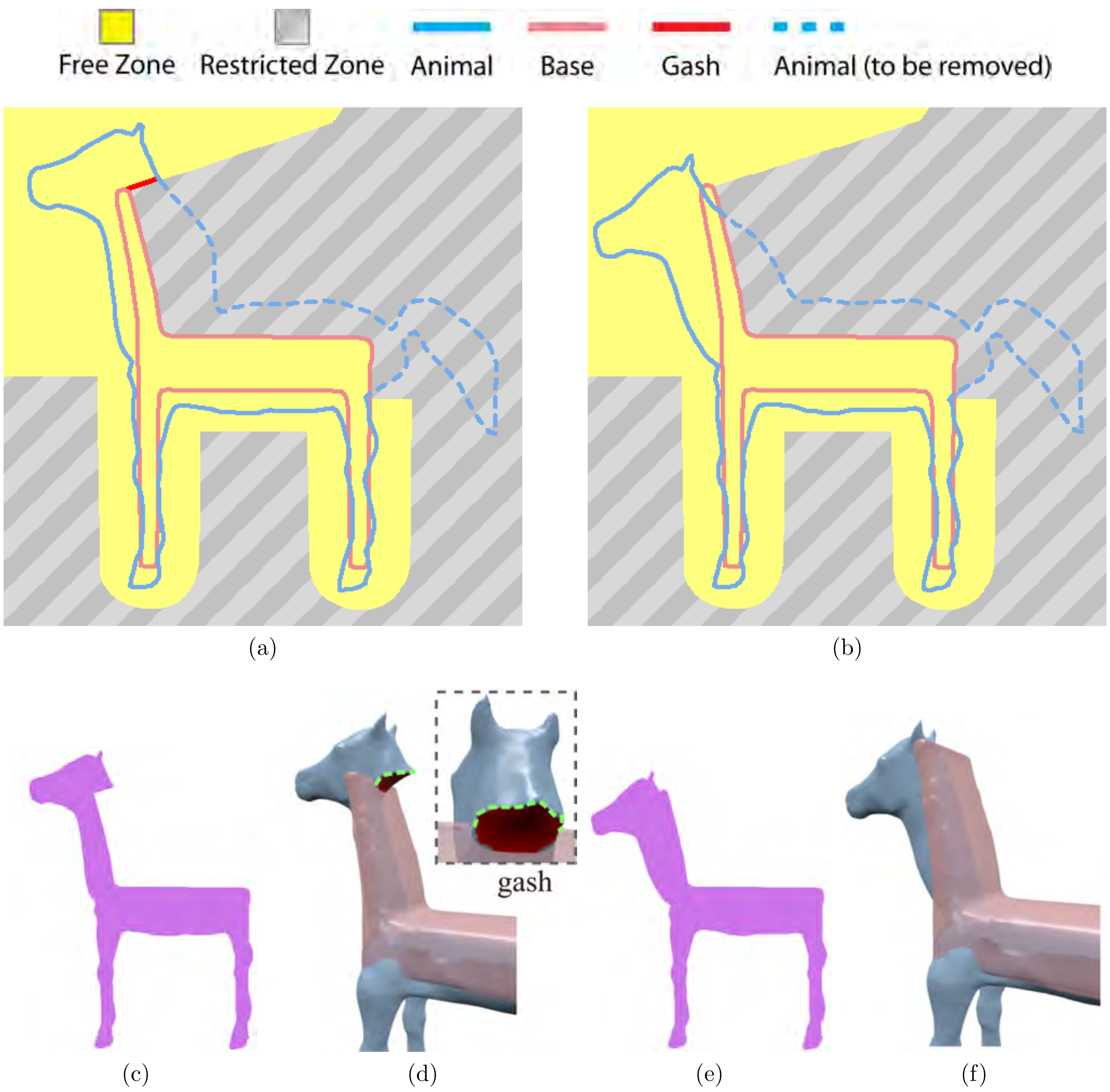


Figure 3.8: Considering gashes when creating a zoomorphic object. (a) After removing the portion of the animal object in the *restricted* zone, a gash (red) will appear. (b) Our optimizer bends the horse's head slightly down and raises the chair's back to avoid the formation of a gash. (c) Zoomorphic object with a gash. (d), (e) Our optimizer removes the gash surface (red) by minimizing the length of the intersection (green) between the gash surface and the animal object's surface. (f) Zoomorphic object without a gash.

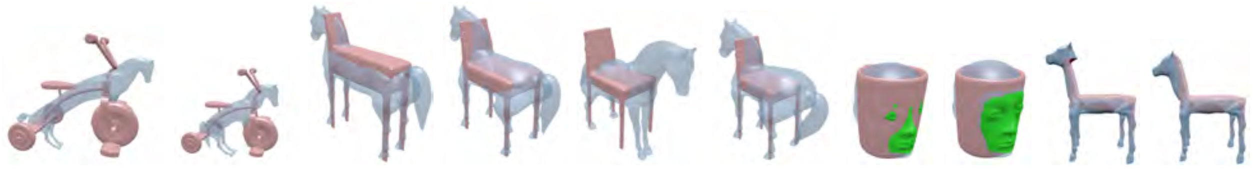

(a) $\mathcal{M}_A$ Deformation (b) $\mathcal{M}_B$ Deformation (c) Registration (d) V. Salience (e) Gash

Figure 3.9: Effects of omitting an energy term. Each subfigure shows the result without (left) and with (right) an energy term. (a) Without the animal object deformation term (left), the horse's body is stretched out excessively to match the tricycle's frame. (b) Without the base object deformation term (left), the chair's base is squeezed to match the horse's body. (c) Without the registration term (left), the horse is not well-aligned with the chair. (d) Without the visual salience term (left), the face's details are not revealed in the merge. (e) Without the gash term (left), the resulting zoomorphic object shows a gash.

surface, which will be a curve whose length is efficiently computable. We can identify these curves by examining the vertices of the animal object $\mathcal{M}_A$. We search for pairs of adjacent vertices $(\mathbf{m}, \mathbf{n})$, where $\mathbf{m}$ is in a *free* zone and $\mathbf{n}$ is in a *restricted* zone, and both vertices are outside the base object. These pairs of vertices are at the locations where the gashes on $\mathcal{M}_A$ occur in the final merge, and are not concealed by the base object from view, ie: the gash is visible (if $\mathbf{m}$ were inside the base object then the gash would not be visible). For each such pair, we mark $\mathbf{m}$ as a *gash vertex*. Let $\mathcal{V}_{\text{g}}$ be the set of all gash vertices. Summing the lengths of all edges connecting gash vertices gives us the length of all intersections, which we use as the gash term:

$$E_{\text{g}}(\phi) = \frac{1}{K_{\text{g}}} \sum_{\substack{\mathbf{v}' \in \mathcal{N}(\mathbf{v}) \\ \mathbf{v}, \mathbf{v}' \in \mathcal{V}_{\text{g}}}} ||\mathbf{v} - \mathbf{v}'||, \tag{3.7}$$

where $\mathcal{N}(\mathbf{v})$ is the set of vertices adjacent to $\mathbf{v}$, and K_{g} is a normalization constant equal to the largest geodesic distance between two vertices in $\mathcal{M}_A$.

Figure 3.9 illustrates the negative effects of dropping any of the terms of the configuration energy function.

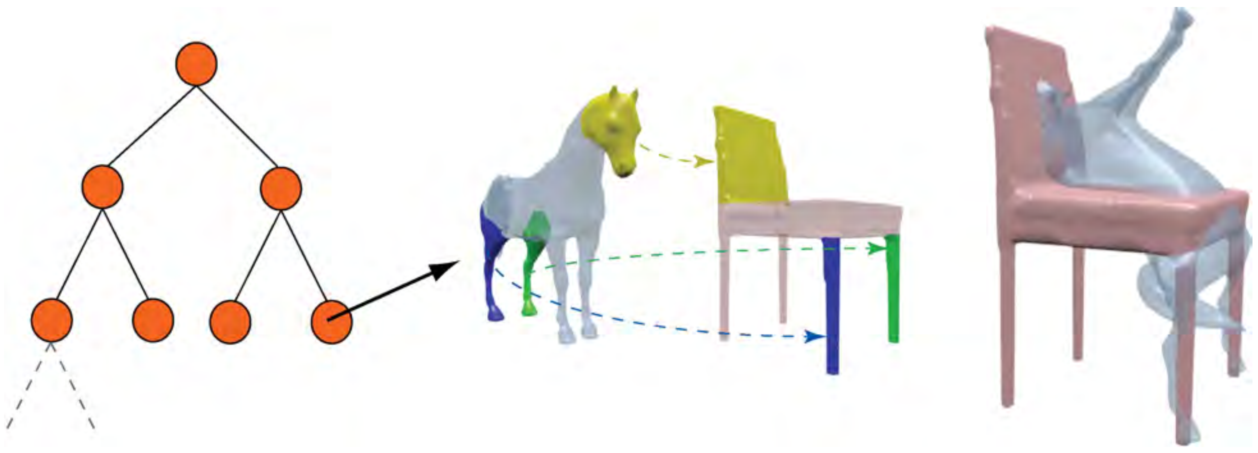
3.4 Zoomorphic Object Creation

Given a base object and animal object pair, we need to search for a desirable way to arrange them before merging them to create a zoomorphic object. We do this by minimizing the energy in Section 3.3.3 in a coarse-to-fine manner. In the coarse stage, we find a good correspondence between the two objects. This coarsely aligns the animal object with the base object and gives us an initial configuration. In the fine stage, we refine the initial configuration with continuous optimization. This stage expands the degrees of freedom of the optimization to resolve situations that require subtle adjustments. Finally, unwanted geometry is removed from the two objects and the resulting objects are merged by a union operation in the volumetric space to create the zoomorphic object.

3.4.1 Correspondence Search

A correspondence is a set of pairs of corresponding segments from the animal object $\mathcal{M}_A$ and base object $\mathcal{M}_B$. We define a correspondence as $c = \{(a_i, b_i), i = 1, \ldots, N\}$, where segment a_i in the animal object $\mathcal{M}_A$ has a corresponding segment b_i in the base object $\mathcal{M}_B$, and N is the total number of pairs of corresponding segments.

The goal of the correspondence search is to coarsely explore the configuration energy landscape, by explicitly associating each correspondence with a configuration. Good correspondences are associated with low-energy configurations, which can be used as initialization points for the fine-scale energy minimization. We justify our search by the observation that many zoomorphic objects have an "implicit correspondence", where parts of the animal object coincide spatially with corresponding parts of the base object. Hence, explicitly searching for a good correspondence provides us with a reasonable strategy for globally exploring the configuration energy landscape. Note that the configurations at this stage need not be optimal. We merely need to identify a rough pose of $\mathcal{M}_A$ that can be refined later (Section 3.4.2).



(a) *Search Tree* **(b)** *Correspondence* **(c)** *Configuration*

Figure 3.10: (a) Each node of the search tree refers to a correspondence. (b) A correspondence between a horse and a chair; corresponding segments have the same color. (c) The configuration induced by this correspondence has a high energy due to the large deformation.

We use a combinatorial tree search similar to (Zhang et al., 2008). As Figure 3.10 illustrates, each node of the tree stores a correspondence between $\mathcal{M}_A$ and $\mathcal{M}_B$. A node is expanded into its child nodes, where each child node has a correspondence c_{child} equal to its parent node's correspondence c_{parent} plus a new pair of corresponding segments (a_k, b_k). That is, $c_{\text{child}} = c_{\text{parent}} \cup \{(a_k, b_k)\}$.

For notational convenience, let us define $\phi(c)$ as the configuration associated with correspondence c. Then $\phi(c)$ should deform $\mathcal{M}_A$ such that each segment a_i of $\mathcal{M}_A$ is aligned with its corresponding segment b_i of $\mathcal{M}_B$. To find $\phi(c)$, we first apply local transformations that align each animal object segment a_i in c to its corresponding base object segment b_i. Since in general, c is a partial correspondence, we still need to find a deformation for the animal object segments that were not in c. We find this deformation by minimizing the ARAP energy (Sorkine and Alexa, 2007a) over the entire animal object with the locally transformed segments constrained to be fixed. Note that the local transformations are completely determined by their segment pair a_i, b_i, so they can be computed in advance for each possible segment pair. In our implementation we use non-rigid ICP to compute the local transformations, but other methods may be used.

Our search starts from the root of the tree. It computes the configuration energy of each

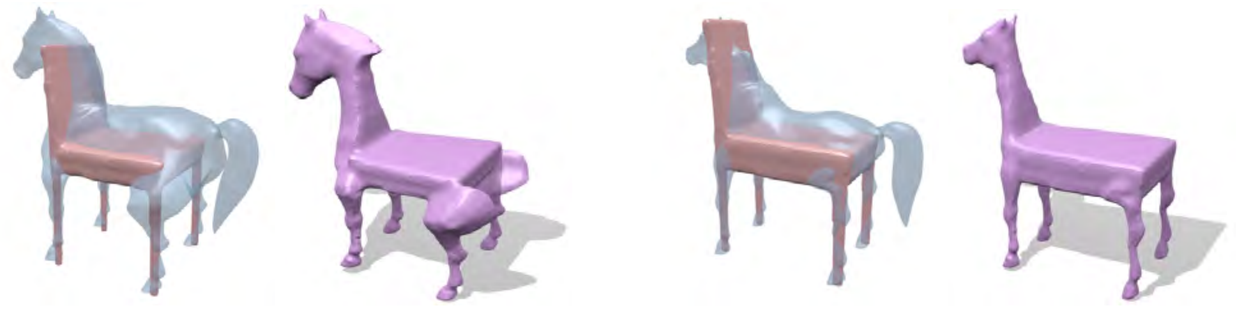

(a) After Correspondence Search (b) After Configuration Refinement

Figure 3.11: Configurations and zoomorphic objects generated after correspondence search and after configuration refinement.

visited node. If a node stores a correspondence that is associated with an infinite energy (this situation occurs when the animal object deformation is infeasible), we prune its subtree. Our search also only considers correspondences that satisfy bilateral symmetry.

The output of this stage is a set of correspondences and their associated configurations. We select the configuration $\hat{\phi}$ with the lowest energy and the associated correspondence $\hat{c}$ for the next stage.

3.4.2 Configuration Refinement

In this step, we refine the rough configuration given from the correspondence search. We optimize the full set of control parameters ϕ_a and ϕ_b of both $\mathcal{M}_A$ and $\mathcal{M}_B$ using $\hat{\phi} = (\hat{\phi_a}, \hat{\phi_b})$ as the initialization point. The registration term $E_{\mathrm{r}}(\phi)$, gash term $E_{\mathrm{g}}(\phi)$ and visual salience term $E_{\mathrm{vs}}(\phi)$ are non-differentiable, and the full energy landscape usually has many local minima. Therefore, we minimize the configuration energy using Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Lozano, 2006). We terminate once the configuration energy has not decreased by more than 10% in the last 25 iterations.

The modifications made to the configuration in this stage are small in absolute terms, but they usually improve the final zoomorphic object's appearance significantly. In particular, this stage has enough degrees of freedom to make the delicate adjustments necessary to lower the gash term $E_{\mathrm{g}}(\phi)$, which has a large impact on the zoomorphic object's appearance. Figure 3.11 shows the configuration and object of the Horse Chair after correspondence search and after configuration refinement.

3.4.3 Removals and Merging

Once the input objects are in their final configuration, we remove certain regions from both objects and then merge them to form a zoomorphic object. To do the removals, we first convert the objects to their volumetric representations.

For the animal object $\mathcal{M}_A$, we remove any of its geometry that lies in the restricted zones given by the VDR. The base object removals are motivated by the observation, that in many cases, the appearance of the zoomorphic object is improved if a segment of the base object does not appear in the zoomorphic object, because it has been "replaced" by a corresponding segment in the animal object. For example, when designing the Horse Chair, we want the original chair legs to be removed, so that they can be completely replaced by the horse's legs. Our formal procedure is to remove a segment b_i on $\mathcal{M}_B$ if all the following conditions are met:

1. Removing b_i will not reveal a gash on $\mathcal{M}_A$.
2. There should be a segment a_i from $\mathcal{M}_A$ that replaces the b_i to be removed (e.g., a horse's leg replacing a chair's leg). This is done by checking if b_i was assigned a corresponding segment a_i from $\mathcal{M}_A$ in the correspondence search.
3. The corresponding segment a_i should also overlap with each of b_i's adjacent segments (e.g., a horse's leg replacing a chair's leg should also overlap with the chair's base). We detect this overlap using the volumetric representation of the segments. Figure 3.13 shows an example which motivates this requirement.

After performing removals, we take the union of the volumetric representations as the final output. To make the transition between the objects appear more natural, we apply several

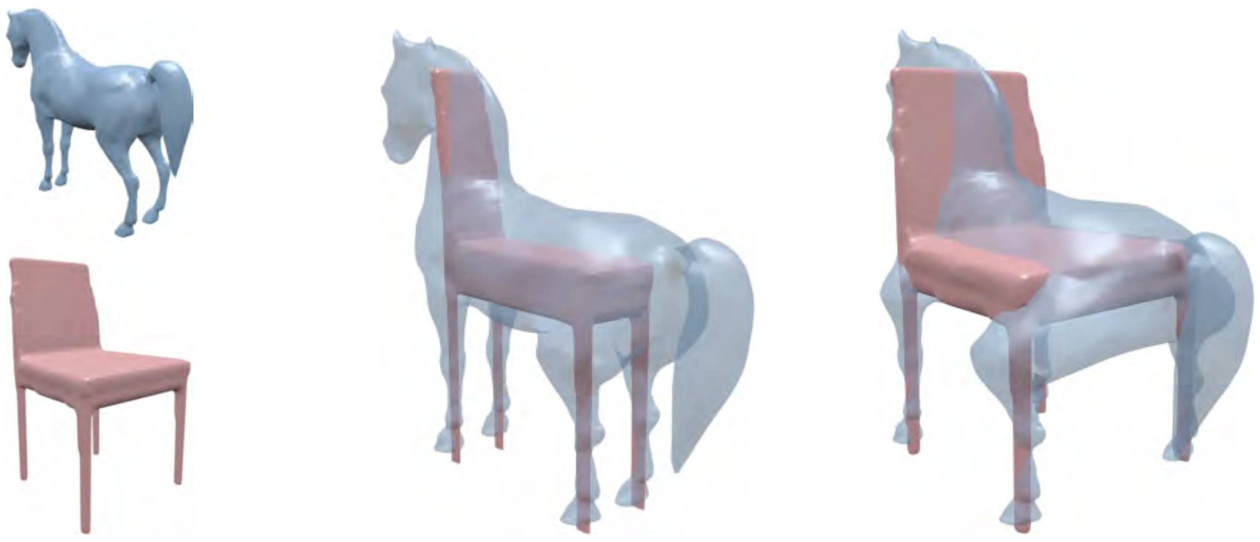

(a) Input objects (b) Small Weight (c) Large Weight

Figure 3.12: Effects of weight w_{df}^{b} in base object control. (a) Input objects. (b) A small weight causes the chair (base object) to deform strongly to match with the horse. (c) A large weight causes the horse to deform strongly to match with the chair.

iterations of implicit smoothing to the volumetric representation of the zoomorphic object at the locations where the base object and animal object intersect. Finally, the volumetric representation is converted to a manifold mesh using a standard method such as Marching Cubes.

3.5 User Control and Enhancements

One of the goals of our approach is to allow the user to direct the high-level design of zoomorphic objects. We provide several ways for the user to do so.

Correspondence Search. The user can explore the design space by adjusting the weight vector $\mathbf{w}$ used to select $\hat{\phi}$. Figures 6.1(i)–(k) shows different Horse Chairs created using different visual salience weights. Figure 6.1(l) shows two Rocking Cows created with different animal object deformation weights.

Base object Control. After the correspondence search returns an initial configuration $\hat{\phi}$ and a correspondence $\hat{c}$, we allow the user to refine the configuration by adjusting the trade-

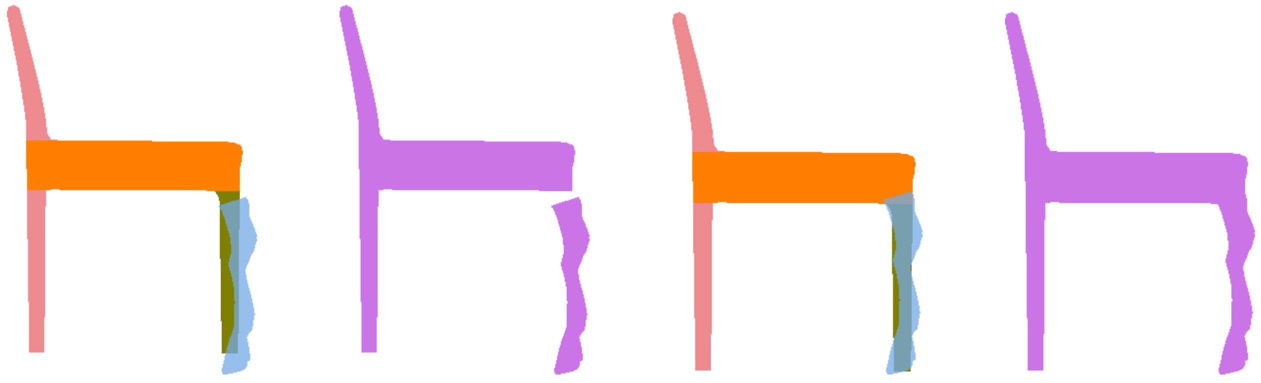
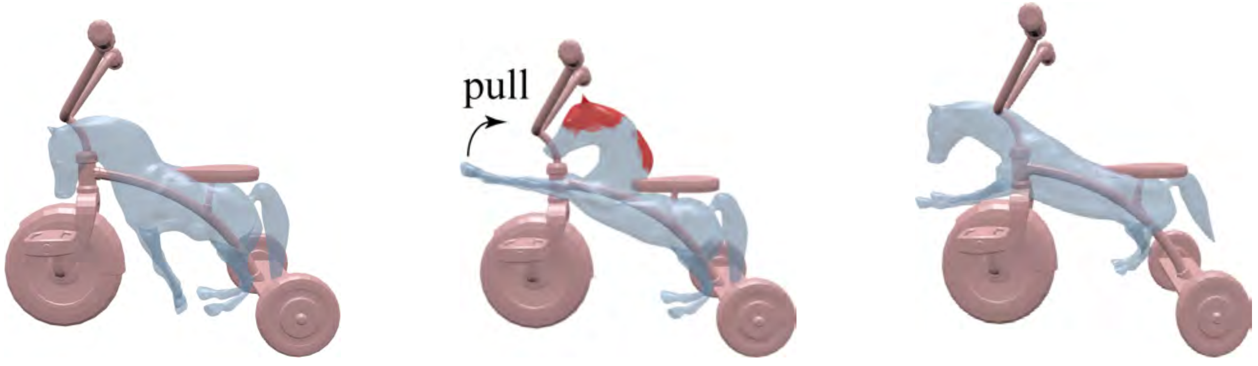


Figure 3.13: Explanation of the overlap criterion for removing base object segments. The chair leg is adjacent to the chair base. (a) The horse leg does not overlap with the chair base. (b) If we remove the chair leg and generate the zoomorphic shape, the horse leg is disconnected. (c, d) When the horse leg does overlap with the chair base, the zoomorphic shape is contiguous.

off between base object and animal object deformation. The user visualizes this trade-off interactively. Our user interface includes a slider that adjusts the base object deformation weight $w^{\mathrm{b}}_{\mathrm{df}}$. After the weight changes, we alter the configuration $\hat{\phi}$ to minimize the configuration energy using gradient descent. We optimize only with respect to the base object control parameters $\hat{\phi}_b$; i.e., the scales of the cuboids. We deform $\mathcal{M}_A$ accordingly whenever $\mathcal{M}_B$ is deformed, such that its segments still align with the corresponding segments in $\mathcal{M}_B$ given by $\hat{c}$. This deformation is obtained by applying the affine transformations from the base object deformation model (Section 3.3.2) to their corresponding segments in the animal object. We show the effects of changing $w^{\mathrm{b}}_{\mathrm{df}}$ in Figure 3.12. Since we only alter the base object part scales and scale the animal object accordingly in this stage, the registration, visual salience, and gash terms will not change significantly from their values in the original configuration $\hat{\phi}$. Therefore we make our optimization more efficient by recomputing only the animal and base object deformation energy terms.

User-Guided Refinement. Inspired by the user interface in (Prévost et al., 2013), we allow the user to stop the optimization process, change the pose of the animal object, and then resume the optimization. Figure 3.14 shows an example of altering a horse's pose.



(a) Initial Configuration (b) User Edit (c) Further Optimization

Figure 3.14: Our optimizer refines the configuration upon user edit. (a) Initial configuration. (b) During user edit, the frontal legs are pulled up by the user, causing intrusion (red) into the *restricted* zone according to the volumetric design restriction. (c) The optimizer further optimizes the configuration to avoid the intrusion.

Segment Removal. For more flexibility, we allow the user to manually specify any segment to be removed in the creation of the zoomorphic object. For example, in creating the horse chair shown in Figure 3.11, the user specified that the segment of the chair's back should be removed. The effect of removing a segment can be easily previewed in our user interface.

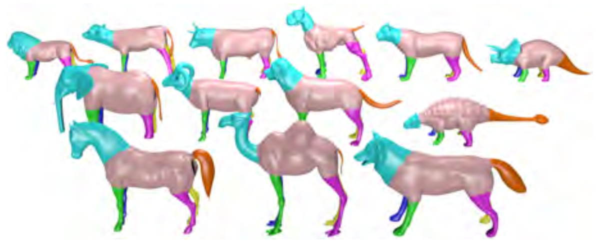
CHAPTER 4

Interchangeable Components

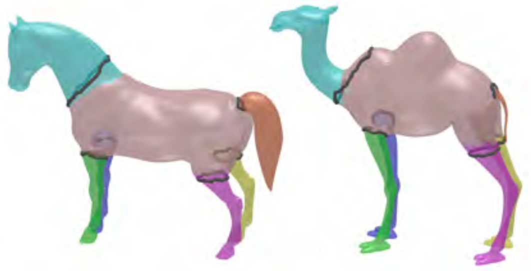
Figure 4.1 shows an overview of our Interchangeable Components approach for hands-on assembly-based modeling. Our goal is to partition a set of input shapes into components which can be physically connected to form novel objects, while minimizing the visual impact of the junctions between components.

4.1 Representation, Formulation, and Approach

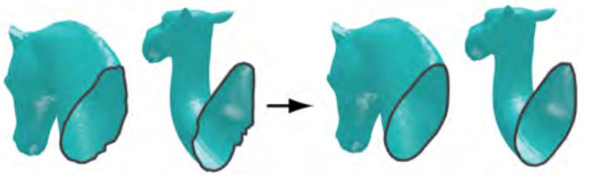
Representation. Our input shapes are represented as triangle meshes. We assume the meshes are aligned with consistent front-back and top-down directions and scaled to a consistent size. Each mesh is assumed to have been segmented into semantically meaningful regions. We clarify that the task of our approach is to *adjust* the borders between semantic regions in order to maximize geometric compatibility. The task of determining what the semantic regions are is a separate, open problem in Computer Graphics. The semantic segmentation can be obtained automatically using a data-driven approach (Kalogerakis et al., 2012), a fully automatic geometric approach (Sidi et al., 2011) or with an interactive tool. The semantic segmentation guides how the input meshes should be partitioned into interchangeable components and specifies the component connectivity. Components corresponding to the same semantic border (e.g., between the body and leg in Figure 4.2) from different meshes should be interchangeable. For example, the camel's leg can be disconnected from the camel's body to replace the horse's leg. For the results shown, the input segmentations for the animals (Section 6.2.1) were obtained with the data-driven approach



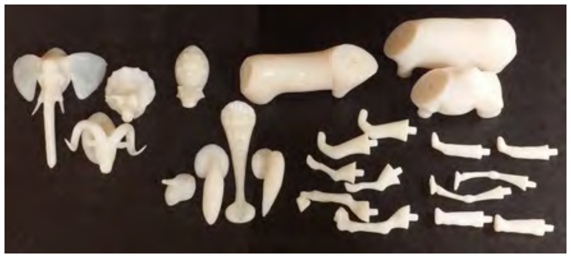
(a) *Input Compatibly Segmented Shapes*



(b) *Find Individual Edge Loops*



(c) *Deform to Common Edge Loops*



(d) *Generate Interchangeable Components*

Figure 4.1: Overview of our Interchangeable Components approach.

of (Kalogerakis et al., 2012), while the others were obtained interactively. Experiments presented in Sections 6.2.2 and 6.2.3 suggest that if the input segmentation is reasonable, the results of our approach will also be reasonable, i.e., the approach is not heavily sensitive to the input segmentation.

Each semantic border (between two segments) on each mesh forms an *individual edge loop*, as depicted in Figure 4.2(a). Note that the individual edge loops for the same semantic border on different meshes are different in shape. We describe the segmentation of each mesh m by a set of individual edge loops, $\mathcal{L}_m = \{l_{m,b}\}$, where individual edge loop $l_{m,b}$ is extracted from semantic border b of mesh m.

To ensure that corresponding components from different meshes are interchangeable, we need corresponding individual edge loops to form a common shape (up to rigid transformation and small differences due to different mesh tessellation). Figure 4.2(b)–(c) shows an illustration. To achieve this common shape, we will create a *common edge loop* to link up all the individual edge loops for each semantic border. We denote the common edge loop for the semantic border b as $l_{c,b}$.

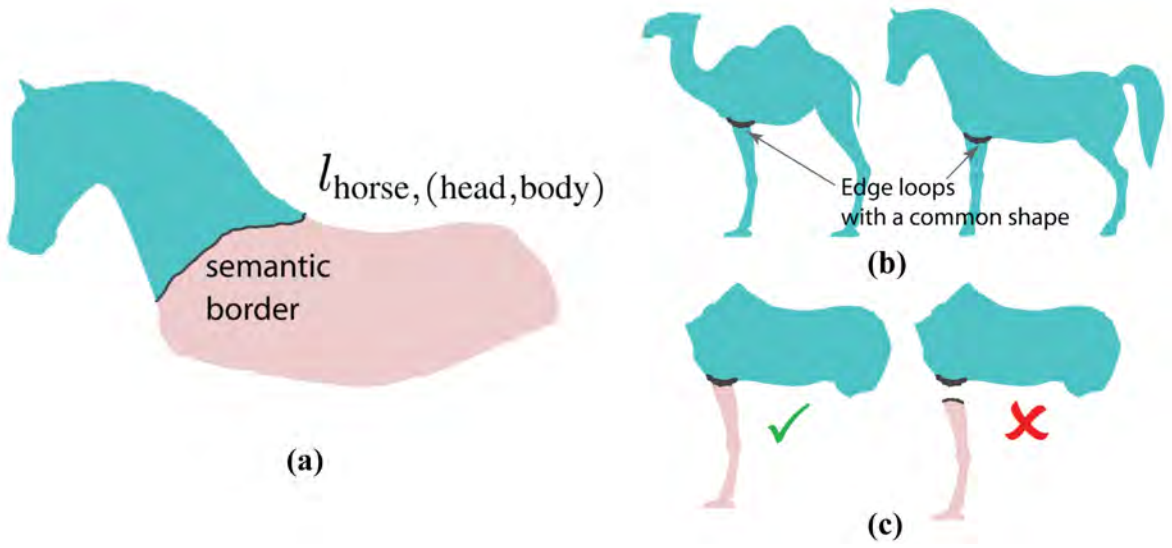

$l_{\text{horse},(\text{head},\text{body})}$
semantic border
(a)
Edge loops with a common shape
(b)
✓
✗
(c)

Figure 4.2: (a) The individual edge loop $l_{\text{horse},(\text{head},\text{body})}$ lies along the semantic border originally. (b) Edge loops with a common shape are used to partition the front leg and body for both the horse and camel. (c) The camel leg can replace the horse leg if their individual edge loops have the same shape. The camel leg cannot replace the horse leg if their individual edge loops have different shapes.

Problem Formulation. Given a set of input meshes $\mathcal{M}$ and a set of individual edge loops $\mathcal{L}_m$ for each mesh $m \in \mathcal{M}$, our approach outputs a set of deformed meshes $\mathcal{M}'$ and a set of modified individual edge loops $\hat{\mathcal{L}}_m$. The modified individual edge loops partition each deformed mesh $m' \in \mathcal{M}'$ into fabricable components. Our outputs should possess the following properties:

1. $\mathcal{M}' \sim \mathcal{M}$. Each deformed mesh $m' \in \mathcal{M}'$ should be similar to its corresponding input mesh $m \in \mathcal{M}$.

2. For any semantic border b and any deformed mesh $m' \in \mathcal{M}'$, we should have $l_{m,b} = \mathbf{R}(l_{c,b})$, where $\mathbf{R}$ is a rigid transformation. That is, all the individual edge loops for a given semantic border should possess a common shape up to rigid transformation. This condition ensures that our components are interchangeable.

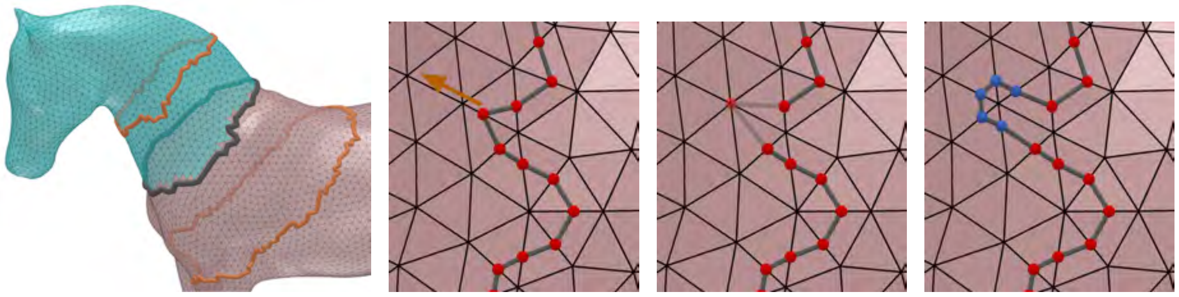
Technical Approach. Our approach proceeds as follows: First, we adjust the individual edge loops at the semantic borders of the input meshes in order to optimize their geometric similarity while not deviating too much from the original segmentation. Next, we deform the input meshes so that the individual edge loops take on the shape of a common edge loop. Finally, the deformed meshes are partitioned by the individual edge loops into interchangeable components, which are sealed and augmented with connectors that allow them to be assembled in the real world.

