Eurographics Symposium on Geometry Processing 2016
Maks Ovsjanikov and Daniele Panozzo
(Guest Editors)

Volume 35 (2016), Number 5

CustomCut: On-demand Extraction of Customized 3D Parts with 2D Sketches

Xuekun Guo[1] and Juncong Lin[2] and Kai Xu[3] and Siddhartha Chaudhuri[4] and Xiaogang Jin[†1]

[1]State Key Lab of CAD&CG, Zhejiang University, China
[2]Software School of Xiamen University, China
[3]National University of Defense Technology, China
[4]Indian Institute of Technology Bombay, India

Abstract

Several applications in shape modeling and exploration require identification and extraction of a 3D shape part matching a 2D sketch. We present CustomCut, an on-demand part extraction algorithm. Given a sketched query, CustomCut automatically retrieves partially matching shapes from a database, identifies the region optimally matching the query in each shape, and extracts this region to produce a customized part that can be used in various modeling applications. In contrast to earlier work on sketch-based retrieval of predefined parts, our approach can extract arbitrary parts from input shapes and does not rely on a prior segmentation into semantic components. The method is based on a novel data structure for fast retrieval of partial matches: the randomized compound k-NN graph built on multi-view shape projections. We also employ a coarse-to-fine strategy to progressively refine part boundaries down to the level of individual faces. Experimental results indicate that our approach provides an intuitive and easy means to extract customized parts from a shape database, and significantly expands the design space for the user. We demonstrate several applications of our method to shape design and exploration.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Modeling—3D Shape Matching

1. Introduction

Sketch-based interfaces provide an intuitive way to create and explore three-dimensional shapes [OSSJ09]. Most shapes of interest can be drawn, and most humans grow up with at least rudimentary drawing skills [Dec88]. Most humans cannot, however, draw very well. Hence, sketch-based interfaces need to interpret a crude drawing and map it to a detailed shape. This problem could be tackled in a data-driven manner: each sketched component is used to search a repository for a high-quality shape with a matching part, which can be extracted and incorporated into the design.

Matching a sketched component to a shape database is a challenging partial matching problem. In the general setting, an arbitrary region of the shape may match the sketch. To reduce the complexity, existing methods leverage a prior segmentation of each shape into predefined semantic parts [LF08, XXM*13]. These parts are individually matched to the sketch, reducing the problem to one of global, rather than partial matching [ERB*12].

In this paper, we present CustomCut, a new end-to-end technique for sketch-based customized part extraction. In contrast to prior work, we require no prior segmentation of the database shapes into predefined parts. Instead, we perform *on-demand* segmentation of retrieved database shapes, to accurately fit the user intent expressed in the sketched stroke (Figure 1). This significantly expands the design space available to the user and enables several applications.

Designing such an algorithm presents hard technical challenges. Since a presegmented (or prelabeled) database is not available to us, we must simultaneously identify and extract parts that match sketched strokes, resulting in a potentially infinite search space. To prune this search space to a manageable size, we design a strategy to carefully balance between returning all possible matches, and providing an overly narrow set of matches. Further, the matching scheme must be fast enough so that appropriate candidate parts can be sought interactively. To address this challenge, we take a projective approach and turn the 3D partial matching problem into a 2D contour-based one. The 2D contours of all database shapes are segmented into fragments and organized based on fragment similarity, using an efficient data structure. Given a query sketch, our method can leverage the structure to quickly return all potentially matched parts via contour matching and match propagation.

Our paper has two main technical contributions, which together form the core of our algorithm:

† Corresponding author: jin@cad.zju.edu.cn

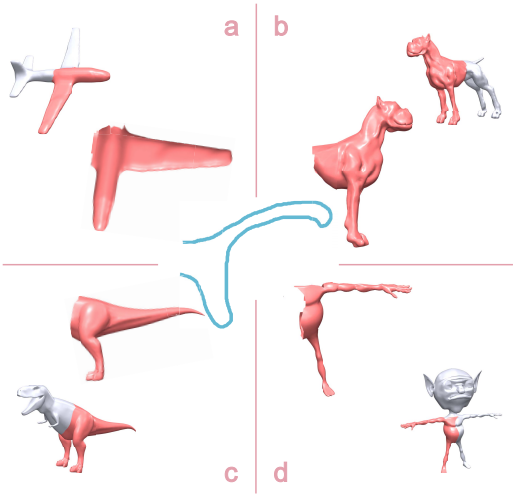


Figure 1: *Given a roughly drawn sketch, our algorithm rapidly searches a shape database to identify and extract customized parts that match the sketch. No presegmentation is required: the retrieved parts can be irregular and not match any standard segmentation.*

- A fast, sketch-based, partial 3D shape matching method based on multi-view projections. The projections are interlinked by matching fragments across different shapes, based on a novel randomized compound k-NN graph representation.
- A novel customized segmentation method based on a super-face graph. The method quickly extracts candidate parts conforming to a user's sketch from a matched shape. The extraction employs a coarse-to-fine strategy to progressively refine part boundaries down to the level of individual faces.

2. Related work

Sketch-Based Shape Retrieval. With the rapid growth of 3D shape data, fast and convenient content-based shape retrieval techniques [TV08] have become important. Content-based methods usually require a user to provide a 3D shape as a query, which introduces a circular dependency in a 3D modeling scenario. In contrast, sketch-based methods [FMK*03, SXY*11, ERB*12] allow the user to roughly draw the outline of the desired shape from one or more viewpoints in 2D. This is typically a more intuitive and convenient way to describe the user's intention.

Lee and Funkhouser [LF08] develop a sketch-based system for retrieving parts from a part database and incorporating them into a 3D model. Xie et al. [XXM*13] propose a similar system for shape editing by context-aware part replacement. These systems assume the parts are pre-generated by automatic segmentation, and hence retrieval reduces to global 2D-to-3D matching. The systems do not allow the user to generate new parts with novel cuts, or retrieve groups of pre-existing parts with one sketch. In contrast, our approach supports arbitrary cuts of exemplar shapes, driven by the sketch. Further, because shapes are not pre-segmented, we must do partial matching from 2D to 3D at interactive rates, which is a significant technical challenge.

Shape Segmentation. Segmenting shapes into meaningful parts is a fundamental problem in many computer graphics tasks and applications, and both automatic and manual approaches have been developed to tackle this problem [Sha08, CGF09]. Recently, several groups have explored methods for jointly segmenting a set of shapes [KHS10, HFL, HKG11, SvKK*11]. By considering a set of shapes as a whole, the segmentation can exploit shared structure and hence yield more coherent results. These works greatly benefit exploratory modeling systems by providing an automatically pre-segmented shape database for part suggestion and re-assembly.

In contrast to these systems, our approach does not assume a pre-segmented and pre-labeled shape database. Instead, we retrieve and segment shapes in real time based on a user sketch. Since, we cannot anticipate the exact boundary of the sketch, an on-the-fly and contour-aware segmentation method is required.

In contrast to sketch-based mesh cutting methods [FML12], we do not know the source shape in advance, and we do not sketch directly on the source shape. Instead, we perform 2D-3D partial matching to automatically retrieve database shapes matching the query sketch.

Exploratory Shape Modeling. The idea of incorporating creativity-inspiring exemplar elements into conventional conceptual modeling has been an active topic for the past few years. Lee et al. [LSK*10] examine the efficicacy of using galleries of examples for creativity support during the design process. Chaudhuri et al. [CK10] mine a 3D shape database to suggest components to creatively extend a base shape, based on geometric compatibility. In subsequent work, the authors develop a statistical model of shape semantics to improve the suggestions [CKGK11]. These works build upon the Modeling by Example system of Funkhouser et al. [FKS*04], which allowed users to query shape databases with 3D proxies for novel parts.

