Probabilistic Reasoning for Assembly-Based 3D Modeling

Siddhartha Chaudhuri* Evangelos Kalogerakis* Leonidas Guibas Vladlen Koltun

Stanford University

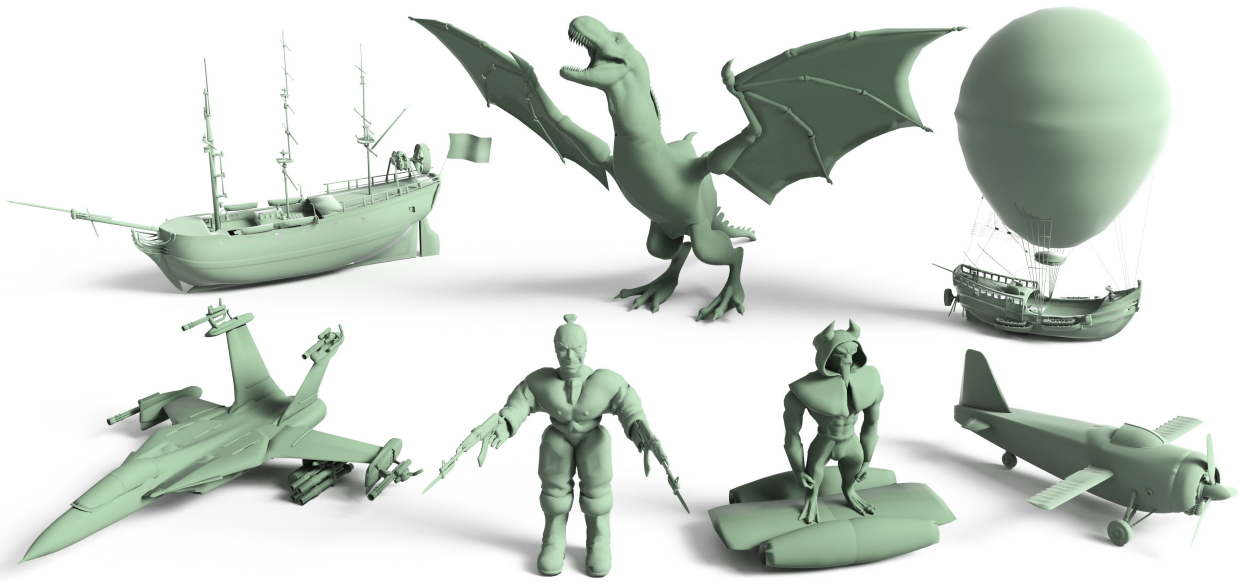

Figure 1: *3D models created with our assembly-based 3D modeling tool.*

Abstract

Assembly-based modeling is a promising approach to broadening the accessibility of 3D modeling. In assembly-based modeling, new models are assembled from shape components extracted from a database. A key challenge in assembly-based modeling is the identification of relevant components to be presented to the user. In this paper, we introduce a probabilistic reasoning approach to this problem. Given a repository of shapes, our approach learns a probabilistic graphical model that encodes semantic and geometric relationships among shape components. The probabilistic model is used to present components that are semantically and stylistically compatible with the 3D model that is being assembled. Our experiments indicate that the probabilistic model increases the relevance of presented components.

CR Categories: I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling;

Keywords: data-driven 3D modeling, probabilistic reasoning, probabilistic graphical models

*S. Chaudhuri and E. Kalogerakis contributed equally to this work.

Links: DL PDF

1 Introduction

> *What remains hard is modeling. The structure inherent in three-dimensional models is difficult for people to grasp and difficult too for user interfaces to reveal and manipulate. Only the determined model three-dimensional objects, and they rarely invent a shape at a computer, but only record a shape so that analysis or manufacturing can proceed. The grand challenges in three-dimensional graphics are to make simple modeling easy and to make complex modeling accessible to far more people.*
>
> — Robert F. Sproull [1990]

Providing easy-to-use tools for the creation of detailed three-dimensional content is a key challenge in computer graphics. With the accessibility of comprehensive game development environments, individual programmers and small teams can build and deploy realistic computer games and virtual worlds [Epic Games 2011; Unity Technologies 2011]. Yet the creation of compelling three-dimensional content to populate such worlds remains out of reach for most developers, who lack 3D modeling expertise.

A promising approach to 3D modeling is assembly-based modeling, in which new models are assembled from pre-existing components. The set of components can be designed specifically for this purpose or derived from a repository of shapes [Funkhouser et al. 2004; Kraevoy et al. 2007; Maxis Software 2008; Chaudhuri and Koltun 2010]. The advantage of assembly-based modeling is that users do not need to spec-

ify new geometry from scratch; modeling reduces to selection and placement of components.

A key challenge in assembly-based 3D modeling is the identification of relevant components to be presented to the user. Unless the modeling task is fixed in advance, an assembly-based modeling tool must use a large and varied database of components. The interface thus needs to provide effective mechanisms for identifying the most relevant available components at any stage of the modeling process. Previous work used either text search or geometric matching to retrieve shapes and components [Funkhouser et al. 2004; Shin and Igarashi 2007; Lee and Funkhouser 2008; Chaudhuri and Koltun 2010]. Such techniques do not take into account the semantic and stylistic relationships between the current model and the components in the database. In this paper, we present an approach that studies a model library to learn how shapes are put together, and uses this knowledge to suggest semantically and stylistically relevant components at each stage of the modeling process. Our work aims to support open-ended 3D modeling [Talton et al. 2009; Chaudhuri and Koltun 2010].

