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

Volume 35 (2016), Number 5

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

Submission ID: paper1050

Abstract

Several applications in shape modeling and exploration require identification and extraction of a shape part matching a 2D sketch. We present CustomCut, an on-demand part extraction method which, given a sketched query, 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 assembly-based modeling, our approach can extract arbitrary parts from input shapes and does not rely on a prior segmentation into predefined 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.

1. Introduction

Sketch-based interfaces provide an intuitive way for both expert and novice users to create and explore three-dimensional shapes [LIAL07]. 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. Typically, this is done in a data-driven manner, using the sketch to search a repository for a matching high-quality shape, which is then incorporated wholly or partly into the user's design [ERB*12a].

A fundamental algorithmic component in such interfaces is the following step: given a 2D query sketch, identify matching database shapes and extract the matched regions to yield customized parts for incorporation in the current design.

In this paper, we present CustomCut, a new end-to-end technique for sketch-based customized part extraction. In contrast to previous assembly-based methods [LF08,XXM*13], we require no prior segmentation of the database shapes into predefined parts. Instead, we perform *on-the-fly* segmentation of retrieved database shapes, to perfectly fit the user intent expressed in the sketched stroke. This significantly expands the design space available to the user and enhances 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 observe that two 3D shapes are potentially similar if their 2D projections are similar. Therefore, given an initial match, we can use multi-view 2D shape matching to quickly retrieve additional matches.

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

- A fast, sketch-based, partial 3D shape matching method based on multi-view projections. The projections are linked by matching sections across different shapes in a novel randomized compound *k*NN graph representation.
- A novel on-the-fly segmentation method employing a super-face graph. The method quickly extracts candidate parts for 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

(SID SAYS: This section will be revised to make it less about modeling and more about sketching, segmentation etc.)

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 efficacy of using galleries of examples for creativity support during the design process. Chaudhuri et

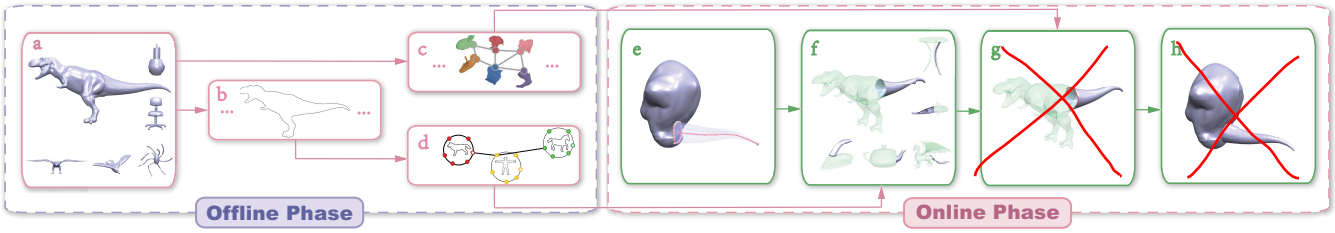


Figure 1: *Pipeline of our system. In the offline phase, for each model in a given shape database (a), we first extract its boundary contours (b). We then construct the super-face graph for each model (c), and organize the boundary contours of all models into a randomized compound kNN graph (d). In the online phase, the user first constructs a 3D proxy (light blue part in the working model) via several sketches (red) to convey his/her design intent (e). Parts matching the proxy are generated on-the-fly via a partial shape matching technique (f), and shown as an ordered list of candidate parts (light blue). After the user chooses a desired part, the part is segmented out of the 3D shape (g) and assembled into the model being constructed (h).*

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 models 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.

In parallel work, researchers have worked on automating the shape generation process, so the user directly explores a space of complete shapes. Talton et al. [TGY*09] describe a data-driven approach for exploring a high-dimensional space of parametric shapes. Xu et al. [XZCOC12] present a fit-and-diverse method for evolving a set of assembled shapes to match user intent. Kalogerakis et al. [KCKK12], Averkiou et al. [AKZM14] and others describe generative probabilistic priors for shape synthesis.