4.2 Finding Individual Edge Loops

Consider a semantic border b (e.g., between the head and body in Figure 4.3(a)). Given the initial edge loops $l_{m,b}$ for each mesh, this step searches for new edge loops $\hat{l}_{m,b}$ that are geometrically similar to one another. We have this goal because the next step will deform the meshes so that the corresponding individual edge loops form a common shape, and we want to minimize this deformation. Note that this step updates the edge loops by modifying the list of vertices specifying the loop, but not by altering the vertex positions.

Contour-based Formulation. The problem of adjusting the individual edge loops is discrete in nature. In order to avoid a combinatorial optimization over possible edge loops, we re-formulate the problem in a continuous setting. In this setting, we have the following problem: given a set of surfaces and a set of closed contours that lie on each surface, adjust the contours to maximize their geometric similarity. The contours are a continuous representation of the edge loops.

To prevent the contours from deviating too severely from the original semantic borders, we create a zone on each surface called the *intermediate zone* in which the contour is constrained to lie. The intermediate zone between two semantic regions represents the area that does not clearly belong to one region or the other. Figure 4.3(a) shows the intermediate zone between a horse's head and body. By default we set the intermediate zone with a simple procedure. We set each semantic region's portion of the intermediate zone equal to the faces in that region that lie within a geodesic distance d of the original semantic border. We determine d by increasing it until the area of the included faces exceeds 50% of the semantic



(a) Contour on mesh (b) Contour in motion (c) Crossing a mesh vertex (d) After crossing

Figure 4.3: (a) The contour for the $l_{\text{horse,(head,body)}}$ semantic border. The contour is shown in grey, the initial semantic regions in cyan and pink, and the intermediate zone boundaries in orange. (b) The contour vertices (red) lie on mesh edges. (c)–(d) After crossing a mesh vertex, new contour vertices (blue) are created on outgoing edges.

region. However, the intermediate zone can be adjusted by the user if the default setting is unsatisfactory.

We use an explicit, piecewise linear representation of the contours in which contour vertices are constrained to lie on mesh edges, following the formulation by Bischoff et al. (2005). In this formulation, vertices are added and removed from the contour as it evolves on the surface to adapt its resolution to the underlying tessellation. Figure 4.3 shows an example of new contour vertices being created when an existing contour vertex crosses a mesh vertex. See the cited paper for further details.

Compatibility-Based Contour Optimization. In previous settings, active contours or "snakes" (Kass et al., 1988) have been optimized with respect to a property of the surface upon which they lie. We deviate from previous work by jointly optimizing a set of contours for geometric similarity with each other. Our measure of geometric similarity considers distances between contour points (C^0 continuity), as well as angles between contour normals (C^1 continuity). Both factors are necessary to achieve a smooth transition between components, as shown in Figure 4.4(d).

Formally, we seek to minimize

$$E_{\text{dis}}(\mathbf{C}) = \sum_{c_p, c_q \in \mathbf{C}} D(c_p, c_q) + D(c_q, c_p), \tag{4.1}$$

the sum of pairwise dissimilarities between contours, with

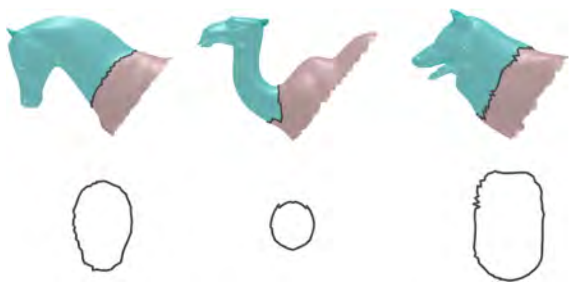
$$D(c_p, c_q) = \int_0^1 \left(\|\mathbf{v}_p^t - \widetilde{\mathbf{v}}_q(\mathbf{v}_p^t))\|^2 - \lambda \cdot \mathbf{n}_p^t \cdot \widetilde{\mathbf{n}}_q(\mathbf{v}_p^t) \right) dt, \tag{4.2}$$

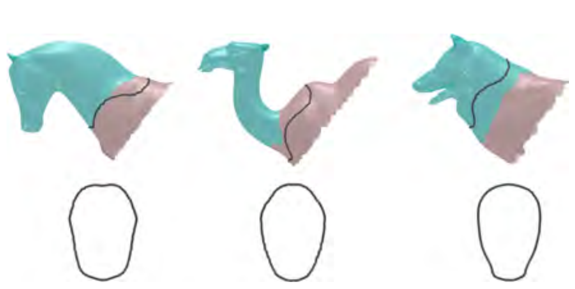
where $\mathbf{v}_p^t$ maps the normalized arclength parameter $t \in [0, 1]$ to a position along the contour c_p and $\mathbf{n}_p^t$ does the same for a normal vector along c_p, $\widetilde{\mathbf{v}}_q(\mathbf{x})$ is the location of the closest point on the contour c_q to the point $\mathbf{x}$, while $\widetilde{\mathbf{n}}_q(\mathbf{x})$ is the normal of the closest point, and λ is a weight controlling the trade-off between optimizing similarity between normals and between positions. Essentially, $D(c_p, c_q)$ sums the average squared distance to the contour c_q along c_p and the average difference in normal vectors between the points on c_p and their closest points on c_q.

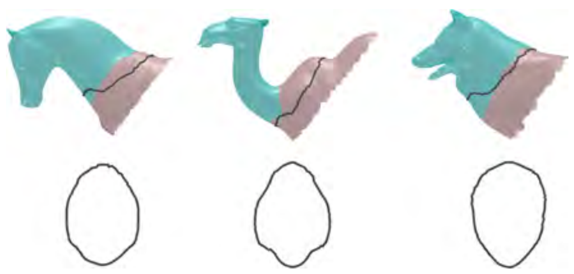
The contour normals are computed by linearly interpolating the vertex normals on the underlying mesh. The weight λ is set equal to αl, where l is the median bounding box diagonal of the initial edge loops. In order to prioritize C^0 continuity over C^1 continuity, we set $\alpha = 0.1$. We approximate the integral in (4.2) by taking the average distance over a set of uniformly spaced points on the contour. The distances and nearest points are efficiently computed with an axis-aligned bounding box tree.

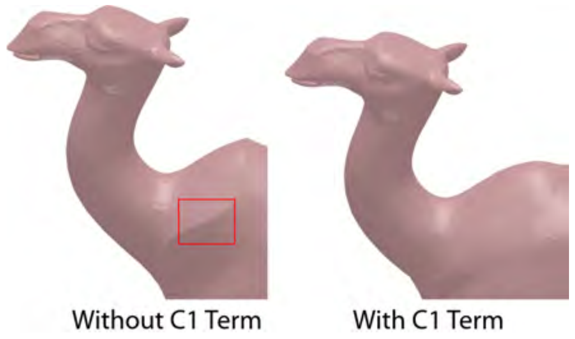
When evaluating the objective, each contour is aligned to a common local frame by finding the minimal transformation that maps its centroid to the origin and maps the plane generated by a least squares fit on its vertices to the $x\,y$-plane.

We initialize the contour to the original semantic border and use Euler's method to minimize the objective. Small time-steps are necessary because the number of vertices in the contour can change when an existing contour vertex runs into a mesh vertex. Nevertheless, the minimization rapidly converges to a solution. We terminate the optimization when the


(a) Semantic Segmentation Only

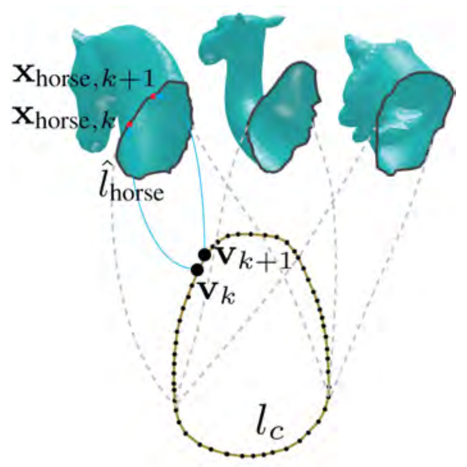

(b) Compatibility-based Optimization Without C^1 Term


(c) Compatibility-based Optimization With C^1 Term



(d) Camel head on Wolf body

Figure 4.4: We compare three ways of choosing the individual edge loops. In (a)–(c) we show the resulting components (top) and loops (bottom). Using the original semantic segmentation (a) results in geometrically dissimilar loops compared to our compatibility-based optimization (b)–(c). In (d) connecting the camel head to the wolf body reveals the effect of the C^1 term. Without the C^1 term, a seam is visible despite C^0 continuity.



(a) Initial Common Edge Loop

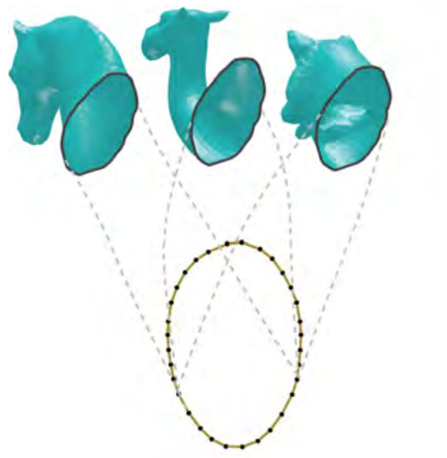

(b) Refined Common Edge Loop

Figure 4.5: (a) Snapping the common edge loop to the individual edge loops. (b) Deforming the meshes to match with the common edge loop. After the deformation, the individual edge loops of the components are identical and the components are interchangeable.

relative improvement in the objective is less than 5% for 100 iterations.

The contour representation is flexible enough to generate new edge loops with complex, non-elliptical shapes that deviate significantly from the initial ones when necessary, as shown in Figure 6.8(c). In Sections 6.2.2, 6.2.3, and 6.2.4, we evaluate the behavior of our contour optimization and its effect on the subsequent steps of the approach.

Our approach applies the above procedure separately for each semantic border. After this step, for each semantic border b of each mesh m, the individual edge loop is updated from $l_{m,b}$ to $\hat{l}_{m,b}$ by snapping the contour vertices to their nearest mesh vertex. Figure 4.4(a)–(b) shows a comparison between the individual edge loops extracted directly from the input semantic borders and those found by our optimization. The loops found by the latter are much more similar in shape. Figure 4.4(c)–(d) shows the necessity of considering C^1 continuity when optimizing the contours.

4.3 Deformation to Common Edge Loops

In Section 4.2, for each semantic border, we obtained a combination of individual edge loops $\mathbf{C} = \{\hat{l}_m\}$ that are similar in shape, one loop for each input mesh m. In this section, we describe how to create a *common edge loop* l_c using this combination of individual edge loops $\mathbf{C}$ for each semantic border. The individual edge loops for each semantic border will assume the shape of l_c in order to produce interchangeable components.

Our approach proceeds in two steps. First, for each semantic border b, we find an initial shape for the common edge loop for b with minimal total shape difference between it and each of the individual edge loops. Then, in the second step we refine the shapes of all the common edge loops simultaneously by solving a global optimization that minimizes the total deformation of all the input meshes, under the constraint that the individual edge loops on each mesh take the shape of their common edge loop. This step considers the total deformation of the meshes rather than just the edge loops.

4.3.1 Common Edge Loops Initialization

Again we focus our discussion on a single semantic border, as we will apply this step independently per semantic border. We describe how to create an initial common edge loop. Figure 4.5(a) shows an illustration. Denote the common edge loop as l_c, which is formed by linking a series of vertices $\mathbf{V}_c = \{\mathbf{v}_k\}$. We want to correspond the vertices $\{\mathbf{v}_k\}$ to the points on the individual edge loop $\hat{l}_m$ (found in Section 4.2) of each mesh m.

We parameterize each individual edge loop $\hat{l}_m$ by its arc length, where $\hat{l}_m(t)$ is a point on $\hat{l}_m$ for $t \in [0, 1]$. We suppose that each vertex $\mathbf{v}_k$ on the common edge loop l_c corresponds to a point $\mathbf{x}_{m,k} = \hat{l}_m(t_{m,k})$ on the individual edge loop $\hat{l}_m$ by $t_{m,k}$.

Our goal in this step is to find the vertices $\mathbf{V}_c = \{\mathbf{v}_k\}$, which define the shape of the common edge loop l_c; and the set of parameters $\mathbf{T} = \{t_{m,k}\}$, which defines the correspondences between the common edge loop l_c and each individual edge loop $\hat{l}_m$. We find $\mathbf{V}_c$ and $\mathbf{T}$ simultaneously by solving a constrained optimization:

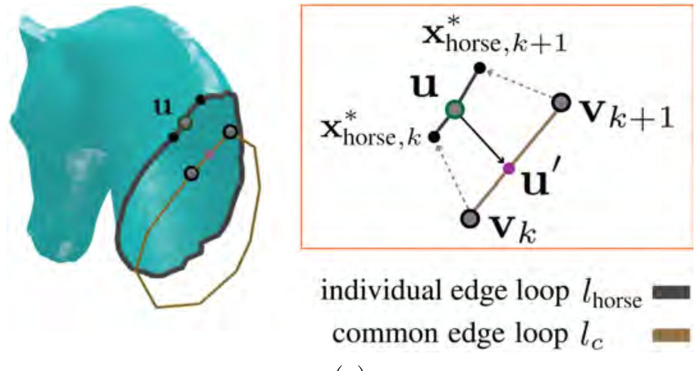
$$\begin{aligned} \min_{\mathbf{V}_c, \mathbf{T}} \quad & \lambda E_{\text{def}}^{\text{loop}}(\mathbf{V}_c, \mathbf{T}) + (1.0 - \lambda) E_{\text{reg}}^{\text{loop}}(\mathbf{T}) \\ \text{subject to} \quad & t_{m,k} < t_{m,k+1},\ \forall m, k. \end{aligned} \tag{4.3}$$

Deformation. $E_{\text{def}}^{\text{loop}}$ penalizes deformation of the common edge loop l_c when its vertices $\{\mathbf{v}_k\}$ are corresponded to points $\{\mathbf{x}_{m,k}\}$ on the individual edge loop $\hat{l}_m$ for mesh m:

$$E_{\text{def}}^{\text{loop}}(\mathbf{V}_c, \mathbf{T}) = \frac{1}{P} \sum_m \sum_k ||(\mathbf{x}_{m,k+1} - \mathbf{x}_{m,k}) - (\mathbf{v}_{k+1} - \mathbf{v}_k)||^2, \tag{4.4}$$

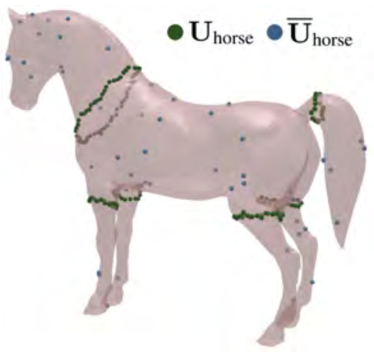
where P is the squared length of the longest individual edge loop.

Regularization. $E_{\text{reg}}^{\text{loop}}$ encourages the corresponded locations of the common edge loop



u
$\mathbf{x}^*_{\text{horse},k+1}$
u
$\mathbf{x}^*_{\text{horse},k}$
$\mathbf{v}_{k+1}$
$\mathbf{u}'$
$\mathbf{v}_k$
individual edge loop l_{horse}
common edge loop l_c

(a)



$\mathbf{U}_{\text{horse}}$
$\overline{\mathbf{U}}_{\text{horse}}$

(b)

Figure 4.6: (a) Representing individual edge loop vertices in terms of common edge loop vertices. Using the correspondence computed from Section 4.3.1, each individual edge loop vertex $\mathbf{u}$ is projected to $\mathbf{u}'$ on the common edge loop, and is represented as a linear combination of common edge loop vertices $\mathbf{v}_k$ and $\mathbf{v}_{k+1}$. (b) $\mathbf{U}_{\text{horse}}$ contains all the individual edge loop vertices (green). $\overline{\mathbf{U}}_{\text{horse}}$ contains all the other (interior) vertices (blue). We show only some of the vertices for clarity.

vertices to spread evenly over the individual edge loops:

$$E^{\text{loop}}_{\text{reg}}(\mathbf{T}) = \sum_m \sum_k ((t_{m,k+1} - t_{m,k}) - \frac{1}{|\mathbf{V}_c|})^2 \tag{4.5}$$

The inequality constraint ($t_{m,k} < t_{m,k+1}$) preserves the ordering of the common edge loop vertices in their correspondences with the individual edge loops. We set the weight λ to 0.9. The optimization can be solved quickly using standard solvers such as IPOPT (Wächter and Biegler, 2006). The optimization computes optimized vertex positions $\mathbf{v}^*_k$ that describe the initial shape of the common edge loop. For each mesh m and each common edge loop vertex k, it computes $\mathbf{x}^*_{m,k}$, the point on individual edge loop $\hat{l}_m$ to which vertex k corresponds.

4.3.2 Shapes and Common Edge Loops Refinement

In the previous step, for each semantic border b, we obtained the initial shape for the common edge loop $l_{c,b}$ and the correspondence between its vertices and points on the individual edge loop $l_{m,b}$.

In this step, we refine the shapes of all the common edge loops jointly, by considering the

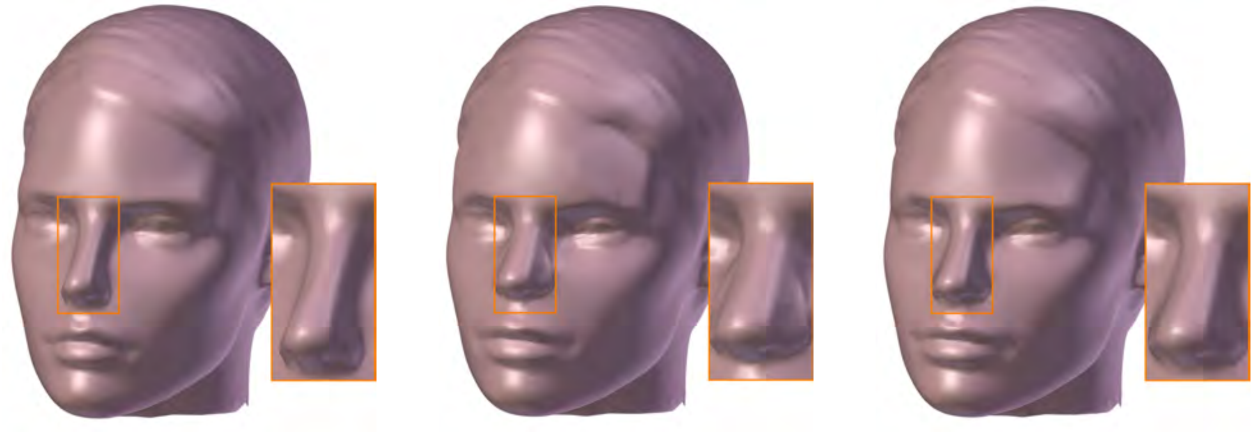


Figure 4.7: Shape-preserving refinement. (a) Input shape. (b) Result produced without refinement by using the common edge loop shapes from Section 4.3.1 as boundary constraints for mesh deformation. (c) Result produced with shape-preserving refinement. Note that while both results can be used to form a set of interchangeable components, the result in (c) more closely resembles the input shape. For example, the nose looks more similar to the nose of the input shape.

deformation induced on the input meshes when the individual edge loops are constrained to assume the shape of their common edge loop.

We use the correspondence obtained in Section 4.3.1 to link the shape of the common edge loops to those of their individual edge loops, by writing each individual edge loop vertex as a linear combination of two common edge loop vertices. Figure 4.6(a) shows this process.

Shape-Preserving Refinement. We minimize the sum of deformation of the input meshes, under the constraint that the individual edge loop vertices are expressed as a linear combination of the common edge loop vertices. This constraint means that corresponding individual edge loops will form the same shape, which is necessary for part compatibility.

Consider a mesh m and its vertices $\mathbf{U}_m \cup \overline{\mathbf{U}}_m$, where $\mathbf{U}_m$ contains the vertices across all the individual edge loops of m, and $\overline{\mathbf{U}}_m$ contains all the other vertices (i.e., interior vertices). See Figure 4.6(b). We can express the individual edge loop vertices $\mathbf{U}_m$ in terms of the common edge loop vertices as shown in Figure 4.6(a). So the deformation of mesh m is specified by $\mathbf{V}_c^{\text{all}} \cup \overline{\mathbf{U}}_m$ instead of $\mathbf{U}_m \cup \overline{\mathbf{U}}_m$, where $\mathbf{V}_c^{\text{all}}$ contains the common edge loop vertices across all the semantic borders; e.g., $\mathbf{V}_c^{\text{all}} = \{\mathbf{V}_{c,(\text{head,body})}, \mathbf{V}_{c,(\text{head,tail})}, \mathbf{V}_{c,(\text{head,left leg})}, ...\}$. We sum the deformation energy over all meshes:

$$E_{\text{def}}^{\text{mesh}}(\mathcal{M}) = \sum_m w_m E_{\text{def}}^m(\mathbf{V}_c^{\text{all}}, \overline{\mathbf{U}}_m), \tag{4.6}$$

where $E_{\text{def}}^m(\mathbf{V}_c^{\text{all}}, \overline{\mathbf{U}}_m)$ is a normalized measure of the deformation of mesh m and $w_m \in [0, 1]$ is a user-specified weight associated with the deformation of mesh m, which is set to 1 by default. The method for minimizing this energy depends on the mesh deformation measure chosen for mesh m. A nice property of our construction is that if $E_{\text{def}}^m(\mathbf{V}_c^{\text{all}}, \overline{\mathbf{U}}_m)$ can be minimized by solving a linear system such as in Laplacian or As-Rigid-As-Possible mesh deformation (Sorkine and Alexa, 2007b), then so can $E_{\text{def}}^{\text{mesh}}(\mathcal{M})$, since the individual edge loop vertices are written as a linear combination of the common edge loop vertices. One can also minimize the energy by alternating between fixing $\mathbf{V}_c^{\text{all}}$ while solving for each E_{def}^m, and fixing $\overline{\mathbf{U}}_m$ while solving for $\mathbf{V}_c^{\text{all}}$. In our implementation we used the latter approach and used As-Rigid-As-Possible mesh deformation for E_{def}^m.

Figure 4.5(b) visualizes the results of this refinement. We obtain a set of refined common edge loops as well as a set of deformed meshes $\mathcal{M}'$. Figure 4.7 shows the advantage of our joint optimization over a more naive approach where the initial common edge loop shapes from Section 4.3.1 are held fixed when deforming the meshes.

4.4 Generating Interchangeable Components

Using the refined individual edge loops obtained in Section 4.3.2, we partition the set of deformed meshes $\mathcal{M}'$ into interchangeable components. We seal the holes on the components resulting from the partition, and then add connectors on the sealed surfaces to make the components connectable.

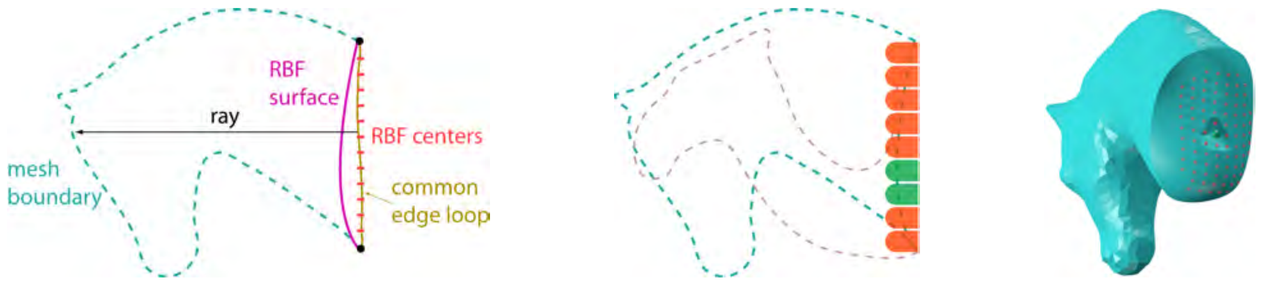
Surface Sealing. As we use the same common edge loop for partitioning the same semantic border of different meshes, we create a 3D surface for sealing for each common edge loop.

Figure 4.8 shows an illustration. We create this surface by using radial basis functions (RBFs) (Carr et al., 2001) fitted over the domain of the common edge loop, with the RBF centers placed uniformly within the loop. We use polyharmonic RBF basis functions. At any location within the loop, the weighted sum of the RBFs gives a height value; hence the RBFs specify a 3D surface over the loop.

In fitting the RBFs to form the 3D surface, the weights of the RBFs are determined by an optimization (Wang and Oliveira, 2003). Our objective minimizes the distance between the RBF surface and the edge loop vertices. It also encourages a smooth and roughly planar RBF surface by minimizing the magnitude of the RBF coefficients. Finally, we add constraints on the height of the RBF surface at a set of uniformly spaced sample points to ensure that it does not penetrate any of the meshes. We determine the value of these height constraints by shooting rays at the mesh, as illustrated in Figure 4.8(a). See Carr et al. (2001) and Wang and Oliveira (2003) for further details of surface completion based on RBFs.

Connector Placement. To allow the components to be conveniently connected and disconnected, our approach automatically adds male and female connectors to each component. The male connectors are simple beveled triangular prism shapes and the females are their complement, sized slightly smaller to create a desirable amount of friction.

Figure 4.8(b)–(c) illustrates the process. Across the sealed surface of each component, we regularly sample candidate locations for adding a connector. We determine the validity of each candidate location by checking whether a connector to be placed there will intersect with the component boundary. We choose the most central valid location to be the location for putting a connector. A male connector and a female connector are added respectively to a pair of compatible components by CSG operations. We experimented with using three connectors per component surface, but found that simply placing a single connector as close as possible to the center of the component surface results in a stable enough connection.



(a) (b) (c)

Figure 4.8: (a) Surface sealing. A surface is created using RBFs fitted over the domain of the common edge loop shown. Rays are shot along positions sampled within the loop to determine the mesh boundary within which the created surface should lie. (b) Connector Placement. Candidate locations (red) are evaluated from a set of regular samples. Valid locations (green) lie within the mesh boundary. (c) The most central valid location (green) is chosen as the connector location, where in this case a female connector of a triangular prism shape is placed.

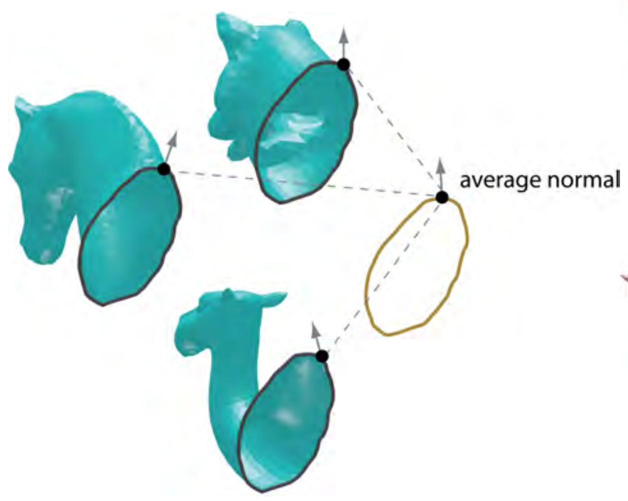
4.5 User Interaction and Enhancements

We present some useful ways the user can direct our approach at the high level and some extensions to our approach which improve the quality and breadth of our results.

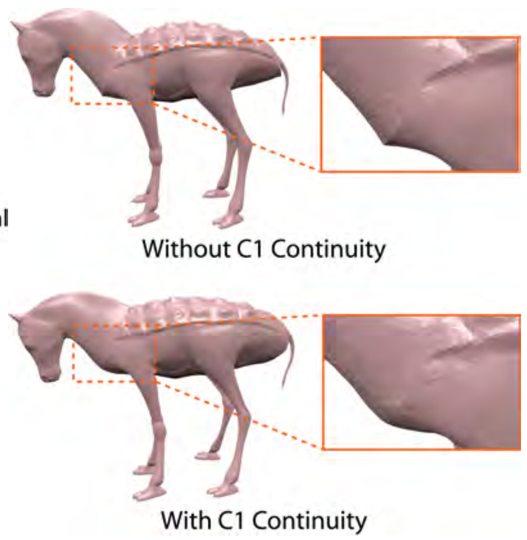
4.5.1 C^1 Component Continuity Deformation

Our contour optimization in Section 4.2 tries to find edge loops with high C^1 continuity between them, but this goal may be unachievable when large discrepancies in the initial geometry are present. In these cases, the deformed components found in Section 4.3 may have poor C^1 continuity. To resolve this issue, we apply an additional deformation step that locally enforces C^1 continuity. Figure 4.9 illustrates the technique and shows its effects. The main idea is to build a representation of the average normal orientation around the set of individual edge loops for each semantic border b, then deform the meshes so that they conform to this orientation.

For each semantic border b, we align the common edge loop $l_{c,b}$ (found in Section 4.3) with $\hat{l}_{m,b}$, the individual edge loop of mesh m for b. We record the normal orientation on the mesh



(a)



(b)

Figure 4.9: Considering C^1 Component Continuity. (a) Computing an average normal. Average normals are recorded at a set of uniformly sampled locations on the common edge loop. (b) The neck transition becomes smoother after incorporating C^1 continuity.

m at a set of uniformly sampled locations on $l_{c,b}$. For each sample location, we record the average orientation across all the meshes.

We use the average orientations to give each individual edge loop vertex a target normal vector and a transformed coordinate frame. We propagate the transformed coordinate frames to the non-boundary part of each mesh using an existing technique (Schmidt and Singh, 2010a), which smoothly deforms the mesh to align its normals with the target normals. This step does not alter the positions of the individual edge loop vertices, so the components will still be interchangeable.

4.5.2 Higher Order Component Connectivity

Our basic approach assumes that the adjacency graph between components is a tree. However, this assumption is not true for some interesting shapes. Consider the problem of making interchangeable arm components in a set of armchairs as shown in Figure 4.10. Our basic approach fails in this situation, because it only guarantees that individual edge loops

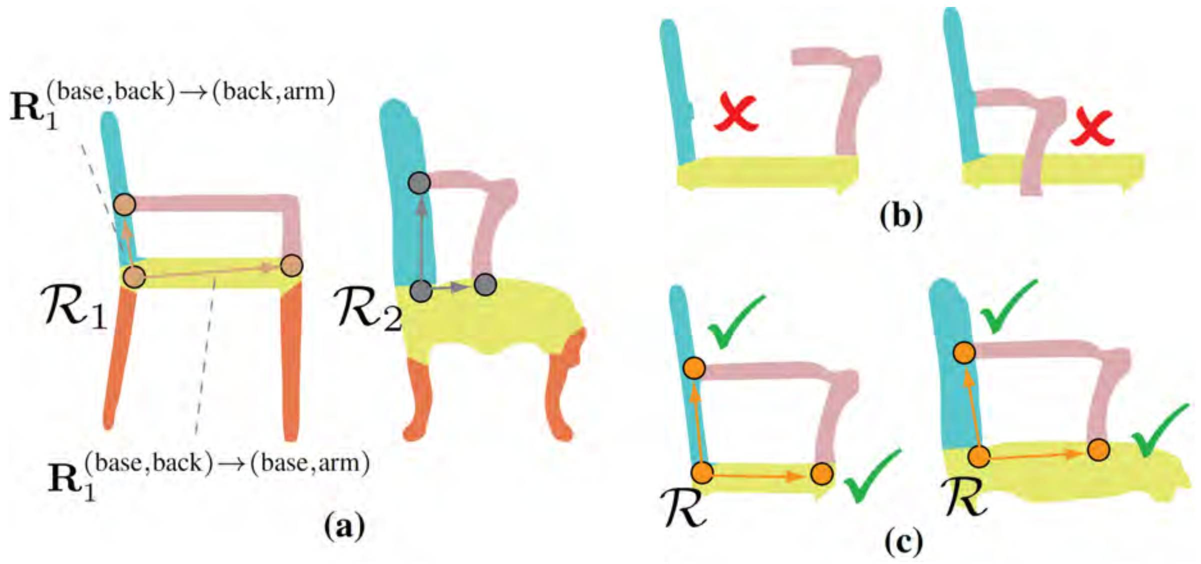

$\mathbf{R}_1^{(\text{base,back})\to(\text{back,arm})}$
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathbf{R}_1^{(\text{base,back})\to(\text{base,arm})}$
(a)
(b)
$\mathcal{R}$
$\mathcal{R}$
(c)

Figure 4.10: Considering higher order connectivity in creating an interchangeable arm component. (a) Input chairs. The relative transformations of the (back,arm) loop and the (base,arm) loop with respect to the (base,back) loop are different for each chair, stored in $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively. (b) The arm component created from Chair 2 using our basic approach fails to connect completely to Chair 1. (c) After the extra optimization step considering higher order connectivity, the arm component can connect properly with both chairs. The chairs and arm are slightly deformed, and the relative transformations among each chair's components are equal to the common relative transformations $\mathcal{R}$.

belonging to a single semantic border are equal. For the arm component this guarantee is not sufficient. As shown in Figure 4.10(b), the arm component can connect along the back border or the base border, but not both.

To resolve this issue we need to generalize the condition for interchangeability described in Section 4.1. We constrain the union of individual edge loops across a *set* of semantic borders to be equal (up to rigid transformation) rather than just a single semantic border, as before. For the example in Figure 4.10, this set would be $\{(\text{back}, \text{arm}), (\text{base}, \text{arm}), (\text{base}, \text{back})\}$. We enforce this constraint by applying an extra optimization step after the procedure described in Section 4.3. We minimize the same mesh deformation energy as in (4.6), but we fix the shapes of the edge loops, and only optimize their orientations and positions.