We use a probabilistic graphical model called a Bayesian network [Pearl 1988; Koller and Friedman 2009] to represent semantic and stylistic relationships between components in a shape database. When new models are assembled, inference in the Bayesian network is used to derive a relevance ranking on categories of components, and on individual components within each category. For example, when a user begins a modeling session by placing an airplane fuselage, the probabilistic model identifies components in the repository that are more likely to be adjacent to a fuselage, such as wings, stabilizers, and engines. The presented components continue to be dynamically updated as the current model is constructed. Thus when the user augments the fuselage with an engine from a military jet, the probabilistic model decreases the probability of propellers and increases the probability of rockets and missiles. After the user adds two wings, the probabilistic model lowers the probability of wings, since the presence of more than two wings on a jet is unlikely.

To evaluate the effectiveness of the presented approach, we have developed an assembly-based 3D modeling tool. Our modeling interface is inspired by the successful Spore creation tools and uses the probabilistic model to dynamically update the presented components. Experiments with this interface indicate that the probabilistic model produces more relevant suggestions than a static presentation of components or a purely geometric approach.

In summary, this paper introduces a probabilistic model of shape structure that incorporates both semantics and geometric style. The model is trained on a shape database and is used to present relevant shape components during a 3D modeling session. The experimental results demonstrate that the model significantly increases the relevance of presented components.

2 Prior Work

Assembly-based 3D modeling was pioneered by Funkhouser et al. [2004], whose Modeling by Example system allows users to retrieve source models using text- or shape-based search. Components are then cut out of some of the retrieved models and glued onto the current one. Subsequent research has investigated sketch-based retrieval of components [Shin and Igarashi 2007; Lee and Funkhouser 2008]. In all these approaches, the user must search for each specific component. This is less appropriate for open-ended 3D modeling, when the composition of the model is not specified in advance and the user can benefit from a variety of suggestions [Chaudhuri and Koltun 2010].

Kraevoy et al. [2007] describe an assembly-based modeling tool in which the user can load a small set of compatible shapes and interchange parts between them. The method assumes that all shapes have the same number of components and does not handle heterogenous shape libraries.

Chaudhuri and Koltun [2010] describe a data-driven technique for presenting components that can augment a given shape. The approach is purely geometric and does not take into account the semantics of components. Our results show that the incorporation of semantic relationships increases the relevance of presented components.

Fisher and Hanrahan [2010] describe a probabilistic model of spatial context for 3D model search. Given a query bounding box placed by the user in a 3D scene, their approach returns database shapes that are appropriate in the context of the surrounding scene. Our probabilistic model is substantially different from that of Fisher and Hanrahan and is designed for interactive shape modeling.

Our use of probabilistic graphical models is influenced by the work of Merrell et al. [2010], who generate residential building layouts using a Bayesian network trained on architectural programs. The probabilistic model of Merrell et al. is designed specifically to represent architectural programs. We introduce a probabilistic model for the structure of general 3D shapes. Our model represents both component categories and a variable number of individual components, and augments existence and adjacency relations with symmetry and geometric style.

The interface of our assembly-based modeling tool is inspired by the Spore creature creator, which allows creatures to be assembled from components such as heads, legs, and arms [Maxis Software 2008]. The Spore creature creator has been used to create over one hundred million creature models in the first year after release. While our interface closely follows the successful approach of Spore, the presented categories and parts are dynamically updated in light of the current model, in order to present the most relevant components from a large heterogenous library. Furthermore, we do not require specially crafted components, but extract them semi-automatically from a shape library.

3 Overview

Our approach consists of two stages, illustrated in Figure 2. The first is an offline preprocessing stage in which a probabilistic model is trained on a shape database. The second is an interactive stage in which inference in the probabilistic model is used to present relevant shape components during a 3D modeling session.

Preprocessing. The input to the preprocessing stage is a repository of segmented and labeled shapes. Each label represents a component category, such as *head*, *arm*, and *torso*. The labeling is hierarchical and each component may contain subparts with associated subcategories; for example, torsos are composed of *lower torso* and *upper torso*. We use the technique of Kalogerakis et al. [2010] to produce the segmentation and labeling from a set of training examples. Thus we preprocess a shape repository into components semi-automatically, relieving the chore of manually segmenting each model and labeling all components.

Given the set of segmented components, we further cluster them based on geometric style. For each category, the components are partitioned into a set of clusters with similar geometric feature vectors. The feature vectors incorporate descriptors such as shape diameter [Shapira et al. 2010], curvature, shape context [Mori et al. 2001], and PCA-based parameters (Appendix A). An example of this clustering is illustrated in Figure 2, where arms

