Interchangeable Components for Hands-On Assembly Based Modelling

Noah Duncan[1*] Lap-Fai Yu[2] Sai-Kit Yeung[3]

[1]University of California, Los Angeles
[2]University of Massachusetts Boston [3]Singapore University of Technology and Design

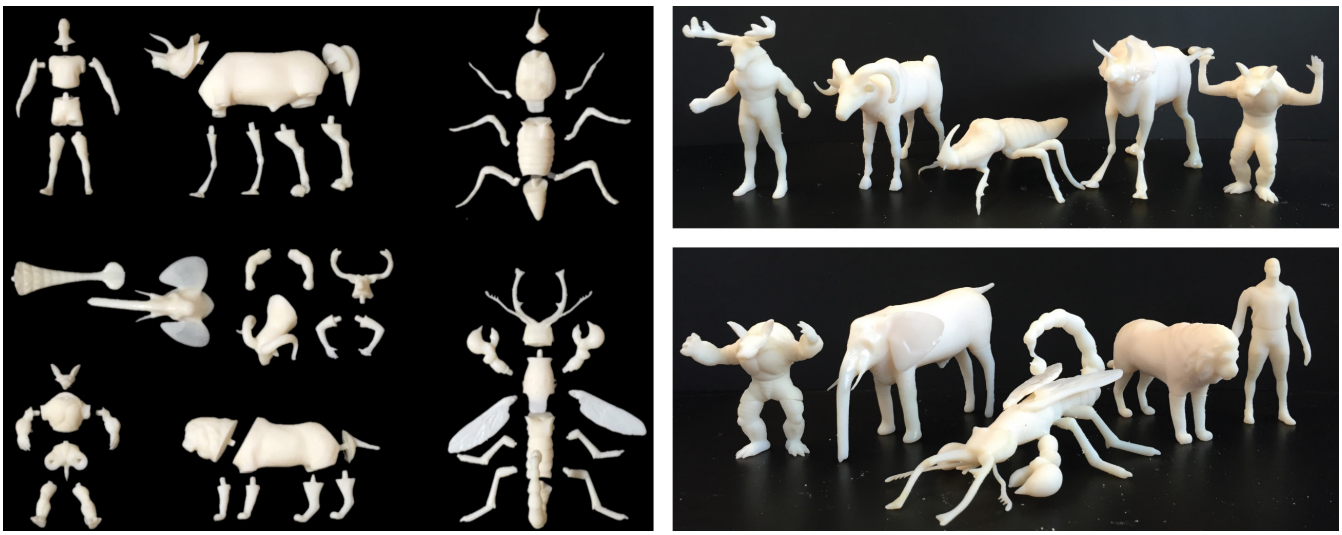


Figure 1: *Our approach deforms and partitions a set of 3D models to fabricate a set of fully interchangeable components, which can be assembled into novel objects with a coherent appearance.*

Abstract

Interchangeable components allow an object to be easily reconfigured, but usually reveal that the object is composed of parts. In this work, we present a computational approach for the design of components which are interchangeable, but also form objects with a coherent appearance which conceals their composition from parts. These components allow a physical realization of Assembly Based Modelling, a popular virtual modelling paradigm in which new models are constructed from the parts of existing ones. Given a collection of 3D models and a segmentation that specifies the component connectivity, our approach generates the components by jointly deforming and partitioning the models. We determine the component boundaries by evolving a set of closed contours on the input models to maximize the contours' geometric similarity. Next, we efficiently deform the input models to enforce both C0 and C1 continuity between components while minimizing deviation from their original appearance. The user can guide our deformation scheme to preserve desired features. We demonstrate our approach on several challenging examples, showing that our components can be physically reconfigured to assemble a large variety of coherent shapes.

Keywords: assembly-based modeling, fabrication, shape creation

Concepts: •**Computing methodologies** → **Mesh geometry models;**

*Part of the work was done when Noah was visiting the Singapore University of Technology and Design and UMass Boston.

SA '16 Technical Papers,, December 05-08, 2016, , Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2982402

ACM Reference Format
Duncan, N., Yu, L., Yeung, S. 2016. Interchangeable Components for Hands-On Assembly Based Modelling. ACM Trans. Graph. 35, 6, Article 234 (November 2016), 14 pages. DOI = 10.1145/2980179.2982402 http://doi.acm.org/10.1145/2980179.2982402.

1 Introduction

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 paper, we focus on an alternate shape creation process in which a set of components is fabricated which 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 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 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 2(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

Figure 2: *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 geometry and clear boundaries between components.*

geometry constructed by these toys is usually extremely simple and possesses an abstract look which makes the component boundaries perceptible as shown in Figure 2(b, c, d).

In this paper, 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 shapes' appearance is challenging with traditional modelling tools. Figure 3 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 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.