3. Overview

The pipeline of our algorithm is illustrated in Figure 2. It consists of two phases: an *offline phase* and an *online phase*. In the offline phase, we construct acceleration structures critical for interactive on-demand part customization. In the online phase, we extract parts in response to user sketches.

Offline Phase. First, we extract *boundary contours* [DFRS03] for each model in the database from different camera views. Then, we extract descriptors for the boundary contours at different scales (Section 5.1).

Next, we organize all boundary contours of all models into a compound k-nearest neighbors graph, for fast retrieval (Section 4).

Finally, we construct a compact representation of each database shape: the super-face graph (Section 6.1). A super-face (analogous to a superpixel [RM03]) is a group of adjacent, similar faces: factoring a shape into its super-faces yields a lower-complexity representation of the raw mesh. We use the super-face graph to quickly extract the part corresponding to a matched query contour.

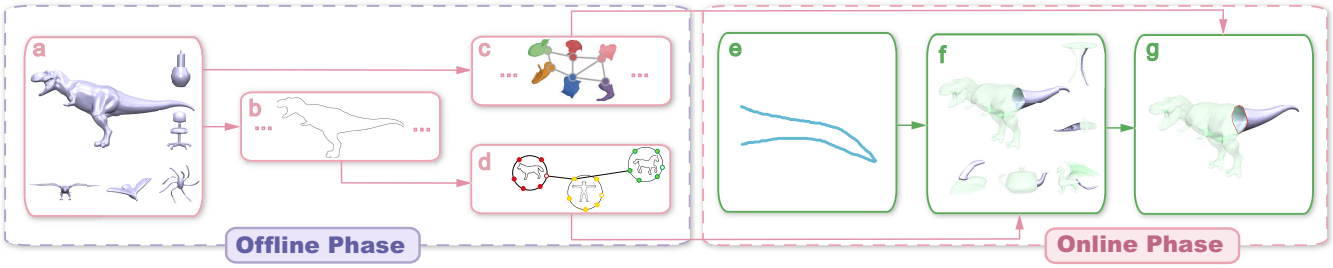


Figure 2: *Pipeline of our system. In the offline phase, for each model in a given shape database (a), we first extract its boundary contours (b). We then construct the* ***super-face graph*** *for each model (c), and organize the boundary contours of all models into a* ***randomized compound k-NN graph*** *(d). In the online phase, the user draws a rough sketch to convey his/her design intent (e). Regions of database shapes matching the sketch are rapidly identified via partial shape matching (f). A selected part is further refined to optimize its boundary and better match the sketch (g)*

Online Phase. The input sketch is used to retrieve partially matching shapes (via the k-NN graph) and identify their regions matching the sketch (via the super-face graph). The boundary of each such region is optimized by a coarse-to-fine segmentation strategy: a rough boundary is first computed at the super-face level and then refined at the raw face level. In contrast to other methods [Sha08], our segmentation method takes contour perception, concavity, and smoothness into consideration (Section 6.3).

4. Data Structure for Fast Partial Matching

At runtime, the user sketches a desired part. The system rapidly searches the database to find parts matching the sketch. We support searching for arbitrarily shaped parts, not just those matching a "standard" presegmentation. Hence, we must process each database shape on the fly, identifying which section of the shape matches the sketch. This requires an extremely fast 2D-3D partial matching method.

To compare a 2D sketch to a larger 3D shape, we have two choices: (a) to infer a 3D proxy from the sketch [IMT99] and match it to the shape, or (b) to compare the sketch to the 2D contours of the shape. We choose to do the latter for three reasons: (i) because it avoids the inherent ambiguity in inferring 3D shape from a 2D outline; (ii) because partial matching in 3D is significantly more expensive; and (iii) because the 2D sketch reflects the user's direct intent, which may be distorted when converting to 3D.

Therefore, we perform 2D partial matching between the sketch and multi-view projections of database shapes. In the offline phase, we render and store boundary contours of each database shape from a large number of camera views. The boundary contours of a database shape are extracted from its depth buffers at different camera views using a contour tracing algorithm [CTSO03]. Then, the problem reduces to matching the sketch to a section of one of these exemplar contours.

It is a significant challenge to quickly search the huge set of exemplar contours for sections that match the sketch. Existing algorithms such as partitioning trees [ML14], hashing [AI08], and k-nearest neighbor graphs [WWZ*12] are designed for global, not partial matches.

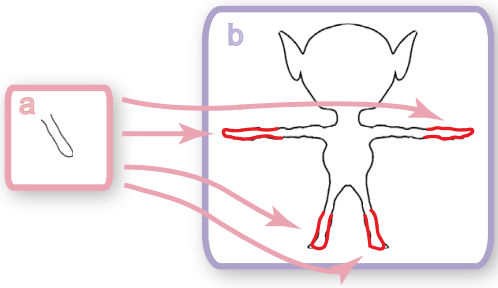


Figure 3: *A single contour may match a query contour in more than one section. The shape contour (b) matches the query contour (a) in four different sections (marked in red).*

To rapidly find database shapes whose contours match the sketch, we propose a new data structure: the *Randomized Compound k-Nearest Neighbors Graph* (RC-kNNG). The RC-kNNG has as its vertices all rendered contours of all database shapes. In a standard k-nearest neighbor graph [WWZ*12], each contour would be connected to its k most similar neighbors. This information is used to quickly retrieve new matches once an initial positive match is found.

In our partial matching scenario, however, a single contour may match a query contour in more than one section, as illustrated in Figure 3. To reflect this, the RC-kNNG allows a contour to be connected to several different sets of k neighbors. Each such set corresponds to a different matched section. Thus, from a partial match in one shape, we can quickly find several other partial matches with similar geometry in other shapes.

Construction of the RC-kNNG. Direct construction of the compound kNN graph by exhaustively comparing all possible sections of all pairs of contours is prohibitively expensive. Instead, we choose to approximate the graph with a double-random strategy:

1. We first generate n sections (we use $n = 6$ in our experiments) from each contour at each scale (Figure 4 (a)). A scale level comprises uniformly spaced points along the contour: a higher

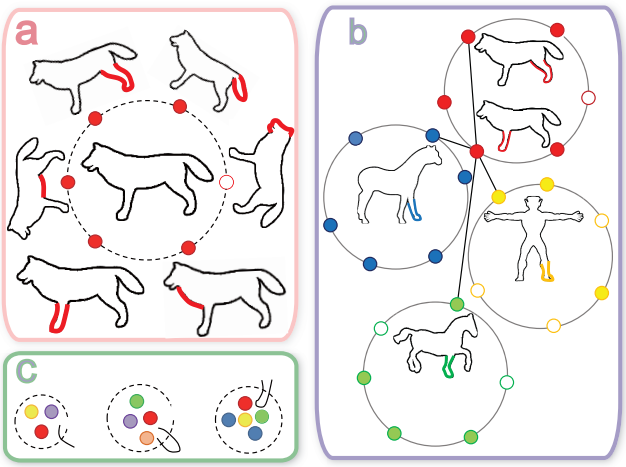


Figure 4: *Illustration of our randomized compound kNN graph. In this figure, a solid circle represents a valid section and a dashed circle represents an invalid one. Given a contour (taken as a node of our randomized compound kNN graph), we first generate several sections (a). Then, we find the nearest neighbors for each section and establish edges between the parent contours of these sections (b). Finally, we find valid sections and cluster them (c).*

level has smaller spacing and hence a larger overall number of points. Our levels place 50, 150 and 250 points along the contour respectively. The contours are sampled uniformly in a counterclockwise order.

To generate a section, we randomly choose an uncovered point with a probability p proportional to its curvature c and distance d to the nearest point covered by other sections: $p \propto c \cdot d$. The section then covers $m = 21$ consecutive points with the selected point at the center.