We enforce this constraint by applying an extra optimization step after the procedure described in Section 4.3. Figure 4.10(c) shows an example. In the extra optimization step, the

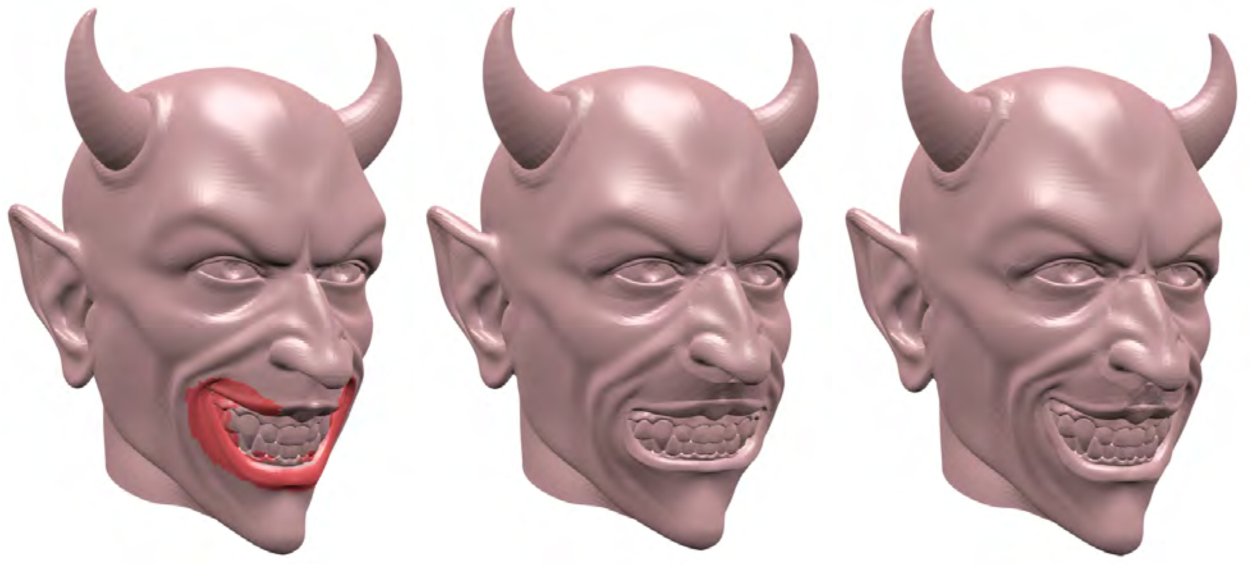

(a) Input (b) No Preservation (c) With Preservation

Figure 4.11: Semantics preservation. (a) Input face. The user paints the mouth region with higher weights to preserve the expression. (b) Without considering semantics preservation (uniform weights), the devil's expression changes from a grin to more of a grimace after deforming to achieve component compatibility. (b) Considering semantics preservation (higher weights on the painted region), the devil still shows a grin after deforming. The deformation of the other shapes did not change significantly as a result of the weight adjustment.

relative transformations (position and orientation) of the loops in each mesh are constrained to be equal to common relative transformations. Suppose $\mathcal{R}_1 = (\mathbf{R}_1^{(\text{base,back})\to(\text{base,arm})}, \mathbf{R}_1^{(\text{base,back})\to(\text{back,arm})})$ is a tuple storing the relative transformations from the (base,back) loop to the (base,arm) loop and to the (back,arm) loop respectively for chair 1. Likewise for $\mathcal{R}_2$ for chair 2. We constrain them to be the same, i.e., we set $\mathcal{R} = \mathcal{R}_1 = \mathcal{R}_2$.

The optimization minimizes the same objective function as in (4.6), but this time with respect to the common relative transformations $\mathcal{R}$ (instead of directly on the loop vertices) and the non-loop (interior) vertices. Note that when $\mathcal{R}$ changes, the loop vertices will undergo a rigid transformation. Figure 4.10(c) shows the result. The arm deforms to become longer to accommodate with chair 1; the back of chair 2 deforms to become slightly shorter; the relative transformations between the back, corner, and base loops of Chair 1 are the same as those of Chair 2.

We minimize (4.6) by alternating between solving for the non-loop (interior) vertex positions and the relative transformations of the loop vertices, while keeping the other set of variables fixed.

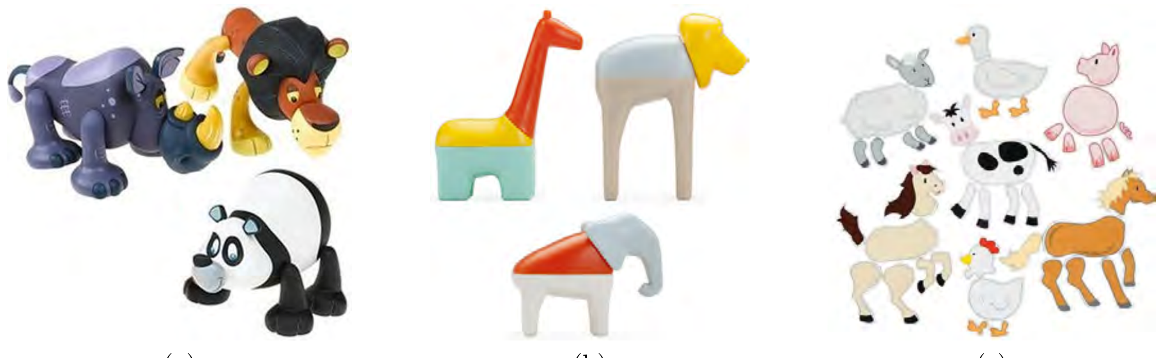
4.5.3 Semantics Preservation

Our core approach is geometric in nature and does not consider semantics. Therefore, when it deforms the input meshes to achieve component compatibility, a mesh may lose some desirable semantic quality. Figure 4.11 shows an example. Because our approach does not explicitly consider the facial expression of the devil, it changes the devil's expression from a grin to more of a grimace. To resolve this problem, we allow the user to interactively adjust the weighting of the shape preservation energies E_{def}^m in (4.6) by painting the surface of the mesh with modified weights. We assume that E_{def}^m is a weighted sum of per-vertex or per-face energies, which is usually the case. By changing these weights we can emphasize the preservation of certain regions in the final deformed meshes.

Figure 4.11(c) shows the result of preserving the grin after the user labels the mouth region to have higher weight in the deformation energy.

4.5.4 Most-Compatible Subset Selection

When constructing a set of interchangeable components using our approach, it may be necessary to restrict the number of input models. For example, when designing a children's toy for assembling animals similar to those in Figure 4.12, the user may only want to include a fixed number of animals in the toy, to satisfy manufacturing and packaging constraints. However, the user may have a much larger database of animal shapes that could be included in the toy. Our approach can conveniently find the subset of shapes that are most compatible with each other. By most compatible, we mean that our approach will have to deform the input shapes the least to create interchangeable components.



(a) (b) (c)

Figure 4.12: Some existing chimera toys. Compared to our results, the products have limited shape variability in (a), (b), low geometric detail in (b), and prominent divisions between components in (c).

To perform this task, we define a dissimilarity metric between input meshes which leverages the compatibility optimization used in Section 4.2. We compute the dissimilarity between mesh m and mesh n as follows:

$$D_{\text{shape}}(m,n) = \frac{1}{|\mathbf{B}|}\sum_{b\in\mathbf{B}} \min_{\substack{c_p\in\mathbf{I}_{m,b},\\ c_q\in\mathbf{I}_{n,b}}} D(c_p,c_q) + D(c_q,c_p), \tag{4.7}$$

where $\mathbf{B}$ contains the semantic borders that exist in both meshes; $\mathbf{I}_{m,b}$ is the intermediate zone for semantic border b of mesh m, as described in Section 4.2, likewise for $\mathbf{I}_{n,b}$, and $D(c_p, c_q)$ measures the distance of contour c_p from contour c_q, as defined in (4.2). Essentially our metric sums the distance between the two closest contours found at each semantic border. The closest contours are found with the minimization procedure in Section 4.2.

Given a user specified value k, we can use our dissimilarity metric D_{shape} and a branch-and-bound search to find the most-compatible size k subset of the shapes in our database. The smaller the maximum dissimilarity between any two members of a subset, the more compatible the subset. We evaluate this technique in Section 4.5.4. For a database which has several models belonging to the same category, we recommend incorporating into the search a geometric diversity metric, as described by Chaudhuri and Koltun (2010), so that the objective of component compatibility in the subset can be balanced against the desire for geometric diversity.

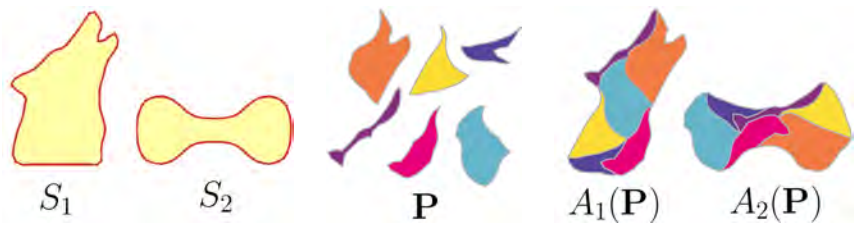
CHAPTER 5

Approximate Dissections

In the Approximate Dissections problem, given a pair of input shapes, we seek to find a single set of pieces that can be assembled into *approximations* of both input shapes. This is a departure from the classic dissection problem, in which the pieces must form the input shapes *exactly*. Below, we formally describe the problems.

Let S_1 and S_2 denote the input shapes, represented as polygons. Let $\mathbf{P}$ denote the set of dissection pieces. Let $A_1(\mathbf{P})$ and $A_2(\mathbf{P})$ denote two non-overlapping arrangements of the pieces $\mathbf{P}$. Figure 5.1 illustrates these variables.

In the classic K-piece dissection problem, given the input shapes and an integer K, we find pieces and arrangements such that $|\mathbf{P}| = K$ and the constraints $A_1(\mathbf{P}) = S_1$ and $A_2(\mathbf{P}) = S_2$ are satisfied. In the approximate K-piece dissection problem, we instead minimize a shape difference measure between the input shapes and the arranged pieces: $||A_1(\mathbf{P}) - S_1|| + ||A_2(\mathbf{P}) - S_2||$.



(a) Input Shapes (b) Pieces (c) Arrangements

Figure 5.1: The variables of the dissection problem.

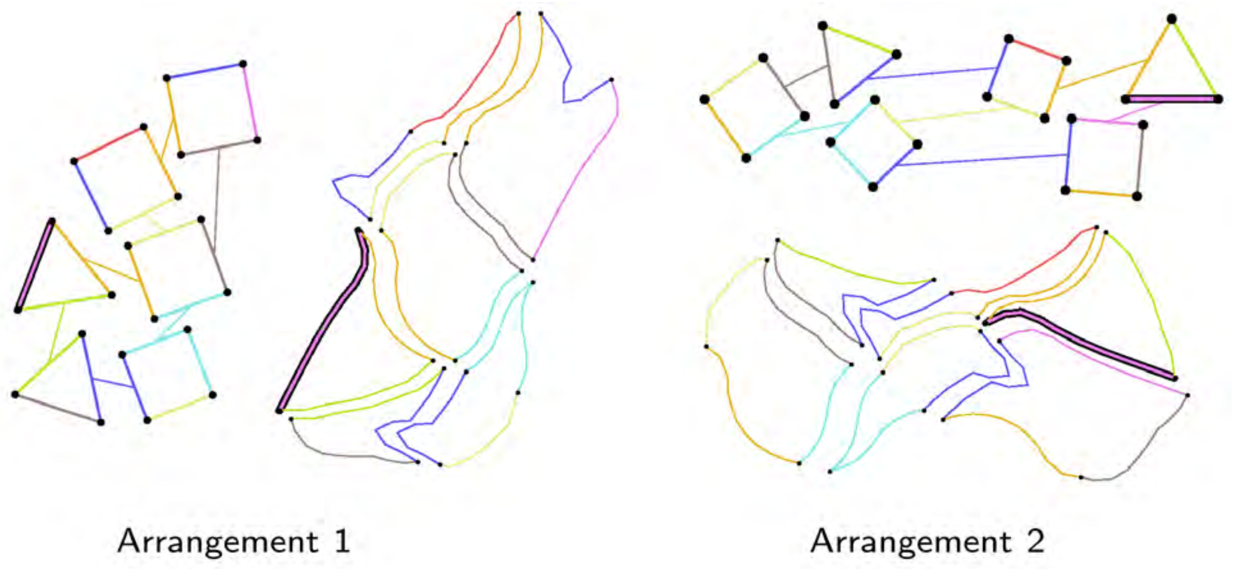


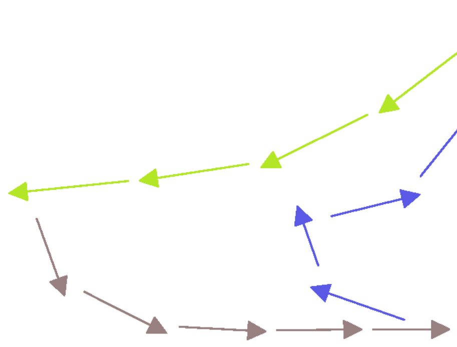
Figure 5.2: A dissection between a dog head and a bone. The boundary intervals are colored according to which connection constraint they belong and are separated by black dots. In the first arrangement, the bolded pink boundary interval is *external* because it is not adjacent to another boundary interval. In the second arrangement, the same boundary interval is no longer external. We show the connectivity representations next to the corresponding geometry.

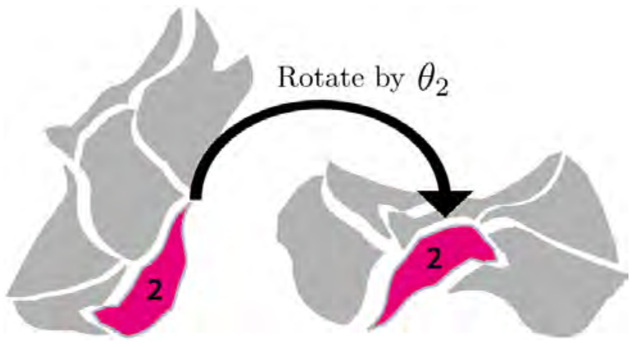
5.1 Representation

Our solution to the approximate dissection problem stems from a novel way of representing dissection solutions that supports the notion of a *lower bound* on the objective value of a partial dissection solution. The lower bound allows us to prune the search space by telling us when a partial solution cannot lead to high quality solutions.

A solution in our representation consists of three components: (1) the connectivity representation (Figure 5.2) describes how the dissection pieces connect, (2) the geometric parameters (Figure 5.3) specify the actual dissection geometry under the constraint of the given connectivity, and (3) the reconstruction mapping (Figure 5.6) describes which regions of the input shapes are reconstructed by which regions of the dissection pieces.

Connectivity Representation. The connectivity representation of a dissection solution specifies how the dissection pieces fit together in each arrangement.


(a) Edge vector sequence



(b) Rotation angle

Figure 5.3: Visualizing the geometric parameters. (a) A dissection piece is represented by a sequence of edge vectors, colored according to their boundary interval. (b) The change in the orientation of each piece from the first arrangement to the second is represented by an angle for that piece.

The boundary of each dissection piece is partitioned into *boundary intervals*. Figure 5.2 shows the pieces for a complete dissection with the boundary intervals delimited. For each arrangement of the pieces, we note the adjacencies between boundary intervals; i.e., when two boundary intervals are touching each other. If a boundary interval is not adjacent to any others in a given arrangement, then it is termed *external* in that arrangement. The external boundary intervals reconstruct the input shape in each arrangement. The boundary intervals are not allowed to partially overlap; i.e., in a given arrangement, all boundary intervals are adjacent to at most one other boundary interval.

The connectivity representation formally consists of a sequence of boundary intervals for each dissection piece specifying their clockwise order and a record of the boundary interval adjacency for each arrangement.

Geometric Parameters. The connectivity representation does not specify the geometry of the dissection. We specify the geometry of each dissection piece as a sequence of edge displacement vectors in clockwise order around the piece. Each edge displacement vector forms a portion of one of the piece's boundary intervals.

To specify how the orientation of each dissection piece changes as it moves from one arrangement to the other, we use a rotation angle for each piece. Specifically, dissection piece

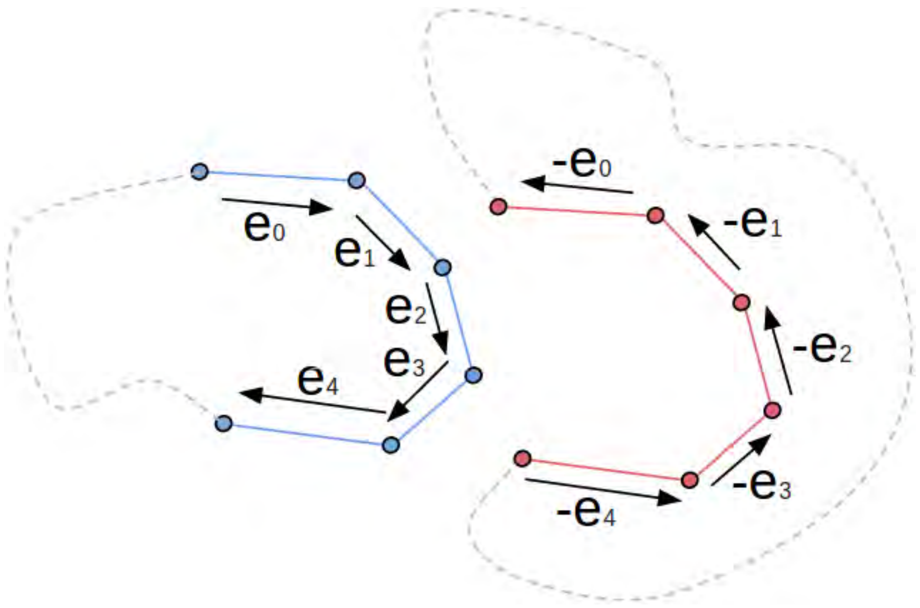


Figure 5.4: Two boundary intervals that connect. The geometry of both boundary intervals is parameterized by 5 edge displacement vectors. The rest of the dissection pieces are in grey.

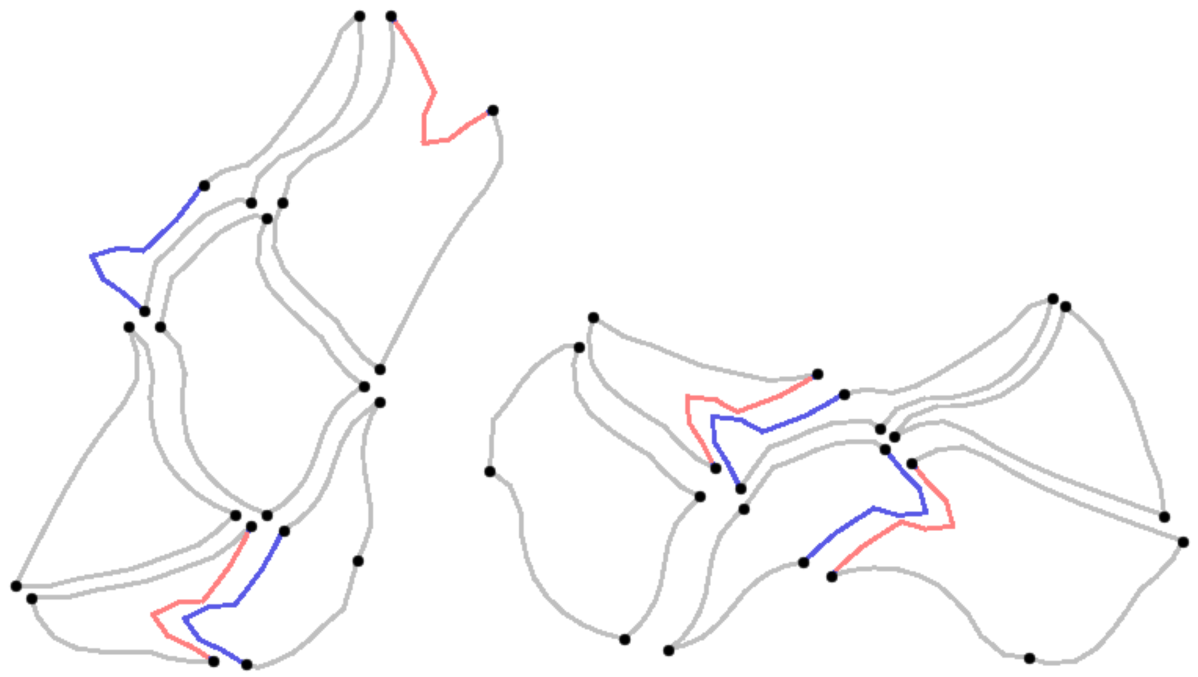
k is rotated θ_k degrees in Arrangement 2 relative to its orientation in Arrangement 1. The geometric parameters are visualized in Figure 5.3.

We determine the value of these geometric parameters for a given solution by optimizing how well they reconstruct the target shapes. This process is described in Section 5.2.

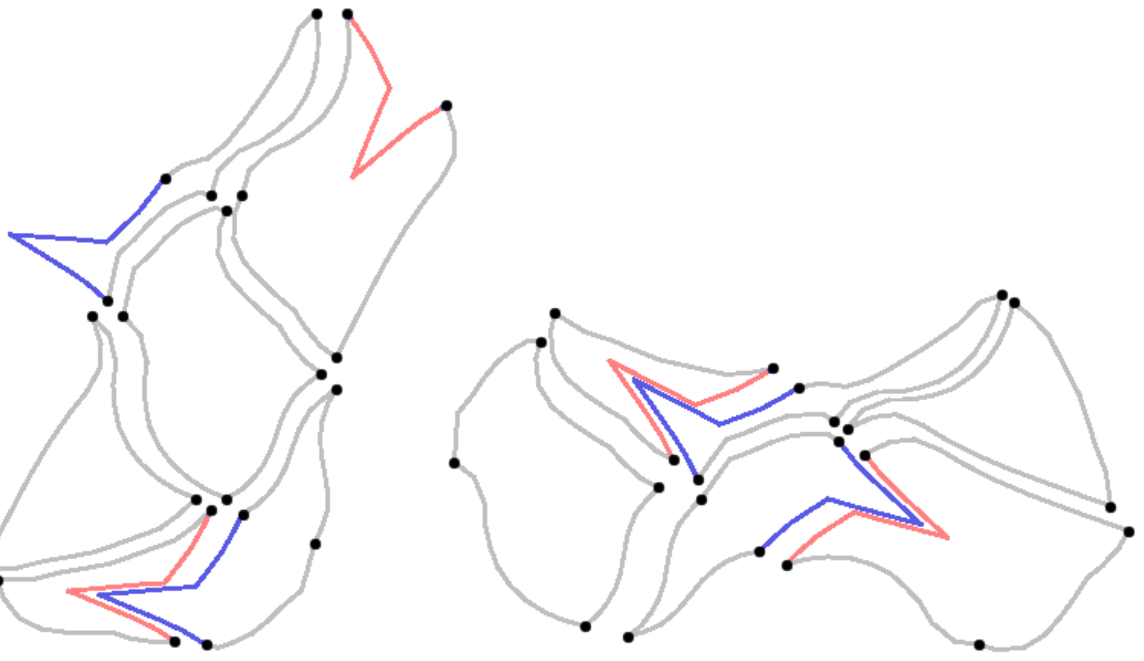
Connection Constraints. The connection constraints are geometric constraints placed on boundary intervals that ensure the dissection pieces can physically connect. They reduce the number of independent geometric parameters.

As mentioned previously, each boundary interval is specified as a sequence of edge displacement vectors in clockwise order along the dissection piece. If a pair of boundary intervals β and β' are adjacent, then for β' to connect flush with β, it must have an edge vector sequence equal to the reversal and negation of β's. Formally if $\beta = \{\vec{e}_1, \ldots, \vec{e}_N\}$ then $\beta' = \{-\vec{e}_N, \ldots, -\vec{e}_1\}$. Intuitively, if β is "male" then β' must be "female". Figure 5.4 visualizes this constraint.

Connection constraints lock the geometry of adjacent boundary intervals into a common

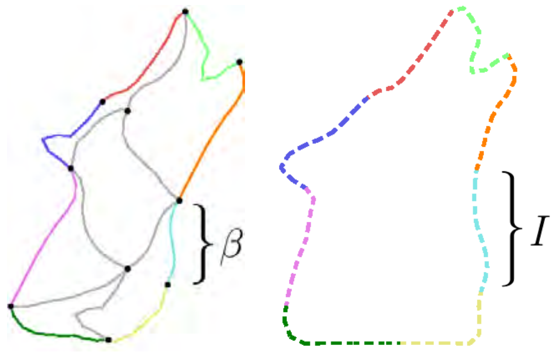


(a)



(b)

Figure 5.5: (a) A connection constraint that controls the geometry of 4 different boundary intervals on the dissection pieces. Blue boundary intervals are “male” connectors, red ones are “female”, and grey ones are not part of the connection constraint. (b) Changes to one boundary interval in the connection constraint affect the others. In this example, if the dog’s mouth becomes deeper, the ear becomes longer.



(a) Dissection Pieces (b) Input Shape

Figure 5.6: Reconstruction mapping for the dog arrangement. The external boundary intervals on the dissection pieces (a) are colored according to which boundary interval on the input shape (b) they reconstruct. Internal boundary intervals (grey) are not involved in the reconstruction mapping. Notation is shown for a dissection piece boundary interval (β) and its corresponding input shape boundary interval (I). Since the dog is the first input shape, this figure visualizes Γ_1.

parameterization. In general, connection constraints parameterize the geometry of *sets* of boundary intervals, rather than pairs. This is because a boundary interval β may be adjacent to β' in Arrangement 1 and β'' in Arrangement 2.

Figure 5.5 shows how a connection constraint controls the geometry of part of the dissection. Here, a single set of edge vectors parameterizes the geometry of four different boundary intervals. Figure 5.5(b) shows how this parameterization propagates edits to the dissection.

We can form a *partition* of the boundary intervals by grouping them according to which connection constraint controls them. Figure 5.2 visualizes this property. Boundary intervals of the same color belong to the same connection constraint.

Reconstruction Mapping. The reconstruction mapping (Figure 5.6) describes which regions of the input shapes are reconstructed by which regions of the dissection pieces. The mapping consists of two bijections $\Gamma_1(I)$ and $\Gamma_2(I)$, where $\Gamma_\alpha(I)$ is a bijection which maps a boundary interval I on input shape α to an external boundary interval β on the dissection pieces in arrangement α.

5.2 Evaluation

Given a candidate dissection solution, we evaluate how well it approximates the input shapes by optimizing over the dissection geometry parameters for the best approximation; i.e., we solve the following constrained optimization problem:

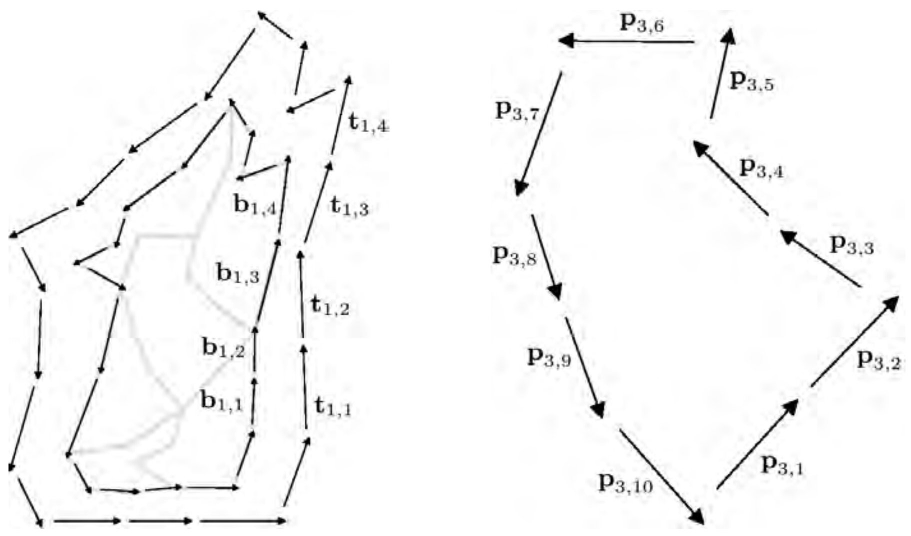
$$\begin{aligned} \underset{\mathbf{E},\boldsymbol{\theta}}{\text{minimize}} \quad & \max_{\alpha,i} \arccos(\hat{\mathbf{b}}_{\alpha,i} \cdot \hat{\mathbf{t}}_{\alpha,i}) + \lambda \sum_{\alpha} \sum_{i} ||\mathbf{b}_{\alpha,i} - \mathbf{t}_{\alpha,i}||^2 \\ \text{subject to} \quad & \sum_{l} \mathbf{p}_{k,l} = \mathbf{0}, \quad \forall k. \end{aligned} \tag{5.1}$$

The objective function measures how tightly the dissection pieces approximate the input shapes, according to the maximum angular error between dissection pieces and input shape edge vectors. Integer i indexes the edges of the reconstructed shape in arrangement α. Vector $\mathbf{b}_{\alpha,i}$ is edge vector i in the reconstruction of input shape α, and $\mathbf{t}_{\alpha,i}$ is the corresponding original input shape edge vector (i.e., the target of $\mathbf{b}_{\alpha,i}$). The value of $\mathbf{t}_{\alpha,i}$ comes from the reconstruction mapping. We set the regularization parameter to $\lambda = 0.01$.

The constraints in (5.1) ensure that the edge vector sequence for each dissection piece forms a closed curve. Integer k indexes dissection pieces while l indexes edges within a dissection piece. Vector $\mathbf{p}_{k,l}$ is edge vector l of dissection piece k. Figure 5.7 illustrates the vectors involved in the objective and constraints.

The optimization variables in (5.1) are $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_K\}$, the set of rotation angles for the K dissection pieces, and $\mathbf{E} = \{\mathbf{E}_1, \ldots, \mathbf{E}_M\}$, the set of edge vector sequences that parameterize the geometry for the M connection constraints. Connection constraint edge vector sequence $\mathbf{E}_i = \{\mathbf{e}_{i,1}, \ldots, \mathbf{e}_{i,N_i}\}$ comprises N_i edge vectors. The vector $\mathbf{e}_{i,j}$ denotes edge vector j within connection constraint i.

We solve the optimization problem (5.1) using IPOPT (Wächter and Biegler, 2006). The optimal objective value determines the quality of the solution. Appendix C describes how we initialize the optimization problem.



(a) Vectors for objective (b) Vectors in a constraint

Figure 5.7: (a) Edge vectors involved in the objective function in (5.1). (b) Edge vectors involved in the constraint for a piece in (5.1).

Geometry Functions. The edge vectors $\mathbf{b}_{\alpha,i}$ and $\mathbf{p}_{k,l}$ may be written as functions of the optimization variables $\mathbf{E}$ and $\boldsymbol{\theta}$. Each of these edge vectors belongs to a specific boundary interval β in an arrangement α. Therefore, we denote these functions as $\mathbf{g}_{\alpha,\beta,i}$, where $\mathbf{g}_{\alpha,\beta,i}(\mathbf{E}, \boldsymbol{\theta})$ is edge vector i within boundary interval β in arrangement α. For brevity, let $\mathbf{G}_{\alpha,\beta}$ denote the sequence of all the edge vectors in boundary interval β in arrangement α; i.e., $\mathbf{G}_{\alpha,\beta} = \{\mathbf{g}_{\alpha,\beta,1}, \ldots, \mathbf{g}_{\alpha,\beta,N}\}$. Figure 5.8 visualizes this notation. The functions can be derived using two geometric properties:

1. For a boundary interval β lying on dissection piece k, we have $\mathbf{G}_{2,\beta} = R(\theta_k, \mathbf{G}_{1,\beta})$ and $\mathbf{G}_{1,\beta} = R(-\theta_k, \mathbf{G}_{2,\beta})$, where $R(\theta, \mathbf{v})$ is an element-wise θ-degree rotation of the vector sequence $\mathbf{v}$.

2. Let β' denote a boundary interval adjacent to β in arrangement α. Then $\mathbf{G}_{\alpha,\beta'} = -\mathrm{Rev}(\mathbf{G}_{\alpha,\beta})$ where $-\mathrm{Rev}(\mathbf{v})$ reverses the vector sequence $\mathbf{v}$ and applies element-wise negation.

From these properties we can write a formula for any boundary interval in terms of any other in the same connection constraint. To obtain formulas in terms of the optimization

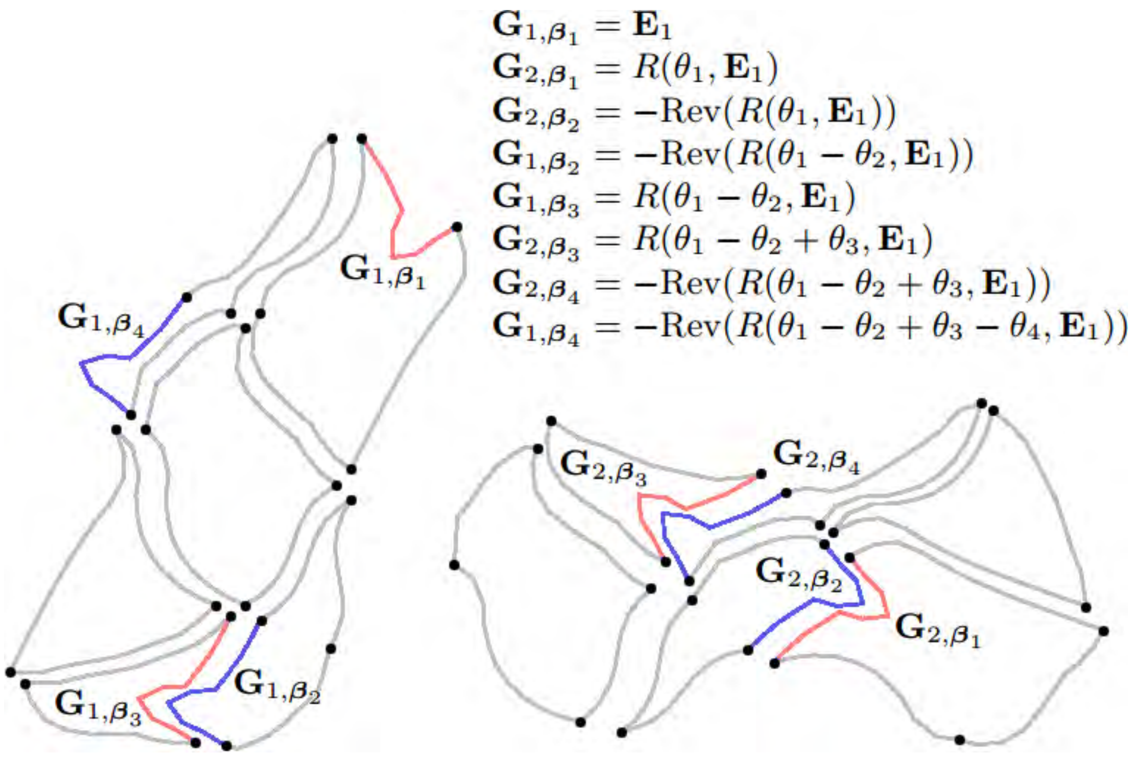

$\mathbf{G}_{1,\beta_1} = \mathbf{E}_1$
$\mathbf{G}_{2,\beta_1} = R(\theta_1, \mathbf{E}_1)$
$\mathbf{G}_{2,\beta_2} = -\mathrm{Rev}(R(\theta_1, \mathbf{E}_1))$
$\mathbf{G}_{1,\beta_2} = -\mathrm{Rev}(R(\theta_1 - \theta_2, \mathbf{E}_1))$
$\mathbf{G}_{1,\beta_3} = R(\theta_1 - \theta_2, \mathbf{E}_1)$
$\mathbf{G}_{2,\beta_3} = R(\theta_1 - \theta_2 + \theta_3, \mathbf{E}_1)$
$\mathbf{G}_{2,\beta_4} = -\mathrm{Rev}(R(\theta_1 - \theta_2 + \theta_3, \mathbf{E}_1))$
$\mathbf{G}_{1,\beta_4} = -\mathrm{Rev}(R(\theta_1 - \theta_2 + \theta_3 - \theta_4, \mathbf{E}_1))$
$\mathbf{G}_{1,\beta_1}$
$\mathbf{G}_{1,\beta_4}$
$\mathbf{G}_{1,\beta_3}$
$\mathbf{G}_{1,\beta_2}$
$\mathbf{G}_{2,\beta_3}$
$\mathbf{G}_{2,\beta_4}$
$\mathbf{G}_{2,\beta_2}$
$\mathbf{G}_{2,\beta_1}$

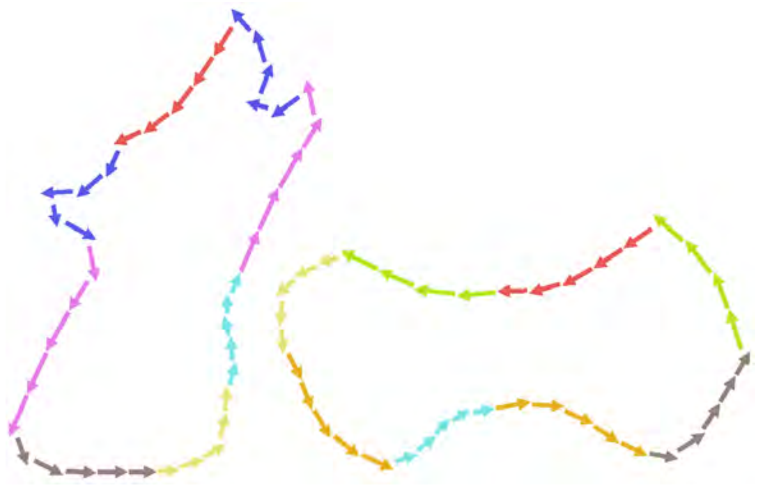
Figure 5.8: Derivation of the formulas for the boundary interval geometry for a single connection constraint.

variables $\mathbf{E}$, we choose one boundary interval, denoted β_r, for each connection constraint i, and one arrangement denoted α_r, to be the "roots" and set $\mathbf{G}_{\alpha_r,\beta_r} = \mathbf{E}_i$. Formulas for the other boundary intervals in the connection constraint can easily be derived by applying the two rules. Figure 5.8 shows an example of this process.