2 Related Work

Mix-and-Match Toys. 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

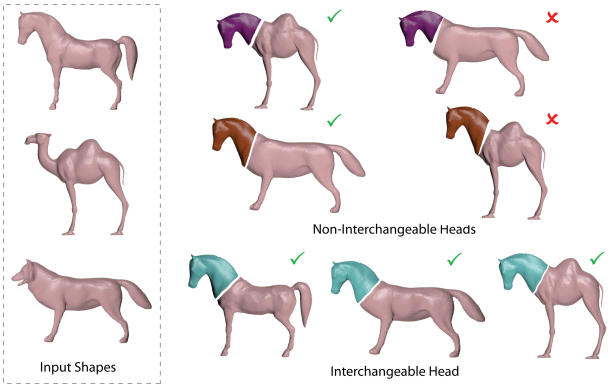


Figure 3: *Without considering interchangeability, the horse's head can fit with either the camel's body or the wolf's body seamlessly, but not both. Considering interchangeability, the horse's head can fit with both the wolf's body and the camel's body seamlessly.*

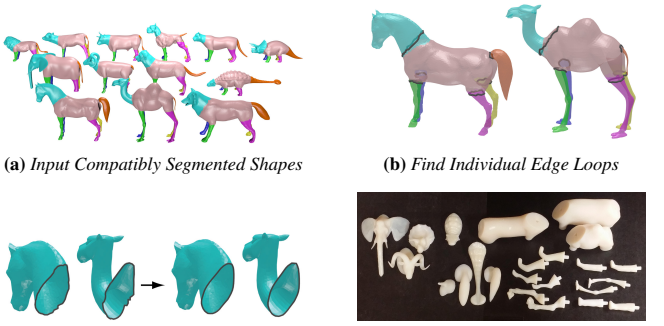
to learn spatial reasoning and express their creativity [Bond 2014; Golinkoff et al. 2004]. Researcher Roberta Golinkoff advises parents to "look for [toys] that children can take apart and remake or reassemble into something different, which builds their imagination." To the best of our knowledge, we are the first to introduce specialized software for the design of these 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 Sheffer et al. [2007] the extraction and composition of components was fully automated. Chaudhuri et al. [2010; 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 modelling 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 [Preost 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



(a) *Input Compatibly Segmented Shapes* **(b)** *Find Individual Edge Loops* **(c)** *Deform to Common Edge Loops* **(d)** *Generate Interchangeable Components*

Figure 4: *Overview of our approach.*

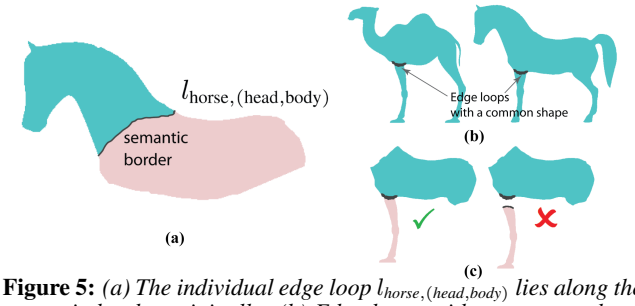


Figure 5: *(a) The individual edge loop $l_{horse,(head,body)}$ lies along the semantic border originally. (b) Edge loops with a common shape are used to partition the front leg and body for both the horse and camel. (c) The camel leg can replace the horse leg if their individual edge loops have the same shape. The camel leg cannot replace the horse leg if their individual edge loops have different shapes.*

which assist the user in creating 3D designs suitable for fabrication. Umetani et al. [2012] introduced an interactive furniture design system which 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 which 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.

3 Overview

Figure 4 shows an overview of our approach. 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.

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 5) 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 in the paper, the input segmentations for the animals (Section 8.1) were obtained with the data-driven approach of [Kalogerakis et al. 2012], while the others were obtained interactively. Experiments in Sections 8.2 and 8.3 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 5(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 5(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 $l_{c,b}$.

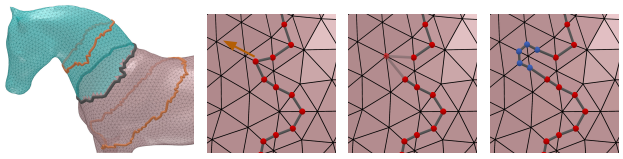
Technical Approach. Our approach proceeds as follows. First, it adjusts the individual edge loops at the semantic borders of the input meshes, to optimize their geometric similarity while not deviating too much from the original segmentation. Next, it deforms the input meshes so that the individual edge 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 which allow them to be assembled in the real world.

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.

4 Finding Individual Edge Loops

Consider a semantic border b (e.g., between the head and body in Figure 6(a)). Given the initial edge loops $l_{m,b}$ for each mesh, this step searches for new edge loops $\hat{l}_{m,b}$, which 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.



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

Figure 6: *(a) The contour for the $l_{horse,(head,body)}$ 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.*

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.

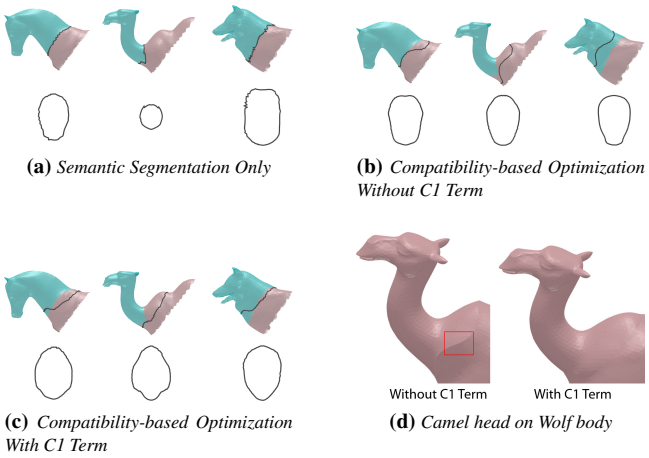
To prevent the contours from deviating too severely from the original semantic borders, we create a zone on each surface called the *intermediate zone* which the contour is constrained to lie in. The intermediate zone between two semantic regions represents the area which does not clearly belong to one region or the other. Figure 6(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 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 in [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 6 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" have been optimized with respect to a property of the surface which they lie upon. 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 (C0 continuity), as well as angles between contour normals (C1 continuity). Both factors are necessary to achieve a smooth transition between components as shown in Figure 7d.