2. We adopt a multiple random divide-and-conquer strategy [WWZ*12] to construct a kNN graph of all the sections (Figure 4 (b)).

We collect all "valid" sections into a global list. A section is considered valid when the distance (which will be defined in Section 5.1) between it and each of its k nearest neighbors is below a threshold $\varepsilon = 0.85$. After that, we cluster the global list to get a sparse set of sections (the cluster centers) which will be used as seed points for queries (Figure 4 (c)). The clustering is based on the same distance defined in Section 5.1. The number of clusters is empirically set to 1708.

5. Candidate Shape Retrieval

In this section, we describe how the RC-kNNG is used to quickly and accurately retrieve candidate parts that match the sketch, from an unsegmented database.

5.1. Contour Descriptor

Each contour is divided into sections. The query contour, which is the sketch, has a single section. Database shape contours have several sections at each scale (Section 4), each of which may resemble the query to yield a partial match.

To compare two contour sections, we generate contour descriptors. The particular descriptor we employ is a matrix of angles [RDB10]. The descriptor is calculated from the relative spatial orientations of lines connecting sampled points on the contour.

A section of a contour has $m = 21$ sampled points on it. For each pair of points (b_i, b_j), we compute an angle metric α_{ij}:

$$\alpha_{ij} = \begin{cases} \langle \overline{b_i b_j}, \overline{b_i b_{i+\Delta}} \rangle & \text{if } i < j, \\ \langle \overline{b_i b_j}, \overline{b_i b_{i-\Delta}} \rangle & \text{if } i > j, \\ 0 & \text{if } \|i - j\| \leq \Delta \end{cases}$$

where $\Delta = 2$ is an offset parameter and $\langle \ell_1, \ell_2 \rangle$ denotes the angle between lines ℓ_1 and ℓ_2. The contour descriptor is then the angle matrix A, where $A_{ij} = \alpha_{ij}$. On each contour section, the first point is the first sample in a counterclockwise order. The distance between two contour sections S and T is the squared Euclidean distance between their angle matrices A^S and A^T:

$$D(S,T) = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \left(A^S_{ij} - A^T_{ij} \right)^2$$

5.2. Shape Retrieval

We need to identify c partially matching shapes from which candidate parts will be presented to the user (our experiments use $c = 9$). To achieve this, we query the RC-kNNG for a larger number ($21c$) of matching contour sections, and compute a score for each parent shape. The c shapes with highest scores are presented in the interface.

To generate the pool of contour sections, we first compare the query contour to all seed sections in the RC-kNNG. The results are sorted in a priority queue according to increasing descriptor distance, allowing us to traverse the graph in a best-first manner. The current best matching section is at the top of the queue. When it is popped off, its neighbors are compared to the query and added to the queue if they are not already there. We continue until we have popped and stored $21c$ matching sections.

A shape with at least one section in the matched pool is assigned the following score, which is used to rank the shapes for selecting the final candidates:

$$s = \frac{\alpha}{t} \sum_{i=1}^{t} D_i + \frac{\beta}{t},$$

where t is the number of matched sections in the shape contour, and D_i is the distance of the i^{th} matched section's descriptor from the query. We empirically set $\alpha = 0.95$, $\beta = 0.05$.

6. Progressive Part Extraction

When a user selects a candidate shape with a contour section matching the sketch, we must quickly and accurately identify and extract the corresponding part. This presents several challenges:

- A single contour may cover more than one semantic part. By "semantic part", we mean one that is typically identified by human labeling or a traditional segmentation algorithm, such as the head of an animal or the back of a chair.

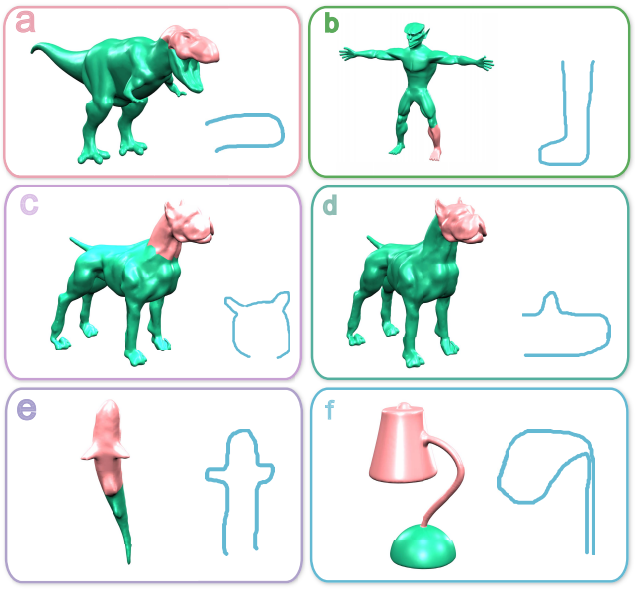


Figure 5: *Different types of parts (red) retrieved by a sketch (blue) using our algorithm. While some parts may be predefined with an automatic segmentation algorithm (d), others are irregular, defined only by the sketch (a,e), and yet others combine multiple "standard" segments (b: shin + foot, c: head + neck, f: shade + neck).*

- The contour boundary may be irregular and not directly correspond to a semantic part boundary. Hence, it can be identified only with reference to the user's sketch.
- The segmentation process should run at interactive speeds.

The first two scenarios are illustrated in Figure 5.

We address these challenges with a new *super-face graph representation* (SFG) for a 3D shape. The super-face graph is a discretized yet fine-grained version of the original model. The vertices of the graph are a set of super-faces, obtained by an oversegmentation of the model that respects strong feature boundaries. While we could also work with the raw faces of the model, their combinatorial search space can be infeasibly large. Hence, we find candidate parts as combinations of super-faces, and subsequently refine the part boundaries down to the level of individual faces (Section 6.3).

Edges of the graph connect pairs of adjacent super-faces. The weight of an edge indicates how frequently the super-face pair co-occurs in a single segment produced by a randomized segmentation algorithm (Figure 6).

The super-face graph gives us a probabilistic prior for merging adjacent superfaces into larger patches to represent an arbitrary but not implausible or disconnected part of the shape. We generate the super-face graph representations of all database models in the offline phase.

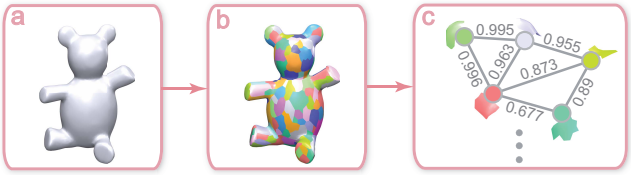


Figure 6: *The construction of the super-face graph for a 3D shape. Given the Teddy model (a), we partition it into a large number of super-faces (b). Each super-face is a node of the graph. Adjacent super-faces are connected by a weighted graph edge (c).*

6.1. Super-Face Graph Representation

To construct the super-face graph for a shape mesh, we oversegment the mesh into N patches [HKG11] (we experimented with $N = 50$, 100 and 200) and take each patch as a super-face (a vertex of the graph). Each pair of adjacent super-faces is connected by a graph edge.

To compute edge weights between nodes, we generate 900 randomized segmentations [GF08] over the shape with the number of segments varying from 2 to 10. Given a segmentation, we construct a histogram for each super-face, describing its segment memberships. Each bin of the histogram corresponds to a segment, and the value of the bin is the fractional area of the super-face lying within the segment (Figure 7). Mathematically, the bin value h_s^g of a super-face s, for a segment g, is defined as:

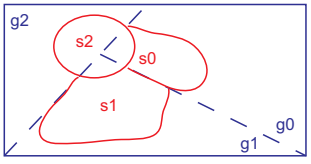
$$h_s^g = \frac{\sum_{f \in Faces(g) \cap Faces(s)} Area(f)}{\sum_{f \in Faces(s)} Area(f)},$$

where $Faces(x)$ is the set of mesh faces in x, and $Area(f)$ is the area of face f.