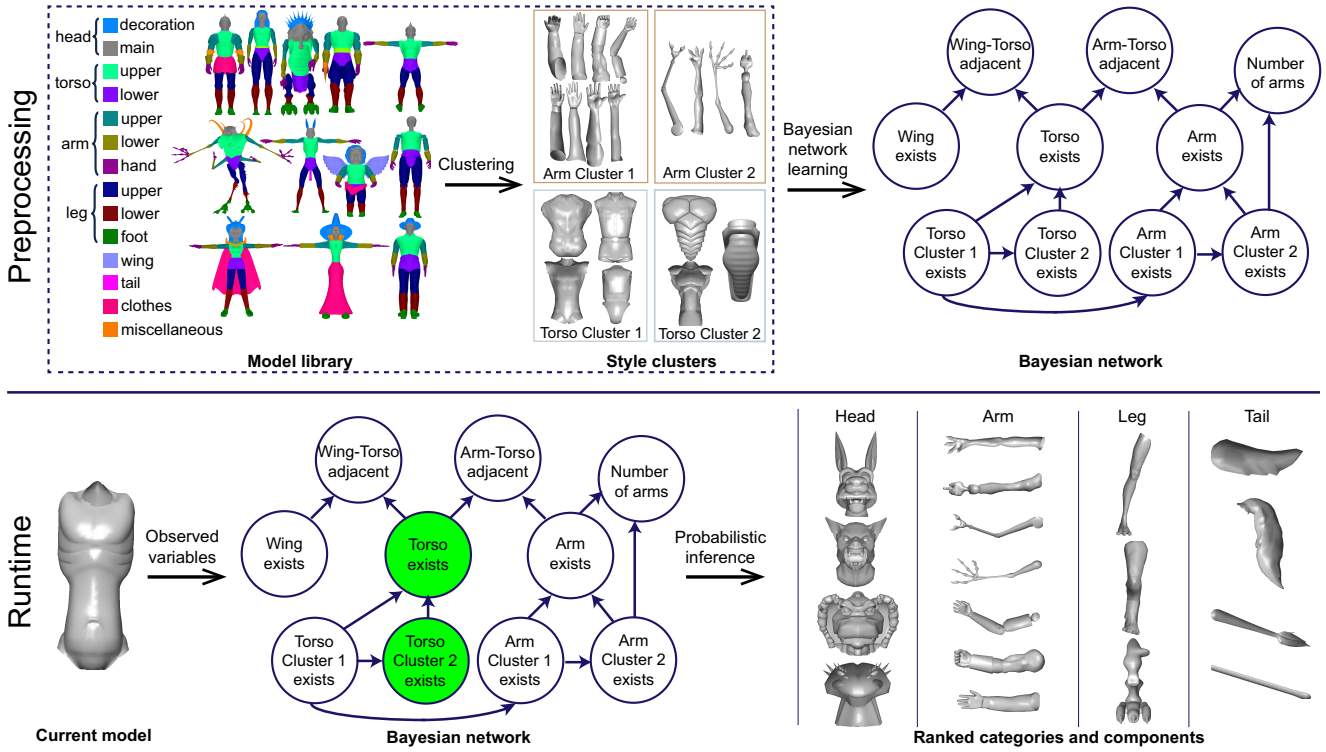

Figure 2: *Overview of our approach. The* **preprocessing** *stage (top) begins with a library of models, segmented and labeled using the technique of Kalogerakis et al. [2010]. The components extracted from the models are further clustered by geometric style. A Bayesian network is then learned that encodes probabilistic dependencies between labels, geometric styles, part adjacencies, number of parts from each category, and symmetries. The figure shows a subset from a real network learned from a library of creature models. The* **runtime** *stage (bottom) performs probabilistic inference in the learned Bayesian network to generate ranked lists of category labels and components within each category, customized for the currently assembled model.*

with a roughly humanoid appearance are grouped into one cluster, while thin and alien arms are grouped into another. The clustering allows the probabilistic model to operate not only on the semantic labels, but also on the geometric appearance of components.

Given the set of category labels and style clusters within each label, the method learns a probabilistic model that encodes dependencies between labels, style clusters, part adjacencies, numbers of parts from each category, and symmetries.

Runtime. During an interactive 3D modeling session, our method performs probabilistic inference in the learned model to generate a ranked list of components that may be useful for augmenting the current shape. The ranking reflects compatibility with components that are already part of the assembled shape.

In order to evaluate the relevance of the suggestions produced by the learned model, we have developed a tool for assembly-based 3D modeling. The modeling interface is illustrated in Figure 4 and in the supplementary video. On the left, the interface presents tabs with semantic labels such as *torso*, *head*, and *arm*. The tabs can expand hierarchically to show subcategories, such as *lower torso*, *upper torso*, and *belt*. When a user selects a tab, the interface shows a list of parts that belong to the associated category. The user can select a component and drag it onto the current model. When a component is added in this way, the lists of categories and components are re-ranked to reflect their semantic and stylistic compatibility with the current 3D model. Using the probabilistic model, the interface also estimates whether the new component should have a symmetric counterpart and computes the symmetry plane. The interface supports sliding any component along the surface of the model, as well as translation, rotation, scaling, duplication, and gluing, thus assisting the assembly of a 3D model from chosen components.

4 Probabilistic Model

Our probabilistic model is designed for the purpose of recommending components that can augment a given 3D model. The probabilistic model is trained on a set of segmented and labeled shapes. Each shape is represented by the existence, adjacencies, geometric style, symmetries, and number of components from each category, as described below. These attributes are represented as random variables. Each shape is thus treated as a sample from a joint probability distribution $P(\mathbf{X})$, where $\mathbf{X}$ contains the following sets of variables:

Existence $\mathbf{E} = \{E_l\}$ where $E_l \in \{0, 1\}$ represents whether a component from category $l \in \mathcal{L}$ exists in a shape, where $\mathcal{L}$ is the set of component labels in the repository. There is one such random variable for each label.

Cardinality $\mathbf{N} = \{N_l\}$ where $N_l \in \{0\} \cup \mathbb{Z}^+$ represents the number of components from category $l \in \mathcal{L}$ in a shape. There is also one such random variable for each label. For tractable inference, we limit the range to $\{0, 1, 2, 3, 4, 5+\}$, where $5+$ represents the case of 5 or more existing components from the category. Note that the random variables $\mathbf{E}$ represent information for the existence of components, whereas the cardinality variables $\mathbf{N}$ make it specific in terms of their exact number. The random variables

$\mathbf{E}$ are still useful in our model as an intermediate representation, since the number of components cannot be directly observed in the user's incomplete shape. In addition, certain dependencies of components are more compactly expressed in terms of their existence rather than their exact cardinality (e.g., a table top exists when table legs exist, regardless of their exact number), which helps decreasing the number of parameters in our model.

Adjacency $\mathbf{A} = \{A_{l,l'}\}$ where $A_{l,l'} \in \{0,1\}$ represents whether a component from category $l \in \mathcal{L}$ is adjacent to a component from category $l' \in \mathcal{L}$ in a shape. There is one such random variable for each pair of distinct labels that appear adjacent in at least one of the shapes in the repository.

Style $\mathbf{S} = \{S_{s,l}\}$ where $S_{s,l} \in \{0,1\}$ represents whether a component of geometric style $s \in \mathcal{S}_l$ exists in a shape, where $\mathcal{S}_l$ is the set of clusters for category l. There is one such random variable for each style cluster within each category.