Our approach lies in the category of part-suggestion methods. However, the major technical contribution of our work is that we do not rely on a pre-segmented (or pre-labeled) database, and can generate arbitrarily shaped segments to perfectly match user sketches. Hence, our approach allows users to efficiently query a much larger design space. Our work also stands in contrast to Modeling by Example, since we construct proxies by 2D sketching, extract parts automatically, and do not require exemplar shapes to be co-aligned.

Semantic 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 methods, our approach does not assume a pre-segmented and pre-labeled shape database. Instead, we segment shapes in real time based on a user sketch. Hence, the segmentation process must be performed at fast, interactive speeds. Further, the extracted segment should match the sketch, whose exact boundary we cannot anticipate. Thus, an on-the-fly and contour-aware segmentation method is required.

Sketch-Based Shape Retrieval. With the rapid growth of 3D shape data, fast and convenient content-based shape retrieval techniques [TV08] 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*12b] 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.

3. Overview

The pipeline of our algorithm is illustrated in Figure 1. It consists of two phases: an *offline phase* and a *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 5.2).

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 is a group of adjacent, similar faces: factoring a shape into its super-faces yields a lower-complexity representation than the raw mesh. We use the super-face graph to quickly extract the part corresponding to a matched query contour.

Online Phase. At runtime, the user roughly sketches a part that could augment a base shape. We interpret the sketch with a 3D proxy, which is used to generate additional contours (from different viewpoints) to guide retrieval. First, this multi-view representation is used to query the RC-kNNG for partially matching shapes. Second, the views are used to identify groups of super-faces in each shape that constitute candidate parts. Each part boundary is refined, for compatibility with the base shape, 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 draws a contour specifying a proxy for a new part. The system rapidly searches the database to find parts matching the proxy. 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 proxy. This requires an extremely fast partial matching method.

If two 3D shapes are similar, they look similar from all viewing angles [CTSO03]. We find that such an observation also holds for a query shape and a *part* of a target object. Therefore, to match a proxy to a portion of a database shape, we can compare their 2D contours. If we use a sketched silhouette from the user and a contour from each database model for partial matching, some parts similar to the user's sketch will be lost; If we perform partial matching between a 3D proxy and portions of 3D database models, it is time-consuming for interactive response. Therefore, we employ a 2D contour partial matching scheme between a contour of the proxy and the contours from multi-view projections of database models to solve our problem. In the offline phase, we render and store boundary contours of each database shape from a large number of camera views. Then, the problem reduces to matching the proxy contour 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 proxy contour. Existing algorithms such as partitioning trees [ML14], hashing [AI08], and k-nearest neighbor graphs [WWZ*12] are designed for global, not partial matches.

To rapidly find database shapes whose contours match the proxy, 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 2. 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 3 (a)). A scale level comprises uniformly spaced points along the contour: a higher level has smaller spacing and hence a larger overall number of points. Our levels place 50, 150 and 250 points along the contour respectively.
   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 $m = 20$ 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 3 (b)).

We collect all "valid" sections into a global list. A section is considered valid when the distance (which will be defined in Section 5.2) between it and each of its k nearest neighbors is below a threshold $\varepsilon = 0.85$. After that, we cluster the list to get a sparse set of sections (the cluster centers) which will be used as seed points for queries (Figure 3 (c)). 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 sketched proxy, from an unsegmented database.

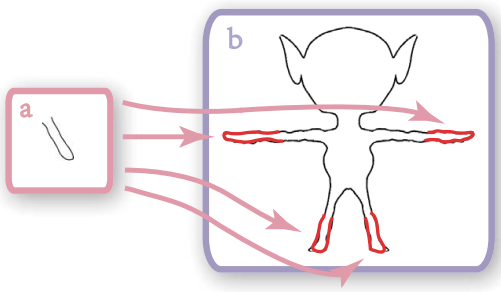


Figure 2: *A single contour may match a query contour in more than one section. The shape contour (b) matches the query contour (a) in four different sections (marked in red).*