Formally, we seek to minimize E_{dis}, the sum of pairwise dissimilarities between contours:

$$E_{\text{dis}}(\mathbf{C}) = \sum_{c_p, c_q \in \mathbf{C}} D(c_p, c_q) + D(c_q, c_p), \qquad (1)$$



(a) *Semantic Segmentation Only* **(b)** *Compatibility-based Optimization Without C1 Term* **(c)** *Compatibility-based Optimization With C1 Term* **(d)** *Camel head on Wolf body*

Figure 7: *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 C1 term. Without the C1 term, a seam is visible despite C0 continuity.*

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

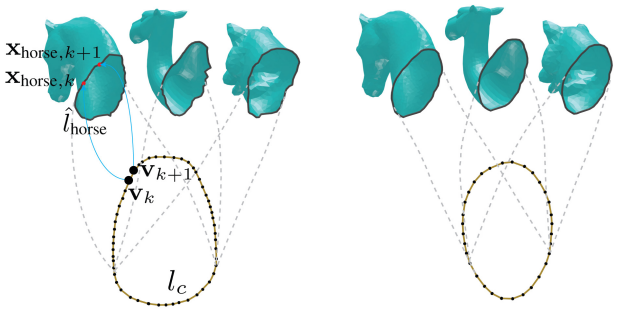
where $\mathbf{v}_p^t$ maps the normalized arclength parameter $t \in [0,1]$ to a position along the contour c_p and $\mathbf{n}_p^t$ does the same for a normal vector along c_p. $\widetilde{\mathbf{v}}_q(\mathbf{x})$ returns the location of the closest point on the contour c_q to the point $\mathbf{x}$ while $\widetilde{\mathbf{n}}_q(\mathbf{x})$ returns the normal of the closest point. 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 C0 continuity over C1 continuity, we set $\alpha = 0.1$. We approximate the integral in equation 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 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 18c. In Sections 8.2, 8.3 and 8.4 we evaluate the behaviour of our contour optimization and its effect on the subsequent steps of the approach.



(a) *Initial Common Edge Loop* **(b)** *Refined Common Edge Loop*

Figure 8: *(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.*

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 7(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 7(c-d) shows the necessity of considering C1 continuity when optimizing the contours.

5 Deformation to Common Edge Loops

In Section 4, 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.

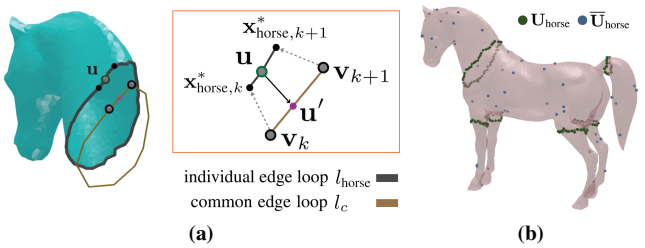
Our approach proceeds in two steps. First, for each semantic border b, we find an initial shape for the common edge loop for b with minimal total shape difference between it and each of the individual edge loops. Then, in the second step we refine the shapes of all the common edge loops simultaneously by solving a global optimization which 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.

5.1 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 8(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) of each mesh m.

We parameterize each individual edge loop $\hat{l}_m$ by its arc length. $\hat{l}_m(t)$ is a point on $\hat{l}_m$, where $t \in [0, 1]$. 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:



(a) **(b)**

Figure 9: *(a) Representing individual edge loop vertices in terms of common edge loop vertices. Using the correspondence computed from Section 5.1, 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 only show some of the vertices for clarity.*

$$\begin{aligned} \min_{\mathbf{V}_c,\mathbf{T}} \quad & \lambda E_{\text{def}}^{\text{loop}}(\mathbf{V}_c, \mathbf{T}) + (1.0 - \lambda) E_{\text{reg}}^{\text{loop}}(\mathbf{T}) \\ \text{subject to} \quad & t_{m,k} < t_{m,k+1},\ \forall m, k. \end{aligned} \tag{3}$$

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:

$$\begin{aligned} E_{\text{def}}^{\text{loop}}(\mathbf{V}_c, \mathbf{T}) = \frac{1}{P} \sum_m \sum_k || & (\mathbf{x}_{m,k+1} - \mathbf{x}_{m,k}) \\ & - (\mathbf{v}_{k+1} - \mathbf{v}_k) ||^2, \end{aligned} \tag{4}$$

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 vertices to spread evenly over the individual edge loops:

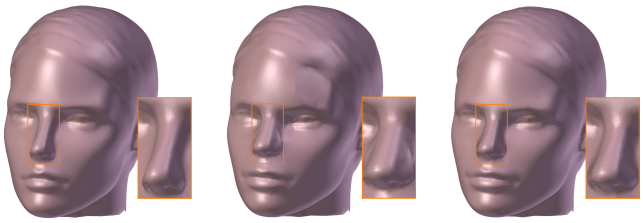
$$E_{\text{reg}}^{\text{loop}}(\mathbf{T}) = \sum_m \sum_k ((t_{m,k+1} - t_{m,k}) - \frac{1}{|\mathbf{V}_c|})^2 \tag{5}$$

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 as 0.9. The optimization can be solved quickly using standard solvers such as IPOPT [Wächter and Biegler 2006]. The optimization computes optimized vertex positions $\mathbf{v}_k^*$ which 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 $\hat{l}_m$ which vertex k corresponds to.