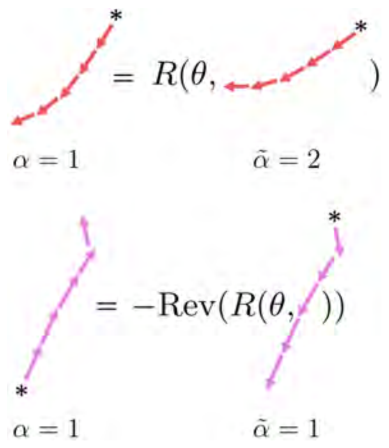
In general, the formula $\mathbf{G}_{\alpha,\beta}$ for a boundary interval in connection constraint i will have the form $R(\pm\theta_{k_1} \cdots \pm \theta_{k_W}, \mathbf{E}_i)$ or $-\mathrm{Rev}(R(\pm\theta_{k_1} \cdots \pm \theta_{k_W}, \mathbf{E}_i))$; i.e., it consists of a series of rotations of the "root" edge vector sequence possibly followed by a reversal and negation.

5.3 Solution Search

The solution search efficiently explores our solution space for high-quality dissections. The main idea is to enumerate partial dissection solutions in a tree structure (Figure 5.10) and prune nodes of the tree when the partial solution they represent has a quality lower bound



(a)



(b)

Figure 5.9: (a) Correspondences for the dog-bone dissection. Edge vectors are colored according to the correspondence to which they belong. Note that correspondences can be between boundary intervals on the same or on different input shapes. (b) Examples of the relationship between edge vector sequences for corresponding boundary intervals. Asterisks mark the beginning of sequences. The indices α and $\tilde{\alpha}$ denote the shape on which each boundary interval lies.

greater than a pruning threshold.

5.3.1 Initializing the Mapping and Constraints

The initialization step divides the boundary of each input shape into intervals. These intervals are the *domain* of the reconstruction mapping—the mapping between input shape boundary intervals and the dissection piece boundary intervals that reconstruct them. The initialization also separates the input shape intervals into corresponding pairs (Figure 5.9). These correspondences restrict the search space to a promising region. Selecting them also initializes the connection constraints that will be used in the dissection. Each correspondence implies the existence of a single connection constraint, as we show next.

Existence of Correspondences. Recall that the reconstruction mapping is a pair of bijections $\Gamma_1(I)$ and $\Gamma_2(I)$, where $\Gamma_\alpha(I)$ maps a boundary interval I on input shape α to a boundary interval on a dissection piece in arrangement α.

We say that an input shape boundary interval I belongs to a connection constraint $\mathbf{C}$ if $\Gamma_\alpha(I) \in \mathbf{C}$. It turns out that the connection constraints partition the input shape boundary intervals into corresponding pairs. Formally, each input shape boundary interval I belongs to some connection constraint $\mathbf{C}$, and there exists exactly one other input shape boundary interval $\tilde{I}$ that also belongs to $\mathbf{C}$.[1]

Any dissection solution will partition the input boundary intervals into corresponding pairs. We choose to specify these corresponding pairs in advance in order to restrict the search space.

Evaluating a Correspondence. Consider a pair of corresponding input shape boundary intervals $(I, \tilde{I})$ that lie on input shapes $(\alpha, \tilde{\alpha})$ and belong to connection constraint $\mathbf{C}$. The eventual dissection solution will map I and $\tilde{I}$ to dissection piece boundary intervals $(\Gamma_\alpha(I), \Gamma_{\tilde{\alpha}}(\tilde{I})) = (\beta, \tilde{\beta})$. Since β and $\tilde{\beta}$ both belong to $\mathbf{C}$ their geometry (in any eventual solution) must obey the constraint $\beta - Q(\tilde{\beta}) = 0$, where $Q(x) = R(\theta, x)$ when $\alpha \neq \tilde{\alpha}$ (intervals lie on different input shapes) and $Q(x) = -\text{Rev}(R(\theta, x))$ when $\alpha = \tilde{\alpha}$ (intervals lie on the same input shape). The value of θ depends on the eventual solution. Figure 5.9(b) visualizes these relations.

To reconstruct the input shapes accurately, the geometry of $(\beta, \tilde{\beta})$ should resemble that of $(I, \tilde{I})$. Thus, in order to allow for the best reconstruction under the connection constraint, the correspondence should minimize $\bar{Q}(I, \tilde{I}) = \min_\theta ||I - Q(\tilde{I})||$; i.e., minimize shape incompatibility given freedom of rotation.

Joint Selection of Intervals and Correspondences. Using this criterion, we jointly select the input shape boundary intervals and correspondences. The input shape boundaries

[1]To see this property, consider an input shape boundary interval I. Without loss of generality, assume I lies on the first input shape. In any eventual dissection solution, I will map to a dissection piece boundary interval β; i.e., $\Gamma_1(I) = \beta$, which belongs to some connection constraint $\mathbf{C}$. So I belongs to $\mathbf{C}$.

In Arrangement 2, β is either external or it connects to a different boundary interval β'. In the first case, there exists a boundary interval on the input shape for Arrangement 2, denoted $\tilde{I}$, for which $\Gamma_2(\tilde{I}) = \beta$. Since β' belongs to $\mathbf{C}$, $\tilde{I}$ belongs to $\mathbf{C}$. So both I and $\tilde{I}$ belong to $\mathbf{C}$. In the second case, the statement just made about β applies to β', except with Arrangement 1 instead of Arrangement 2. Since the number of boundary intervals is finite, we must eventually reach the first case.

are discretized into a large number of uniformly sized segments so that input shape boundary intervals can be defined discretely. Given user-specified minimum and maximum interval lengths, we generate a set of M candidate boundary intervals and $\binom{M}{2}$ candidate correspondences. We seek a set of correspondences $\{(I_1, \tilde{I}_1), \ldots, (I_N, \tilde{I}_N)\}$ which form a partition of the input shape boundaries and minimize the following objective:

$$\sum_{i=1}^{N} \bar{Q}(I_i, \tilde{I}_i), \tag{5.2}$$

which sums the shape incompatibility across the chosen correspondences. The shape incompatibility measure $\bar{Q}$ is defined in the previous subsection. We search for solutions using a simple enumerative approach, ignoring correspondences with poor similarity.

5.3.2 Tree Search

Once the input shape correspondences have been chosen, we enumerate the dissection solutions derived from them using a search tree. This step operates in the discrete connectivity space rather the continuous geometric parameter space.

Each node of the tree corresponds to a partial dissection solution with the root node being the empty solution. A child of a node is generated by extending the node's partial solution by inserting one or more boundary intervals into one or more dissection pieces. The newly created boundary intervals can be connected to other dissection piece boundary intervals or to a boundary interval on an input shape. For the second case, connecting a boundary interval β on a dissection piece to a boundary interval I on input shape α means we set $\Gamma_\alpha(I) = \beta$. In the leaf nodes (complete solutions), every boundary interval will have a connection in both arrangements. Figure 5.10 shows how the tree search incrementally extends partial dissection solutions into complete solutions.

Search Ordering. The choice of the *order* in which to insert the boundary intervals affects the ease of pruning partial solutions. An obvious choice is to order by dissection piece; i.e., a

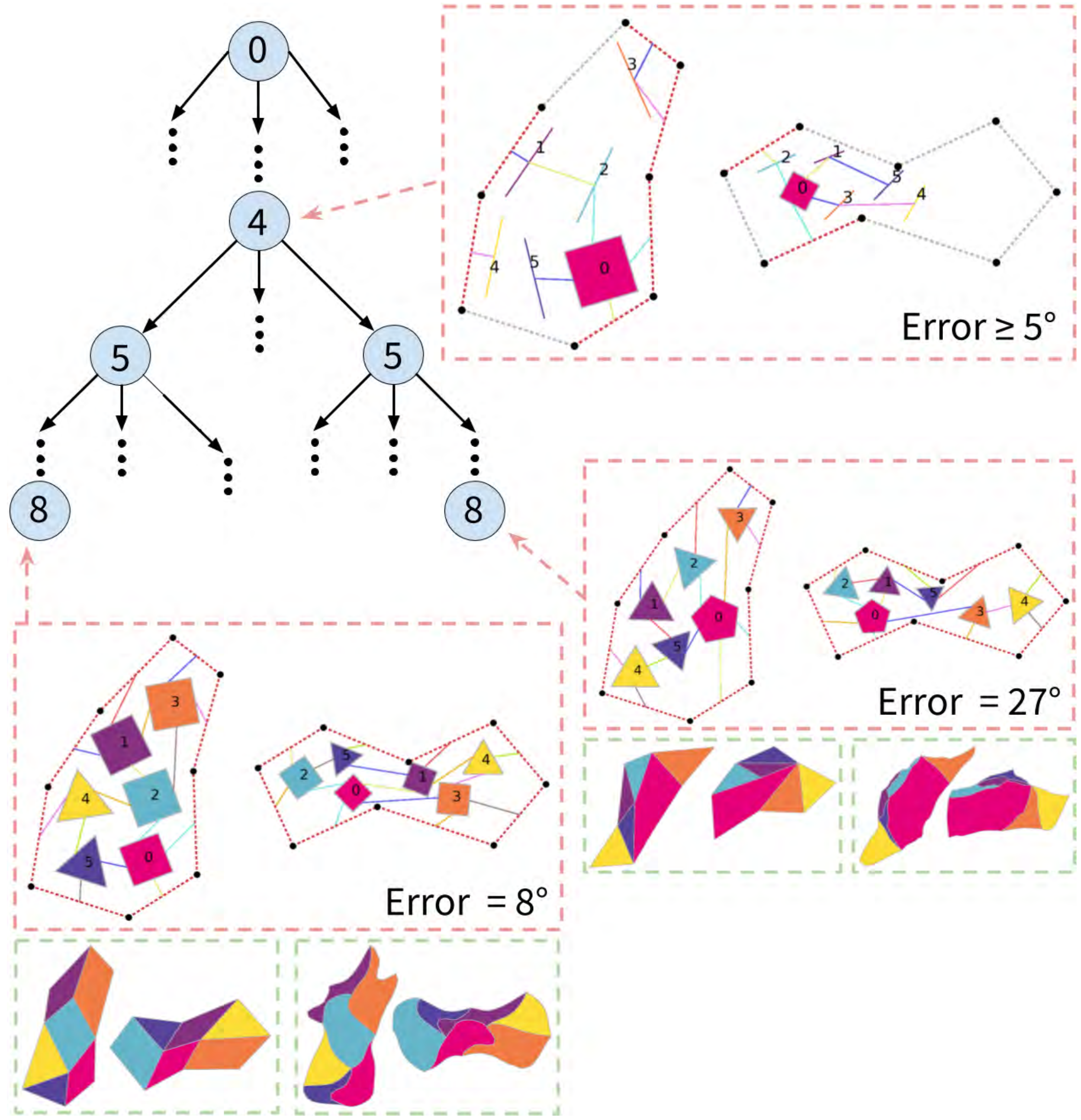
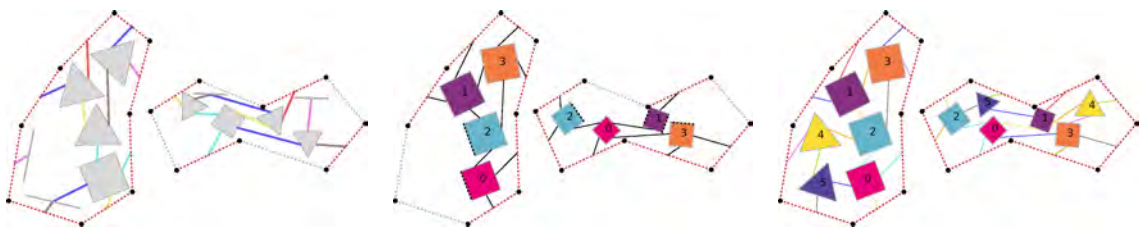


Figure 5.10: Visualizing the tree search. Tree nodes (partial solutions) are labeled with the number of connection constraints added (equal to the search depth). Pink boxes show the connectivity representation at each state along with the associated error for complete solutions and the error bound for partial solutions. Input shapes (at low resolution) are drawn around the connectivity representations in dashed lines. Input shape boundary intervals that have their connection constraint assigned are red, others are grey. Green boxes show the dissection geometry at low (left) and high (right) resolutions. In the partial solution shown at Depth 4, pieces that have only one or two boundary intervals are drawn as line segments.



(a) Connection constraint ordering (Constraints 0 - 5) (b) Dissection piece ordering (Pieces 0 - 3) (c) Complete solution

Figure 5.11: Two different partial solutions (a), (b) which lead to the same complete solution (c). Ordering by connection constraint means that all the boundary intervals in the partial solution have connections. Ordering by dissection piece means that the connectivity of several boundary intervals (dashed lines) is unknown. Boundary interval connections in (a) are colored according to the connection constraint.

node at depth i specifies the boundary intervals for dissection pieces $1, \ldots, i$ (Figure 5.11(b)). However, this ordering has the problem that after we insert the boundary intervals for a given piece, we do not necessarily know which boundary intervals they will connect to, since they may connect to boundary intervals on pieces that we have not yet reached. This uncertainty makes it difficult to formulate a bound on the quality of partial solutions.

Therefore, we order the boundary intervals by connection constraint rather than by dissection piece (Figure 5.11(a)). That is, a node at depth i specifies the boundary intervals that belong to connection constraints $1, \ldots, i$. To generate the children of a node at depth i, we consider all possible ways of generating the boundary intervals for the connection constraint $i + 1$. Each way creates an additional child.

This ordering avoids the problem with the dissection-piece-based ordering, because a boundary interval connects only to other boundary intervals in its connection constraint. In other words, for every boundary interval β in a partial solution, we know that the connectivity of β will not change in any subsequent solution. Figure 5.11 compares the two ways of ordering the search.

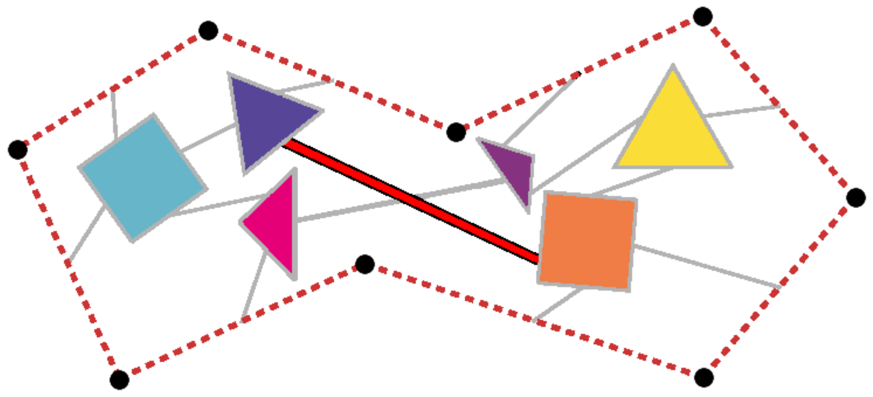


Figure 5.12: An example of a topologically invalid connection (shown in red) between boundary intervals.

5.4 Pruning the Search Space

We now discuss the several ways in which we restrict the search space to make the problem tractable.

Boundary Interval Limit. To ensure that the search terminates, we limit the total number of boundary intervals that can be present in a solution. For the results shown herein, we set this limit to four times the number of connection constraints.

Connectivity. We restrict the search to topologically valid connectivities. We maintain a half-edge data structure throughout the search that detects when connecting a pair of boundary intervals is an invalid operation. Figure 5.12 shows an example.

5.4.1 Orientation Based Pruning

In orientation based pruning, we solve a relaxed version of the optimization problem described in Section 5.2 that gives a lower bound on the solution quality. The main idea is to solve for the dissection geometry in terms of edge vector *orientations* while ignoring magnitudes. Figure 5.13 visualizes this concept.

As in the full optimization problem, we want to minimize the maximum angular difference

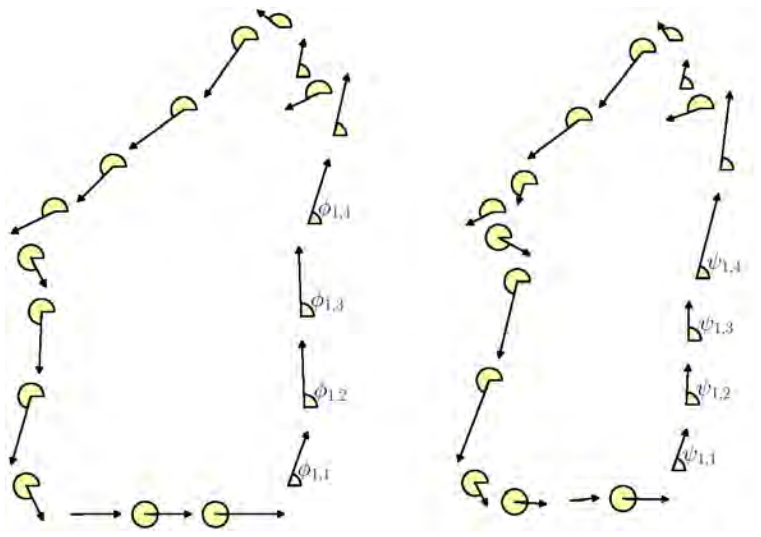


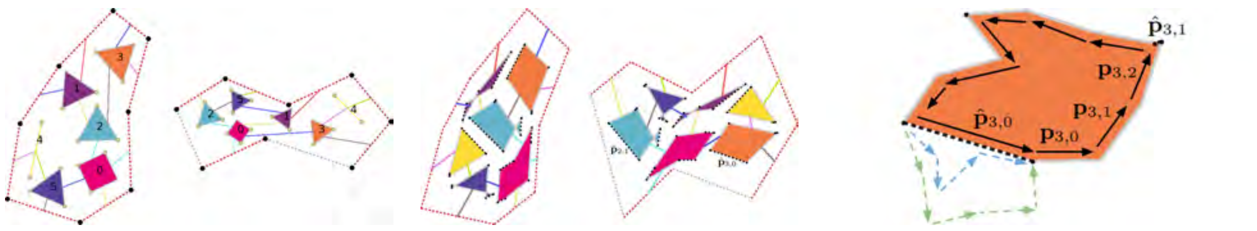
Figure 5.13: Visualizing the angular optimization problem. Vector magnitudes are ignored and we optimize only over vector angles. The notation is shown for some of the angles (ϕ and ψ) used in the optimization problem.

between the input shape edge vectors and their matching dissection piece edge vectors:

$$\underset{\boldsymbol{\theta}_E, \boldsymbol{\theta}, \mathbf{n}}{\text{minimize}} \quad \max_{\alpha, i} |\phi_{\alpha,i} - \psi_{\alpha,i}(\boldsymbol{\theta}_E, \boldsymbol{\theta}) + 2\pi n_{\alpha,i}|, \tag{5.3}$$

where α indexes over arrangements, i indexes over the input shape edge vectors, $\phi_{\alpha,i}$ and $\psi_{\alpha,i}$ are matching input shape and dissection piece edge vector orientations, $\boldsymbol{\theta}_E$ is the set of edge vector orientation sequences for the connection constraints, and $\boldsymbol{\theta}$ is the dissection piece rotation angles. The integer variables $\mathbf{n}$ make the angular differences modulo 2π. This objective is similar to that in (5.1), but expressed only in terms of vector angles. Thus, $\boldsymbol{\theta}_E$ in the angular problem corresponds to $\boldsymbol{E}$ in the full problem, while $\boldsymbol{\theta}$ has the same meaning in both problems.

The dissection piece edge vector orientations $\boldsymbol{\psi}$ can be written as linear functions of $\boldsymbol{\theta}_E$ and $\boldsymbol{\theta}$, by converting the functions for the edge vector geometry in Section 5.2 to angular terms. In general the formulas have the form $\pm\theta_{k_1} \cdots \pm \theta_{k_W} + \boldsymbol{\theta}_{E_i}$ or $\pm\theta_{k_1} \cdots \pm \theta_{k_W} + \pi + \text{Rev}(\boldsymbol{\theta}_{E_i})$. The second form corresponds to the form in the full problem where we reverse and negate the edge vector sequence, because negating a vector adds π to its orientation.



(a) Partial Solution Connectivity (b) Partial Solution Geometry (Low Resolution) (c) Piece 3 Edge Vectors (High Resolution)

Figure 5.14: Visualizing the geometry optimization for a partial solution. (a) The connectivity representation of a partial solution where the boundary intervals for the last connection constraint have not been inserted. Locations where new boundary intervals can be inserted are marked with gold dots. (b) The corresponding (low resolution) geometry after optimization. Potential boundary intervals are drawn as dotted lines. (c) The edge vectors for Piece 3 at higher resolution. The edge vectors for boundary intervals in two different complete solutions are shown in blue and green.

The optimization may be written as a mixed integer optimization, where the variables $\mathbf{n}$ are constrained to be integers, and $\boldsymbol{\theta}_E$ and $\boldsymbol{\theta}$ are continuous. We solve it using a commodity solver (Gurobi Optimization, Inc., 2016).

This optimization immediately generalizes to partial solutions. In (5.3), we maximize only over the edge vectors belonging to connection constraints that are specified in the partial solution. Due to the connection constraint-based ordering, the orientation formulas for these edge vectors will be defined and will not change in any subsequent solution. When a new connection constraint is added to the partial solution, the maximization is taken over additional terms.

5.4.2 Full Geometry Pruning

The full geometry pruning generalizes the dissection geometry optimization in Section 5.2 to partial dissection solutions. The objective value given by this optimization is a lower bound, meaning that no complete solution derived from the partial solution will attain a better objective value. Figure 5.14 visualizes this optimization. The main idea is to relax the discreteness of the problem by allowing for the insertion of "fractional" boundary intervals.

The structure of the objective function and constraints in this optimization are identical to that in (5.1). However, they are expressed differently, due to the incomplete information in a partial solution.

Objective for Partial Solutions. Recall that the objective function compares each input shape edge vector $\mathbf{t}_{\alpha,i}$ with its corresponding dissection piece edge vector $\mathbf{b}_{\alpha,i}$ (Figure 5.7(a)). Each $\mathbf{t}_{\alpha,i}$ lies on an input shape boundary interval I that belongs to a connection constraint $\mathbf{C}$. If the partial solution has not yet added the boundary intervals for $\mathbf{C}$, then the vector $\mathbf{b}_{\alpha,i}$ is unknown. Next, we show how to express it in terms of other variables.

Suppose I lies on input shape α. We know that in any complete solution derived from the partial solution, I will be reconstructed by some boundary interval β. Formally, we know there will exist a β for which $\Gamma_\alpha(I) = \beta$. As discussed in Section 5.3.1, I has a corresponding boundary interval $\tilde{I}$ on input shape $\tilde{\alpha}$. Thus β will have a corresponding dissection piece boundary interval $\tilde{\beta} = \Gamma_{\tilde{\alpha}}(\tilde{I})$. The boundary intervals β and $\tilde{\beta}$ will belong to the same connection constraint, so they both can be expressed as functions of an edge vector sequence $\mathbf{E}_i$. Denoting these functions as $\bar{\mathbf{G}}_\beta(\mathbf{E}_i)$ and $\bar{\mathbf{G}}_{\tilde{\beta}}(\mathbf{E}_i)$, we define them as $\bar{\mathbf{G}}_b(\mathbf{E}_i) = \mathbf{E}_i$ and $\bar{\mathbf{G}}_{\tilde{\beta}}(\mathbf{E}_i) = -\text{Rev}(R(\hat{\theta}, \mathbf{E}_i))$ if $\alpha = \tilde{\alpha}$, and $\bar{\mathbf{G}}_{\tilde{\beta}}(\mathbf{E}_i) = R(\hat{\theta}, \mathbf{E}_i)$ otherwise. $\hat{\theta}$ is added to the optimization as a free variable; i.e., it is independent of the rotation angles for the dissection pieces. Note that these functions are like $\mathbf{G}_{\alpha,\beta}$ from Section 5.2, except that the sum of dissection piece rotation angles is replaced by the free variable $\hat{\theta}$. These functions provide the values for any unknown $\mathbf{b}_{\alpha,i}$.

Constraint for Partial Solutions. The constraint for dissection piece l ensures that the edge vectors around the piece $\mathbf{p}_{k,l}$ form a closed curve (Figure 5.7(b)). This formulation does not work for a partial solution, because some of the pieces will have new boundary intervals inserted in any complete solution descended from the partial solution. Thus, it is incorrect to take the sum only around the existing $\mathbf{p}_{k,l}$.

We deal with this issue by inserting *potential* boundary intervals into each dissection piece at points on the piece's boundary where it would be legal to insert a new boundary interval

(Figure 5.14). Each potential boundary interval represents geometry that could potentially be present in an eventual complete solution. We represent the potential geometry for dissection piece k at insertion point v with a single edge vector, denoted $\hat{\mathbf{p}}_{k,v}$. These edge vectors are free variables in the optimization.

More specifically, $\hat{\mathbf{p}}_{k,v}$ represents the *net translation* of zero or more potentially inserted boundary intervals. By net translation, we mean the vector obtained by summing the edge vectors in the boundary interval. It suffices to consider only the net translations of the potential boundary intervals, because they are connected to nothing in this setting. Figure 5.14(c) illustrates this principle. The single edge vector $\hat{\mathbf{p}}_{3,0}$ has the same net translation as the two edge vector sequences shown in blue and green.

The constraints from (5.1) are modified to take the potential geometry into account, as follows:

$$\sum_{l} \mathbf{p}_{k,l} + \sum_{v} \hat{\mathbf{p}}_{k,v} = \mathbf{0}, \quad \forall k. \tag{5.4}$$

We can tighten the bound by constraining the total length of the potential geometry. Suppose our search has an upper limit of S boundary intervals in the solution, and our partial solution has already inserted S' boundary intervals. Then, we can have at most $S - S'$ additional boundary intervals in any eventual complete solution. Denoting the *net length* of the edge vector sequence for connection constraint i as $L_i = ||\sum_j \mathbf{e}_{i,j}||$ (i.e., L_i is the magnitude of the net translation of the edge vector sequence for connection constraint i), and letting $L_{\max} = \max_i L_i$, we can add the constraint

$$\sum_{k} \sum_{v} ||\hat{\mathbf{p}}_{k,v}|| \leq (S - S')L_{\max}. \tag{5.5}$$

This constraint reflects the fact that we can add at most $S - S'$ boundary intervals and each boundary interval consumes at most $L_{\max}$ length.

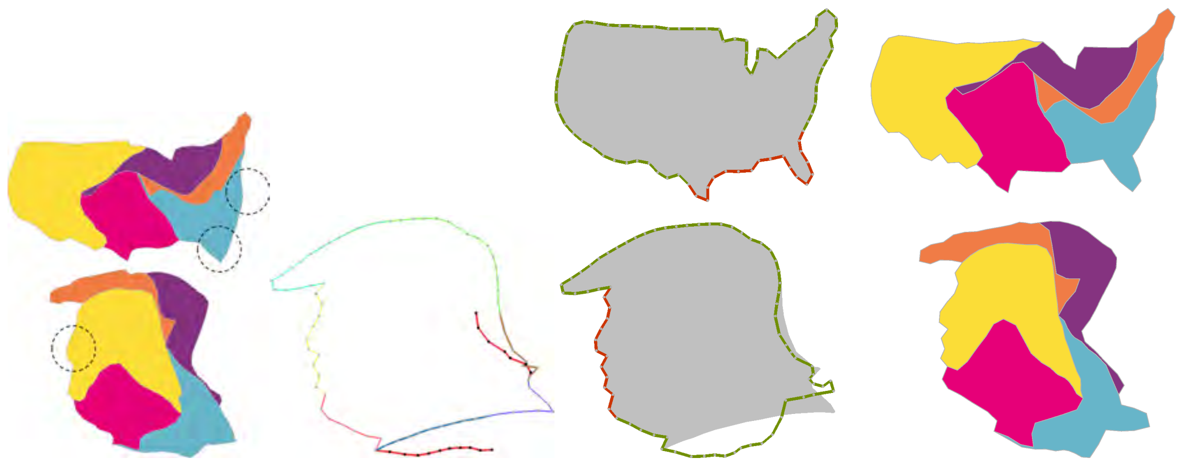
The use of potential boundary intervals is conceptually similar to a continuous relaxation of an integer programming problem. The number of boundary intervals inserted into a piece is a discrete property, like the value of an integer variable. The potential boundary intervals relax the discreteness by allowing for the insertion of a "fraction" of a boundary interval, just like continuous relaxation allows for assigning fractional values to an integer variable.

5.4.3 Pruning Usage

We now describe how the pruning tests are incorporated into the tree search.

The two pruning tests described in the previous subsection require the solution of an optimization problem, which incurs a significant computational expense, especially for the full geometry pruning (Section 5.4.2). Additionally, at a sufficient depth in the search tree, a significant proportion of tree nodes (partial solutions) do not expand into *any* complete solutions, because it is impossible to form a solution with valid connectivity. Expanding a tree node (by inserting boundary intervals for a given connection constraint and validating their connectivity) is extremely cheap relative to the pruning tests. Therefore, for these nodes it is more efficient (on average) to check if the node expands to any complete solutions, and run the pruning tests only if it does. For even deeper depths of the search tree, it is most efficient to refrain from running the pruning tests at all.

Procedure Details. For all the results shown, we used the following search procedure. Let M equal the number of connection constraints we obtain from the correspondence search described in Section 5.3.1. First, all partial solutions at depth $M/2$ are generated and the pruning tests are applied to them. Next, the remaining partial solutions are assigned between P processes that run in parallel. Each process expands its partial solutions serially. At depth $3M/4$, we apply the pruning tests again, but only after verifying that the node contains complete solutions as described above. This simple procedure could likely be improved, but we found it performed acceptably in practice. We set the angular error threshold for pruning to $\min(15°, 1.5 \cdot \tau_{\min})$, where $\tau_{\min}$ is the lowest error among the solutions found so far.



(a) Initial Design (b) Suggested Edits (c) Edited Input Shapes (d) Final Design

Figure 5.15: A workflow in our user interface. (a) The initial design from the automatic approach. Problematic areas are circled. The details of the face have been smoothed out and the region around the state of Florida is distorted. (b) The original input shape with edit suggestions in red. (c) The input shapes after the editing and painting of salient regions (red). The original input shape is shown in gray. (d) The final design after user edits. The details of the face have been restored and Florida is no longer distorted.

5.5 User Interaction

Our solution search uses a geometric criterion to evaluate the quality of a dissection. This criterion works well for weeding out poor solutions, but it has difficulty discriminating between high-quality solutions because it does not explicitly consider human perception. We resolve this shortcoming by allowing user guidance towards a final dissection solution, after the automatic approach described in Section 5.3 has identified a small set of promising candidate solutions.

The user selects one of the candidate solutions, which they can refine by adjusting the terms of the optimization in (5.1) through a graphical user interface. Figure 5.15 shows an example of this process.

In this phase of our approach, the optimization objective uses a least squares term instead of the maximum angular error in order to control the tradeoff in deformation between different parts of the mesh. The new objective is $\sum_\alpha \sum_i w_{\alpha,i} ||\mathbf{b}_{\alpha,i} - \mathbf{t}_{\alpha,i}||^2$, where $w_{\alpha,i}$ is the preservation weight of edge i on input shape α.

Input Shape Editing Suggestions. Our approach targets input shapes that have a fuzzy geometric specification. Thus, we allow the user to alter the input shapes to simplify the optimizer's job. Such edits are suggested to the user by overlaying dissection piece boundary intervals on top of their corresponding input shape boundary intervals as red curves. The user can follow the suggestions by moving the input shape edges towards these curves. The user often makes additional edits to preserve the shape's identity or coherency. Figure 5.15 shows an example of this process where the user reshapes the bottom and side of the head.

Salient Region Painting. Some areas on the input shapes are *salient* in that human perception is especially sensitive to distortions in them. The user can paint salient parts of the input shape and the optimizer will prioritize them by increasing the preservation weights for their edges. In Figure 5.15(c) the user prioritizes the preservation of the protrusions forming the states of Texas and Florida as well as the nose and lips of the head.

Direct Editing. The user can also directly edit the dissection pieces. To satisfy the connection constraints, these edits will propagate to the geometry of other pieces. This propagation is visualized interactively.