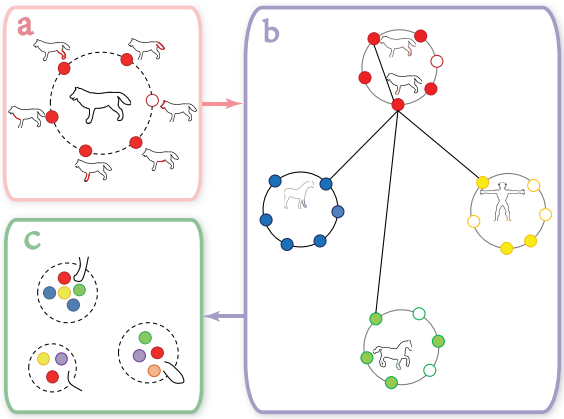


Figure 3: *Illustration of our randomized compound kNN graph. In this figure, a solid circle represents a valid section and a dashed circle represents an invalid one. Given a contour (taken as a node of our randomized compound kNN graph), we first generate several sections (a). Then, we find the nearest neighbors for each section and establish edges between the parent contours of these sections (b). Finally, we find valid sections and cluster them. (c).*

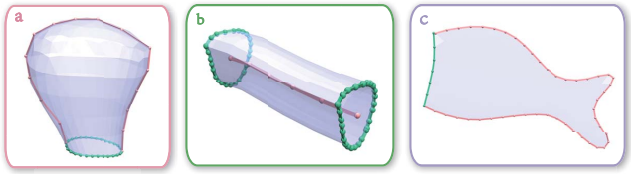


Figure 4: *An illustration of three parametrized proxies. (a) Rounded shape. (b) Generalized cylinder. (c) Sheet.*

5.1. Proxy Construction

In the most general setting, the user's sketched contour can be directly used to retrieve matching parts from database shapes. However, we found that introducing an intermediate 3D *proxy* allowed a better expression of the user's intent.

The sketch is interpreted using one of three parametrized proxies: rounded shapes, generalized cylinders, and sheets (see Figure 4). Rounded shapes are constructed by the approach of Nealen et al. [NISA07]. A generalized cylinder is constructed by two strokes: a loop on the base shape and a stroke tracing the medial axis. The free end of the cylinder may be resized to create a cone or frustum. To construct a sheet-like shape, the user first draws the connecting edge as a stroke on the base shape, and then sketches the outline of the sheet. Various editing operations (scaling, deleting, and dragging) on these primitives are supported. A variety of proxies, along with the strokes used to sketch them, are shown in Figure 4 and in the accompanying video.

During the construction of the proxy, we record the volume change of the proxy and its corresponding camera view. After the proxy shape is completely constructed, we extract the 2D contour for the 3D proxy under the camera view with a maximum volume change as our *proxy contour* to better reflect intent of the user. Here, the volume change is defined as the difference of the proxy volume after/before a sketching operation.

5.2. Contour Descriptor

Each contour is divided into sections: the query contour of the sketched proxy has a single section, whereas 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 $m = 20$ 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 $A_{ij} = \alpha_{ij}$. The distance between two contour sections S and T is the squared Euclidean distance between their angle matrices A^S and A^T:

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

5.3. 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 $\alpha = 0.95$, $\beta = 0.05$.