Symmetry $\mathbf{R} = \{R_{l,l'}\}$ where $R_{l,l'} \in \{0,1\}$ represents whether a component from category $l \in \mathcal{L}$ has a symmetric counterpart with respect to a component from category $l' \in \mathcal{L}$. For example, an arm has a symmetric counterpart with respect to the torso. There is one such random variable for each distinct pair of labels that have such a symmetry relationship in at least one of the shapes in the repository.

The probabilistic model encodes the joint distribution $P(\mathbf{X})$. The purpose of the model is to support estimation of compatibility between each component from the repository and a given 3D model. The given model is the current 3D model observed during the runtime stage. The given model imposes specific values on some random variables in $\mathbf{X}$. For example, if the current model contains a component from category l and style s, the corresponding random variables E_l and $S_{s,l}$ are set to 1. Similarly, if a component with label l is adjacent to a component with label l', the corresponding random variable $A_{l,l'}$ is set to 1. Such variables are said to be *observed*. The rest of the random variables are unobserved. In addition, the cardinality random variables are never observed, since we also do not know the final number of components in the assembled shape; although unobserved, cardinality variables are important for querying our model, as explained below.

By performing inference in the probabilistic model, we will compute probability distributions on some of the unobserved variables given the observed ones, and thus estimate the compatibility of various components with the currently assembled shape.

Specifically, the compatibility of a component i from category l_i and style s_i is evaluated based on (a) how likely its category l_i is to be adjacent to any of the categories $\mathcal{L}' \subseteq \mathcal{L}$ existing in the shape, (b) how likely the shape is to include more parts of category l_i, and (c) how likely the shape is to include components of style s_i. Thus, we need to evaluate, for each component i in the repository, the probability of the following expression:

$$\left(\bigvee_{l' \in \mathcal{L}'} (A_{l_i,l'} = 1) \right) \wedge (N_{l_i} > n_{l_i}) \wedge (S_{s_i,l_i} = 1), \qquad (1)$$

where n_{l_i} is the number of components from category l_i in the current shape. The probability of this query can be evaluated using the conditional probability distribution $P(\mathbf{X}_q \mid \mathbf{X}_e = \mathbf{e})$, where $\mathbf{X}_q \in \mathbf{X}$ is the set of the random variables involved in the above query, $\mathbf{X}_e \in \mathbf{X}$ is the set of observed random variables, and $\mathbf{e}$ are their observed values. We define the *compatibility score* of component i as

$$\text{comp}(i) = \sum_{\mathbf{q} \in Q} P(\mathbf{X}_q = \mathbf{q} \mid \mathbf{X}_e = \mathbf{e}), \qquad (2)$$

where Q is the set of instantiations of $\mathbf{X}_q$ that satisfy Expression 1. The compatibility score measures the suitability of component i for augmenting the currently assembled shape.

The compatibility score is expressed in terms of the conditional probability distribution $P(\mathbf{X}_q \mid \mathbf{X}_e = \mathbf{e})$. From the definition of conditional probability, we have:

$$P(\mathbf{X}_q \mid \mathbf{X}_e = \mathbf{e}) = \frac{P(\mathbf{X}_q, \mathbf{X}_e = \mathbf{e})}{P(\mathbf{X}_e = \mathbf{e})}, \qquad (3)$$

which can be computed from the joint distribution $P(\mathbf{X})$ by summing over all possible assignments to the unobserved random variables:

$$P(\mathbf{X}_q \mid \mathbf{X}_e = \mathbf{e}) = \frac{\sum_{\mathbf{u}} P(\mathbf{X}_q, \mathbf{X}_e = \mathbf{e}, \mathbf{X}_u = \mathbf{u})}{\sum_{\mathbf{u},\mathbf{q}} P(\mathbf{X}_q = \mathbf{q}, \mathbf{X}_e = \mathbf{e}, \mathbf{X}_u = \mathbf{u})}, \qquad (4)$$

where $\mathbf{X}_u = \mathbf{X} - \mathbf{X}_e - \mathbf{X}_q$ is the set of random variables that are neither query nor evidence. However, an explicit summation of this form is generally intractable, since the number of possible assignments is in general exponential in the number of variables. The exponential complexity can be reduced by assuming that all random variables in $\mathbf{X}$ are independent. However, this assumption is generally incorrect. For example, if the user's shape contains a quadruped torso, it is likely that the shape would have other quadruped parts, such as legs, a head, and a tail. The existence, number, adjacencies, and style of these parts are dependent on the existence and style of the torso.

Instead of manually specifying which random variables are probabilistically dependent on which others, we learn the factorization of the joint distribution from data. This factorization makes the computation of the compatibility score tractable. The factorized distribution is represented with a directed graphical model called a Bayesian network [Pearl 1988; Koller and Friedman 2009]. A Bayesian network represents the random variables and their conditional dependencies using a directed acyclic graph. Nodes represent random variables and directed edges represent conditional dependencies between the corresponding variables. Each random variable is associated with a probability function that takes as input the set of values of the node's parent variables and outputs the probability of this random variable given the input values. Since our random variables are discrete, these probability functions are represented as Conditional Probability Tables (CPTs). The procedure for learning the connectivity of the network and the entries of all the CPTs is described in Section 5. Information on the learned Bayesian networks used in our experiments is provided in Table 1.

Inference. Given a learned Bayesian network, the joint distribution can be expressed as $P(\mathbf{X}) = \prod_{x \in \mathbf{X}} P(x \mid \pi(x))$, where $\pi(x)$ is the set of parent variables of x. Inference in the Bayesian network can be used to answer conditional probability queries such as (3). Since exact inference in probabilistic graphical models is NP-hard, we use approximate inference. We had experimented with loopy belief propagation [Frey and MacKay 1997], but with limited success, as it often returned poor approximations to the query distributions. Instead, we implemented a likelihood-weighted sampling technique that stochastically samples the joint distribution and weighs each sample based on the likelihood of the observed nodes accumulated throughout the process [Fung and Chang 1990].