These histograms serve as feature vectors for determining the probability that two super-faces may lie in the same segment. We define $P(s,s')$ as the probability that two adjacent super-faces s and s' lie in the same segment, and it is defined via the χ^2 distance between their histograms H and H':

$$P(s,s') = 1 - \chi^2(H,H') = 1 - \frac{1}{2}\sum_{g_i \in G} \frac{\left(h_s^{g_i} - h_{s'}^{g_i}\right)^2}{\left(h_s^{g_i} + h_{s'}^{g_i}\right)^2},$$

where $G = \{g_i \,|\, 1 \le i \le N\}$ is the set of segments generated by the randomized segmentation, g_i is the ith segment, and N is the number of segments.



		g0	g1	g2
Histogram(s0)	=	1.0	0.0	0.0
Histogram(s1)	=	0.0	1.0	0.0
Histogram(s2)	=	0.2	0.15	0.65

Figure 7: *Distributions of super-faces on segments. s_0, s_1, and s_2 are the super-faces. g_0, g_1, and g_2 are the segments generated by the randomized segmentation on the shape.*

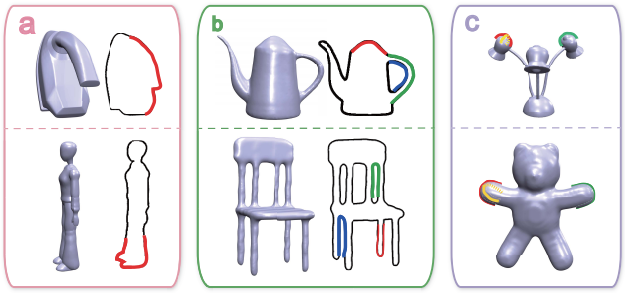


Figure 8: *Illustrations of the complexity of contours and their relationship. Depth-discontinuous contour sections are shown in (a). Topologically different contour sections are illustrated in (b). Isolated contour sections are shown in (c).*

To calculate the edge weight between two adjacent super-faces, we first accumulate the consistency scores over all the different randomized segmentations, and then normalize it using the number of randomized segmentation operations.

6.2. Fuzzy Part Identification

As a first step, we use the SFG to extract a rough approximation of the part corresponding to a matched contour section. We can convert an open section to a closed one by drawing a line between its ends. This forms a natural perceptual boundary for the projection of the desired part.

Now, we start with any super-face adjacent to the center of the contour and perform a flood fill, gathering all super-faces whose projections lie within the closed contour obtained above.

We must handle several complications here. First, the contour section may be depth-discontinuous, spanning multiple isolated parts in 3D (Figure 8(a)). Second, the contour section may enclose empty space that is not accounted for in the sketch (the blue and green sections in Figure 8(b)). Third, the same sketch may match multiple instances of the same part (Figure 8(c)). To address these issues, we filter out invalid parts and collapse multiple instances into a single part. The invalid parts generated with depth-discontinuous contour sections are identified by a group of isolated super-faces. Such parts are discarded. Multiple instances of the same part (Figure 8(c)) are identified by matching the sets of super-faces. Multiple instances of the same parts (Figure 8(c)) are identified, if the two sets of super-faces are equal. Such repetitions are also discarded.

6.3. Coarse-to-Fine Boundary Refinement

Once an approximate part is identified, we segment it from the model and refine its boundary to better match the sketch and respect local geometric cues. We take the following constraints into consideration:

- **Contour closure.** As noted above, the line joining the ends of a contour section forms a natural perceptual boundary for the part. Hence, the part's projection should respect this line as much as possible.
- **Super-face co-occurrence.** If two super-faces regularly co-occur in the same segment of a random segmentation, they should probably both be retained or both excluded from the part. This acts as a shape prior.
- **Concavity.** Shape concavity information plays an important role in achieving high-quality segmentation as concave creases and seams are generally regarded as natural segmentation boundaries by humans [AZC*12].
- **Smoothness.** The boundary should avoid sharp zigzags.

We propose a novel coarse-to-fine strategy that optimizes the fuzzy part boundary in two steps: (1) optimize the set of boundary superfaces according to perceptual and co-occurrence priors; and (2) further refine the boundary at the level of individual faces, using the remaining constraints.

6.3.1. Coarse Level Extraction

We formulate the coarse level part extraction as a binary labeling problem which can be solved by minimizing a Gibbs energy E_c:

$$E_c = \sum_{i \in V} E_1(l_i) + \sum_{(i,j) \in E} E_2(l_i, l_j),$$

where V and E represent the nodes and edges of the super face graph $\mathcal{G}$, respectively, $E_1(l_i)$ is the likelihood energy encoding the cost when the label of node i is l_i, and $E_2(l_i, l_j)$ is the prior energy denoting the cost when the labels of adjacent nodes i and j are l_i and l_j, respectively. E_1 is defined as follows:

$$E_1(l_i) = \begin{cases} -\ln \mathcal{P}(i), & l_i = 1, \\ -\ln(1 - \mathcal{P}(i)), & l_i = 0, \end{cases}$$

where $\mathcal{P}(i)$ is the probability of assigning label 1 to node i. According to the *contour closure* constraint, $\mathcal{P}(i)$ is defined as the fractional area of the ith super-face lying within the contour closure:

$$\mathcal{P}(i) = \frac{Area^{in}_{proj}(i)}{Area_{proj}(i)},$$

where $Area^{in}_{proj}(i)$ is the area of the projection of the ith super-face overlapping with the contour closure, $Area_{proj}(i)$ is the area of the projection of the ith super-face. E_2 is defined according to the *super-face co-occurrence* constraint:

$$E_2(l_i, l_j) = \begin{cases} 0, & l_i = l_j, \\ e_{ij}, & l_i \neq l_j, \end{cases}$$

where e_{ij} is the edge weight of the SFG. We solve E_c using binary graph cut [BK04].

6.3.2. Fine Level Extraction

After obtaining the coarse level part, we map each boundary super face of $\mathcal{G}$ onto a vertex in the original surface that is closest to the center of the super face. For an edge e in $\mathcal{G}$, let v_1 and v_2 be the projections of its nodes on the original surface. We propagate from v_1 toward v_2 by iteratively finding the neighbour whose projection on the line v_1v_2 is closest to v_2.

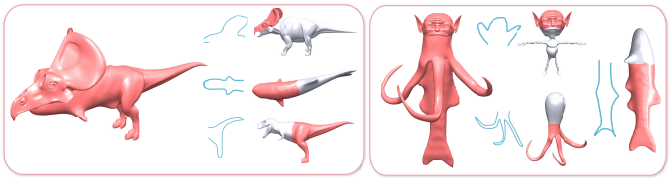


Figure 9: *Examples of sketch-driven assembly-based modeling. In each subfigure, we show the designed models (red), the user's sketches (blue), and the part suggestions selected by the user.*

We then refine the obtained boundary in an iterative manner to minimize the following energy:

$$E_f = E_v + E_s,$$

where E_v is the concavity energy, and E_s is the smoothness energy. E_v is defined as the sum of all boundary edge's concavity energy [KT03]:

$$E_v = \sum_{e \in \partial Q} \eta(1+\cos\alpha_e)|e|,$$

where e represents an edge on the boundary of the candidate part Q, $|\cdot|$ represents the length of e, α_e is the dihedral angle of e, $\eta = 0.1$ when e is concave, otherwise $\eta = 1.0$. E_s is defined as:

$$E_s = \sum_{v \in \partial Q} |\sin\langle e_l, e_r\rangle|,$$

where v is a vertex on the boundary Q, e_l is an edge on Q pointing to v, and e_r is another edge on Q starting from v. We apply snake operations on the boundary vertices to minimize the energy E_f [JLCW06].