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



(a) *Input* **(b)** *No Refinement* **(c)** *Refinement*

Figure 10: *Shape-preserving refinement. (a) Input shape. (b) Result produced without refinement by using the common edge loop shapes from Section 5.1 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.*

We use the correspondence obtained in Section 5.1 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 9(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 9(b). We can express the individual edge loop vertices $\mathbf{U}_m$ in terms of the common edge loop vertices as shown in Figure 9(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:

$$E_{\text{def}}^{\text{mesh}}(\mathcal{M}) = \sum_m w_m E_{\text{def}}^m(\mathbf{V}_c^{\text{all}}, \overline{\mathbf{U}}_m), \tag{6}$$

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 2007], 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 8(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 10 shows the advantage of our joint optimization over a more naive approach where the initial common edge loop shapes from Section 5.1 are held fixed when deforming the meshes.

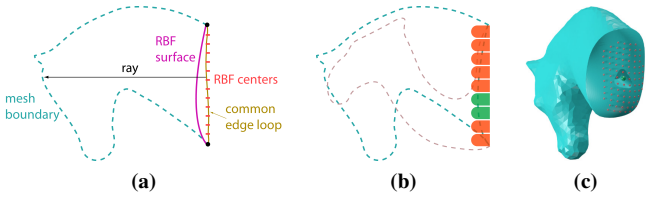


Figure 11: *(a) Surface sealing. A surface is created using RBFs fitted over the domain of the common edge loop shown. Rays are shot along positions sampled within the loop to determine the mesh boundary, which the created surface should lie within. (b) Connector Placement. Candidate locations (red) 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.*

6 Generating Interchangeable Components

Using the refined individual edge loops obtained in Section 5.2, 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 11 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 11(a). Please refer to the literature [Carr et al. 2001; 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 11(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.

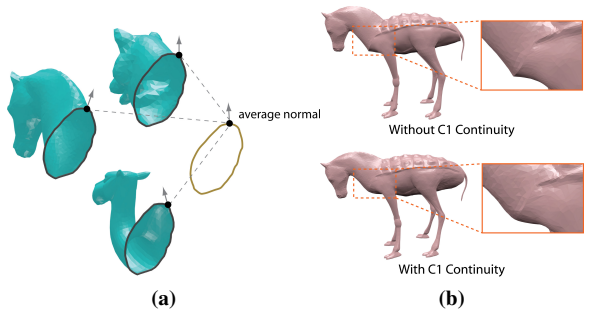


Figure 12: *Considering C1 Component Continuity. (a) Computing an average normal. Average normals are recorded at a set of uniformly sampled locations on the common edge loop. (b) The neck transition becomes smoother after incorporating C1 continuity.*

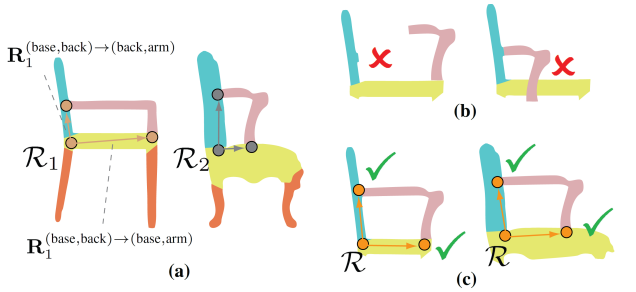


Figure 13: *Considering higher order connectivity in creating an interchangeable arm component. (a) Input chairs. The relative transformations of the (back,arm) loop and the (base,arm) loop with respect to the (base,back) loop are different for each chair, stored in $\mathcal{R}_1$ and $\mathcal{R}_2$ 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 $\mathcal{R}$.*

7 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.

7.1 C1 Component Continuity Deformation

Our contour optimization in Section 4 tries to find edge loops with high C1 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 5 may have poor C1 continuity. To resolve this issue, we apply an additional deformation step that locally enforces C1 continuity. Figure 12 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 5) with $\hat{l}_{m,b}$, the individual edge loop of mesh m for b. We record the normal orientation on the mesh 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 2010], 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.

7.2 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 13. Our basic approach fails in this situation, because it only guarantees that individual edge loops belonging to a single semantic border are equal. For the arm component this guarantee is not sufficient. As shown in Figure 13(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 3. We constrain the union of individual edge loops across a *set* of semantic borders to be equal

Figure 14: *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.*

(up to rigid transformation) rather than just a single semantic border, as before. For the example in Figure 13, 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 5. Figure 13(c) shows an example. In the extra optimization step, the 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 Equation (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 13(c) shows the

(a) (b) (c)

Figure 15: *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).*

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 Equation (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.

7.3 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 14 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^m_{def} in Equation(6) by painting the surface of the mesh with modified weights. We assume that E^m_{def} 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 14(c) shows the result of preserving the grin after the user labels the mouth region to have higher weight in the deformation energy. Please refer to our supplementary video for a demonstration of this feature.

7.4 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 15, 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 which could be included in the toy. Our approach can conveniently find the subset of shapes which 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.