Category ranking. While the compatibility score determines the ranked order of individual components, the part categories also need to be ranked as a whole, so that category labels can be presented in order of relevance by the interface. We achieve this by evaluating a compatibility score similar to Equation 2 that ignores

the part-specific style term. The score of a part category l is defined as

$$\text{comp}(l) = \sum_{\mathbf{q} \in Q'} P(\mathbf{X}_q = \mathbf{q} \mid \mathbf{X}_e = \mathbf{e}), \quad (5)$$

where Q' is the set of instantiations of $\mathbf{X}_q$ that satisfy

$$\left(\bigvee_{l' \in \mathcal{L}'} (A_{l,l'} = 1) \right) \wedge (N_l > n_l), \quad (6)$$

where $\mathcal{L}' \subseteq \mathcal{L}$ is the set of categories of components in the current shape, and n_l is the number of existing components from category l.

Initial ranking. The assembled shape is initially empty. Therefore none of the random variables in $\mathbf{X}$ are observed initially. In this case, we rank the parts and part categories by querying the prior probabilities $P(E_{l_i} = 1, S_{s_i,l_i} = 1)$ for each part i with label l_i and style s_i, and $P(E_l = 1)$ for each label l. The resulting category ranking is determined by the frequency of occurrence of the categories in the repository: the most popular categories appear first. For example, for a repository of creatures, the top categories are torsos, legs, and heads. Similarly, the initial ranking of parts is derived from the frequency of occurrence of style clusters in the repository.

Symmetry. The probabilistic model also provides information on whether a chosen component should have a symmetric counterpart. Our implementation supports planar reflective symmetry, which is among the most prevalent in real-world objects [Podolak et al. 2006]. For a chosen component i with label l_i, we evaluate, for each observed category l' in the current shape, the probability $P(R_{l_i,l'} = 1 \mid \mathbf{X}_e = \mathbf{e})$, and take the observed category l_{max} that maximizes this probability. If this maximal probability is greater than 0.5, the interface generates the symmetric counterpart of i about the symmetry plane of the component with label l_{max}. If there are multiple such components, we select the largest one. If there are multiple symmetry planes for that component, we select the one that maximizes the projection of the assembled shape onto it. Symmetry planes are computed using the approach of Simari et al. [2006].

5 Training

We now describe the offline procedure for learning the entries of the conditional probability tables for the Bayesian network, as well as the structure of the network. The input to this procedure is a set of shapes from a repository. We assume that a small set of manually segmented and labeled models is provided as a training set for a supervised segmentation and labeling procedure that propagates the segmentations and labelings to the rest of the dataset [Kalogerakis et al. 2010]. The sizes of the training sets used in our experiments are reported in Table 1.

We modified the segmentation and labeling algorithm of Kalogerakis et al. [2010] to support hierarchical labeling. For each class of shapes, the modified algorithm learns a conditional random field (CRF) at each level of the segmentation hierarchy, and generates a sequence of increasingly refined segmentations of each shape by successively applying the CRFs in order. The set of geometric descriptors is the same as in the original algorithm. The geometric descriptors are computed at surface samples rather than mesh faces to accommodate polygon soups. A JointBoost classifier [Torralba et al. 2007] is used to output the probability distribution of labels over each sample. Then the distribution is assigned to each mesh face, by averaging the probabilities of labels over its nearest point samples. The unary and pairwise terms of the CRFs are defined as in the original algorithm.

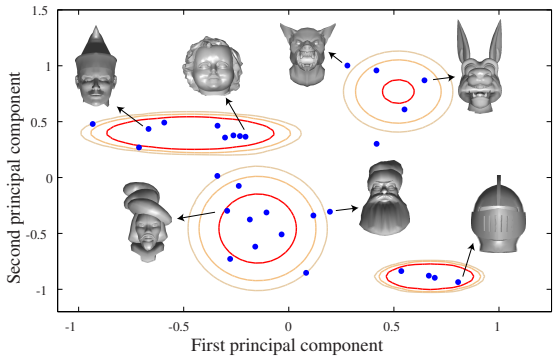

Figure 3: *Clusters of head parts, based on a Gaussian mixture model. The feature space is visualized by projection onto the two principal axes.*

Given a segmented and labeled dataset, the method performs clustering to identify groups of geometrically similar parts within each part category. The method then learns the structure and parameters of the Bayesian network. The balance of this section describes these two steps in more detail.

Style clustering. For each component i we compute a geometric feature vector $\mathbf{z}_i$. In order to capture a variety of geometric attributes, this feature vector incorporates shape diameter, curvature, shape context, and PCA descriptors (Appendix A). The clustering is based on a Gaussian mixture model in the feature space of $\mathbf{z}$, so that the mixture is a superposition of Gaussian distributions that have their own covariance structure. The number of Gaussians of the mixture is estimated by penalizing the complexity of the fitted mixture model. Specifically, for each part category l, we maximize the following objective function with respect to the number and parameters of the mixture model:

$$J_l = \sum_{i=1}^{m_l} \log \left(\sum_{s=1}^{k_l} w_{s,l} \cdot \mathcal{N}(\mathbf{z}; \mu_{s,l}, \Sigma_{s,l}) \right) - \frac{1}{2} P \log(m_l), \quad (7)$$

where m_l is the number of parts in category l, k_l is the number of Gaussians of the mixture for l, $\mathcal{N}(\mathbf{z}; \mu_{s,l}, \Sigma_{s,l})$ is a Gaussian with mean $\mu_{s,l}$ and covariance matrix $\Sigma_{s,l}$, $w_{s,l}$ is the corresponding mixing coefficient, and P is the number of parameters in the mixture model. The second term penalizes model complexity based on the Bayesian Information Criterion [Schwarz 1978].