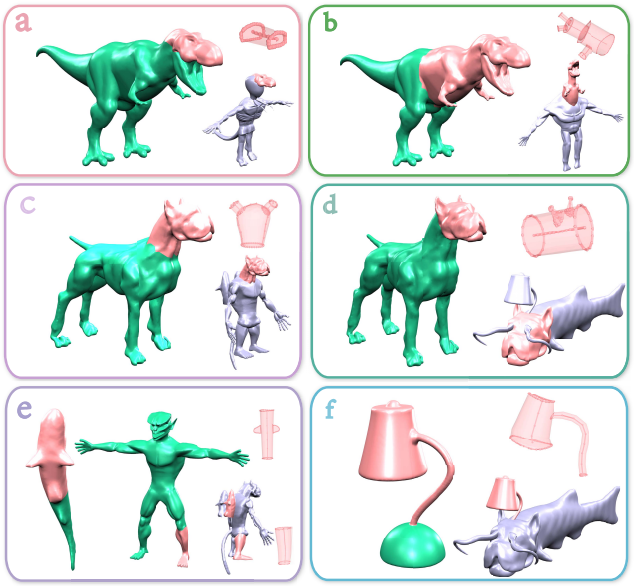


Figure 5: *Illustrations of irregular segmentations. In each subfigure, we show the segmentation(s) of the database model(s) on the left. The composed shapes are shown in the bottom right.*

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.
- 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. 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 7).

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.

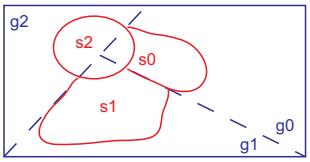
6.1. Super-Face Graph Representation

To construct the super-face graph for a shape mesh, we oversegment the mesh into 50 patches [HKG11] 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 6). 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.



		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 6: *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.*

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.

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 the 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

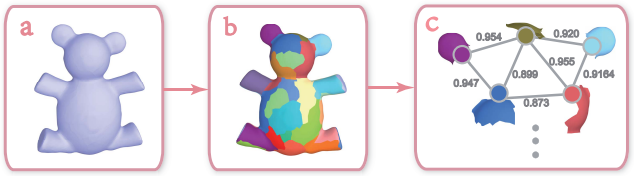


Figure 7: *The construction of the super-face graph for a 3D shape. Given the Teddy model (a), we partition it into* 50 *patches (b). Each super-face patch is a node of the graph. Adjacent super-faces are connected by a weighted graph edge (c).*

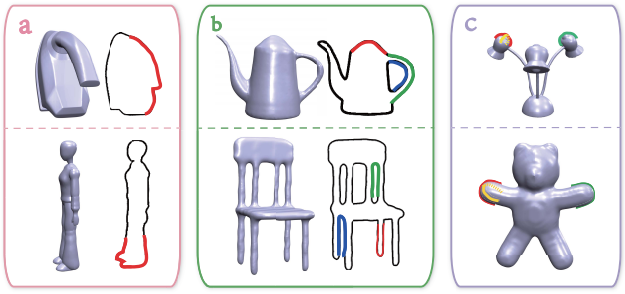


Figure 8: *Illustrations of the complexity of contours and their relationship. Depth-discontinuous contour sections are shown in (a). Topologically different contour sections are illustrated in (b). Isolated contour sections are shown in (c).*

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.

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 sketched proxy. 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 $v_1 v_2$ is closest to v_2.

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]:

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

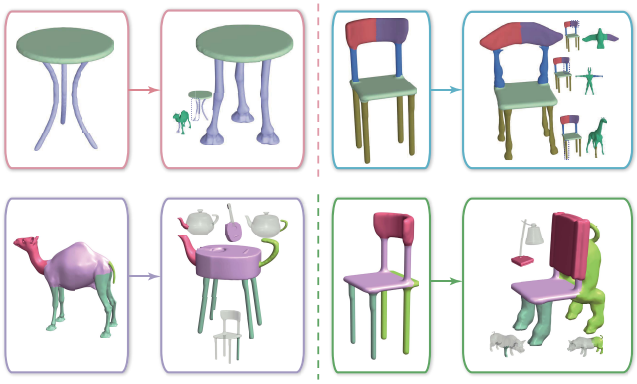


Figure 9: *Examples of shape variations created with our method. Given a segmented model (left), we can use one part as the query to retrieve similar parts from the database. The selected retrieved part is composited with the non-query parts to create a new shape (right) with the same layout as the original shape.*

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 an other edge on Q starting from v. We apply snake operations on the boundary vertices to minimize the energy E_f [JLCW06].

Fine level extraction is optional for part extraction because it will not affect the design significantly.