To perform this task, we define a dissimilarity metric between input meshes which leverages the compatibility optimization used in Section 4. We compute the dissimilarity between mesh m and mesh n as follows:

$$D_{\text{shape}}(m,n) = \frac{1}{|\mathbf{B}|}\sum_{b\in\mathbf{B}} \min_{\substack{c_p\in\mathbf{I}_{m,b},\\ c_q\in\mathbf{I}_{n,b}}} D(c_p,c_q) + D(c_q,c_p), \quad (7)$$

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, likewise for $\mathbf{I}_{n,b}$. $D(c_p,c_q)$ measures the distance of contour c_p from contour c_q, as defined in Equation (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.

(a) (b) (c)

Figure 16: *(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.*

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 is. We evaluate this technique in Section 7.4. For a database which has several models belonging to the same category, we recommend incorporating a geometric diversity metric as described by Chaudhuri and Koltun [2010] into the search so that the objective of component compatibility in the subset can be balanced against the desire for geometric diversity.

8 Results and Experiments

8.1 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.

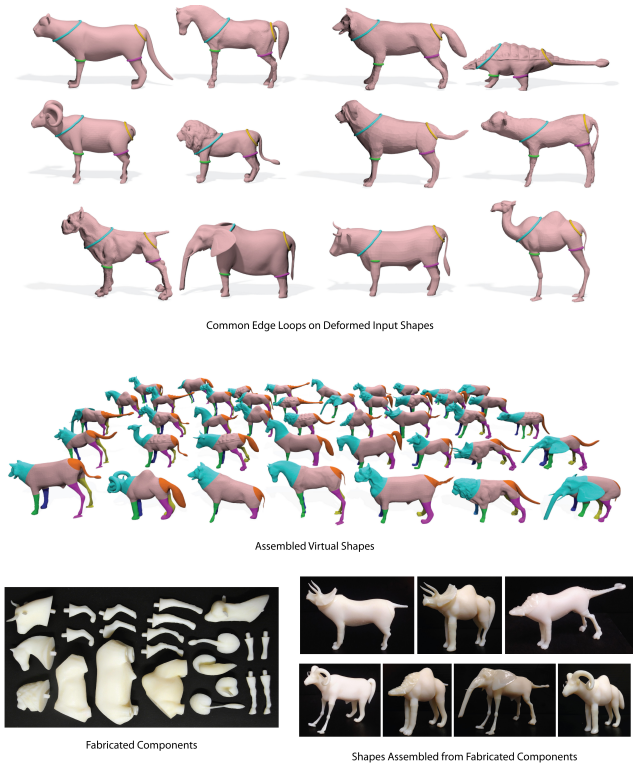
Animals. In this experiment we used our approach to generate a set of components for constructing chimeric four-legged creatures. Twelve animal shapes were used as the input to our approach.

We show the edge loops chosen by our compatibility-based optimization in Figure 17(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 20 shows a diverse set of animals assembled by our interchangeable components.

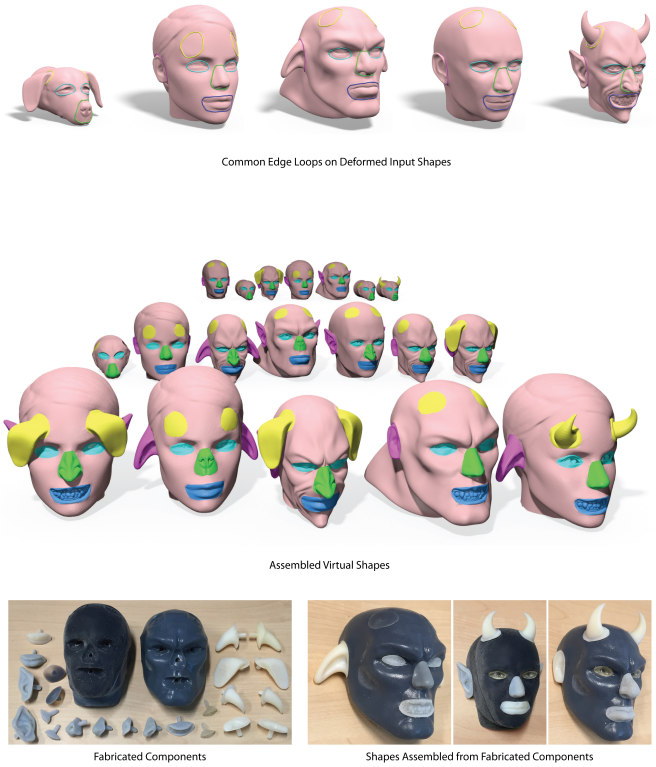
This result was inspired by several similar commercial products. However, all these products either possess a much more limited set of constructible shapes than our result or make it very obvious that the shapes are composed of components (Figure 15).