The objective J_l is maximized as follows. Starting from $k_l = 2$, we run expectation-maximization (EM) to estimate the parameters $w_{s,l}$, $\mu_{s,l}$, and $\Sigma_{s,l}$. If the number of parts k_l is less than twice the number of parameters, we restrict the covariance matrix to be diagonal to prevent overfitting. EM is initialized with centers computed by k-means. We run EM separately for $k_l = 3, 4, \ldots$. If there is no increase in the objective function for 5 successive repetitions, we stop and accept the fitting computed by the EM iteration with the highest objective function value. The clusters are defined as follows: each part i is assigned to the cluster s that corresponds to the Gaussian that maximizes $w_{s,l} \cdot \mathcal{N}(\mathbf{z}_i \mid \mu_{s,l}, \Sigma_{s,l})$. Figure 3 illustrates the clustering of some of the head components from the creature dataset.

Bayesian network learning. Given training data $\mathbf{D}$, our method learns the graph structure G and parameters Θ of the Bayesian network by maximizing the Bayesian Information Criterion score

$$\log P(\mathbf{D} \mid G) \approx \log P(\mathbf{D} \mid G, \Theta) - \frac{1}{2} v \log(n), \qquad (8)$$

where v is the number of independent parameters in the network and n is the number of shapes in the dataset. The first term is the likelihood of the parameters Θ and the second term penalizes model complexity.

Structure learning is NP-hard and exact algorithms are super-exponential in the number of variables. Following Merrell et al. [2010], we use a local search heuristic that explores the space of network structures by adding, removing, and flipping edges.

To assist the structure learning procedure, we enforce the existence of a set of edges that represent obvious conditional dependencies. For each category l, we establish links from the existential node E_l to the cardinality node N_l, as well as to the style nodes $\{S_{s,l}\}_{s \in \mathcal{S}_l}$ and the symmetry and adjacency nodes involving this category. If category l has subcategories, we also establish links to the existential nodes of the subcategories.

6 Modeling Interface

To evaluate the presented approach to suggestion generation, we have implemented an assembly-based 3D modeling interface. Categories of suggested parts are shown on the left, with individual parts arranged within categories (Figure 4). The categories and the parts within each category are ordered based on their compatibility with the current shape, as computed by the probabilistic model. The order of the categories and parts is automatically updated when the composition of the assembled shape changes. The user can select a part and drag it onto the assembly area on the right, where it is snapped and glued to the current shape [Schmidt and Singh 2010].

After adding a component, the user can further adjust its position, orientation, and scale. During these transformations, glued connections are maintained so that the part slides, rotates and scales on the surface of the rest of the assembly. This reduces the need for painstaking 6-DOF manipulation of parts. This snap-dragging can be disabled when desired, which is useful for assemblies with narrow, irregular or hard-edged boundaries.

When the user drags in a part that has a symmetric counterpart, the counterpart is automatically generated and transformed with respect to the corresponding symmetry plane. This makes it easier to position pairs of legs, arms, wings, and wheels. The user can delink the symmetric pair and position each part individually. Parts can also be duplicated, reflected and deleted. To support confident exploration, the interface provides unlimited undo/redo functionality. The interface also provides a search box that supports textual search for category labels, but participants in our experiments almost never used this functionality, preferring visual exploration of part categories.

The modeling interface is further demonstrated in the supplementary video. In our experiments, new users became proficient with the tool after 5 minutes of demonstration and 5 additional minutes of hands-on exploration. All models in this paper were created after this minimal training period.

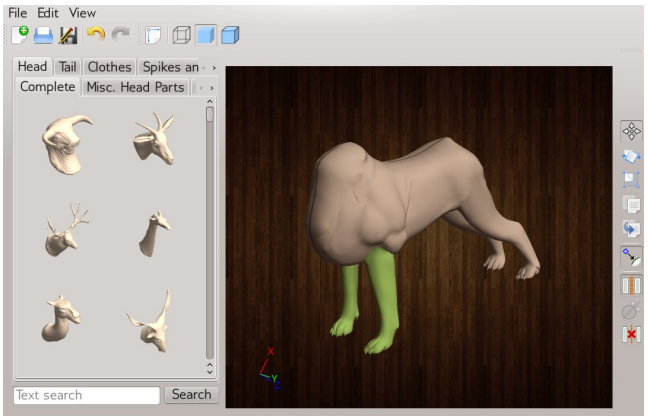

Figure 4: *Modeling interface. The model is assembled from presented parts.*

7 Evaluation

In this section, we describe an experimental evaluation of the relevance of the components presented by the probabilistic model. The goal of the experiments was to evaluate the relevance of the presented components during open-ended 3D modeling.

7.1 Experimental Setup

The presentation of components using the probabilistic model was compared to two alternative approaches. The first alternative was a static ordering of categories and parts. The ordering is based on the prior probabilities of categories and style clusters in the training data. Thus more frequently used categories and geometric styles appear first. Static ordering is the approach used in the Spore creature creator [Maxis Software 2008], where the order of presented categories and components does not adapt to the current model. The second alternative was the purely geometric suggestion generation approach of Chaudhuri and Koltun [2010]. We have modified the published technique to use the same segmentations as in the other two conditions, thus the set of components was identical and only the order of presentation varied across conditions. The same modeling interface was used in all conditions, so the components suggested by the approach of Chaudhuri and Koltun [2010] were also presented by category.

To evaluate the relevance of components presented by the three approaches, we recruited 42 volunteers from the student body of a computer science department in a research university. The students were recruited through departmental mailing lists. Most of the participants had little or no prior exposure to 3D modeling. Each participant was given a 5-10 minute orientation and was then asked to perform four modeling tasks. The tasks were of two types:

Toy. A neighborhood toy company asked you to design a children's toy that they will manufacture. Create such a toy using the provided application.