7. Applications

(SID SAYS: This section needs a lot of work. The greater the variety of applications and shape domains, the better. Examples from non-fantastical, non-creature domains would be great.)

Our on-the-fly sketch-based segmentation method is a general technique that can drive various modeling applications. Here we describe some potential applications.

Shape variation. Our pipeline can be used to generate shape variations by simple adaption, as shown in Figure 9. Given a segmented model, we can use any part of the model as the query to retrieve similar parts from the model database. The chosen retrieved part can then be used to replace the original part to create shape variations. To construct the proxy contour of a query part, we first perform Principal Component Analysis (PCA) on the part. Then, we take the view perpendicular to the plane containing the first and second principal directions of the part to calculate the query contour.

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 "proxy contour" and feed it to the pipeline. 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 extending the shape. The suggested parts can be irregular ones or from different shape families (see Figure 10).

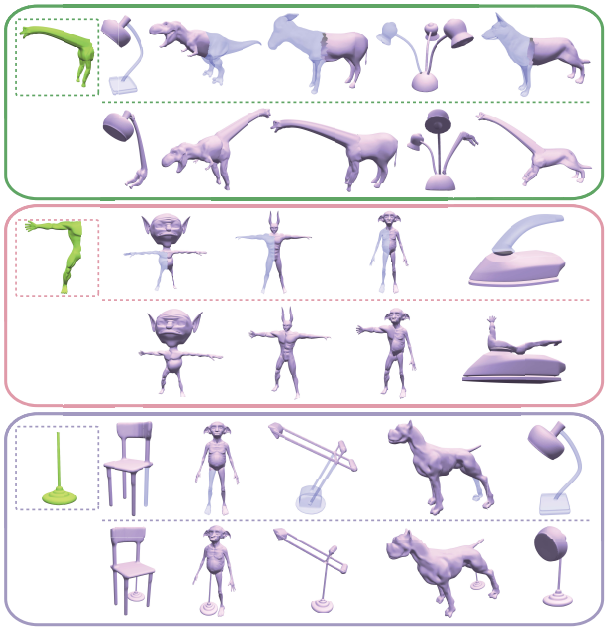


Figure 10: *Examples of part suggestions. Given the query shapes (in green), our system presents a list of part suggestions (in purple). The assembled shapes are shown below the suggested parts.*

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

$$T_n = \frac{w(Vol(P_M) - Vol(p_p))}{Vol(p_p)},$$

where $Vol(\cdot)$ is the volume of an object, P_M is the retrieved database model, p_p is the part in the database model corresponding to the query, and the weight $w = 2$. We constrain T_n to lie between 1 and 7. We rank the segments in ascending order by their distance to the corresponding part. Now, given a scale S, we can suggest the segments with this scale. An example is given in Figure 11.

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.

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

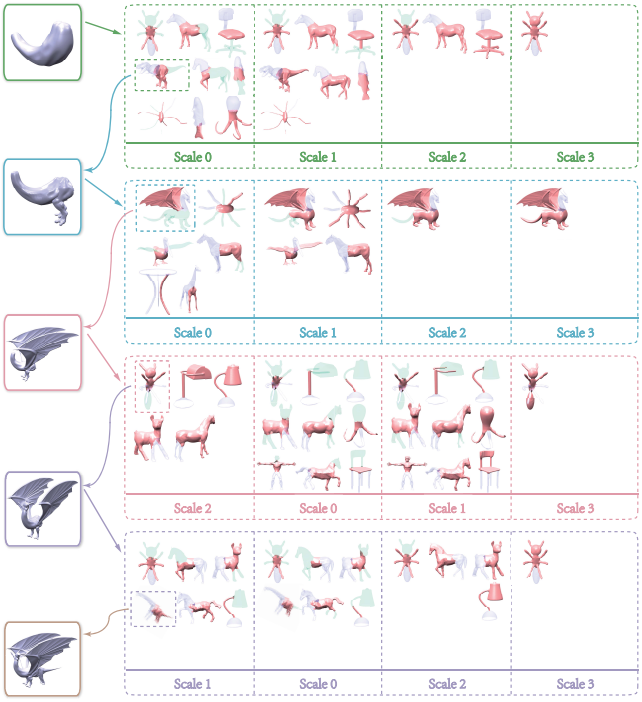


Figure 11: *An example of multi-scale part suggestion. The initial shape and subsequent working models, used as queries, are listed in the left column. On the right, we show the suggested parts (red) at different scales for each modeling step. The parts corresponding to the queries are marked in light blue. The other parts are marked in light green.*