Most Compatible Subset of Animals: We applied our most-compatible subset technique (Section 7.4) to find the most compatible size 4 subset of the full set of animals. We show the subset and some results in Figure 16. 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 17(b)). This result was inspired



(a) *Animals*



(b) *Faces*

(c) *Chairs*

(d) *Humanoids*

Figure 17: *Generating interchangeable components for different types of shapes.*

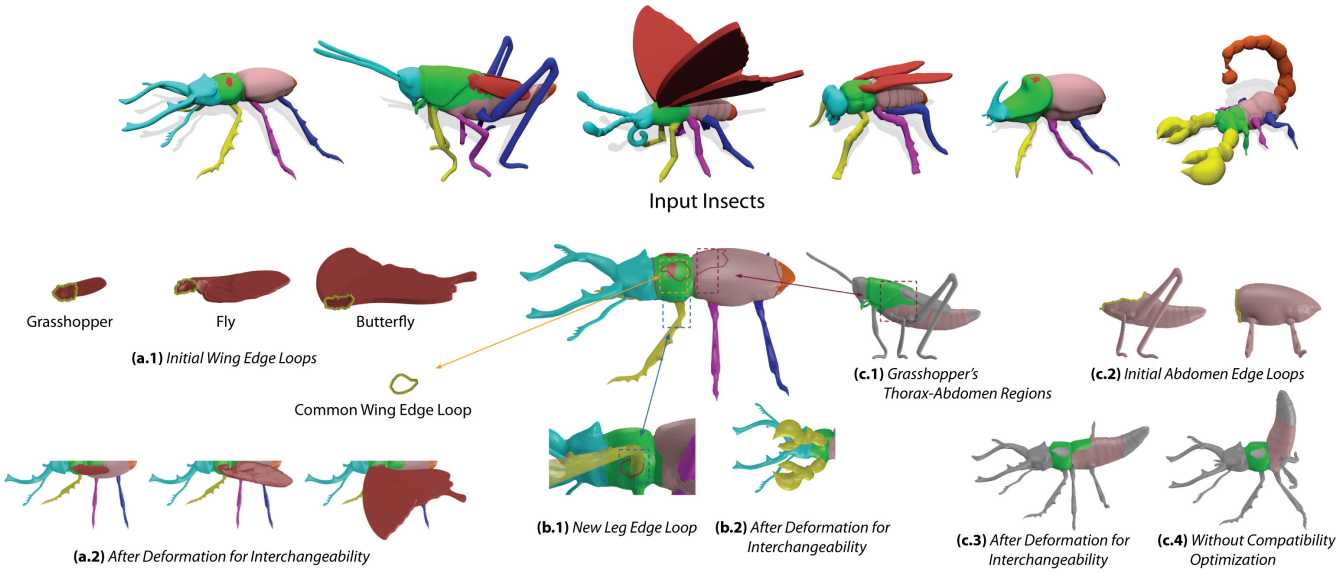


Figure 18: *Some of the challenges involved in generating insect components. Refer to the text for explanation.*

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 7.3).

By segmenting meshes appropriately, our approach allows the user to connect components to an object which 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 17(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 7.2) 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 17(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 26 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 18. 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 18 we highlight the challenges involved in creating interchangeable components for the Stag Beetle. The initial semantic regions of the Stag Beetle and the edge loops found by our compatibility optimization are shown in the center.

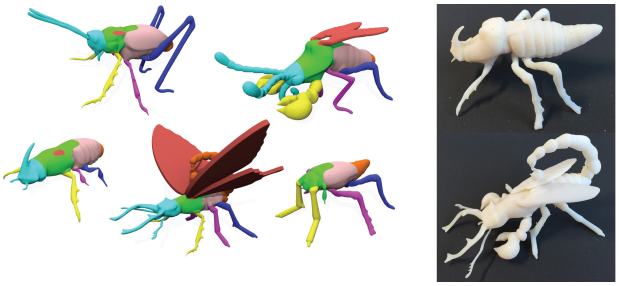


Figure 19: *Some assembled insects.*

In Figure 18(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 18(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 18(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 (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

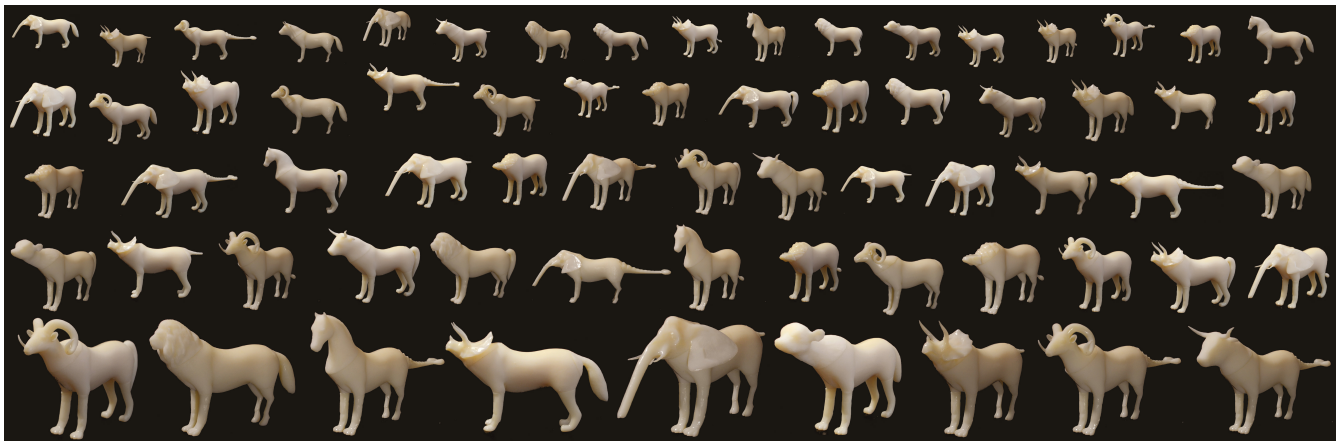
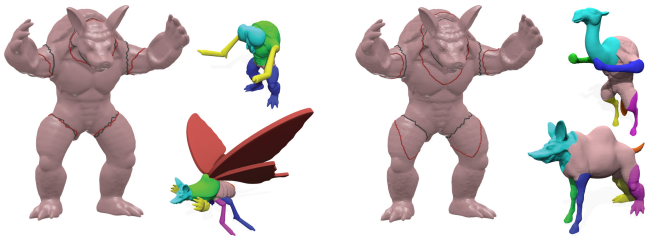
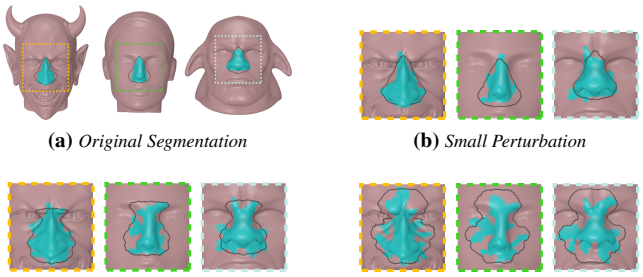