Creature. A neighborhood game company asked you to design a large number of creatures for an upcoming fantasy game. Create one such creature using the provided application.

For the Toy assignment, the tool used a large model library with 491 models obtained from the Digimation ModelBank and Archive collections. The library included models of creatures, aircraft, watercraft, and furniture. For the Creature assignment, the tool used a smaller library with 173 creature models. Table 1 provides further

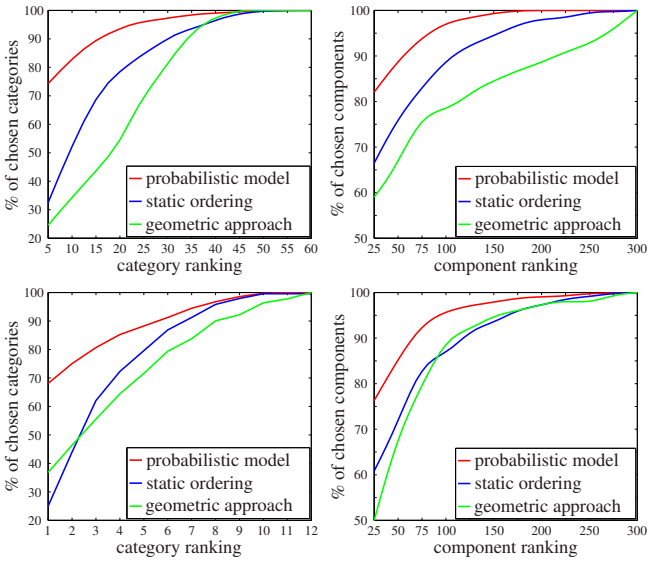

Figure 5: *Results for the Toy task (top) and Creature task (bottom). The plots on the left show the cumulative distribution of categories from which parts were chosen, as a function of the ranking of the category at the time of selection. The plots on the right show the cumulative distribution of components used by participants, as a function of the ranking of the component within its category at the time of selection. The probabilistic model presented more relevant categories and components than the static ordering or the geometric approach of Chaudhuri and Koltun [2010].*

information on the two datasets.

	Generic	Creature
# of database shapes	491	173
# of hand-segmented shapes	180	64
# of categories	62	24
# of unique components	3702	1598
# of style clusters	248	115
# of nodes in Bayesian network	595	265
# of edges in Bayesian network	1224	632

Table 1: *Datasets used in the evaluation.*

Each participant performed two toy tasks and two creature tasks. Each task was performed in a randomly chosen condition: components presented by the probabilistic model, static ordering, or components suggested by the approach of Chaudhuri and Koltun. The modeling interface was identical in all conditions. The conditions were not disclosed to the participants, although some differences were apparent, since in the static condition the order of the presented components never changed, and in the geometric condition the suggestion generation process was much longer due to the intensive computational demands of that technique. Several participants performed less than four tasks due to time constraints. In total, 141 modeling tasks were completed. The average modeling time per session with the probabilistic model was 21.4 minutes, the median was 19 minutes, the shortest was 7 minutes and the longest was 56 minutes. The average modeling time was 25.4 minutes with the static ordering and 34.3 minutes with the approach of Chaudhuri and Koltun.

The speed of component presentation by the probabilistic model compared favorably with the approach of Chaudhuri and Koltun.

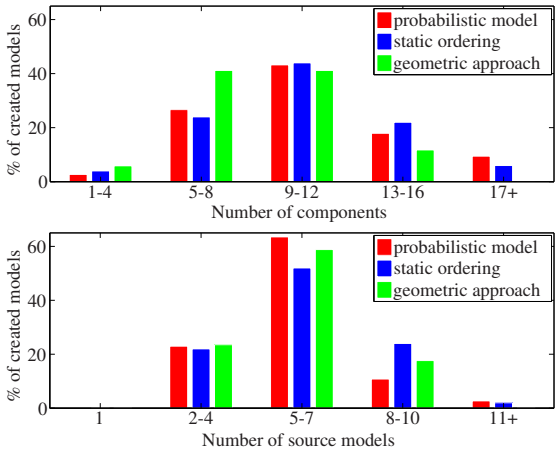

Figure 6: *Number of components used in a single assembled model (top) and number of library models that the employed components originate from (bottom).*

We ran parallel modeling sessions during the evaluation, using an eight-core 2.53 GHz Xeon workstation and a quad-core 3.2 GHz Core i7 workstation. Inference in the Bayesian network used a single core and completed in 2.8 seconds on average. The approach of Chaudhuri and Koltun used all available cores and required roughly 90 seconds to generate suggestions.

7.2 Results

To evaluate the relevance of components presented by the three approaches, we have analyzed logs generated by the application for all modeling sessions. Relevance was evaluated both for the ordering of categories and for the ordering of parts within categories. Figure 5 (left) presents the cumulative distribution of categories from which parts were chosen, as a function of the ranking of the category at the time of selection. Thus for the Toy task, 64.6% of the chosen components were selected from the first 2 categories in the probabilistic model condition, compared to 23.5% in the static condition and 21.3% in the geometric condition. For the Creature task, 75.05% of the components were chosen from the first 2 categories presented by the probabilistic model, compared to 43.9% in the static condition and 46.4% in the geometric condition. Figure 5 (right) presents the cumulative distribution of the components used by participants, as a function of the ranking of the component within its category at the time of selection. For both tasks, the probabilistic model outperformed the static ordering and the geometric approach.

Figure 6 (top) shows the distribution of the number of components used by participants in their models. On average, participants used 9.9 components for 3D models created with the probabilistic model. Figure 6 (bottom) shows the distribution of the number of library models that served as sources of components for each assembled 3D model. On average in the probabilistic model condition, the components used in single assembled model originate from 5.7 library models. The average number of components per assembled model and the average number of source library models per assembled model was similar across the three conditions.