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 $\sim$0.56 seconds (with GPU and multi-core acceleration) on average for our system to present part suggestions.

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. When searching for partial matches to a proxy 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.

Algorithm \ Performance	R-time	C-time
Brute force kNNG	4.54	51.7
Wang's kNNG approximation method	5.68	1.15
RC-kNNG with the brute force method	0.053	33.89
RC-kNNG with Wang's method	0.058	3.52

Table 1: *Detailed evaluations of retrieval performance for different methods on our test dataset. R-time represents the retrieval time for a proxy contour (in seconds). C-time represents the construction time for the kNNG data structure (in hours).*

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.

(SID SAYS: Performance with varying super-face counts, and varying other parameters, goes here. Is it possible to have a benchmark for how good the retrieved results are? Even reporting the final error (between the sketched query and the contour of the final part) would be very helpful. Currently we have speed stats but not quality stats. If we're pitching this as an algorithm paper, I think the latter would be very nice. We don't have time to create a proper ground truth set, but even reporting the errors for different superface counts would help.)

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 perfectly 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, and advances the state-of-the-art in designing detailed models using only rough sketches.

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 would like to augment our proxy construction technique with the ability to semantically exaggerate the geometry of parts. 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 combine both 2D projection information and 3D information (e.g. shape diameter) when searching with an inferred 3D

proxy. The view under which the proxy contour is extracted is important for our system. 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.

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

[AKZM14] AVERKIOU M., KIM V., ZHENG Y., MITRA N. J.: Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. In *Proc. Eurographics* (2014). 2

[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, 7

[CKGK11] CHAUDHURI S., KALOGERAKIS E., GUIBAS L., KOLTUN V.: Probabilistic reasoning for assembly-based 3D modeling. *ACM TOG 30*, 4 (2011). 2, 7

[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*12a] EITZ M., RICHTER R., BOUBEKEUR T., HILDEBRAND K., ALEXA M.: Sketch-based shape retrieval. *ACM TOG 31*, 4 (July 2012), 31:1–31:10. 1

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

[GF08] GOLOVINSKIY A., FUNKHOUSER T.: Randomized cuts for 3D mesh analysis. *ACM TOG 27*, 5 (2008). 5, 7

[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

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

[KCKK12] KALOGERAKIS E., CHAUDHURI S., KOLLER D., KOLTUN V.: A probabilistic model for component-based shape synthesis. *ACM TOG 31*, 4 (2012), 55:1–55:11. 2

[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). 6

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

[LIAL07] LAVIOLA J. J., IGARASHI T., ALVARADO C., LIPSON H.: Sketch-based interfaces: techniques and applications. In *SIGGRAPH Course Notes* (2007). 1

[LSK*10] LEE B., SRIVASTAVA S., KUMAR R., BRAFMAN R., KLEMMER S. R.: Designing with interactive example galleries. In *Proc. CHI* (2010). 1

[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. 8

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

[NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: FiberMesh: Designing freeform surfaces with 3D curves. *ACM TOG 26*, 3 (2007). 4

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

[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

[TGY*09] TALTON J. O., GIBSON D., YANG L., HANRAHAN P., KOLTUN V.: Exploratory modeling with collaborative design spaces. *ACM TOG 28*, 5 (2009). 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, 8

[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

[XZCOC12] XU K., ZHANG H., COHEN-OR D., CHEN B.: Fit and diverse: set evolution for inspiring 3D shape galleries. *ACM TOG 31*, 4 (2012). 2