7. Applications

Our on-demand sketch-based segmentation algorithm is a general technique that can drive various modeling applications. Here we describe some potential applications. Some of these applications directly use 2D contour queries (sketches). Other applications use our algorithm as an efficient partial matching method, where a 3D query is represented by an informative 2D contour.

Sketch-driven assembly-based modeling. A direct application of our technique is sketch based part composition [LF08]. As our method does not rely on a pre-segmented database, it can retrieve more diverse parts for composition. Our method can also improve the expressiveness of such a system, as the user can draw complex sketches while still retrieving correctly matching parts. As shown in Figure 9, our method can create detailed creative shapes.

Contour-driven shape completion. Our algorithm can be used to suggest possible completions for occluded objects in images, an important application in computer vision and image editing. Given a photograph with an occluded object, we manually or automatically trace the boundary of the visible portion and use this trace as input to our algorithm, which retrieves possible completions of the shape from the database. Here, 2D-3D partial matching directly helps us infer the occluded portion. The process is illustrated in Figure 10. Alternatively, instead of starting with a photo, the user may directly draw a novel sketch with some occluded components, for example when constructing a complex 3D scene by sketching [XCF*13]. Here, too, our algorithm can be used to help recover hidden geometry.

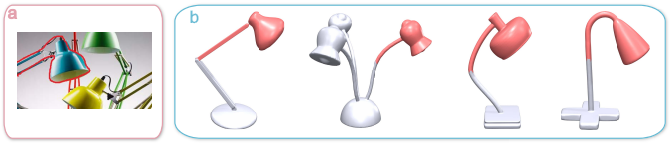


Figure 10: *Example of contour-driven shape completion. Given a photograph of an occluded lamp, the user traces the outer boundary of the visible portion. This trace is used as the query contour. Our algorithm retrieves possible completions for the object from a database of lamps. (Photo: Xavier Young)*

Figure 11: *Example of image-driven part-based modeling. Given the image (on the top left corner), the user sketches the contours of the parts of the shape. Our method extract parts from the database. The user selects the parts and assembles the 3D shape (middle) in the image. The body part of the generated shape is segmented from an existing horse shape.*

Image-driven part-based modeling. Our algorithm can be used to assemble 3D shapes from images. Given an image of an object, the user sketches contours of parts of the object. Our method retrieves candidate parts from the database according to the user's sketch. The user selects appropriate parts and assembles the 3D shape to match the image. Figure 11 shows an example.

Shape variation. Our pipeline can be used to generate shape variations by simple adaption, as shown in Figure 12. Given a segmented shape, we can use any part of the shape as a query to retrieve similar parts from the shape database. The chosen retrieved part can then be used to replace the original part to create shape variations. To extract a query contour from the source part, to be used as the 2D search key in our algorithm, we first perform Principal Component Analysis (PCA) on the part. Then, we take the view perpendicular

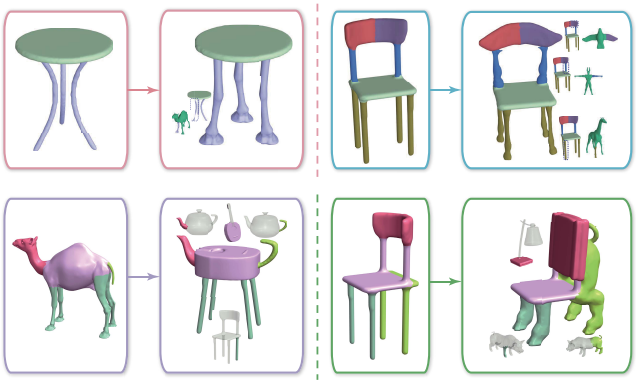


Figure 12: *Examples of shape variations created with our method. Given a segmented model (left), we can use one part as the query to retrieve similar parts from the database. The selected retrieved part is composited with the non-query parts to create a new shape (right) with the same layout as the original shape.*

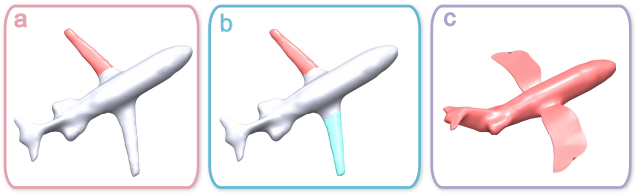


Figure 13: *Example of symmetry-aware selection and editing. The user selects an arbitrary region of a shape (a). Note that the region is not the complete wing, i.e. it is not a "standard" segment. We use our algorithm as a fast partial matching technique, to find other parts in the same shape with the same contour. This yields the corresponding region of the other wing as a symmetric counterpart (b). Finally, the symmetric pair may be replaced by another, retrieved from a database using our "shape variations" approach (c).*

to the plane containing the first and second principal directions of the part to calculate the query contour.

Symmetry-aware selection and editing. Our method can also be used to select a group of similar elements in a single shape for further editing. For example, in Fig 13, the user picks an arbitrary region of the leg of a table. The corresponding regions of the other legs will be automatically identified, by partial matching of the contour of the selected region with the rest of the shape itself. Subsequently, the entire selected group can be replaced by similar parts retrieved from the database, as in the "shape variations" applicaton.

Part suggestion. Our on-the-fly part extraction technique can be used to suggest parts to extend a base shape [CK10, CKGK11]. In contrast to previous methods, we do not require a pre-segmented database. Given a query shape, we treat its outline as the query contour and feed it to the algorithm. Our method matches the contour to parts of exemplar shapes, and suggests maximal connected components of the *rest* of the exemplar as suggestions for creatively

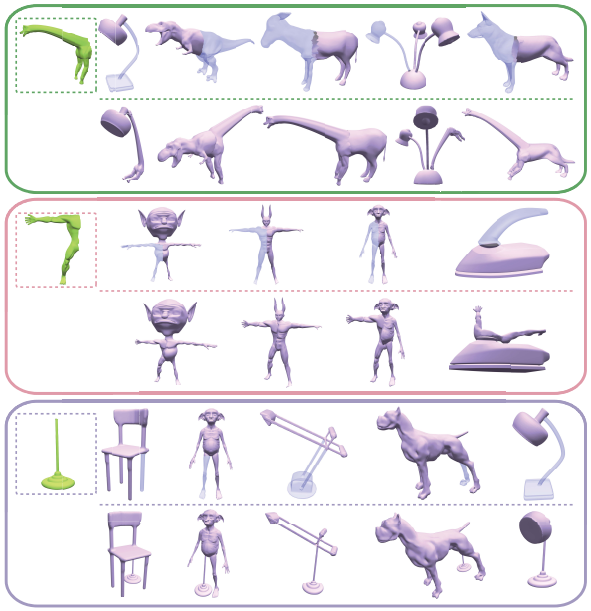


Figure 14: *Examples of part suggestions. Given the query shapes (in green), our system presents a list of part suggestions (in purple). The assembled shapes are shown below the suggested parts.*

extending the shape. The suggested parts can be irregular ones or from different shape families (see Figure 14).

Multi-scale part suggestion. Our system can suggest parts at various scales. Given a part retrieved by the method above, we perform normalized cuts [GF08] on the SFG corresponding to the retrieved part to generate T_n segments, each of which consists of a group of adjacent super-faces. T_n is defined as:

$$T_n = \frac{2Vol(P_r)}{Vol(P_p)},$$

where $Vol(\cdot)$ is the volume of an object, p_r is the retrieved part, p_p is the part in the database model corresponding to the query ("corresponding part"). We constrain T_n to lie between 1 and 7. The segments are sorted by increasing Euclidean distance of their centers from the center of the corresponding part. Given a scale parameter S, we return the union of the first S segments in the sorted order as the suggested part. With increasing S, larger and larger parts adjacent to the corresponding part are returned. An example is given in Figure 15.