CHAPTER 6

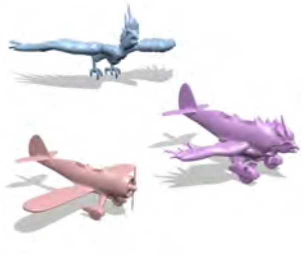
Experiments and Results

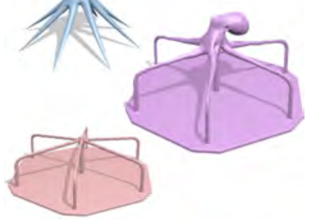
In this chapter, we evaluate our solutions to the three problems covered in this thesis and analyze of the results. First, in the Zoomorphic Design problem, we generate a wide variety of zoomorphic shapes, run our approach under different constraints and parameter settings, and compare our results with zoomorphic shapes designed by artists. We also test the perception of the zoomorphic shapes with a simple user study. Second, in the Interchangeable Components problem, we apply our method to a wide variety of shapes, and test it under challenging scenarios, such as making interchangeable components from input shapes from very different categories, and perturb the initial semantic segmentation. Finally, for the Approximate Dissection problem, we use our approach to create dissections between a wide variety of complex shapes. We measure the extent to which our search tree pruning improves the method's performance and how much the approximation accuracy improves as the number of pieces in the dissection grows. We also perform a simple user study demonstrating that the dissections can be used to make a challenging puzzle.

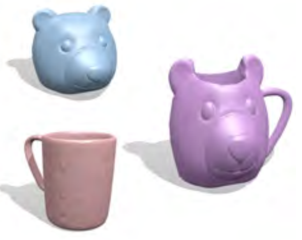
6.1 Zoomorphic Design

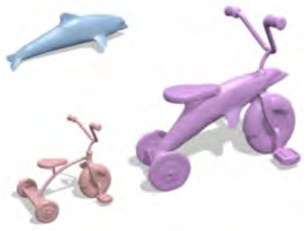
We have applied our zoomorphic design technique to a variety of models taken from free 3D model repositories on the internet[1], the Princeton Shape Benchmark (Shilane et al., 2004)

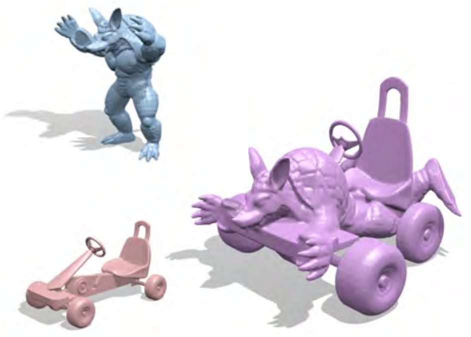
[1]3DVIA, Archive 3D and Autodesk 123D

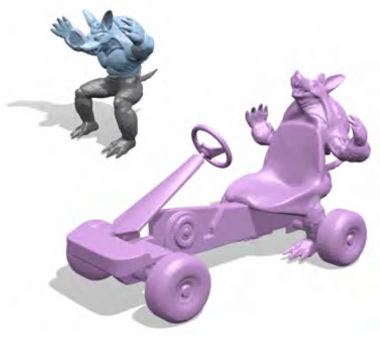

(a) Phoenix Plane

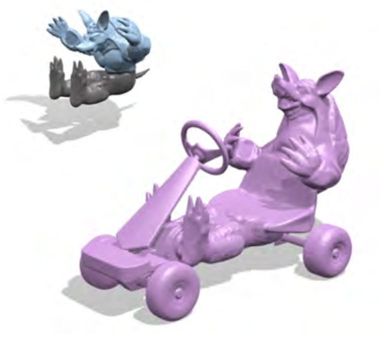

(b) Octopus Carousel

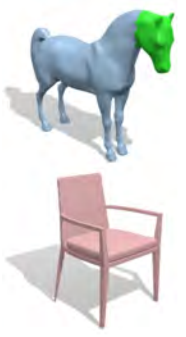

(c) Bear Mug

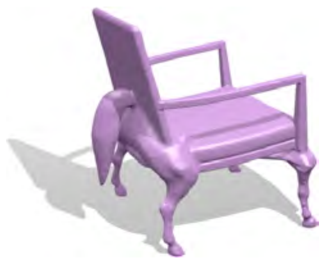

(d) Dolphin Tricycle

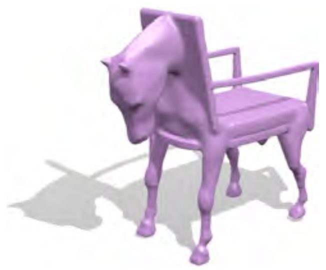

(e) No User Constraint

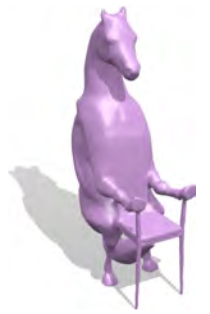

(f) User Constraint 1

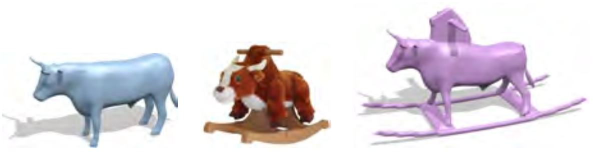

(g) User Constraint 2

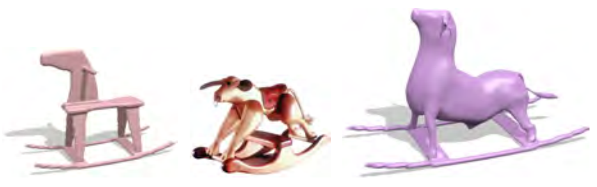

(h)


(i) Low Visual Salience


(j) Medium Visual Salience


(k) High Visual Salience


Small Animal object Deformation


Large Animal object Deformation

(l) Rocking Cow

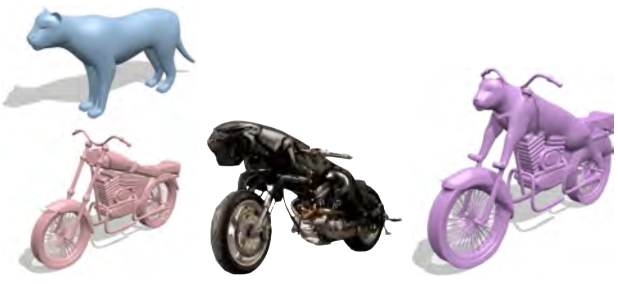

(m) Jaguar Motorcycle

Figure 6.1: Zoomorphic designs created by our system.

and the Shape COSEG Dataset.[2] Figures 6.1, 3.14, and 6.2 show the results.

Figures 6.1(a)–(d) shows a variety of zoomorphic designs created using our system. The volumetric design restriction ensures that the airplane's cockpit and wheels are preserved, that the octopus leaves room to occupy the carousel, that the Bear Mug can contain fluid, and that the dolphin's fin doesn't prevent sitting on the tricycle.

6.1.1 Pose Constraints.

We allow the user to rigidly constrain parts of the animal object and optimize the non-constrained parts of the animal object and base object. We show examples of this process in Figures 6.1(e)–(g). Different constraints lead to different proportions in the base object. In each case, the VDR ensures that there is room for riders to place their feet on the front of the go-kart. For the non-constrained example, the presence of the armadillo's hands causes the front of the go-kart to widen considerably.

6.1.2 Changing Weights.

Figures 6.1(i)–(k) shows the effects of using different visual salience weights to produce interesting yet reasonable changes in the design. As the visual salience weight increases, the horse's head grows increasingly prominent in the zoomorphic object. In the least salient configuration, the horse's head is completely removed. In the next most salient configuration, it is visible, but blocked from some viewpoints by the back of the chair. In the most salient configuration, it remains visible from all view angles. Here, the VDR ensured that the horse's head is high enough that someone can sit underneath it. Figure 6.1(l) shows the effects of changing the animal object deformation weight. Using a large deformation weight for the cow results in a small deformation. This creates a Rocking Cow in which the cow remains close to its initial pose, and the handles from the base object are used. In contrast, using a

[2] `http://web.siat.ac.cn/~yunhai/ssl/ssd.htm`

small deformation weight for the cow causes the cow to sharply bend its head such that its horns replace the base object handles.

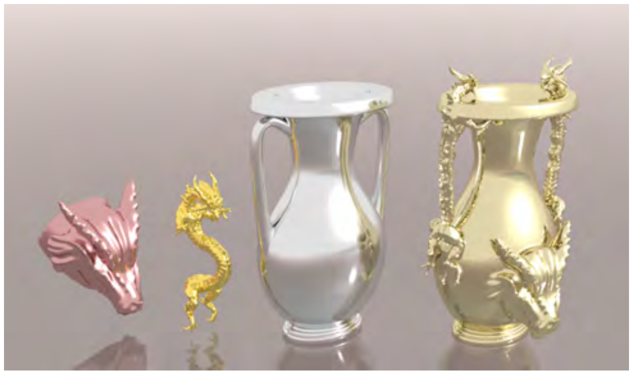
6.1.3 User Guidance.

Zoomorphic objects are often enhanced when the animal object is placed in a pose of semantic significance, which our approach cannot recognize automatically. Fortunately, the fine scale optimization step can naturally incorporate user guidance which can provide such semantics. In Figure 3.14, the initial solution found by our optimizer is altered by the user, who wants the horse's legs stretched out in a leaping pose. The manual adjustment puts the horse's head in a pose that creates gashes, increasing the energy. This prompts our optimizer to find a configuration close to the user's adjustment, but with lower energy because the gashes are removed. The gashes arose because the horse's head blocked the rider's view and therefore intruded into the restricted zone of the VDR.

Note that most of our results required a non-trivial set of changes in order to create aesthetically pleasing zoomorphic objects that satisfied the design restrictions of the base object. For example, to create the Horse Tricycle in Figure 3.14(a), we need to ensure that the horse's body merges naturally with the seat and main support, while not blocking the view of the rider or their ability to reach the pedals. In satisfying these objectives, the optimization could not deform the horse unnaturally. To get a good visual salience energy, it also needed to place the horse's head in a prominent location. The optimization process slightly raises the seat to match with the horse body while bending the neck and legs to avoid blocking the rider.

6.1.4 Other Results.

Our approach can be easily applied to create zoomorphic designs similar to real-world designs. Figure 6.1(l) shows our design of Rocking Cows and Figure 6.1(m) shows our Jaguar


(a) Dragon Vase

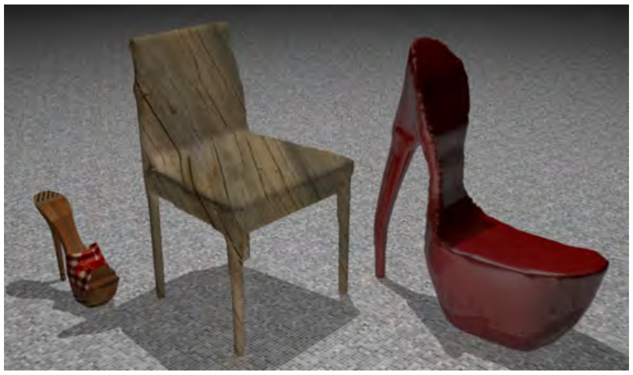

(b) Shoe Chair

Figure 6.2: (a) A simple extension of our approach allows creating a zoomorphic object using multiple animal objects. A Dragon Vase is created by merging a vase with several dragons. (b) Our approach is not limited to animal objects. In this example, a shoe and a chair are used to create a Shoe Chair.

Motorcycle, along with their real-world counterparts. In designing a zoomorphic object, it is also possible to merge multiple animal objects with a base object. Figure 6.2(a) shows the design of a Dragon Vase by merging three dragons with a vase. Figure 6.2(b) shows an example of merging a non-animal object (a shoe) with a base object (a chair), which results in a Shoe Chair. Figure 6.3 shows an application of our approach to virtual scene modeling. We create a "zoomorphic restaurant" furnished with a Manatee Chair, an Octopus Lamp, a Dolphin Bottle, and Fish Plates.

Our approach produces physically realizable results in some cases. Figure 6.4 shows a fabricated version of the Horse Chair from Figure 3.11.

6.1.5 Performance.

We tested an unoptimized implementation of our approach on a 2.4 GHz laptop. By far, the most time-consuming tasks are the correspondence search and configuration refinement, which on average take about 1.5 minutes and 1.0 minute, respectively. Our approach can provide immediate feedback in some situations where it would be highly desirable, such as seeing how the results of the correspondence search change after changing the weights used

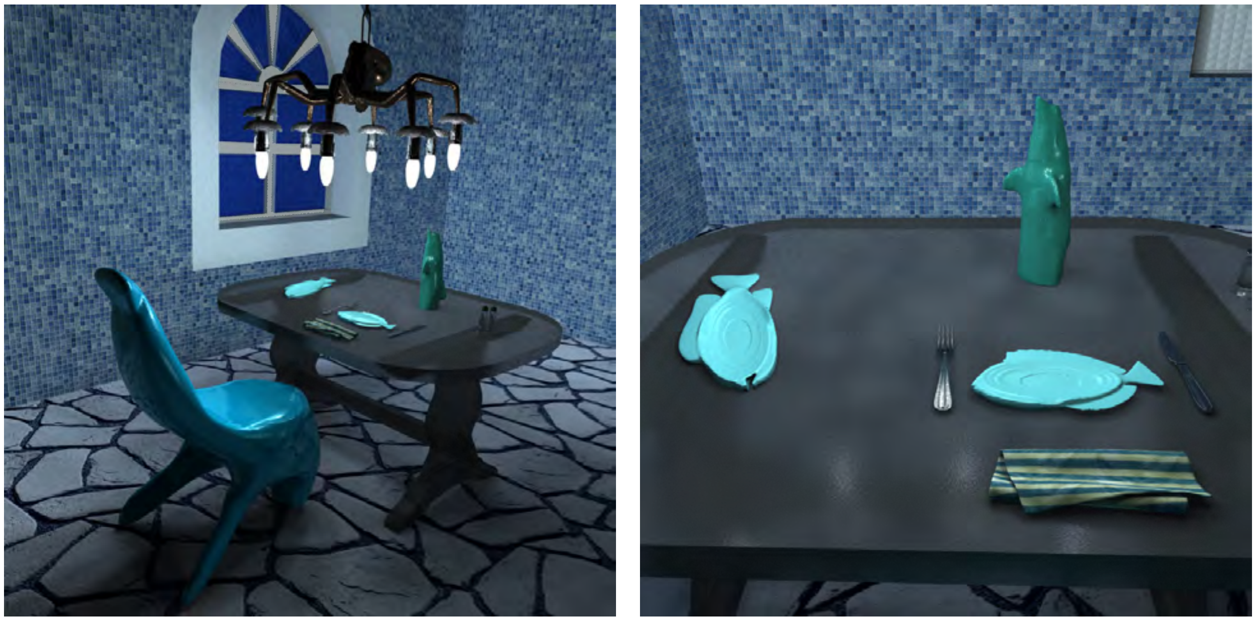


Figure 6.3: A corner of a “zoomorphic restaurant” furnished with a Manatee Chair, an Octopus Lamp, a Dolphin Bottle, and Fish Plates.

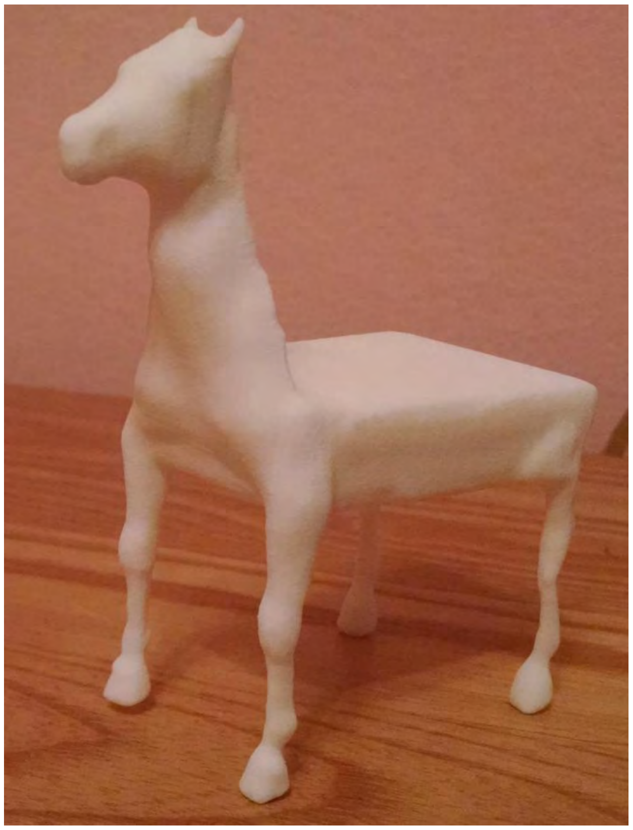


Figure 6.4: A 3D-printed horse chair

to select the best correspondence, and seeing how the input objects deform after adjusting the slider in the base object control.

6.1.6 Evaluation

We conducted informal studies to evaluate our results. The settings of our studies are similar to those in other recent creative 3D modeling efforts (Kalogerakis et al., 2012; Alhashim et al., 2014; Zheng et al., 2013). Volunteers, who were university students from different majors, were recruited to participate in our studies. Each participant was shown a sequence of images via a web browser. Each participant was required to answer a question about each image. There were two tests in our studies.

The first test aimed at determining whether the results are zoomorphic—the main theme of our work. At the beginning, the participant was given the definition of zoomorphism[3] and was shown several images of zoomorphic objects. Next, the participant was randomly presented a sequence of 20 images. Ten images showed a base object and the other 10 images showed a zoomorphic object (a result). For each image, the participant had to decide whether or not the object shown was zoomorphic, or indicate they could not decide.

The second test aimed at determining whether the zoomorphic objects are plausible examples of the type of base object from which they were derived. The participant was presented a random sequence of 20 images, each of which showed a zoomorphic object (a result) created with or without the volumetric design restriction. For each image, the participant had to answer the question "Is this a plausible *category*?", where *category* refers to the object category of the base object from which the zoomorphic object in question was created. For example, for the Horse Chair, the participant was asked "Is this a plausible chair?". The answer could be either "yes", "no", or "undecided".

We collected 880 responses from 44 participants for our first test. 89.55% of the results were

[3]From Wikipedia: "the shaping of something in animal form or terms".

regarded as zoomorphic, while 86.14% of the base objects were regarded as non-zoomorphic. From informal interviews after the studies, we found that some of the results were not chosen as zoomorphic because the animal quality was subtle. The dolphin tricycle in Figure 6.1(d) is an example; some participants were unaware that a dolphin has been merged with the tricycle's frame.

We collected 820 responses from 41 participants for the second test. 85.85% of the results created with the volumetric design restriction were regarded as visually plausible. Participants pointed out a few problems in the results that make them seem implausible despite the VDR—the Horse Chair with high visual salience may not be structurally stable; the sitting area for the Jaguar Motorcycle may be too small. On the other hand, only 36.59% of the results created without the volumetric design restriction were regarded as visually plausible. These results support our belief that the VDR can preserve in the zoomorphic object certain essential features from the base object that would otherwise be lost.

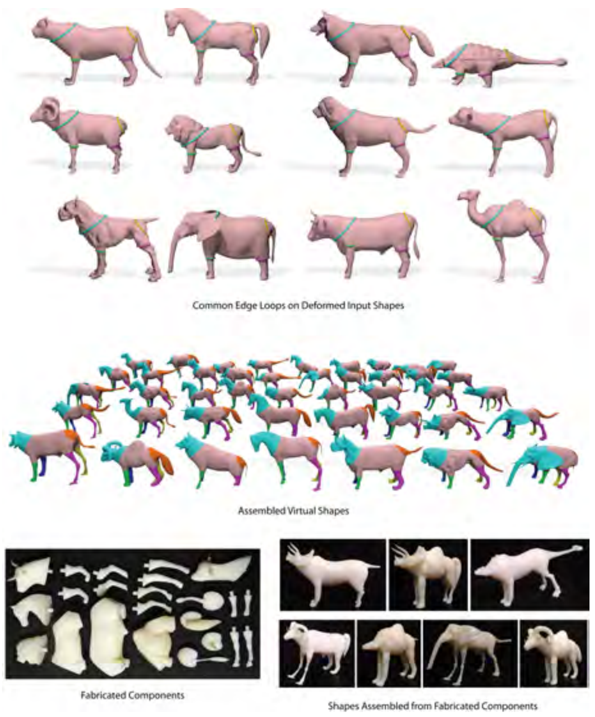
We emphasize that our studies are meant only as an informal preliminary evaluation rather than a scientific validation. However, they do provide us with useful insights about the design of zoomorphic objects. Appendix A provides further details and a list of all the objects used.

6.2 Interchangeable Components

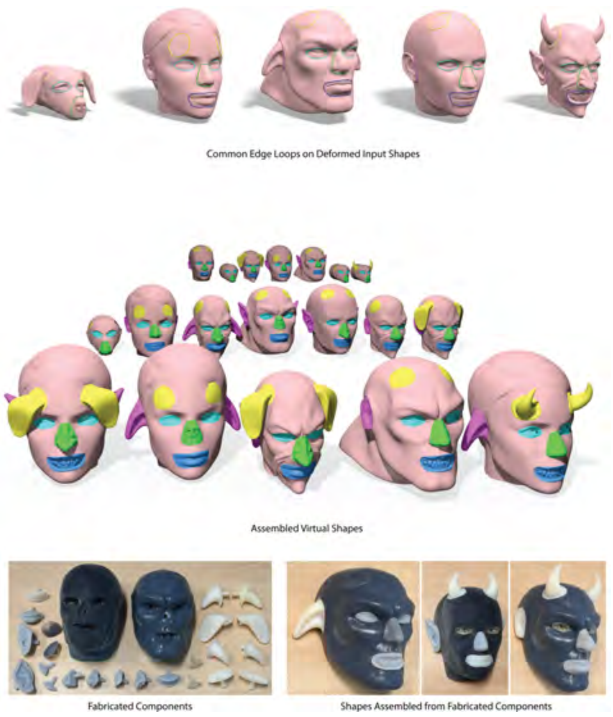
6.2.1 Different Categories

We applied our approach to generate components for five types of shapes: animals, faces, chairs, humanoids, and insects. The input shapes were taken from free 3D model repositories on the internet. Appendix B provides an alternate visualization of these results.

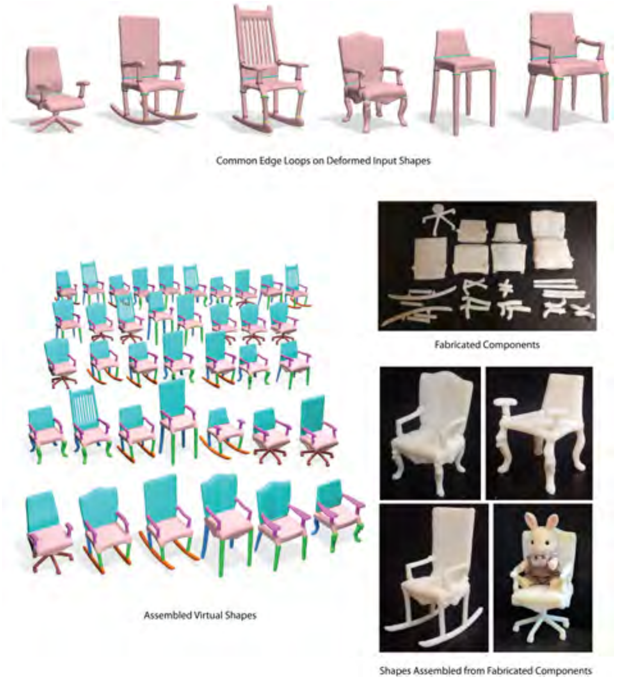
Animals. In this experiment we used our approach to generate a set of components for constructing chimeric four-legged creatures. Twelve animal shapes were used as the input to our approach.


Common Edge Loops on Deformed Input Shapes
Assembled Virtual Shapes
Fabricated Components
Shapes Assembled from Fabricated Components

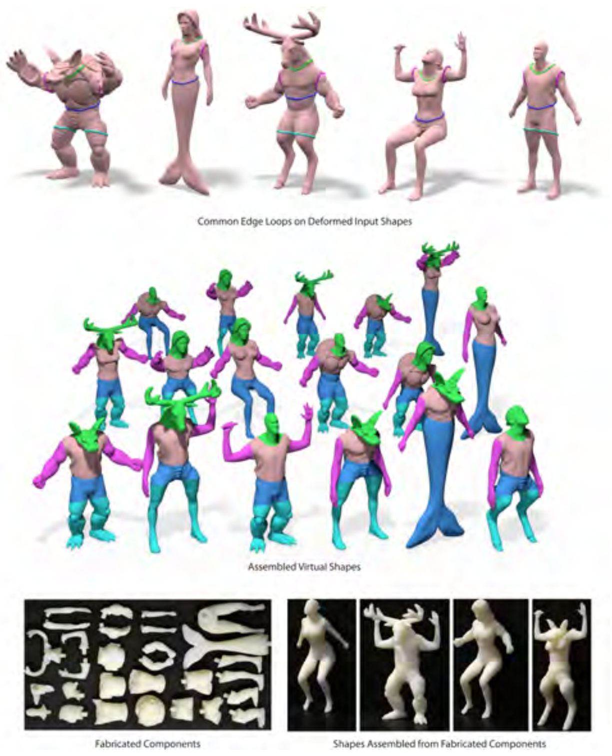
(a) Animals


Common Edge Loops on Deformed Input Shapes
Assembled Virtual Shapes
Fabricated Components
Shapes Assembled from Fabricated Components

(b) Faces


Common Edge Loops on Deformed Input Shapes
Fabricated Components
Assembled Virtual Shapes
Shapes Assembled from Fabricated Components

(c) Chairs


Common Edge Loops on Deformed Input Shapes
Assembled Virtual Shapes
Fabricated Components
Shapes Assembled from Fabricated Components

(d) Humanoids

Figure 6.5: Generating interchangeable components for different types of shapes.

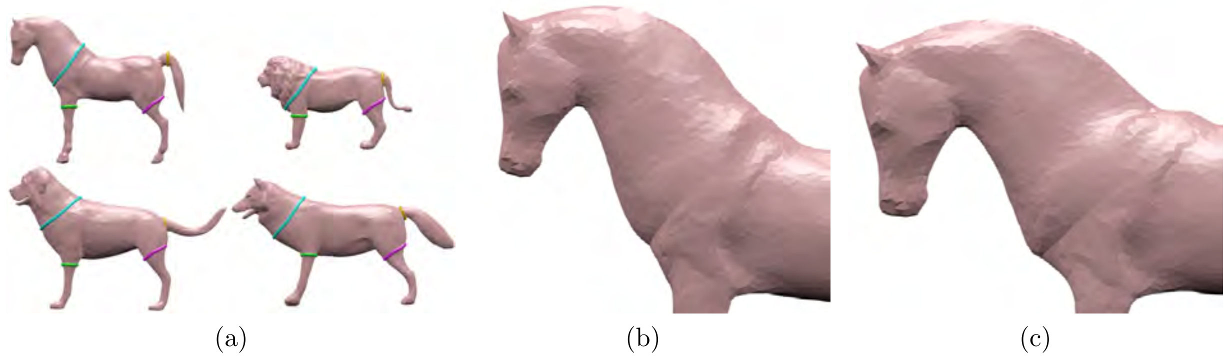


Figure 6.6: (a) Initial edge loops selected for the most-compatible size 4 subset of animals. (b) Deformation of the horse using the most-compatible subset. (c) Same when using the full set of animals.

We show the edge loops chosen by our compatibility-based optimization in Figure 6.5(a). The loops for the tail component often chop off a small portion of the animal's rear, because the loops on the actual tail did not possess enough geometry diversity. Despite the wide geometric variation in the input set, our approach chooses edge loops and deformations that make the assembled shapes look coherent. Figure 6.14 shows a diverse set of animals assembled by our interchangeable components.

This result was inspired by several similar commercial products. However, all these products either possess a much more limited set of constructible shapes than our result or make it very obvious that the shapes are composed of components (Figure 4.12).

Most Compatible Subset of Animals: We applied our most-compatible subset technique (Section 4.5.4) to find the most compatible size 4 subset of the full set of animals. We show the subset and some results in Figure 6.6. The algorithm selected the lion, horse, wolf, and dog. Note that the selected edge loops for the (tail, body) border actually partition the tail, unlike the selected edge loops in the full set of animals. We note that these shapes have to undergo less deformation for component compatibility than when they are used in the full set.

Faces. We used our approach to generate components for assembling various stylized faces (Figure 6.5(b)). This result was inspired by the Mr. Potato Head toy. The Mr. Potato Head toy consisted of abstract faces, but ours possess full detail.

Faces are a challenging example in our problem setting because humans are sensitive to minor distortions in facial appearance. Indeed, the initial deformation from our approach caused the devil face to loose its characteristic grin. Fortunately, our incorporation of interactivity into the approach allows us to resolve the problem with a simple user edit (Section 4.5.3).

By segmenting meshes appropriately, our approach allows the user to connect components to an object that did not originally have them. For example, we connect horns to the ogre and human face, which originally had no horns. Even in narrow, concave regions, like the eyes, our approach generates components that connect physically.

Chairs. We used our approach to generate components for assembling several types of chairs (Figure 6.5(c)). These could be used to furnish a doll house or even as real furniture if they were fabricated at a large enough scale.

The extension to our approach to higher order connectivity (Section 4.5.2) made the arm and rocking chair legs interchangeable. These more stringent constraints created noticeable deformations in some of the chairs, but none of that affected their functionality.

The ability of our components to turn any chair into a rocking chair is a simple example of how interchangeable components can alter the functional properties of shapes.

Humanoids. We generate components for several figurines which allow the user to replace the legs with a mermaid's tail and incorporate pose variation (Figure 6.5(d)). Despite a challenging amount of diversity in the input models, such as the armadillo's lack of a neck, our approach arrives at a solution that makes the assembled humanoids appear plausible. Figure 6.7 shows a diverse set of humanoids assembled by our interchangeable components.

Insects. We generate components from the set of five insects and a scorpion shown in Figure 6.8. All insects have a head, thorax, abdomen and six legs, but these body parts possess an extraordinary amount of diversity, making insects an especially challenging test for our approach. In Figure 6.8 we highlight the challenges involved in creating interchangeable components for the Stag Beetle. The initial semantic regions of the Stag Beetle and the edge

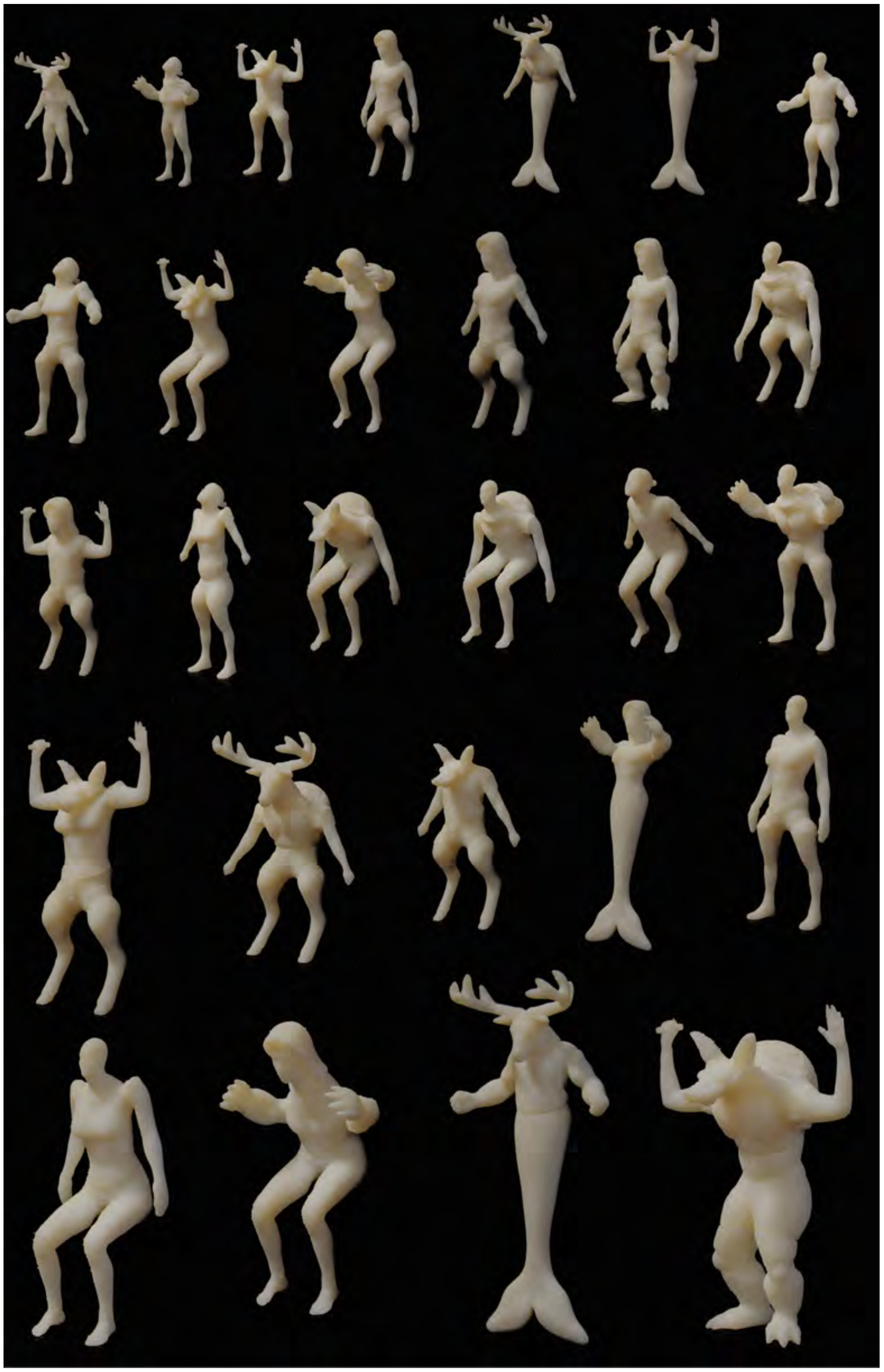
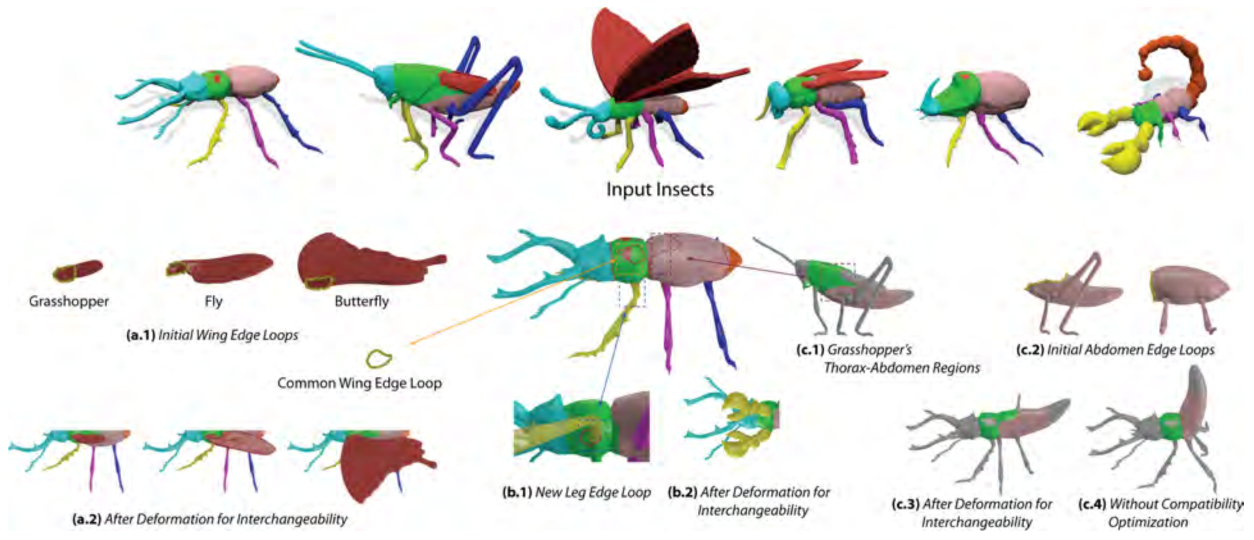


Figure 6.7: Humanoids assembled by our components.