Figure 20: *Animals assembled by our interchangeable components. Note that each assembled animal is different and even more variations are possible.*



(a) *With Insects* **(b)** *With Animals*

Figure 21: *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.*



(a) *Original Segmentation* **(b)** *Small Perturbation*

(c) *Medium Perturbation* **(d)** *Large Perturbation*

Figure 22: *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.*

approach jointly considers these factors for six models. Conducting this process by hand would be a tedious and difficult task. Figure 19 shows some assembled insects and their prints.

8.2 Cross-Category Components

We demonstrate that our approach is flexible enough to generate interchangeable components which 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 21 shows how our contour optimization scheme (Section 4) generates significantly different edge loops depending on which category we target. For example, because the 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 21.

8.3 Sensitivity to Initial Segmentation

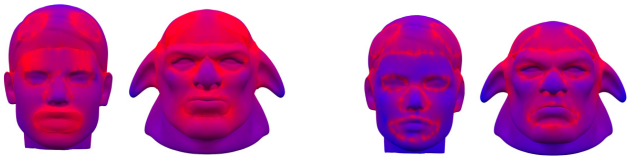
We examine our approach's dependence on the input semantic segmentation. For three faces, we run our contour optimization (Section 4) on the border between the nose and face. We apply three different levels of perturbation to the initial segmentation. Figure 22 visualizes the results. The resulting contours are all semantically valid unless the perturbation is severe, which causes them to cross into the eyes.

8.4 Compatibility Optimization and Mesh Deformation

We quantitatively evaluated the improvement of the component's quality resulting from the compatibility-based optimization (Section 4). We ran our full approach and measured the extent of the deformation for compatibility (Section 5) using the ARAP mesh energy. Next, we re-ran the approach with the compatibility-based optimization omitted and compared the energies. See Figure 25 for the results. The extent of our improvement ranges from over 100% for the faces to about 17% for the humanoids. Figure 23 visualizes how the distortion is distributed over the mesh.

8.5 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 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 5.1 and for finding an RBF surface in Section 6 can be solved in a few seconds using standard solvers. The time required to minimize the total mesh deformation energy in Section 5.2 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 input model faces. Please refer to the supplementary material for detailed per-



(a) *Without Compatibility* **(b)** *With Compatibility*

Figure 23: *Visualizing the distortion caused by satisfying component interchangeability. Red indicates distortion level. Compatibility optimization reduces the distortion considerably.*

formance data.

8.6 Fabrication

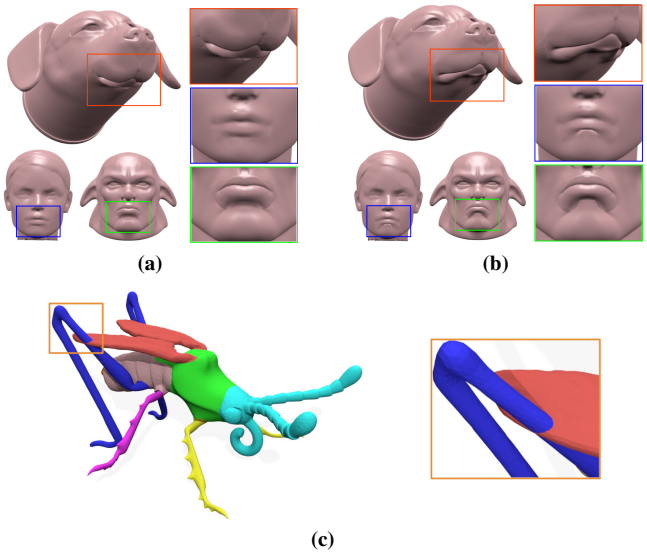
Our results were fabricated on an Objet Connex 3D printer in solid material. The printer possessed sufficient precision to make the components fit together smoothly, making most seams virtually unnoticeable, although components with highly curved connections will have more visible seams. We expect that 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 20, 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.

8.7 Limitations

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 by our approach may be semantically incorrect. The user interaction discussed in Section 7.3 can mitigate these issues, but may be unable to resolve cases where the components of the input meshes differ fundamentally in their geometry. Figure 24(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 24(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. 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 5.2 with one that incorporated template parameters.

Finally, our approach only considers geometric, not physical properties of the assembled objects. For example, it is possible to construct a humanoid (Section 8.1) which does not stand stably on its legs. Combining our geometric problem with the physical problems posed by works like [Preost et al. 2013] offers an interesting direction for future work: guaranteeing that *any* object assembled from a set of components possesses some physical properties.