8. Evaluation

We have validated our prototype system, written in C++, on a 64-bit desktop machine with a 3.5 GHz Intel Core I7-3770K processor, 8GB memory, and an Nvidia GeForce GTX 660 GPU video card. Figure 16 shows retrieval results generated with our method.

There are 513 3D shapes in our database. From these shapes, we extracted 10,773 contours. It took ~3.5 hours to organize these contours into an RC-kNNG. The average time to construct the

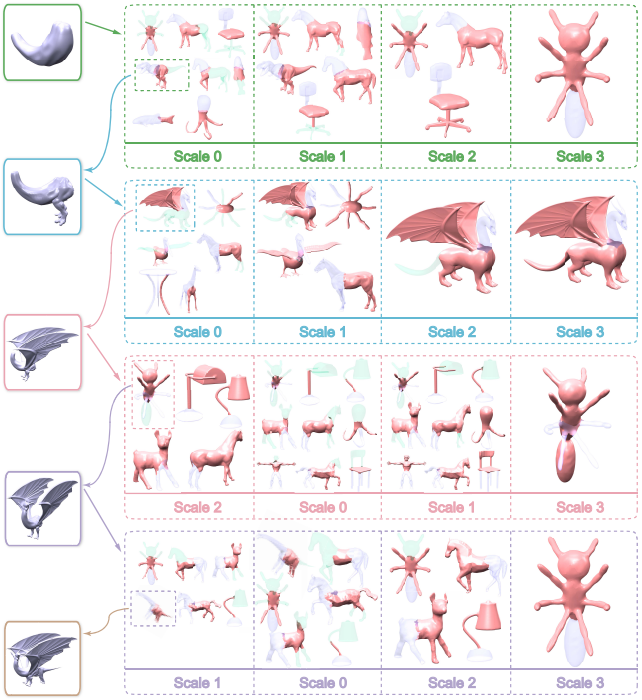


Figure 15: *An example of multi-scale part suggestion. The initial shape and subsequent working models, used as queries, are listed in the left column. On the right, we show the suggested parts (red) at different scales for each modeling step. The parts corresponding to the queries are marked in light blue. The other parts are marked in light green.*

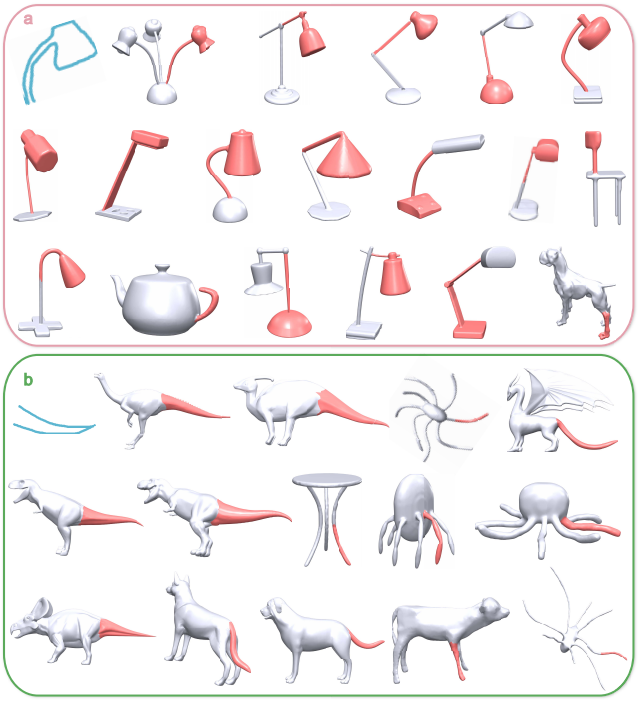


Figure 16: *Retrieval results generated with our method. The retrieved parts are ranked from top to bottom, from left to right.*

super-face graph of a shape is 20 seconds. Because of the super-face graph representation, our part extraction is real-time. The time for the coarse and fine level extractions for one model is 0.15ms and 10.3ms on average, respectively. Thanks to the two acceleration structures (RC-kNNG and SFG) and the coarse-to-fine part extraction strategy, our system can run interactively during the design process. Once a user finishes drawing the sketch, it takes 0.56-0.61 seconds (with GPU and multi-core acceleration) on average for our system to present part suggestions.

RC-kNNG Retrieval Performance. To evaluate the performance of our RC-kNNG, we compare the following methods:

1. Traditional (single-layer) kNNG constructed by brute force.
2. Wang's randomized kNNG approximation [WWZ*12].
3. RC-kNNG constructed by brute force.
4. RC-kNNG with Wang's randomized approximation (**our method**).

For the two kNN graph methods (1 and 2), nodes of the graph are complete shape contours. Node adjacencies and edge weights are identified by global comparison of shape contours. To compute the descriptor of the shape contour, the latter is first sampled at 3 different scales. The contour descriptor is then computed for the sampled points. When searching for partial matches to a query contour, we must extract similar contour sections explicitly, significantly slowing down both methods.

For the two RC-kNNG methods (3 and 4), nodes of the graph are contour sections (Section 4). In our evaluations, we set $k = 20$ and extract 6 sections from each shape contour.

The retrieval performance for each method is shown in Table 1. It is clear that RC-kNNG with Wang's method, as proposed in this paper, has the best performance. For retrieval, it is about 97 times faster than Wang's kNNG, with only 3 times the construction time. The kNNG approaches, which require explicit partial matching at runtime, are much slower. We quantitatively evaluate the quality of the extracted parts by comparing the contour of the extracted parts with the user's sketch in terms of the contour descriptor introduced in Section 5.1. We visualize the retrieved parts in Figure 17. The precision recall curves for the retrieval methods are shown in Figure 18(a). It is clear that the two RC-kNNG methods perform better than the two kNNG methods. The reason is that the kNNG methods, comparing shape contours in a global manner, may match obviously dissimilar contour sections together.

Increasing Super-Face Counts. In Table 2, we show the timing statistics for different super-face counts. The coarse-level part extraction time is strongly dependent on the number of super-faces. However, this is a relatively fast step so this dependence does not significantly hurt overall performance. Overall, despite quadrupling the number of super-faces, part extraction remains interactive, completing in well under 1s.