Input Insects
Grasshopper
Fly
Butterfly
(a.1) Initial Wing Edge Loops
Common Wing Edge Loop
(c.1) Grasshopper's Thorax-Abdomen Regions
(c.2) Initial Abdomen Edge Loops
(a.2) After Deformation for Interchangeability
(b.1) New Leg Edge Loop
(b.2) After Deformation for Interchangeability
(c.3) After Deformation for Interchangeability
(c.4) Without Compatibility Optimization

Figure 6.8: Some of the challenges involved in generating insect components. Refer to the text for explanation.

loops found by our compatibility optimization are shown in the center.

In Figure 6.8(a) we focus on the wings. The approach must find an area on the beetle's thorax for the grasshopper, fly, and butterfly wings to attach, without straying into adjacent semantic regions. Despite this issue and the different shapes for the initial wing boundaries (a.1), our approach finds a common edge loop shape that does not distort the wings and allows all of them to connect to the beetle. Since the attachment point needs to accommodate the large butterfly wings, it barely fits onto the thorax.

Figure 6.8(b) shows how by extending the leg-thorax edge loop to the beetle's thorax our approach allows it to accommodate the much larger scorpion pincer.

Figure 6.8(c) highlights an unexpected, yet beneficial deviation from the original semantic regions in determining the individual edge loops for the abdomen-thorax connection. The grasshopper possesses a very different layout of the wings, thorax and legs than the beetle (c.1). The initial edge loops are very different in shape (c.2). Our algorithm recognizes that the grasshopper does not have the flexibility to adjust its edge loop very much, and instead modifies the beetle's to get closer, which cuts a large section of its abdomen. Despite this deviation from the proper anatomy, the components connect to form a plausible shape

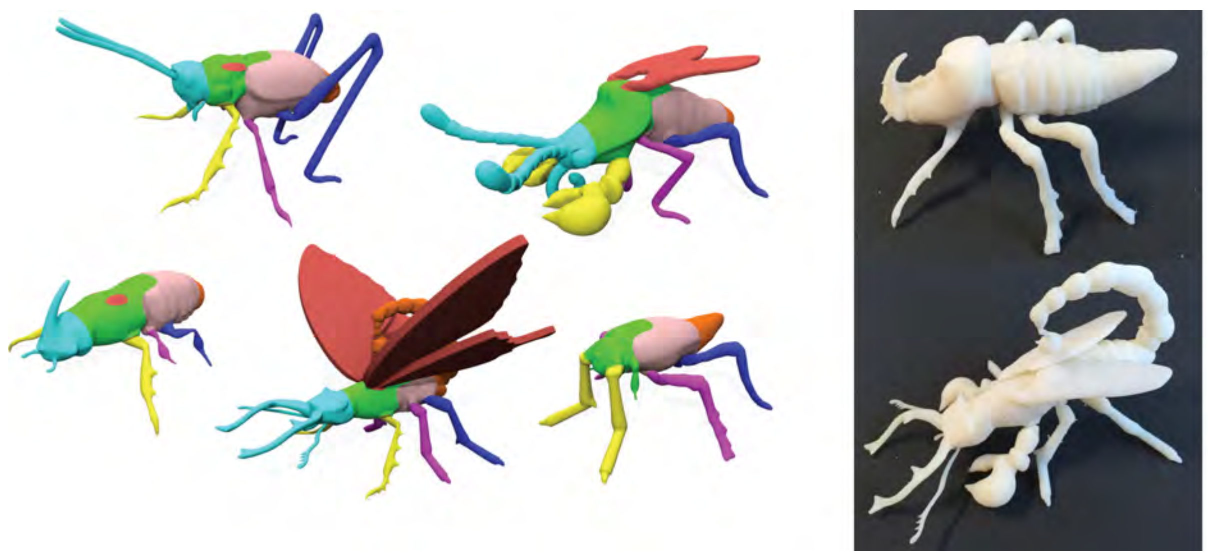


Figure 6.9: Some assembled insects.

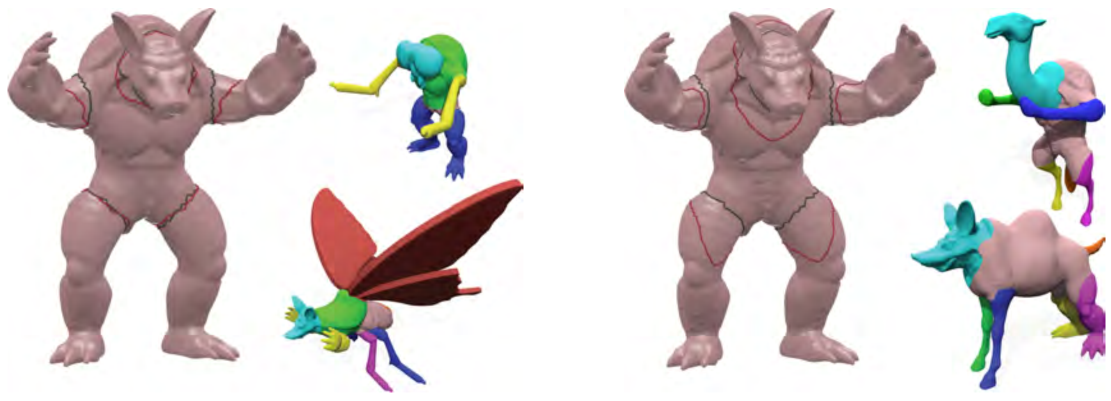
(c.3). Without making this cut, the deformation for compatibility unnaturally raises the grasshopper's abdomen (c.4).

Note that we have only shown the process for a single model. Our approach jointly considers these factors for six models. Conducting this process by hand would be a tedious and difficult task. Figure 6.9 shows some assembled insects and their prints.

6.2.2 Cross-Category Components

We demonstrate that our approach is flexible enough to generate interchangeable components that produce coherent shapes from input models belonging to different categories. We combined the Armadillo model from the humanoids collection with three insects, and three four-legged animals. The Armadillo's original semantic segmentation from the humanoid example is directly compatible with the insects. To make it compatible with the animals, we merged the chest and abdomen into a single semantic region. No other changes to the segmentation were made.

Figure 6.10 shows how our contour optimization scheme (Section 4.2) generates significantly different edge loops depending on which category we target. For example, because the



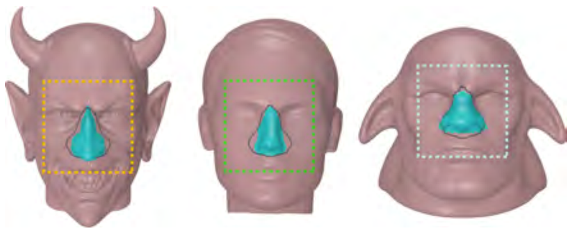
(a) With Insects (b) With Animals

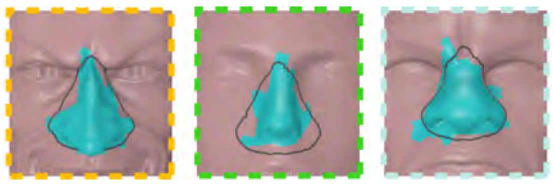
Figure 6.10: The change in edge loops made by our approach for maximizing the armadillo's compatibility with (a) insects and (b) four-legged animals. The original edge loops are shown in gray and the new edge loops in red. Some shapes constructed from the resulting components are shown on the right. The components can form both upright and crawling shapes.

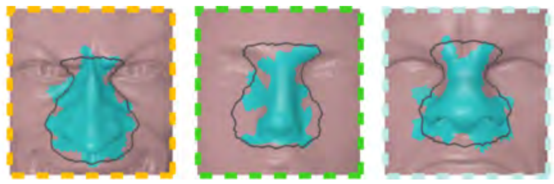
insects tend to have relatively narrow, circular shaped necks, the edge loop for the armadillo is tightly closed around its neck. For the animals, which have larger, oval-shaped chests, the armadillo's edge loop cuts into its chest. The resulting components are versatile, capable of generating creatures which walk upright or on four or six legs, as shown in the constructed shapes in Figure 6.10.

6.2.3 Sensitivity to Initial Segmentation

We examine our the dependence of our approach on the input semantic segmentation. For three faces, we run our contour optimization (Section 4.2) on the border between the nose and face. We apply three different levels of perturbation to the initial segmentation. Figure 6.11 visualizes the results. The resulting contours are all semantically valid unless the perturbation is severe, which causes them to cross into the eyes.


(a) Original Segmentation


(b) Small Perturbation


(c) Medium Perturbation

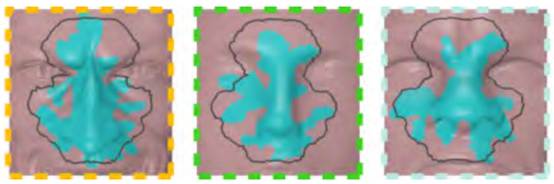

(d) Large Perturbation

Figure 6.11: The effect of perturbing the input segmentations on the contours found by our optimization. Only upon large perturbation to the inputs do the output contours become non-viable.

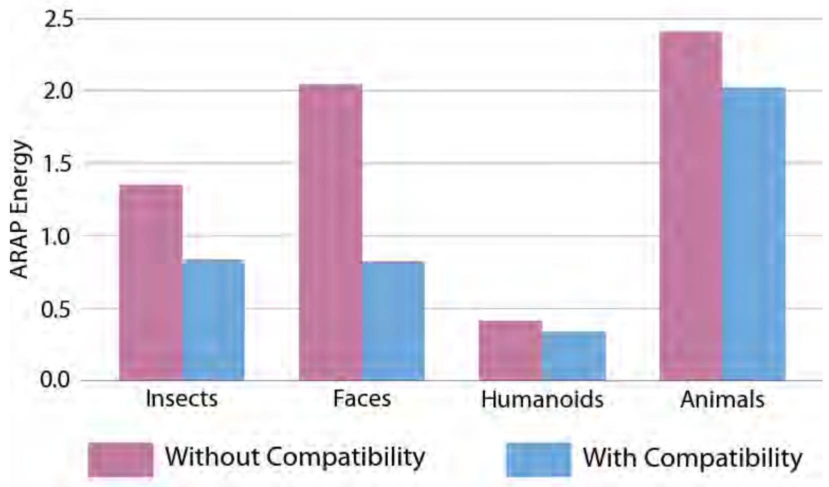


Figure 6.12: The improvement in shape preservation from compatibility-based optimization.

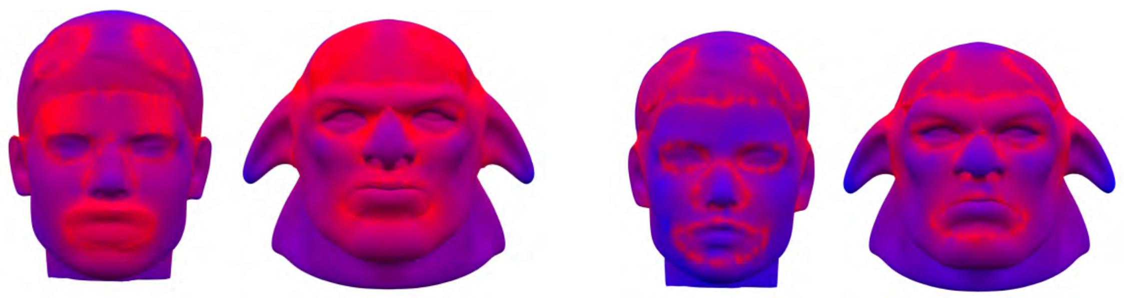

(a) Without Compatibility (b) With Compatibility

Figure 6.13: Visualizing the distortion caused by satisfying component interchangeability. Red indicates distortion level. Compatibility optimization reduces the distortion considerably.

6.2.4 Compatibility Optimization and Mesh Deformation

We quantitatively evaluated the improvement of the component's quality resulting from the compatibility-based optimization (Section 4.2). We ran our full approach and measured the extent of the deformation for compatibility (Section 4.3) using the ARAP mesh energy. Next, we re-ran the approach with the compatibility-based optimization omitted and compared the energies. See Figure 6.12 for the results. The extent of our improvement ranges from over 100% for the faces to about 17% for the humanoids. Figure 6.13 visualizes how the distortion is distributed over the mesh.

6.2.5 Performance

We tested a single-threaded implementation of our approach on a 2.4 GHz laptop. The optimization for compatible edge loops in Section 4.2 is the longest step. It took less than 4 minutes in all our experiments. The constrained optimization problems for finding the initial common edge loops in Section 4.3.1 and for finding an RBF surface in Section 4.4 can be solved in a few seconds using standard solvers. The time required to minimize the total mesh deformation energy in Section 4.3.2 scales linearly with the total number of vertices in the input set. In all our experiments it took less than 30 seconds to converge. In total, the approach took about 8 minutes for the 13 input model animals and 6 minutes for the 5

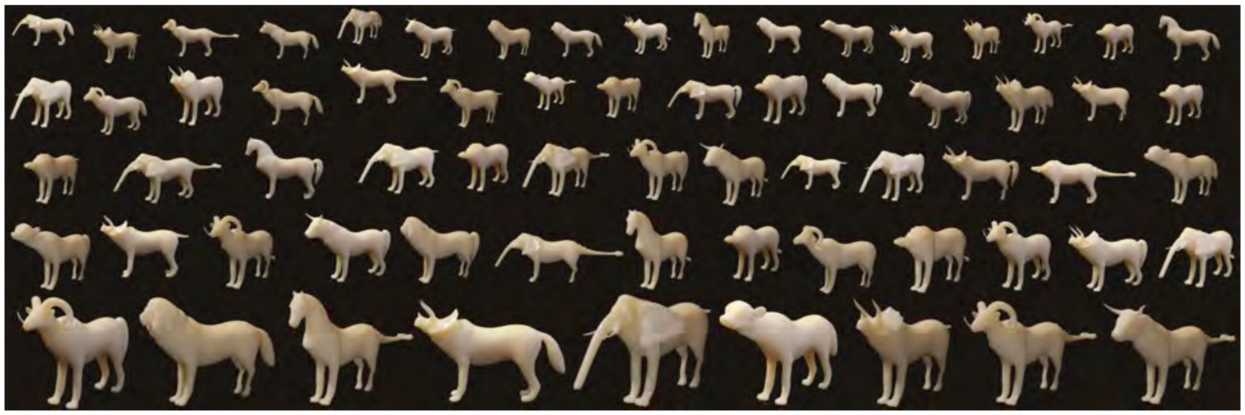


Figure 6.14: Animals assembled by our interchangeable components. Note that each assembled animal is different and even more variations are possible.

input model faces.

Appendix B provides detailed performance data.

6.2.6 Fabrication

We fabricated our Interchangeable Components results on an Objet Connex 3D printer in solid material. The printer possessed sufficient precision to enable the components to fit together smoothly, making most seams virtually unnoticeable, although components with highly curved connections will have more visible seams. We expect that fabricating the parts with a higher precision technique, such as molding, should make the seams less visible. Some parts of the fabricated components were not covered by support material, which gave them a shinier appearance than parts that were covered. This effect, which is visible in Figure 6.14, may interfere with the perception of seamlessness. Components that lack any graspable protrusion and lie in concave regions like the mouth and eyes are easier to extract with a small flat-head screwdriver than by hand. Components involved in higher-order constraints, like the arms in the armchairs, require a greater degree of precision from the printer for a perfectly seamless connection. Since the printer lacks this precision, these components often have more visible seams, though they still connect.

	Pieces	**J. Constraints**	**Runtime**
Dog-Bone	6	8	438 min
Cat-Fish	6	8	410 min
Dove-Bomb	6	8	368 min
Bunny-Egg	6	9	583 min
Serpent-Apple	5	8	96 min

Table 6.1: Performance on several inputs. For each test, we list the number of pieces used, the number of connection constraints, and the total runtime.

6.3 Approximate Dissections

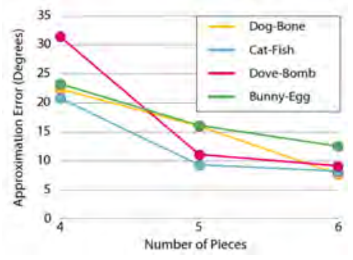
6.3.1 Performance

We implemented our approximate dissections method in C++ and tested it on a 2.6 GHz laptop. Due to the exponential nature of the problem, our solution search takes a substantial amount of time. However, the search procedure described in Section 5.4.3 is embarrassingly parallelizable, and we would expect our runtimes to decrease substantially with the addition of more processors.

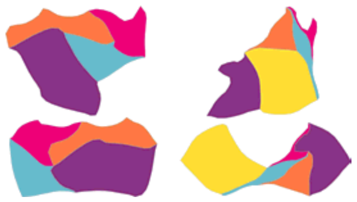
Table 6.1 summarizes the performance of our method. We note that it is not unusual to see similar performance characteristics when solving geometric problems in a vast combinatorial search space. For example, recent work by Kwan et al. (2016) that tackled the 2D collage problem had comparable timing, taking about 12 hours to generate their most complex result.

6.3.2 Approximation Accuracy

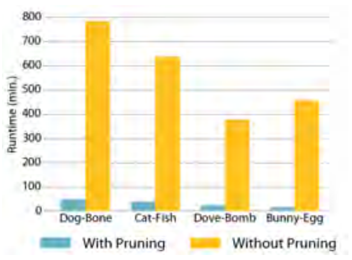
We tested how increasing the piece count improves the accuracy of approximating the input shapes. Figure 6.15(a) shows the results across four dissections. We used the maximum angular error from our optimization (5.1) to measure the approximation accuracy. For the three piece counts tested, we observed a roughly linear relationship.



(a) Approximation Error vs. Piece Count



(b) Dog-bone dissection with four and five pieces



(c) Effect of Pruning For 5-Piece Dissections

Figure 6.15: (a) Decline in approximation error as the number of pieces used is increased. (b) The best solutions for the dog-bone dissection obtained with four and five pieces. (c) Performance gains obtained from the pruning tests.

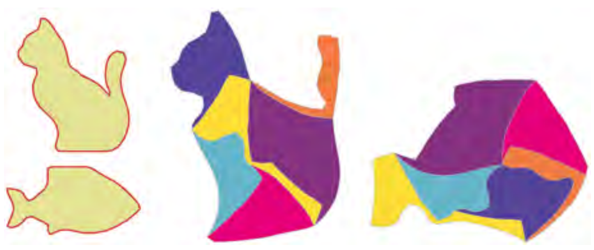
6.3.3 Pruning Efficiency Gain

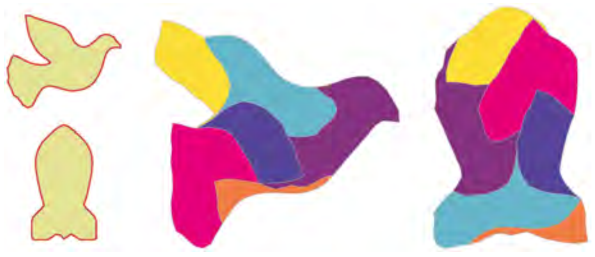
We measured the extent to which the pruning tests (Section 5.4) improve the performance of the solution search. Figure 6.15(b) compares the runtimes for pruning versus no pruning, for four dissections, using a piece count of five. At this piece count, the pruning improves the performance by a factor of roughly 15. For six pieces, all the trials run without pruning did not finish after 24 hours.

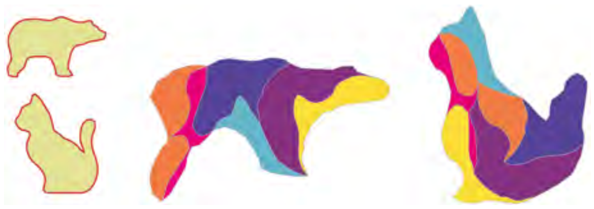
6.3.4 Results

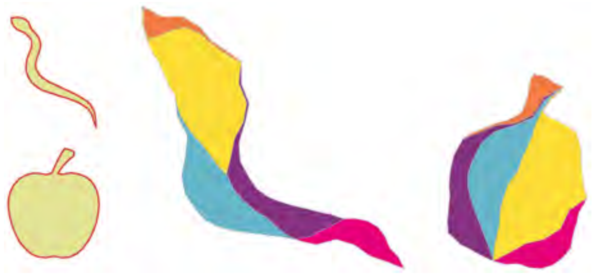
We used our approach to generate several dissections, which are shown in Figure 6.16. All the input shapes were obtained by taking one of the first results from a Google image search. Appendix C provides an alternate visualization of these results.

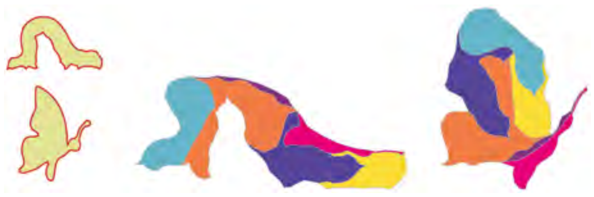
Our method supports organic objects (Figure 6.16(a)–(e)), man-made objects (Figure 6.16(b), (g)), the outlines of geographic entities (Figure 6.16(f), (h)), and abstract shapes (Figure 6.16(f), (g)). In most cases, six pieces were needed to form a satisfactory approximation of the input shapes. The two exceptions were the serpent-apple and Trump-USA dissections, which used only five pieces. The Trump-USA result (Figure 6.16(h)) shows how simple texturing can enhance a dissection's appearance.

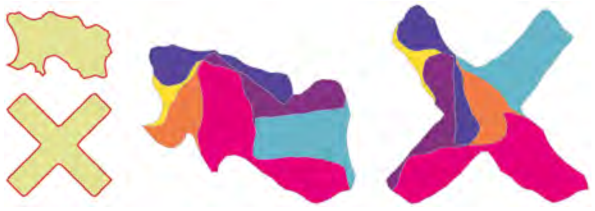

(a) Cat to Fish

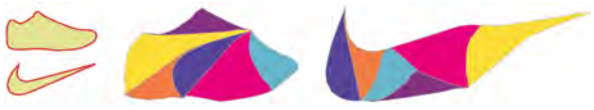

(b) Dove to Bomb


(c) Bear to Cat


(d) Serpent to Apple


(e) Caterpillar to Butterfly


(f) Map of Island to X


(g) Nike Logo to Shoe

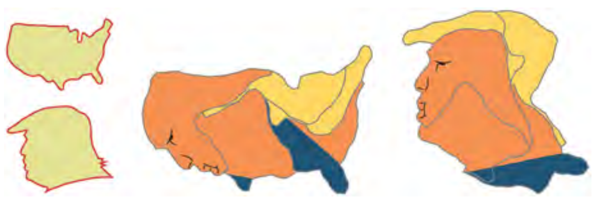
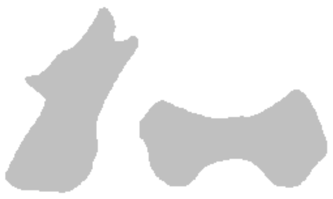
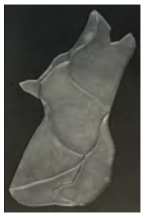
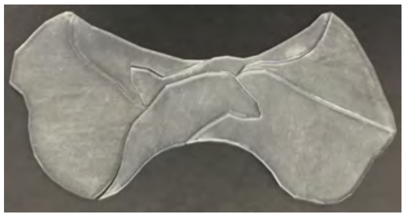

(h) United States to Trump

Figure 6.16: Generating dissections between different shapes. Input shapes are shown on the left. Assembled pieces are shown on the right.


(a) Arrangement Outlines


(b) Dog Arrangement


(c) Bone Arrangement

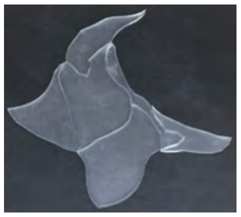

(d) Incorrect Arrangement

Figure 6.17: The fabricated dog bone puzzle. (a) Outlines of the arranged pieces that were shown to the participant.

6.3.5 Application to Puzzles

Our Approximate Dissections technique can be applied to create a type of puzzle as shown in Figure 6.17. We fabricated several of our results and conducted an informal user study in which participants were shown outlines of the two shapes and instructed to assemble the pieces into those shapes. We observed that the puzzles are substantially, albeit not overwhelmingly, difficult. On average, it took a participant about twenty minutes to solve the puzzles for both shapes. The pieces can form coherent shapes that are not one of the intended ones (Figure 6.17(d)), which increases the difficulty of the puzzle.

We can introduce a post-processing step that partitions the original set of dissection pieces into smaller ones, allowing the user to tune the difficulty of the puzzle. The partition could aim for high similarity between the pieces, as in traditional jigsaw puzzles.

CHAPTER 7

Conclusion

7.1 Summary

This dissertation introduced three novel geometric open-ended design problems and proposed viable solutions to them. Open-ended design problems are challenging because it is unclear how one should formalize them. On the one hand, opting for a highly mathematically satisfying formalization may result in an elegant solution, but the formalization may not sufficiently capture the nuances of the original problem. On the other hand, a formalization that tries to capture every subtlety of the original problem might result in an unwieldy, ad hoc solution that only works for a narrow range of inputs. In this thesis we sought to avoid these pitfalls by developing a concise formalization to tackle some "core aspect" of the problem, and then introducing extensions to the formalization to address the subtleties. In many cases these extensions provided user input through a graphical interface, thus tackling situations that are difficult for a computer but trivial for a human.

First, we introduced the problem of zoomorphic object creation to computer graphics and offered a novel approach to solve it. This problem has the unique challenge of combining very different objects while preserving key properties and structures. Our approach creates a zoomorphic object by altering and then merging a man-made and animal object. We developed a novel technique to ensure that the design restrictions of the man-made object are still satisfied in the zoomorphic object. We incorporated this technique into an optimization process that jointly deforms the two objects to improve the appearance of the resulting zoomorphic object, according to high level preferences given by the user.

Second, we presented an Interchangeable Components approach to convert 3D models into interchangeable components that form shapes with a coherent appearance. Our algorithm chooses how to partition the models into components and how to deform the components for interchangeability. Both steps consider C^0 and C^1 continuity between the components in order to minimize the visual impact of component junctions. This process would be extremely tedious to perform manually for complex shapes or large numbers of shapes. Our optimization-based approach generates components which produce shapes of greater complexity and diversity than those of commercial products while naturally incorporating user guidance to preserve desired features in the components. The increasing availability of high quality 3D models and cheap 3D printing services has motivated a recent trend in Computer Graphics Research, which focuses on allowing casual users to create customized fabricable objects that possess a desirable property from initial meshes that lacked this property. Some example properties include stability (Prévost et al., 2013), spinnability (Bächer et al., 2014), and aerodynamic characteristics (Umetani et al., 2014). Our work contributes to this trend, with the specific property being the interchangeability of the components.

Third, we introduced the approximate dissection problem, a relaxation of the exact dissection problem, which allows us to capture the essence of a pair of complex shapes using a small number of pieces. We developed a combinatorial search strategy for the problem that prunes the solution space to identify high-quality dissection solutions in a tractable amount of time. Since our geometric approach cannot take into account the perceptual significance of a given deformation, we developed a novel user interface that allows a casual user to refine the dissection solution. Our interface suggests alterations to the input shapes that would make the dissection generation easier, while allowing the user to add additional alterations that preserve the identity of the input shape. Our system enables the creation of dissections that are qualitatively different than previous ones. Dissections between complex, naturalistic shapes possess a visceral quality that is lacking in dissections between abstracted or geometrically ideal shapes. These dissections can be used to convey a message in a unique fashion (Figure 7.1) or as a puzzle.

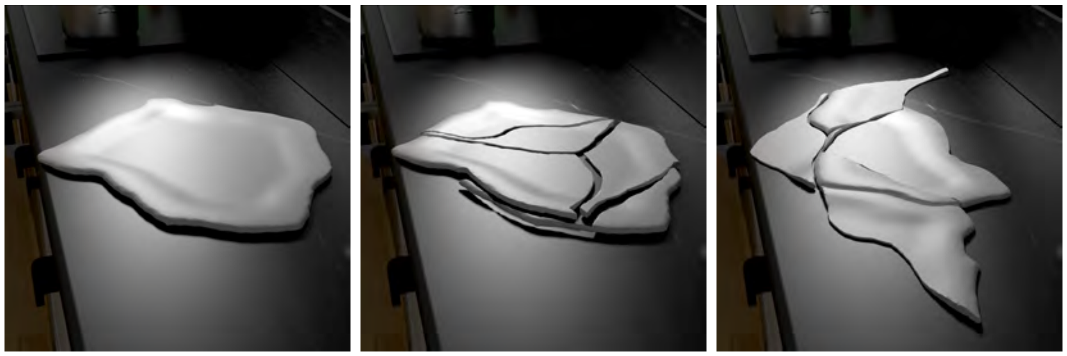

(a) (b) (c)

Figure 7.1: The application of our method to a horror movie scene. (a) A plate on a countertop. (b) The plate begins to vibrate and splits into pieces. (c) The pieces rearrange themselves to form a devilish visage.

7.2 Limitations and Future Work

In our Zoomorphic Design approach, our volumetric design restriction cannot preserve the functionality of the base object in the final zoomorphic object in all cases. We do not guarantee that the zoomorphic object will be aerodynamic, physically stable, or capable of bearing a given weight. An interesting direction for future work is to integrate our optimization framework with other approaches that handle these more complex types of functionality (see (Prévost et al., 2013) and (Umetani et al., 2014)).

Our approach considers the animal object as a coherent entity, which constrains its parts to locations that will not induce a large global deformation. However, some real world designers relax this constraint, which allows them to create viable zoomorphic objects that our approach cannot. For example, the designer of the scorpion chair in Figure 7.2(a) chose to place the scorpion arms above and behind the scorpion head such that they overlap with the chair arms. The designer effectively disconnected the arms from the rest of the object, prioritizing semantic correspondence over physical validity. A real scorpion cannot deform in this manner, yet the Scorpion Chair is visually provocative. A scorpion chair generated using our approach cannot place the arms in this manner, as it would incur an extremely high deformation energy. Instead we place the scorpion arms at the sides of the chair

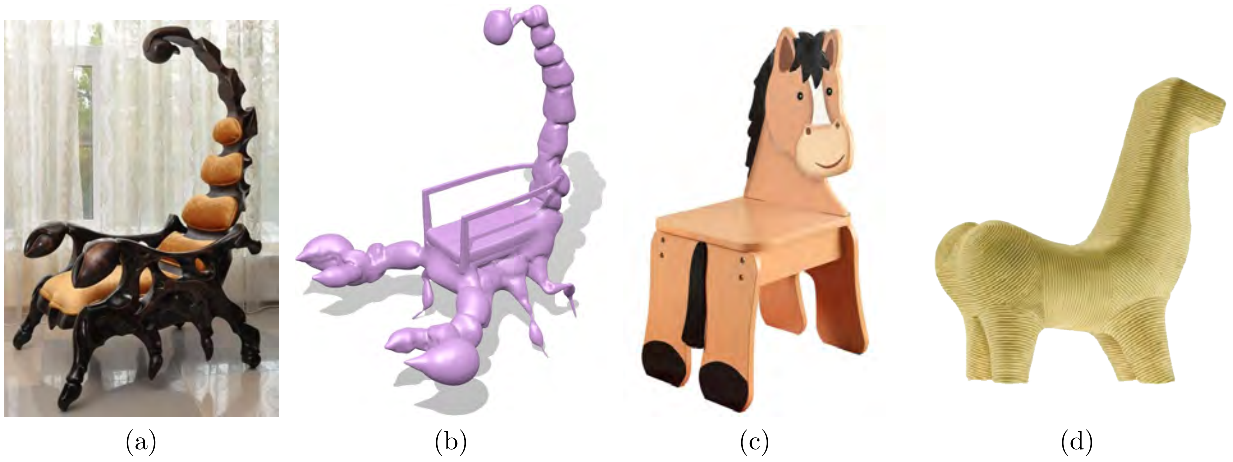


Figure 7.2: Limitations. Our system can only place the scorpion arms at the sides (a) while an artist places them to coincide with the chair arms (a). (c)–(d) Zoomorphic objects that could not be created by our system, suggesting alternative models based on texture synthesis guided by geometry (c) or shape abstraction (d).

(Figure 7.2(b)), which may be undesirable since the arms take up much space. We believe our approach could be extended naturally to handle this type of deformation by allowing parts of the animal object to disconnect themselves should the deformation energy become too large.

There are other styles of zoomorphic art that our approach is not designed to handle. Figures 7.2(c)–(d) show two examples. The development of approaches to handling these styles offers interesting directions for future work in zoomorphic design.