(a) **(b)** **(c)**

Figure 24: *(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.*

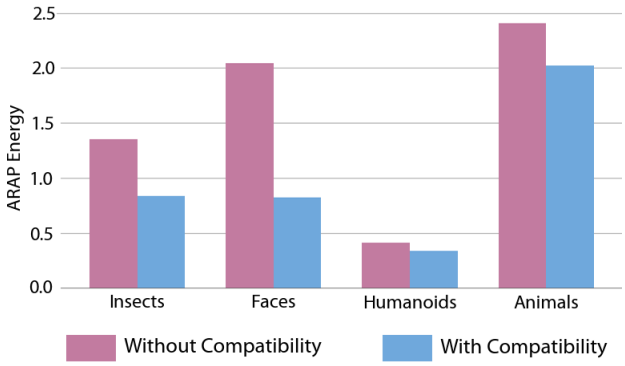


Figure 25: *The improvement in shape preservation from compatibility-based optimization.*

9 Summary and Future Work

We have presented a computational 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 C0 and C1 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 [Preost et al. 2013], spinnability [Bächer et al.

Figure 26: *Humanoids assembled by our components.*

2014], and aerodynamic characteristics [Umetani et al. 2014]. Our work contributes to this trend, with the property being the interchangeability of the components.

We see several directions from which our paper can lead to future work. 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 relaxing constraints on the appearance is an intriguing problem for future work.

10 Acknowledgements

We are grateful to the anonymous reviewers for their constructive comments. We also thank Michael S. Brown for narrating the video; Benjamin Kang Yue Sheng for assistance with fabricating the results; and Ibraheem Alhashim and Alec Jacobson for providing source code for ARAP mesh deformation. Lap-Fai Yu is supported by the University of Massachusetts Boston StartUp Grant P20150000029280 and by the Joseph P. Healey Research Grant Program provided by the Office of the Vice Provost for Research and Strategic Initiatives & Dean of Graduate Studies of the University of Massachusetts Boston. This research is supported by the National Science Foundation under award number 1565978. We also acknowledge NVIDIA Corporation for graphics card donation. Part of the work was done when Noah was visiting SUTD, and when he was visiting UMass Boston under the support of the Joseph P. Healey Research Grant. Sai-Kit Yeung is supported by Singapore MOE Academic Research Fund MOE2013-T2-1-159 and SUTD-MIT International Design Center Grant IDG31300106. We acknowledge the support of the SUTD Digital Manufacturing and Design (DManD) Centre which is supported by the National Research Foundation (NRF) of Singapore. This research is also supported by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative.

References

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.

BISCHOFF, S., WEYAND, T., AND KOBBELT, L. 2005. Snakes on triangle meshes. In *Bildverarbeitung für die Medizin 2005*. Springer, 208–212.

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, Mar. Accessed: 1-2-2016.

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*, ACM, 67–76.

CHAUDHURI, S., AND KOLTUN, V. 2010. Data-driven suggestions for creativity support in 3d modeling. In *ACM Trans. Graph.*, vol. 29, ACM, 183.

CHAUDHURI, S., KALOGERAKIS, E., GUIBAS, L., AND KOLTUN, V. 2011. Probabilistic reasoning for assembly-based 3d modeling. In *ACM Trans. Graph.*, vol. 30, ACM, 35.

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.

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. In *ACM Trans. Graph.*, vol. 23, ACM, 652–663.

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.

HU, R., LI, H., ZHANG, H., AND COHEN-OR, D. 2014. Approximate pyramidal shape decomposition. *ACM Trans. Graph. 33*, 6, 213.

JAIN, A., THORMÄHLEN, T., RITSCHEL, T., AND SEIDEL, H.-P. 2012. Exploring shape variations by 3d-model decomposition and part-based recombination. In *Computer Graphics Forum*, vol. 31, Wiley Online Library, 631–640.

KALOGERAKIS, E., CHAUDHURI, S., KOLLER, D., AND KOLTUN, V. 2012. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph. 31*, 4, 55.

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.

LAU, M., OHGAWARA, A., MITANI, J., AND IGARASHI, T. 2011. Converting 3d furniture models to fabricatable parts and connectors. In *ACM Trans. Graph.*, vol. 30, ACM, 85.

LUO, L., BARAN, I., RUSINKIEWICZ, S., AND MATUSIK, W. 2012. Chopper: partitioning models into 3d-printable parts. *ACM Trans. Graph. 31*, 6, 129.

PREOST, R., WHITING, E., LEFEBVRE, S., AND SORKINE-HORNUNG, O. 2013. Make it stand: Balancing shapes for 3d fabrication. *ACM Trans. Graph. 32*, 4.

SCHMIDT, R., AND SINGH, K. 2010. Drag, drop, and clone: An interactive interface for surface composition. Tech. rep., Citeseer.

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.

SHEFFER, V. K. D. J. A. 2007. Shuffler: Modeling with interchangeable parts. *Visual Computer journal*.

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*, vol. 30. ACM.

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SGP '07, 109–116.

UMETANI, N., IGARASHI, T., AND MITRA, N. J. 2012. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph. 31*, 4, 86.

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.

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.

WANG, J., AND OLIVEIRA, M. M. 2003. A hole-filling strategy for reconstruction of smooth surfaces in range images. In *Computer Graphics and Image Processing, 2003. SIBGRAPI 2003. XVI Brazilian Symposium on*, IEEE, 11–18.

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.