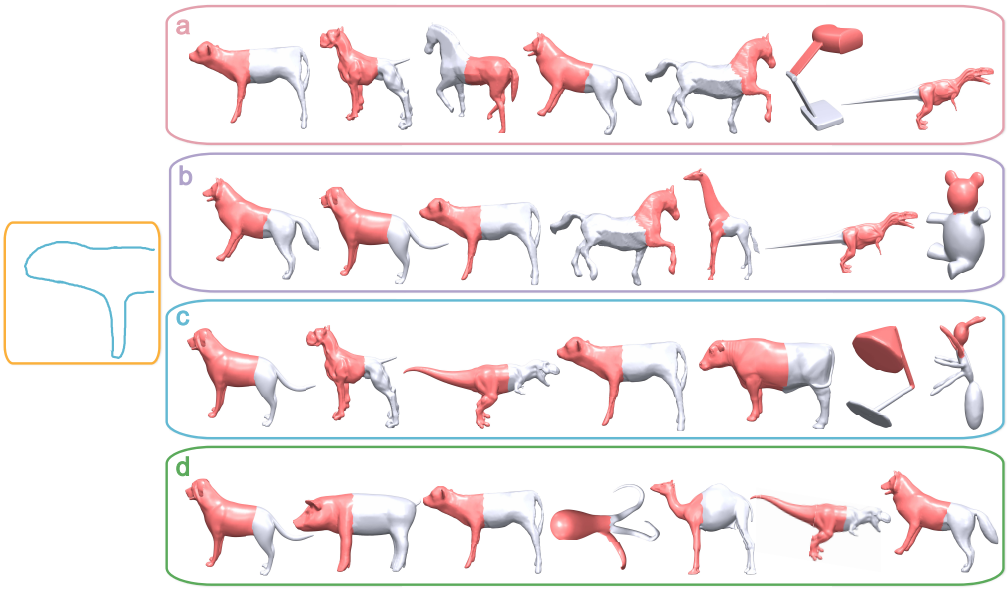


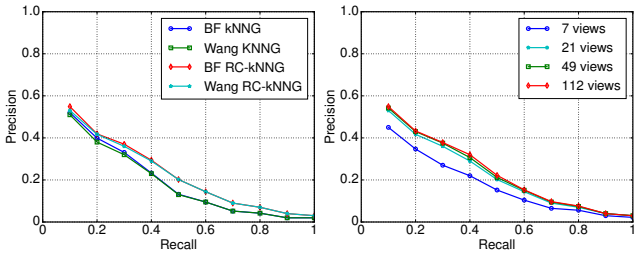
Figure 17: *Retrieved parts generated with different methods: the brute force kNNG method (a), the Wang's randomized kNNG approximation method (b), the RC-kNNG with the brute force method (c), and the RC-kNNG with Wang's method (d).*

Performance / Algorithm	RT	CT	ME
BF *k*NNG	4.54s	51.7h	0.0799
Wang *k*NNG	5.68s	1.15h	0.081
BF RC-*k*NNG	0.053s	33.89h	0.07
Wang RC-*k*NNG	0.058s	3.52h	0.07

Table 1: *Retrieval performance for different methods on our test dataset. RT is the retrieval time for a query contour. CT is the construction time for the data structures. ME is the matching error for the retrieved parts. BF KNNG represents the Brute force kNNG method. Wang kNNG represents the Wang's kNNG approximation method. BF RC-kNNG represents the RC-kNNG with the brute force method. Wang RC-kNNG represents the RC-kNNG with Wang's method.*

In Figure 19, we show the qualitative improvement in part extraction as we increase the number of super-faces. With fewer super-faces, the search space is relatively coarse and the extracted parts need not perfectly match the sketch. As we increase the number of super-faces, the search space becomes more fine-grained. The part contours match the sketch better and better, and the part boundaries are less constrained to follow suboptimal cuts. Table 3 shows the matching error statistics for different super-face counts. Note that the error decreases substantially as the shape representation becomes more fine-grained.

Comparison to pre-segmentation approach. We compare our method to an approach based on a pre-segmented database of



(a) Different retrieval methods (b) Different camera view settings

Figure 18: *Averaged precision recall curves. (a) shows the precision recall curves generated with different retrieval methods on a set of 35 test sketches. (b) shows the precision recall curves generated with our method under different camera view settings. In (a), BF kNNG represents the Brute force kNNG method; Wang kNNG represents the Wang's kNNG approximation method; BF RC-kNNG represents represents the RC-kNNG with the brute force method; Wang RC-kNNG represents the RC-kNNG with Wang's method (our method).*

SF Count / Step	50	100	200
Construction of SFG	20s	14.01s	19.07s
Coarse extraction (per shape)	0.15ms	0.77ms	3.91ms
Full part extraction	560ms	577ms	605ms

Table 2: *Timing statistics with different super-face counts. Full part extraction measures the time from launch of a query to generation of a final part.*

SF Count	50	100	200
Matching error	0.45	0.29	0.07

Table 3: *Matching error statistics with different super-face counts, averaged over a set of 20 test sketches.*

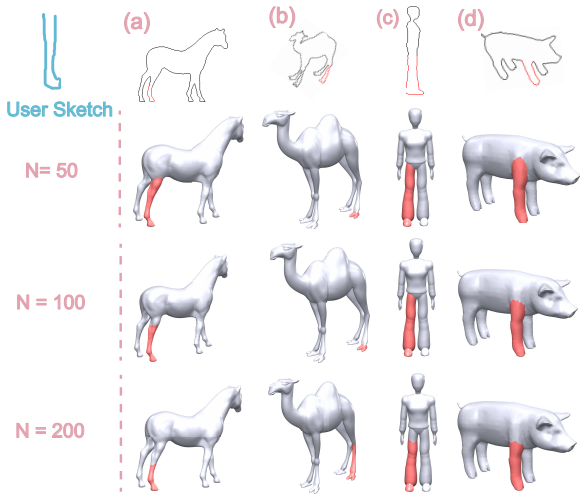


Figure 19: *Improvement in part quality with increasing super-face count (N). With more super-faces, the extracted part fits the user's sketch better and better.*

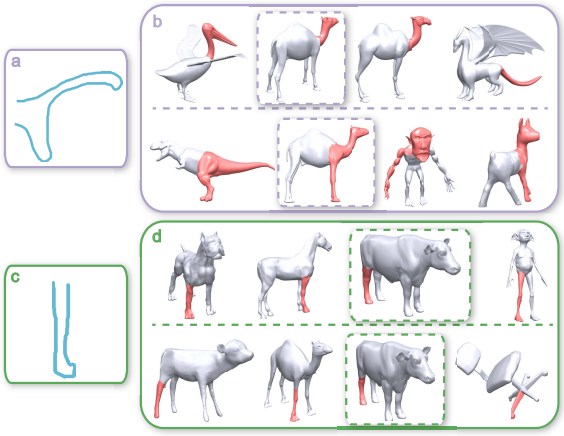


Figure 20: *Retrieval results generated with our method and the pre-segmentation method. Given the user's sketch, we show the retrieval results generated with the two methods. In (b) and (d), results generated with the pre-segmentation method are on the upper row; results generated with our method are on the lower row. The pre-segmentation method retrieves the predefined parts (the head of the camel model in (b) and the leg of the cow model in (d)), which are most similar to the user's sketch. Our method extracts parts (the "head+leg" part of the camel model in (b) and the lower leg of the cow model in (d)) on-the-fly according to the user's sketch.*

parts. The pre-segmentation method proceeds as follows: 1) Each database model is pre-segmented into regular (typical semantic) parts. For example, a human model is decomposed into four parts: head, torso, arms, and legs. 2) We extract boundary contours for each part under different camera views. 3) We construct the *k*NN graph for the part contours. Each node in the *k*NN graph represents the whole-part contour. The edge is established by global matching between part contours. 4) In the runtime stage, given the user's sketch, candidate parts are retrieved through a similar procedure as our method with the only difference that contour matching is performed in a global manner. The matching error for the pre-segmentation method is 0.624 averaged over the set of sketches used to generate Table 3. Apart from the higher matching error, visual examination also suggests that our method produces better results than the pre-segmentation method. We visualize the retrieval results generated with our method and the pre-segmentation method in Figure 20.

Camera view settings. We evaluate the influence of the different camera view settings under which we extract contours for the database models. Figure 18(b) shows the precision recall curve generated with different camera view settings. The 7 views include 3 canonical side views and 4 corner views. The 21 views include 3 canonical side views, 4 corner views, and 14 uniformly sampled views [FWX*13]. The 49 views include 3 canonical side views, 4 corner views, and 42 uniformly sampled views. The 112 views include 3 canonical side views, 4 corner views, and 105 uniformly sampled views. To balance between efficiency and effectiveness, we use 21 views for all the experiments and applications.

9. Conclusion

We have introduced a sketch-based customized part extraction algorithm for 3D shape modeling. Our approach queries a database in real time and retrieves parts matching the user's sketch. In contrast to previous methods, our approach does not rely on a pre-segmented database. Instead, it generates customized parts on-the-fly to accurately match the sketch, thus significantly enriching the design space. Candidate parts are identified and segmented by a fast 2D-to-3D partial matching technique. Our algorithm enables several applications.