We see several directions for future work related to our Interchangeable Components approach. The components generated by the current approach correspond to semantically meaningful regions of the input models. While the semantic constraint makes shape assembly more intuitive, it also limits the geometric diversity of the components. An interesting problem would be to synthesize a set of components from scratch that can approximate the geometry of a set of models, without any semantic considerations on the components. The resulting components would form a kind of 3D tangram puzzle. The present approach dealt with objects which are reconfigurable between a set of states specified by *appearance*. Specifying the states by a higher level goal such as functionality or a physical property while

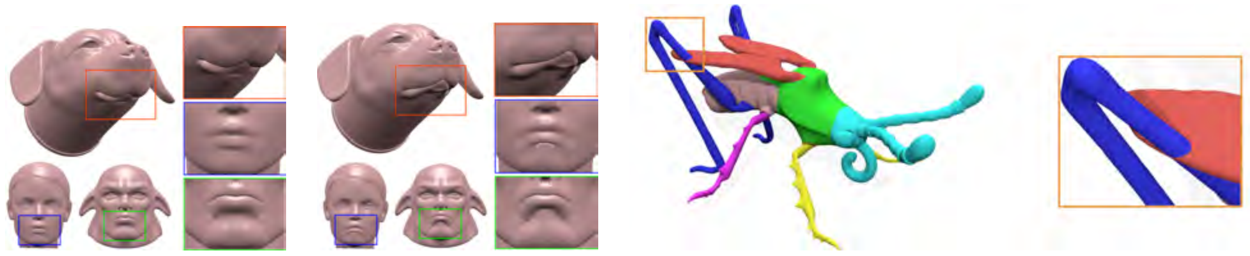

(a) (b) (c)

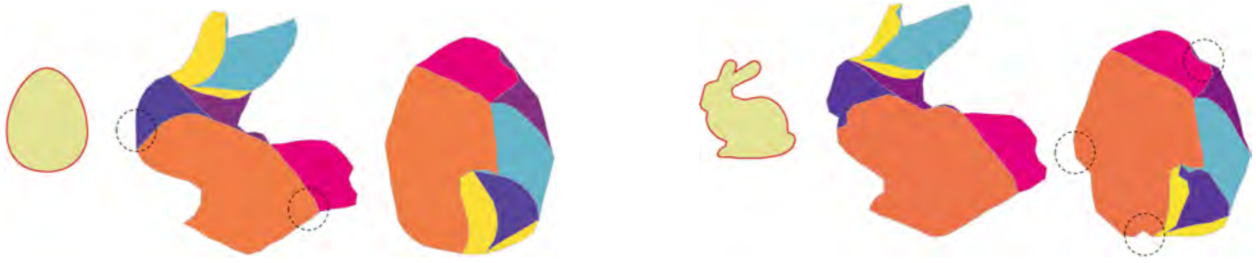
Figure 7.3: (a-b) A failure case due to severe differences in the initial component geometry. (a) The default deformation produced by our approach distorts the dog's mouth. (b) Adjusting the weights to preserve the dog's mouth creates undesirable deformations in the other shapes. (c) Interference between components prevents physical assembly.

relaxing constraints on the appearance is an intriguing problem for future work.

Our approach deforms the input meshes to achieve part compatibility, but does not guarantee anything about the extent of the deformation. In some cases the deformations introduced may be semantically incorrect. The user interaction discussed in Section 4.5.3 can mitigate these issues, but may be unable to resolve cases where the components of the input meshes differ fundamentally in their geometry. Figure 7.3(a-b) shows such a case. In a few cases the diversity of our component geometry creates configurations where components interfere with one another, as shown in Figure 7.3(c). We leave the problem of automatically deforming the components to eliminate these cases for future work. The relative alignment between components when they are connected is determined by the local coordinate frame for the contours, whose computation is described in Section 4.2. Since the automatic local frame computation may not always produce a satisfactory component alignment, future work could allow the user to adjust the frame if desired.

When dealing with man-made shapes, our approach could benefit from using parametrized shape templates (Schulz et al., 2014) instead of simple triangle meshes. To incorporate shape templates, we would replace the mesh based deformation energy in Section 4.3.2 with one that incorporated template parameters.

Our approach only considers geometric, not physical properties of the assembled objects. For



(a) Design with egg prioritized (b) Design with bunny prioritized

Figure 7.4: A failure. The input shapes are shown in gold. Even with user interaction, problematic regions (circled) in both shapes cannot be eliminated in a single design.

example, it is possible to construct a humanoid (Section 6.2.1) that does not stand stably on its legs. Combining our geometric problem with the physical problems posed by works like (Prévost et al., 2013) offers an interesting direction for future work: guaranteeing that *any* object assembled from a set of components possesses some physical properties.

Our Approximate Dissections approach offers several opportunities for future work. First, we could generalize it to support dissections between more than two shapes. Second, the search time for the solution is substantial, even with the pruning tests. Since we currently use a commodity solver (Wächter and Biegler, 2006), we can potentially boost performance by developing a special procedure for the dissection geometry optimization (Section 5.2). Developing additional ways of pruning partial solutions could provide even bigger gains. Our approach does not naturally generalize to 3D, because we represent dissection pieces as a ring of one-dimensional boundary intervals and do not take piece interlocking into account. Hence, a substantially different method would be needed to tackle the approximate dissection problem in three dimensions.

In some cases our approach fails to generate a satisfactory dissection. Figure 7.4 shows an example. Even with manual editing, the constraints of the design preclude capturing the details of the Stanford bunny without causing unsightly bulges and cavities on the egg. In such cases, the trade-off between the two options is left to the user.

Our method does not consider the physical properties of the pieces, so it can generate pieces with extremely narrow sections that would be weak if fabricated. In our current implementation, the user may manually edit the solution to remove these sections. A term penalizing such sections may be added to the optimization (5.1). We also ignore the semantic significance of the pieces. For example, in the cat-fish dissection (Figure 6.16(a)), one of the pieces corresponds almost exactly to the cat's head. These cases may be undesirable when designing a puzzle as they reduce its difficulty. Also, allowing pieces to flip between arrangements, in addition to rotating, would enable us to generate a wider range of solutions.

Finally, this thesis discussed open-ended problems whose solution is ultimately judged by human perception. In our geometrically-formulated solutions to these problems, geometry was used as a rough approximation of human perception. To deal with this gap we incorporated user guidance into our approaches, allowing users to tweak the solution in places their perception flagged as incorrect. However, recent advances in machine learning have built rich models of the relationship between perception and geometry. For a notable example in the graphics community, see Yumer et al. (2015). Future work could incorporate such models, leading to semantically-formulated solutions to open-ended design problems.

APPENDIX A

Supplemental Material on Zoomorphic Design

This appendix provides more details about the shape graph construction, the graph kernel evaluation, the automatic transfer of VDR labels, and the informal studies.

A.1 Suggesting Objects to Blend

A.1.1 Shape Graph

We describe how to construct the shape graph for an input object which has been pre-segmented. Figure A.1 shows a chair separated into its segments. Each segment corresponds to a node of the shape graph. Each adjacency between segments corresponds to an edge of the shape graph. For each node of the shape graph we record the following attributes:

1. $\gamma_1, \gamma_2, \gamma_3$ - The three dimensional attributes of the segment.

2. μ_1, μ_2, μ_3 - The relative segment scales used in (Xu et al., 2010a), which measure how planar, linear and spherical a segment is.

3. $y_{\min}, y_{\max}$ - The minimum and maximum y-coordinate of all the vertices in the segment.

4. κ - The average centricity of the segment. Centricity is defined as the average geodesic distance from a vertex to all the other vertices in the mesh. The geodesic distances are normalized between 0 and 1.

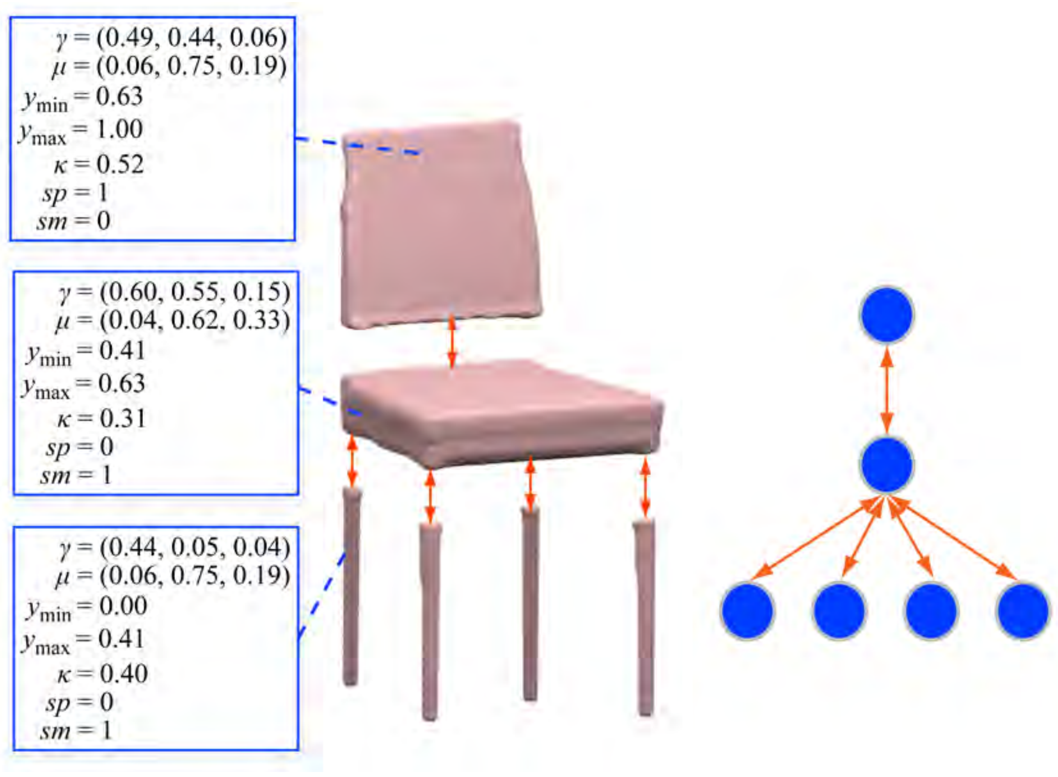


Figure A.1: Shape graph representation of a chair. Each segment is represented as a node. Blue boxes show the attributes stored in each node. Orange arrows refer to the edges connecting each pair of adjacent nodes.

5. *sp* - A binary attribute denoting whether the segment is a supporting segment. We classify a segment as supporting if $y_{\min}$ is within 10% of the minimum y-coordinate across the entire mesh. This attribute assumes that the mesh is oriented upright, which can be automatically inferred by Fu et al. (2008).

6. *sm* - A binary attribute denoting whether the segment is positioned in the center of the object's bilateral symmetry plane or if it has a symmetric counterpart on the other side of the plane.

A.1.2 Graph Kernel

We use a graph kernel to measure the similarity between two shape graphs. Graph kernels are a general tool for measuring the similarity between two graphs. An example use is measuring the similarity between scenes (Fisher et al., 2011). We use a graph kernel which compares two shape graphs by evaluating the similarity of all pairs of length p walks on the graphs ($p = 3$ in our experiments). The similarity between two walks is evaluated by using a node kernel between corresponding nodes and an edge kernel between corresponding edges of the walks. The evaluation can be done efficiently through a dynamic programming algorithm (Shawe-Taylor and Cristianini, 2004).

Node Kernel. For any node u, let u_a denote the value of its attribute a. Our node kernel $\mathcal{K}_{\text{node}}$ measures the similarity between the input nodes u and v:

$$\begin{aligned}\mathcal{K}_{\text{node}}(u,v) &= \delta(u_{sp},v_{sp})\delta(u_{sm},v_{sm}) \\ &\times\frac{1}{2}\left(\mathcal{K}_{\text{centricity}}(u,v)+\mathcal{K}_{\text{scale}}(u,v)\right),\end{aligned} \tag{A.1}$$

$$\mathcal{K}_{\text{centricity}}(u,v) = \exp\left(-\frac{(u_\kappa - v_\kappa)^2}{{\sigma_\kappa}^2}\right), \tag{A.2}$$

$$\begin{aligned}\mathcal{K}_{\text{scale}}(u,v) &= \prod_{i=1}^{3}\exp\left(-\frac{(u_{\gamma_i}-v_{\gamma_i})^2}{{\sigma_\gamma}^2}\right) \\ &\times\prod_{i=1}^{3}\exp\left(-\frac{(u_{\mu_i}-v_{\mu_i})^2}{{\sigma_{\mu_i}}^2}\right),\end{aligned} \tag{A.3}$$

where δ is the Kronecker delta function, σ_κ is the largest centricity difference, σ_γ is the largest scale difference between all the segments of the two objects and σ_{μ_i} is the largest difference in μ_i.

Edge Kernel. We measure the horizontalness of an edge e connecting two nodes u and v (corresponding to two adjacent segments), by the following formula:

$$h(u,v) = \frac{\max(0,\min(u_{y_{\max}},v_{y_{\max}})-\max(u_{y_{\min}},v_{y_{\min}}))}{0.5\cdot((u_{y_{\max}}-u_{y_{\min}})+(v_{y_{\max}}-v_{y_{\min}}))}. \tag{A.4}$$

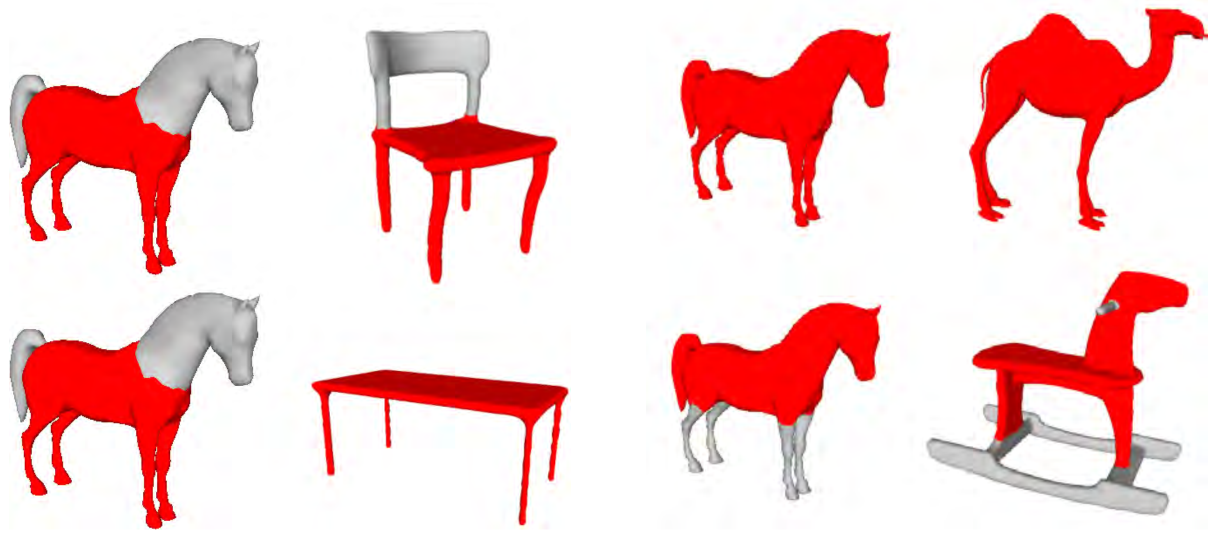


Figure A.2: Common substructures (red) between the two objects found by the graph kernel.

For simplicity, denote $e_h = h(u, v)$. Our edge kernel $\mathcal{K}_{\text{edge}}$ measures the difference in horizontalness between the input edges e and f:

$$\mathcal{K}_{\text{edge}}(e, f) = \exp\left(-\frac{(e_h - f_h)^2}{{\sigma_h}^2}\right), \tag{A.5}$$

where σ_h is the largest difference in horizontalness between all the edges of the two objects.

A.1.3 Graph Kernel Evaluation

We can visualize the operation of our graph kernel to confirm that it gives reasonable results. The length p graph walk kernel will compute a score for every pair of length p walks in the two graphs being compared. For each node in the graphs we record the maximum walk score across all pairs of walks that involved that node. In Figure A.2 we color the segments of the objects by this score. The red areas of a shape indicate substructures that have corresponding substructures in the other shape. The case where we compare a horse and a rocking-horse is a partial failure, because we fail to detect similarity between the horse's legs and the central supports of the rocking-horse. This failure is because the horse's legs have symmetrical counter-parts, but the central supports do not.

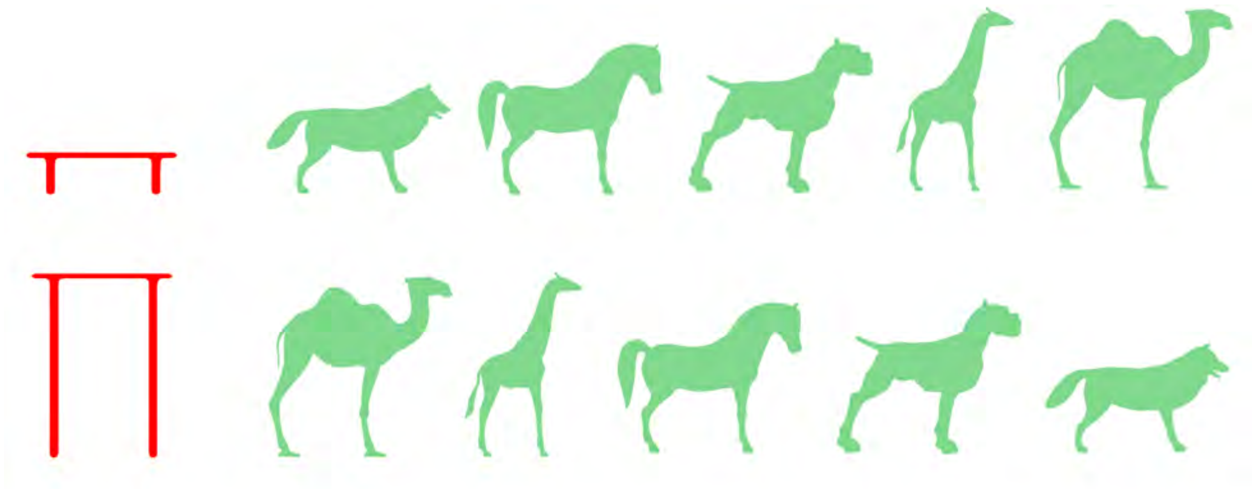


Figure A.3: Query base objects (tables) and the top five returned animal objects.

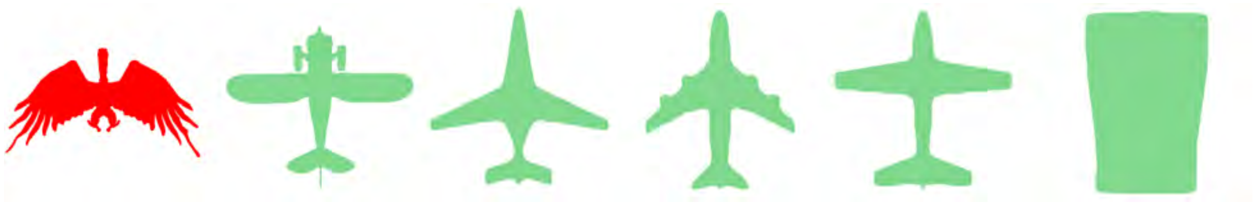


Figure A.4: Query animal shape (a phoenix) and the top five returned base objects.

In Figure A.3, we test the graph kernel's ability to discriminate between geometric characteristics at the part-level by querying on two tables which are identical except for the length of their legs. The top five results are all quadrupeds which possess similar structure to the table. However, the order of these results depends on their legs' geometric similarity to those of the table.

Figure A.4 shows an example of using a phoenix as the query animal shape and the returned base object results. Because there were only four airplanes present in the database, the fifth result is a cup which matches the body of the phoenix.

In Figure A.5, we use an armchair as the query base shape and show the returned animal objects. Four-legged animals come first, followed by the armadillo which has two arms and two legs, and then by the eight-legged octopus. The hummingbird which has no legs and the ant whose legs extend more horizontally then downwards receive the lowest rankings.

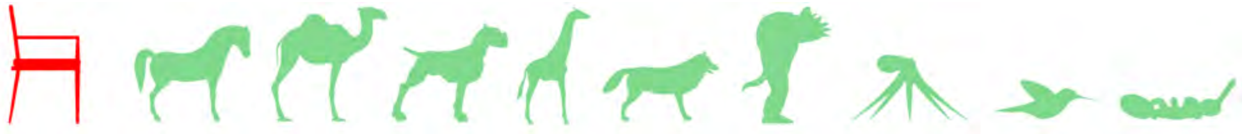


Figure A.5: Query base object (an armchair) and the suggested animal objects for merging. Objects are sorted from left to right in descending order of graph kernel match.

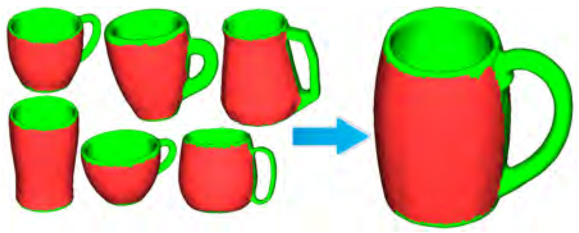


Figure A.6: Left: Manually labeled training meshes (labels in green). Right: Automatically labeled mesh. The geometry of the automatically labeled mesh is significantly different from that of the training set.

A.2 Automatic Transfer of VDR Labels

An advantage of the VDR is that a classifier can be trained to transfer VDR labels automatically for meshes in a given category. We conducted some simple experiments to verify that the approach of Kalogerakis et al. (2012) can be applied transfer them successfully.

We test the labeling method of Kalogerakis et al. (2012) on our problem by comparing the automatic VDR labeling to the ground-truth labeling across four different object categories and four different training data set sizes. In Figure A.6 we show manually labeled cups used as training examples and an automatically labeled cup.

Table A.1 shows the relationship between the size of the training data set and the labeling

	Training Set Size			
	3	6	12	19
Chairs	72.13%	89.07%	94.42%	94.84%
Cups	79.83%	89.93%	89.78%	90.70%
Airplanes	81.34%	80.14%	96.78%	98.634%
Tables	90.78%	88.31%	92.49%	96.05%

Table A.1: Training set size and the corresponding automatic VDR labeling accuracies.

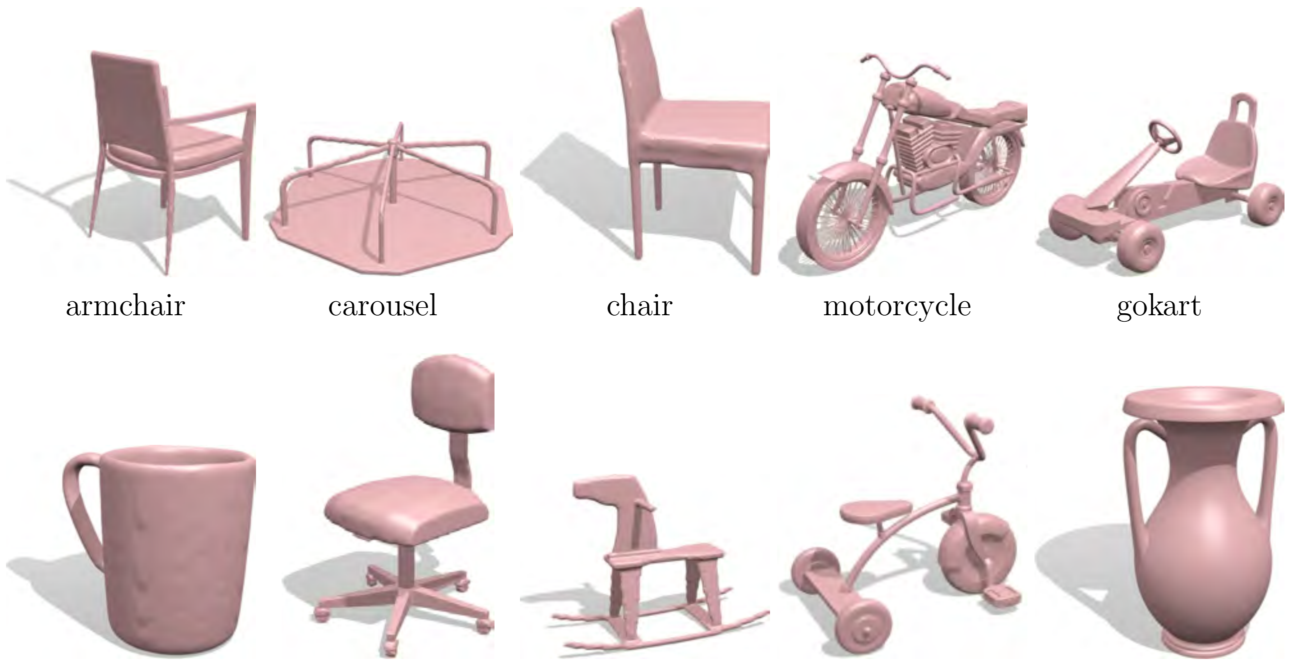


Figure A.7: Base objects used in our informal studies.

accuracy computed using the Segment-Weighted Error used by Kalogerakis et al. (2012). Their method gives reasonable results when applied to our problem. We find that in general a training data set of about 12-19 meshes is enough to obtain a classification accuracy of about 90% or above.

A.3 Informal Studies Details

Figure A.7 and A.8 show the base and animal objects used to generate the zoomorphic objects in our informal studies. Figure A.7 and A.9 show the objects used in the first test. Figure A.9 and A.10 show the objects used in the second test. Note that in our actual studies, all the 3D models were rendered in the same color, so that our test participants could not determine the categories of the objects from their colors. Tables A.2 and A.3 summarize the responses.

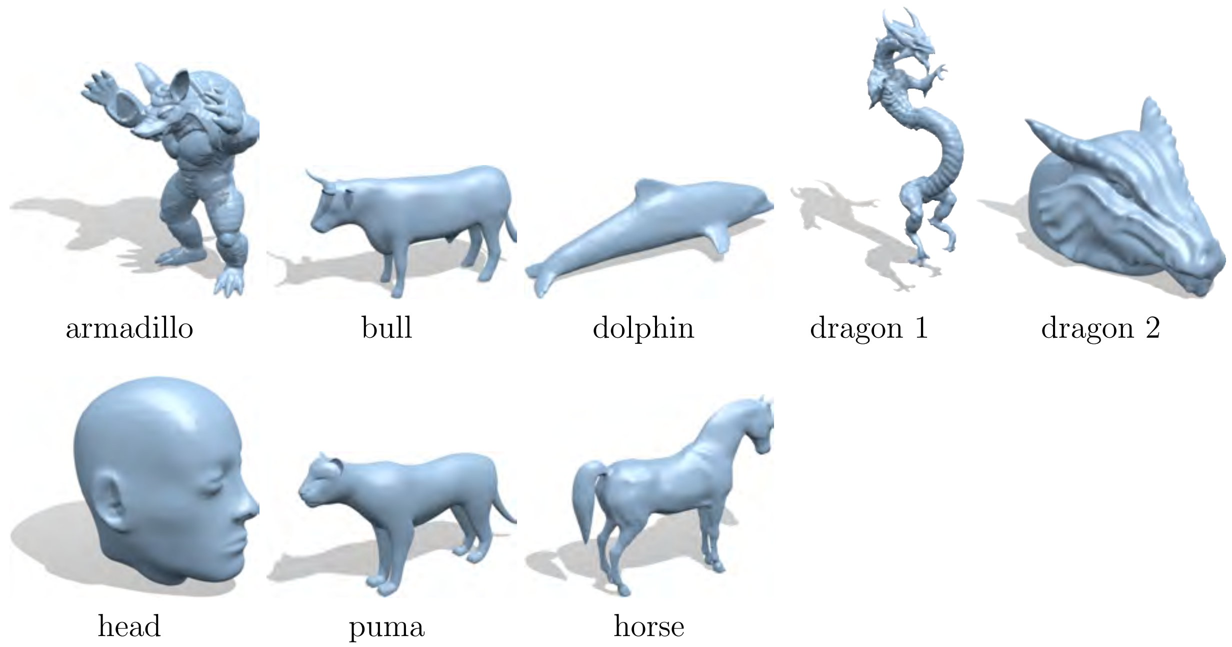


Figure A.8: Animal objects used to generate the zoomorphic objects in our informal studies.

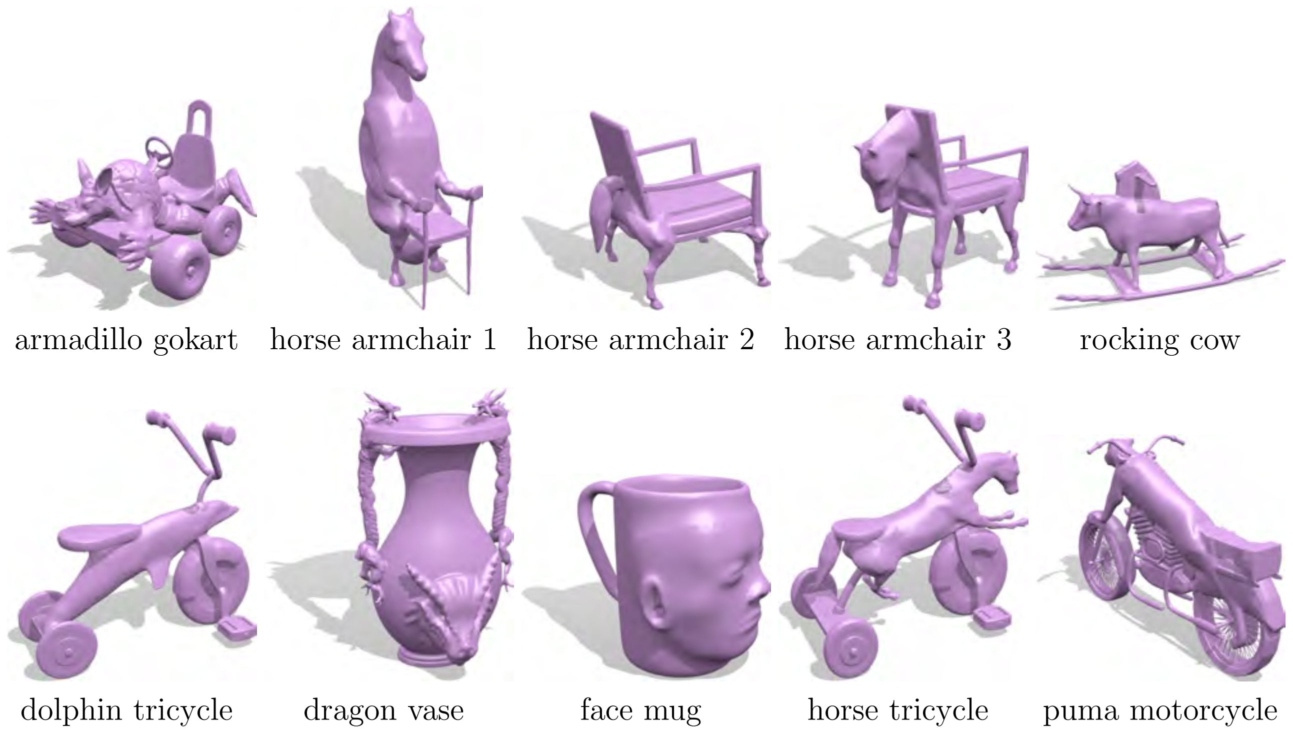


Figure A.9: Zoomorphic objects used in our informal studies, generated with the volumetric design constraint.

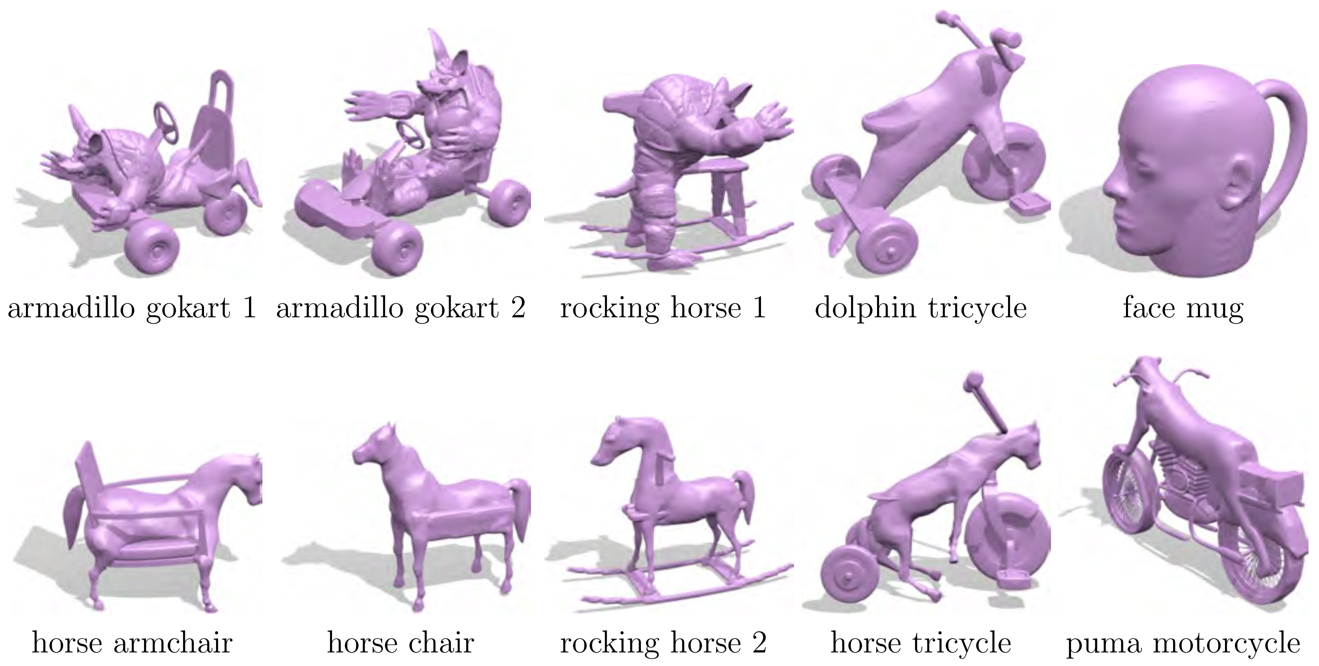


Figure A.10: Zoomorphic objects used in our informal studies, generated without the volumetric design constraint.

	Yes	No	Undecided
Our Result	89.55%	9.09%	1.36%
Base Shape	11.36%	86.14%	2.50%

Table A.2: Responses for the first test. The participant was asked "Is this object zoomorphic?".

	Yes	No	Undecided
With VDC	85.85%	12.20%	1.95%
Without VDC	36.59%	59.02%	4.39%

Table A.3: Responses for the second test. The participant was asked "Is this a plausible *category*?", where *category* refers to the object category of the base object from which the zoomorphic object shown was created.

APPENDIX B

Supplemental Material on Interchangeable Components

This appendix provides details on the computational performance of our Interchangeable Components approach and visualizes the distortions incurred in enforcing interchangeability between components.

B.1 Performance and Combinations Data

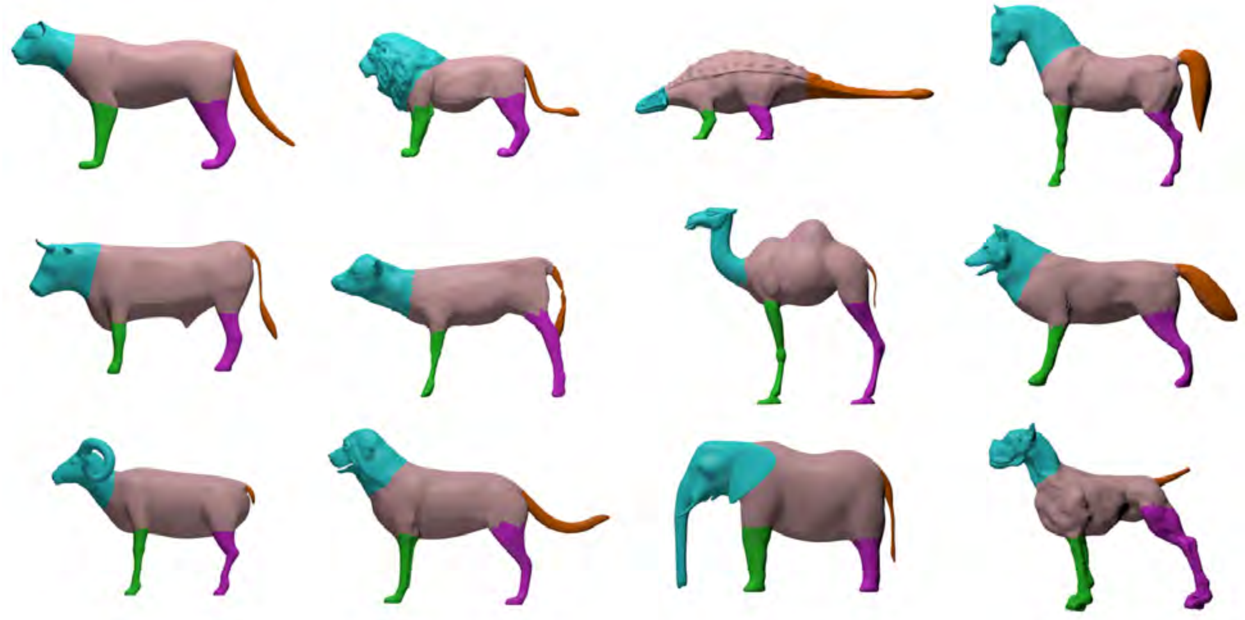
In Table B.1, we show the total runtimes to generate each of our main results, along with the number of input models, total number of vertices in the input models, number of components and the number of possible shapes which can be assembled from the components. For the simple case of N input models, each with K components, this number is simply N^K.

	Num. Models	**Num. Components**	**Possible Shapes**	**Num. Vertices**	**Runtime (sec.)**
Insects	6	70	302M	105144	229
Faces	5	42	250K	79975	370
Humanoids	5	35	78K	95500	293
Animals	13	91	62M	122764	465
Chairs	6	47	1.6M	152648	148

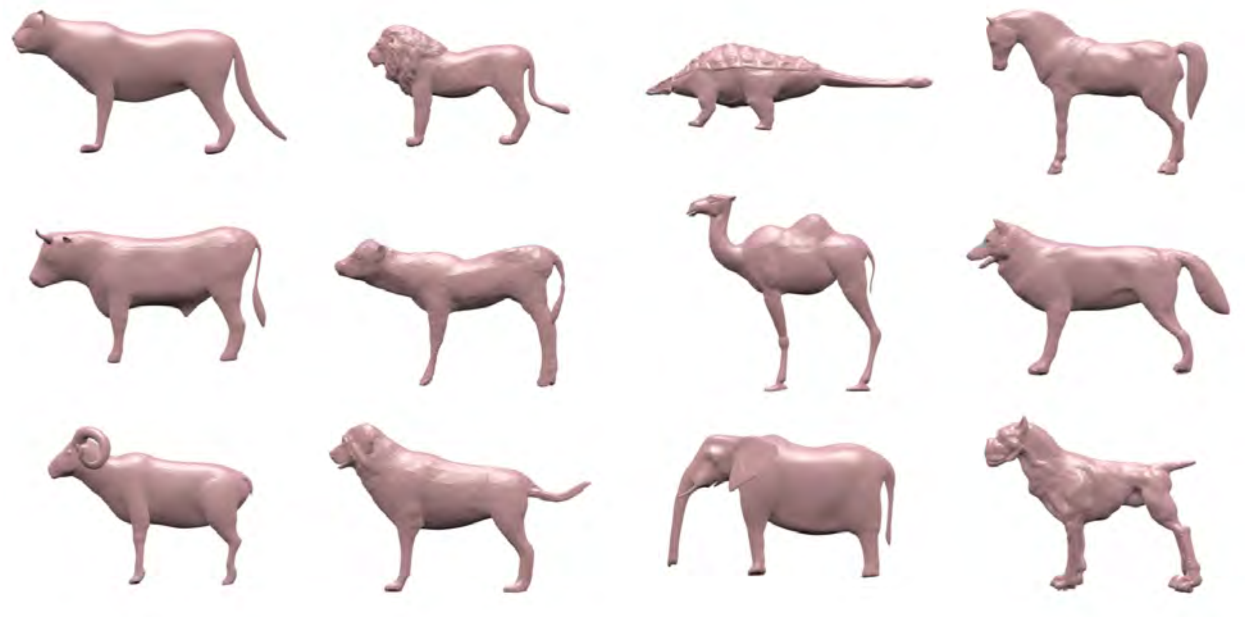
Table B.1: Interchangeable components runtimes.

B.2 Deformation Comparison

The following figures show the initial input shapes colored by their semantic regions and the input shapes after deformation for component compatibility. The deformations were created using the default uniform weights, without any user guidance.



(a)



(b)

Figure B.1: (a) The original input animals colored by their segmentations. (b) The same animals after deformation.

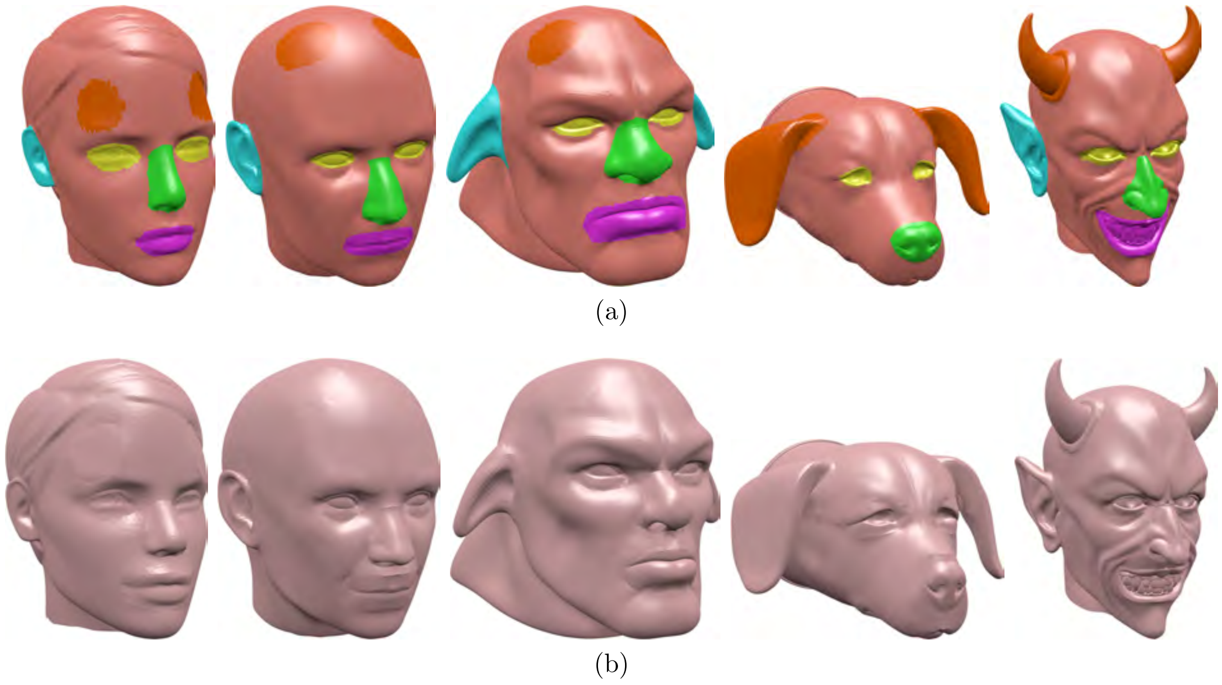


Figure B.2: (a) The original input faces colored by their segmentations. (b) The same faces after deformation.

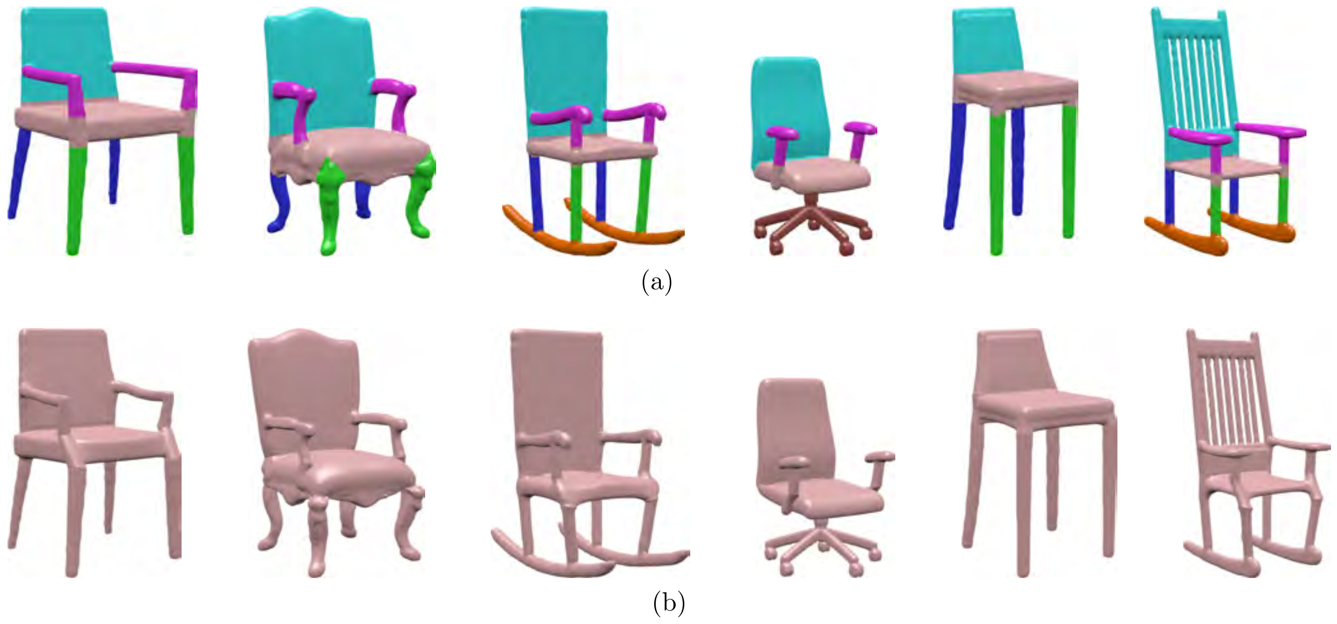
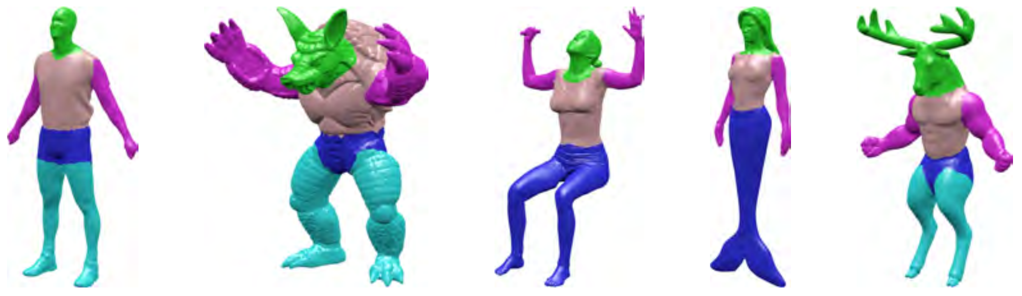
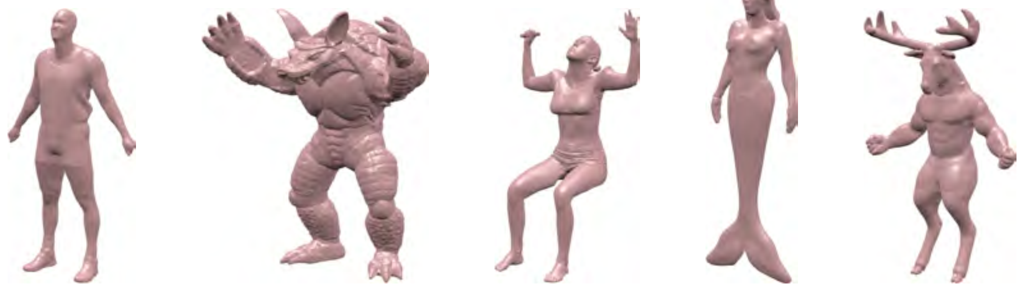


Figure B.3: (a) The original input chairs colored by their segmentations. (b) The same chairs after deformation.

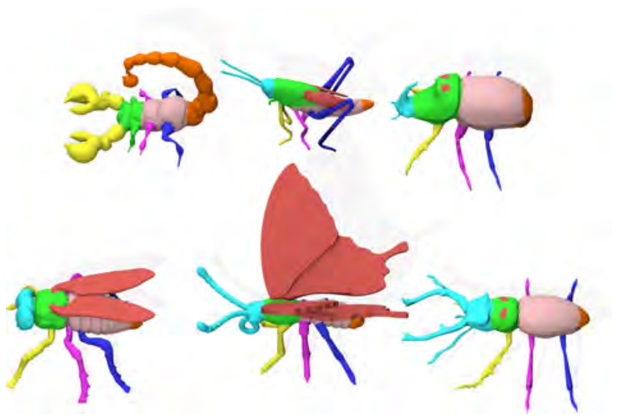


(a)



(b)

Figure B.4: (a) The original input humanoids colored by their segmentations. (b) The same humanoids after deformation.


(a)

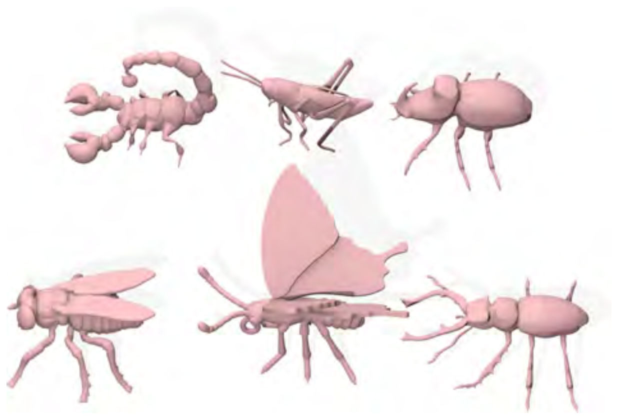

(b)

Figure B.5: (a) The original input insects colored by their segmentations. (b) The same insects after deformation.

APPENDIX C

Supplemental Material on Approximate Dissections

This appendix visualizes the closeness of the dissection pieces' approximation of the original shapes and describes how to initialize the geometry optimization.

C.1 Comparing the Input and Output Shapes

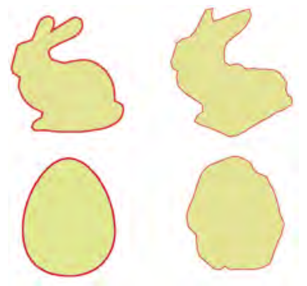
In Figure C.1 we show each input shape next to its corresponding arrangement of dissection pieces without showing the interior dissection piece boundaries. This style lets us more easily see how accurately the dissection pieces approximate the input shapes.

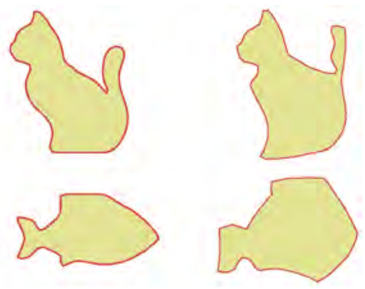
C.2 Initializing the Full Geometry Optimization

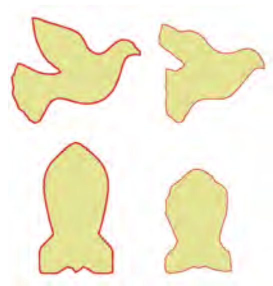
The geometry optimization in Section 5.2 is non-convex and hence the final objective value is dependent on the starting point. We use the optimization result from the orientation-based pruning described in Section 5.4.1 to obtain a reasonable starting point. The orientation-based pruning is posed as a mixed-integer linear programming problem and does not need a starting point of its own to find the global optimum.

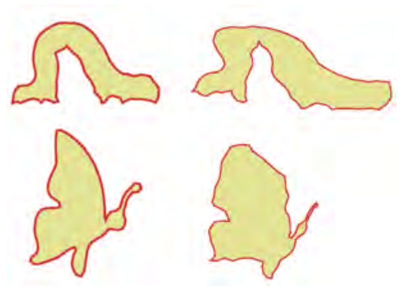
The optimization variables for the full geometry optimization are the dissection piece rotation angles $\boldsymbol{\theta}$ and the junction constraint edge vector sequences $\mathbf{E} = \{\mathbf{E}_1, ..., \mathbf{E}_M\}$.

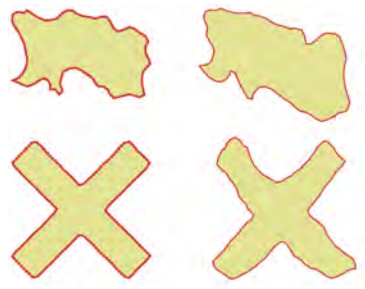
The initial values for $\boldsymbol{\theta}$ are simply set to their corresponding values from the orientation-

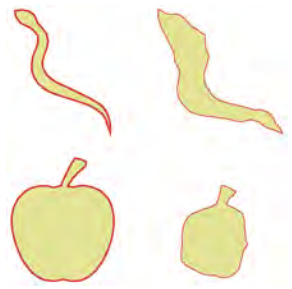

(a) Bunny to Egg

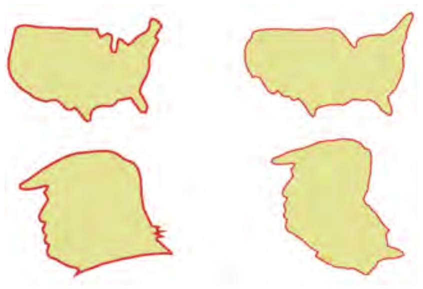

(b) Cat to Fish


(c) Dove to Bomb


(d) Caterpillar to Butterfly


(e) Map of Island to X


(f) Snake to Apple


(g) USA to Trump

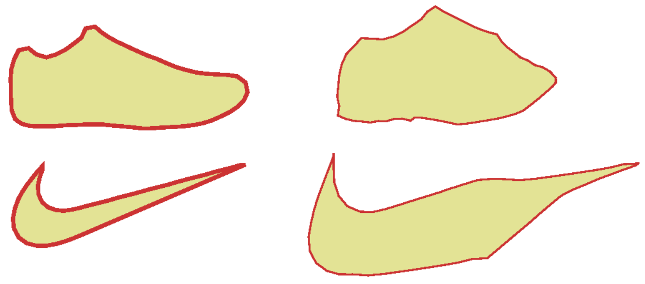

(h) Shoe to Nike Logo

Figure C.1: The input shapes (left) next to the arranged dissection pieces (right).

based optimization's solution.

The initial values for $\mathbf{E}$ are obtained by solving the optimization in Section 5.2 with $\boldsymbol{\theta}$ fixed to the initial values, ignoring the constraints and using the least-squares regularization term instead of the maximum angular error. This simplified optimization only requires a linear solve.

This procedure is not guaranteed to produce the "optimal" starting point, but was effective in practice.

REFERENCES

Aldersey-Williams, H. (2003). *Zoomorphic: New Animal Architecture*. HarperDes. 4

Alhashim, I., Li, H., Xu, K., Cao, J., Ma, R., and Zhang, H. (2014). Topology-varying 3D shape creation via structural blending. *ACM Trans. Graph.* 89

Aliaga, D. G., Vanegas, C. A., and Beneš, B. (2008). Interactive example-based urban layout synthesis. *ACM Trans. Graph.*, 27(5):160. 1, 15

Bächer, M., Whiting, E., Bickel, B., and Sorkine-Hornung, O. (2014). Spin-it: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph.*, 33(4):96. 14, 16, 106

Baran, I. and Popović, J. (2007). Automatic rigging and animation of 3D characters. *ACM Trans. Graph.*, 26(3):72. 28

Bischoff, S., Weyand, T., and Kobbelt, L. (2005). Snakes on triangle meshes. In *Bildverarbeitung für die Medizin 2005*, pages 208–212. Springer. 46

Bond, A. (2014). Hands-on toys help kids prep for school and life, research says. `http://www.huffingtonpost.com/2014/03/21/blocks-puzzles-help-kids_n_5008358.html`. Accessed: 1-2-2016. 15

Bosboom, J., Demaine, E., Demaine, M., Lynch, J., Manurangsi, P., Rudoy, M., and Yodpinyanee, A. (2015). k-piece dissection is NP-hard. In *Abstracts from the 18th Japan Conf. on Discrete and Computational Geometry and Graphs*. 10

Bramston, D. (2008). *Basics Product Design 01: Idea Searching*. AVA Publishing. 4

Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., and Evans, T. R. (2001). Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 67–76. ACM. 54

Chaudhuri, S., Kalogerakis, E., Guibas, L., and Koltun, V. (2011). Probabilistic reasoning for assembly-based 3D modeling. *ACM Trans. Graph.*, 30(4):35. 13, 15

Chaudhuri, S. and Koltun, V. (2010). Data-driven suggestions for creativity support in 3D modeling. *ACM Trans. Graph.*, 29(6):183:1–10. 13, 15, 60

Chen, X., Zhang, H., Lin, J., Hu, R., Lu, L., Huang, Q., Benes, B., Cohen-Or, D., and Chen, B. (2015). Dapper: Decompose-and-pack for 3D printing. *ACM Trans. Graph.*, 34(6):213. 16

Coates, M., Brooker, G., and Stone, S. (2009). *The Visual Dictionary of Interior Architecture and Design*. Fairchild Books. 4

Cohn, M. (1975). Economical triangle-square dissection. *Geometriae Dedicata*, 3(4):447–467. 10

Duncan, N., Yu, L.-F., and Yeung, S.-K. (2016). Interchangeable components for hands-on assembly based modelling. *ACM Transactions on Graphics (TOG)*, 35(6):234:1–14. 3

Duncan, N., Yu, L.-F., Yeung, S.-K., and Terzopoulos, D. (2015). Zoomorphic design. *ACM Transactions on Graphics (TOG)*, 34(4):95:1–13. 2

Duncan, N., Yu, L.-F., Yeung, S.-K., and Terzopoulos, D. (2017). Approximate dissections. *ACM Transactions on Graphics (TOG)*, 36(6):182:1–13. 3

Fisher, M., Savva, M., and Hanrahan, P. (2011). Characterizing structural relationships in scenes using graph kernels. *ACM Trans. Graph.*, 30(4):34. 114

Frederickson, G. (2002). *Hinged Dissections: Swinging and Twisting*. Cambridge University Press. 9

Frederickson, G. (2003). *Dissections: Plane and Fancy*. Cambridge University Press. 17

Fu, H., Cohen-Or, D., Dror, G., and Sheffer, A. (2008). Upright orientation of man-made objects. *ACM Trans. Graph.*, 27(3). 113

Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., and Dobkin, D. (2004). Modeling by example. *ACM Transactions on Graphics (TOG)*, 23(3):652–663. 13, 15

Gardner, R. (1985). A problem of Sallee on equidecomposable convex bodies. *Proc. American Mathematical Society*, 94(2):329–332. 10

Golinkoff, R. M., Hirsh-Pasek, K., and Eyer, D. (2004). *Einstein Never Used Flashcards: How Our Children Really Learn – And Why They Need to Play More and Memorize Less.* Rodale Books. 15

Gurobi Optimization, Inc. (2016). Gurobi optimizer reference manual. 77

Hu, R., Li, H., Zhang, H., and Cohen-Or, D. (2014). Approximate pyramidal shape decomposition. *ACM Trans. Graph.*, 33(6):213. 16

Huang, Y.-J., Chan, S.-Y., Lin, W.-C., and Chuang, S.-Y. (2016). Making and animating transformable 3D models. *Computers & Graphics*, 54:127–134. 18

Igarashi, T., Matsuoka, S., and Tanaka, H. (1999). Teddy: A sketching interface for 3D freeform design. *ACM Trans. Graph.*, pages 409–416. 13

Igarashi, Y., Igarashi, T., and Mitani, J. (2012). Beady: Interactive beadwork design and construction. *ACM Trans. Graph.*, 31(4):49. 1, 15

Jacobson, A., Baran, I., Kavan, L., Popović, J., and Sorkine, O. (2012). Fast automatic skinning transformations. *ACM Trans. Graph.*, 31(4):77. 28

Jain, A., Thormählen, T., Ritschel, T., and Seidel, H.-P. (2012). Exploring shape variations by 3D-model decomposition and part-based recombination. *Computer Graphics Forum*, 31(2pt3):631–640. 13, 15

Kalogerakis, E., Chaudhuri, S., Koller, D., and Koltun, V. (2012). A probabilistic model for component-based shape synthesis. *ACM Trans. Graph.*, 31(4):55. 13, 16, 25, 42, 43, 89, 117, 118

Kalogerakis, E., Hertzmann, A., and Singh, K. (2010). Learning 3D mesh segmentation and labeling. *ACM Trans. Graph.*, 29(4):102:1–102:12. 19, 20

Kaplan, C. and Salesin, D. (2000). Escherization. In *Proc. ACM SIGGRAPH '00 Conf.*, pages 499–510. 17

Kashima, H., Tsuda, K., and Inokuchi, A. (2004). Kernels for graphs. In *Kernel Methods in Computational Biology*, pages 155–170. 21

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331. 46

Koo, B., Li, W., Yao, J., Agrawala, M., and Mitra, N. J. (2014). Creating works-like prototypes of mechanical objects. *ACM Trans. Graph.*, 33(6). 16

Kraevoy, V., Julius, D., and Sheffer, A. (2007). Shuffler: Modeling with interchangeable parts. *The Visual Computer*. 13, 15

Kranakis, E., Krizanc, D., and Urrutia, J. (2000). Efficient regular polygon dissections. *Geometriae Dedicata*, 80(1):247–262. 10

Kwan, K., Sinn, L., Han, C., Wong, T.-T., and Fu, C.-W. (2016). Pyramid of arclength descriptor for generating collage of shapes. *ACM Transactions on Graphics (TOG)*, 35(6):229. 18, 101

Laga, H., Mortara, M., and Spagnuolo, M. (2013). Geometry and context for semantic correspondences and functionality recognition in man-made 3D shapes. *ACM Trans. Graph.*, 32(5):150. 14

Lau, M., Ohgawara, A., Mitani, J., and Igarashi, T. (2011). Converting 3D furniture models to fabricatable parts and connectors. *ACM Trans. Graph.*, 30(4):85. 16

Lee, C. H., Varshney, A., and Jacobs, D. W. (2005). Mesh saliency. *ACM Trans. Graph.*, 24(3):659–666. 31

Li, H., Alhashim, I., Zhang, H., Shamir, A., and Cohen-Or, D. (2012). Stackabilization. *ACM Trans. Graph.*, 31(6):158. 14

Lidwell, W. (2014). Power to the peeps: Empowering children through eyewear design. `http://www.2020mag.com/story/49775/`. 4

Lidwell, W. and Manacsa, G. (2011). *Deconstructing Product Design: Exploring the Form, Function, Usability, Sustainability, and Commercial Success of 100 Amazing Products.* Rockport Publishers. 4

Löffler, M., Kaiser, M., van Kapel, T., Klappe, G., van Kreveld, M., and Staals, F. (2014). The connect-the-dots family of puzzles: Design and automatic generation. *ACM Transactions on Graphics (TOG)*, 33(4):72. 17

Lozano, J. A. (2006). *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms*, volume 192. Springer. 37

Luo, L., Baran, I., Rusinkiewicz, S., and Matusik, W. (2012). Chopper: Partitioning models into 3D-printable parts. *ACM Trans. Graph.*, 31(6):129. 16

Manurangsi, P., Rudoy, M., and Yodpinyanee, A. (2016). Dissection with the fewest pieces is hard, even to approximate. In *18th Japan Conference on Discrete and Computational Geometry and Graphs (JCDCGG 2015)*, volume 9943, page 37. 10

Merrell, P., Schkufza, E., Li, Z., Agrawala, M., and Koltun, V. (2011). Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.*, 30(4):87. 1, 15

Mitra, N., Wand, M., Zhang, H. R., Cohen-Or, D., Kim, V., and Huang, Q.-X. (2013). Structure-aware shape processing. In *SIGGRAPH Asia 2013 Courses*, page 1. ACM. 14

Prévost, R., Whiting, E., Lefebvre, S., and Sorkine-Hornung, O. (2013). Make it stand: Balancing shapes for 3D fabrication. *ACM Trans. Graph.*, 32(4). 14, 16, 40, 106, 107, 110

Schmidt, R. and Singh, K. (2010a). Drag, drop, and clone: An interactive interface for surface composition. Technical report, Citeseer. 56

Schmidt, R. and Singh, K. (2010b). Meshmixer: An interface for rapid mesh composition. In *ACM SIGGRAPH 2010 Talks*, page 6. ACM. 13

Schulz, A., Shamir, A., Levin, D. I., Sitthi-Amorn, P., and Matusik, W. (2014). Design and fabrication by example. *ACM Trans. Graph.*, 33(4):62. 16, 109

Shapira, L., Shalom, S., Shamir, A., Cohen-Or, D., and Zhang, H. (2010). Contextual part analogies in 3D objects. *International Journal of Computer Vision*, 89(2-3):309–326. 14

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis.* Cambridge University Press. 21, 22, 114

Shilane, P., Min, P., Kazhdan, M., and Funkhouser, T. (2004). The Princeton shape benchmark. In *Proceedings of Shape Modeling Applications*, pages 167–178. IEEE. 83

Sidi, O., van Kaick, O., Kleiman, Y., Zhang, H., and Cohen-Or, D. (2011). Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. Graph.*, 30(6). 42

Song, P., Fu, C.-W., Jin, Y., Xu, H., Liu, L., Heng, P.-A., and Cohen-or, D. (2017). Reconfigurable interlocking furniture. *ACM Transactions on Graphics (TOG)*, 36(6). 18

Sorkine, O. and Alexa, M. (2007a). As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*, volume 4. 30, 36

Sorkine, O. and Alexa, M. (2007b). As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 109–116, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association. 53

Tagliasacchi, A., Alhashim, I., Olson, M., and Zhang, H. (2012). Mean curvature skeletons. *Computer Graphics Forum*, 31(5):1735–1744. 28

Takayama, K., Schmidt, R., Singh, K., Igarashi, T., Boubekeur, T., and Sorkine, O. (2011). Geobrush: Interactive mesh geometry cloning. *Computer Graphics Forum*, 30(2):613–622. 13

Umetani, N., Igarashi, T., and Mitra, N. J. (2012). Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, 31(4):86. 16

Umetani, N., Kaufman, D. M., Igarashi, T., and Grinspun, E. (2011). Sensitive couture for interactive garment editing and modeling. *ACM Trans. Graph.*, 30(4). 1, 15

Umetani, N., Koyama, Y., Schmidt, R., and Igarashi, T. (2014). Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Trans. Graph.*, 33(4):65. 14, 16, 106, 107

Vanegas, C. A., Garcia-Dorado, I., Aliaga, D. G., Benes, B., and Waddell, P. (2012). Inverse design of urban procedural models. *ACM Trans. Graph.*, 31(6). 1, 15

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57. 51, 67, 110

Wang, J. and Oliveira, M. M. (2003). A hole-filling strategy for reconstruction of smooth surfaces in range images. In *Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003)*, pages 11–18. IEEE. 54

Xu, J. and Kaplan, C. (2007). Image-guided maze construction. *ACM Transactions on Graphics (TOG)*, 26(3). 17

Xu, K., Li, H., Zhang, H., Cohen-Or, D., Xiong, Y., and Cheng, Z.-Q. (2010a). Style-content separation by anisotropic part scales. *ACM Trans. Graph.*, 29(6):184. 112

Xu, X., Zhang, L., and Wong, T.-T. (2010b). Structure-based ASCII art. *ACM Transactions on Graphics (TOG)*, 29(4). 17

Yao, M., Chen, Z., Luo, L., Wang, R., and Wang, H. (2015). Level-set-based partitioning and packing optimization of a printable model. *ACM Trans. Graph.*, 34(6):214. 16

Yu, L.-F., Yeung, S. K., Tang, C.-K., Terzopoulos, D., Chan, T. F., and Osher, S. (2011). Make it home: Automatic optimization of furniture arrangement. *ACM Trans. Graph.*, 30(4):86. 1, 15

Yumer, M. E., Chaudhuri, S., Hodgins, J. K., and Kara, L. B. (2015). Semantic shape editing using deformation handles. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2015)*, 34. 111

Zhang, H., Sheffer, A., Cohen-Or, D., Zhou, Q., Van Kaick, O., and Tagliasacchi, A. (2008). Deformation-driven shape correspondence. *Computer Graphics Forum*, 27(5):1431–1439. 14, 36

Zheng, Y., Cohen-Or, D., and Mitra, N. J. (2013). Smart variations: Functional substructures for part compatibility. *Computer Graphics Forum (Eurographics)*, 32(2pt2):195–204. 27, 89

Zheng, Y., Liu, H., Dorsey, J., and Mitra, N. J. (2016). Ergonomics-inspired reshaping and exploration of collections of models. *IEEE Transactions on Visualization and Computer Graphics*, 22(6):1732–1744. 14

Zhou, Y., Sueda, S., Matusik, W., and Shamir, A. (2014). Boxelization: Folding 3D objects into boxes. *ACM Trans. Graph.*, 33(4):71. 1, 15, 17

Zhou, Y., Wang, R., et al. (2012). An algorithm for creating geometric dissection puzzles. In *Proc. Bridges Conf.*, pages 49–58. 17

Zhu, L., Xu, W., Snyder, J., Liu, Y., Wang, G., and Guo, B. (2012). Motion-guided mechanical toy modeling. *ACM Trans. Graph.*, 31(6):127. 1, 15

Zou, C., Cao, J., Ranaweera, W., Alhashim, I., Tan, P., Sheffer, A., and Zhang, H. (2016). Legible compact calligrams. *ACM Transactions on Graphics (TOG)*, 35(4):122. 17