Limitations. In our current implementation, we assume that the models in the database are manifold. Also, the contour descriptor we adopt is not scale-invariant.

Future work. We plan to develop more powerful scale-invariant contour descriptors. Since there have been several improvements to the angle descriptor [ML11], we would like to incorporate these more powerful descriptors in our framework. In addition, it would be nice to incorporate cues from surrounding context to help disambiguate sketches. Local symmetry/similarity matching could benefit from the context-based search. The view under which the sketch is drawn is important: some views are more discriminative than others. Helping users to draw optimal views and/or helping them interactively resolve ambiguous sketches, is an important research direction. The pose of the partial shape used for computing the

boundary contours is important for our method. If a suboptimal view is chosen, the method can return inappropriate candidate parts. It would be interesting to explore the problem of best view selection in the future. We plan to further investigate alternative formulations of the partial matching problem, together with associated metrics and benchmarks. It is also of interest to test our method on a much larger database.

Acknowledgments

We would like to thank Yutong Wang, Debing Zhang, Xuan Cheng, Tian Qiu, Kuan Wang, Yu Huang, Wanxuan Sun, and the reviewers for their constructive comments. Xiaogang Jin was supported by the National Natural Science Foundation of China (No. 61472351, and 61272298). Juncong Lin was supported by the National Natural Science Foundation of China (No. 61202142), and the National Key Technology R&D Program Foundation of China (No. 2015BAH16F00/F02). Kai Xu was supported by the National Natural Science Foundation of China (No. 61572507, and 61532003). Siddhartha Chaudhuri was partially supported by a gift from Adobe Systems.

References

[AI08] ANDONI A., INDYK P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM 51*, 1 (2008). 3

[AZC*12] AU O.-C., ZHENG Y., CHEN M., XU P., TAI C.-L.: Mesh segmentation with concavity-aware fields. *IEEE TVCG 18*, 7 (2012), 1125–1134. 6

[BK04] BOYKOV Y., KOLMOGOROV V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *TMAPI 26*, 9 (2004), 1124–1137. 6

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *ACM TOG 28*, 3 (2009). 2

[CK10] CHAUDHURI S., KOLTUN V.: Data-driven suggestions for creativity support in 3D modeling. *ACM TOG 29*, 6 (2010). 2, 8

[CKGK11] CHAUDHURI S., KALOGERAKIS E., GUIBAS L., KOLTUN V.: Probabilistic reasoning for assembly-based 3D modeling. *ACM TOG 30*, 4 (2011). 2, 8

[CTSO03] CHEN D.-Y., TIAN X.-P., SHEN Y.-T., OUHYOUNG M.: On visual similarity based 3D model retrieval. *Comp. Graph. For. 22*, 3 (2003), 223–232. 3

[Dec88] DECKER C. A.: *Children: The Early Years*. Goodheart-Willcox, 1988. 1

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *ACM TOG 22*, 3 (2003), 848–855. 2

[ERB*12] EITZ M., RICHTER R., BOUBEKEUR T., HILDEBRAND K., ALEXA M.: Sketch-based shape retrieval. *ACM TOG 31*, 4 (2012). 1, 2

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM TOG 23*, 3 (2004), 652–663. 2

[FMK*03] FUNKHOUSER T., MIN P., KAZHDAN M., CHEN J., HALDERMAN A., DOBKIN D., JACOBS D.: A search engine for 3d models. *ACM TOG 22*, 1 (2003), 83–105. 2

[FML12] FAN L., MENG M., LIU L.: Sketch-based mesh cutting: a comparative study. *Graphical Models 74*, 6 (2012), 292–301. 2

[FWX*13] FAN L., WANG R., XU L., DENG J., LIU L.: Modeling by drawing with shadow guidance. *Comp. Graph. For. 32*, 7 (2013), 157–166. 11

[GF08] GOLOVINSKIY A., FUNKHOUSER T.: Randomized cuts for 3D mesh analysis. *ACM TOG 27*, 5 (2008). 5, 8

[HFL] HU R., FAN L., LIU L.: Co-segmentation of 3D shapes via subspace clustering. *Comp. Graph. For. 31*, 5, 1703–1713. 2

[HKG11] HUANG Q., KOLTUN V., GUIBAS L.: Joint shape segmentation with linear programming. *ACM TOG 30*, 6 (2011). 2, 5

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3D freeform design. In *Proc. SIGGRAPH* (1999), pp. 409–416. 3

[JLCW06] JI Z., LIU L., CHEN Z., WANG G.: Easy mesh cutting. *Comp. Graph. For. 25*, 3 (2006). 7

[KHS10] KALOGERAKIS E., HERTZMANN A., SINGH K.: Learning 3D mesh segmentation and labeling. *ACM TOG 29*, 4 (2010). 2

[KT03] KATZ S., TAL A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM TOG 22*, 3 (2003). 7

[LF08] LEE J., FUNKHOUSER T.: Sketch-based search and composition of 3D models. In *Proc. SBIM* (2008), pp. 97–104. 1, 2, 7

[LSK*10] LEE B., SRIVASTAVA S., KUMAR R., BRAFMAN R., KLEMMER S. R.: Designing with interactive example galleries. In *Proc. CHI* (2010). 2

[ML11] MA T., LATECKI L.: From partial shape matching through local deformation to robust global shape similarity for object detection. In *Proc. CVPR* (2011), pp. 1441–1448. 11

[ML14] MUJA M., LOWE D.: Scalable nearest neighbor algorithms for high dimensional data. *TPAMI 36*, 11 (2014). 3

[OSSJ09] OLSEN L., SAMAVATI F. F., SOUSA M. C., JORGE J. A.: Sketch-based modeling: a survey. *Computers & Graphics 33*, 1 (2009), 85–103. 1

[RDB10] RIEMENSCHNEIDER H., DONOSER M., BISCHOF H.: Using partial edge contour matches for efficient object category localization. In *Proc. ECCV* (2010). 4

[RM03] REN X., MALIK J.: Learning a classification model for segmentation. In *Proceedings of the Ninth IEEE International Conference on Computer Vision* (Washington, DC, USA, 2003), ICCV '03, IEEE Computer Society, pp. 10–17 vol.1. 2

[Sha08] SHAMIR A.: A survey on mesh segmentation techniques. *Comp. Graph. For. 27*, 6 (2008), 1539–1556. 2, 3

[SvKK*11] SIDI O., VAN KAICK O., KLEIMAN Y., ZHANG H., COHEN-OR D.: Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM TOG 30*, 6 (2011). 2

[SXY*11] SHAO T., XU W., YIN K., WANG J., ZHOU K., GUO B.: Discriminative sketch-based 3D model retrieval via robust shape matching. *Comp. Graph. For. 30*, 7 (2011), 2011–2020. 2

[TV08] TANGELDER J. W., VELTKAMP R. C.: A survey of content based 3D shape retrieval methods. *Multimedia Tools and Appl. 39*, 3 (2008), 441–471. 2

[WWZ*12] WANG J., WANG J., ZENG G., TU Z., GAN R., LI S.: Scalable k-NN graph construction for visual descriptors. In *Proc. CVPR* (2012). 3, 4, 9

[XCF*13] XU K., CHEN K., FU H., SUN W.-L., HU S.-M.: Sketch2Scene: sketch-based co-retrieval and co-placement of 3D models. *ACM TOG 32*, 4 (2013). 7

[XXM*13] XIE X., XU K., MITRA N. J., COHEN-OR D., GONG W., SU Q., CHEN B.: Sketch-to-Design: Context-based part assembly. *Comp. Graph. For. 32*, 8 (2013), 233–245. 1, 2