Figures 1 and 7 show 3D models created by participants with the probabilistic model. The models were created entirely with the assembly-based 3D modeling interface.

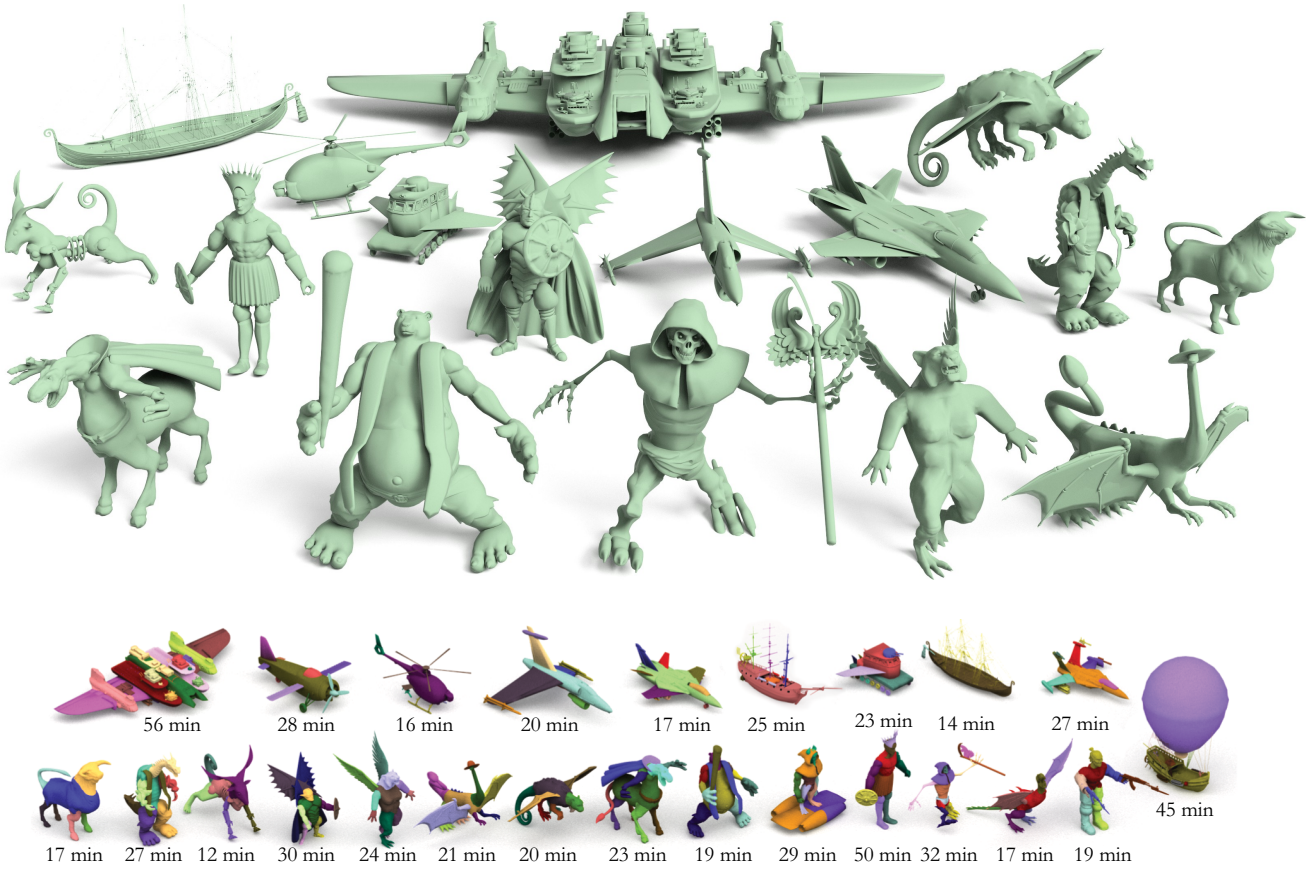

Figure 7: *Top: models created by participants with our assembly-based 3D modeling tool. Bottom: models shown above and in Figure 1, with individual components and modeling times.*

8 Discussion

We have described a probabilistic reasoning approach to the presentation of components in assembly-based 3D modeling. Our approach learns a probabilistic graphical model that encodes conditional dependencies between shape components. The model operates on both semantics and geometric style. During an interactive modeling session, inference in the probabilistic model is used to present relevant components.

The probabilistic model can be augmented in a number of ways. The model represents conditional dependencies in a directed graph. This captures directed dependencies in the data: when a fuselage from a commercial jet is added, it "activates" the existence of two wings and their adjacency to the fuselage, as well as the existence of two or four engines adjacent to the wings. However, certain relationships can be alternatively modeled by undirected links, which are more appropriate for representing mutual constraints; for example, a fuselage from a commercial jet is unlikely to co-exist with missiles. Future work could investigate the use of undirected graphical models, such as Markov random fields, or unified models such as factor graphs. In addition, random variables that represent existence information could be omitted in favor of a tree-based representation for the CPDs at the cardinality variables. Also, the presented model uses discrete variables, which simplify both learning and inference; however, geometric style can be more precisely modeled with continuous variables, which can further increase the relevance of presented parts. The model can also be augmented to support a greater variety of symmetries, as well as spatial and functional relationships between components.

Assembly-based 3D modeling can benefit from improved techniques for consistent shape segmentation [Golovinskiy and Funkhouser 2009; Kalogerakis et al. 2010] and for gluing and cutting shapes [Sharf et al. 2006; Schmidt and Singh 2010]. Sketch-based shape manipulation algorithms can be integrated to support editing of individual components [Nealen et al. 2005] and careful meshing techniques can yield structurally sound models that can be fabricated in physical form [Shen et al. 2004].

The data-driven approach to three-dimensional content creation also calls for improved interactive techniques for texturing, rigging, and animating shapes. Our implementation produces static shapes with no material properties. The ability to easily texture the assembled models would significantly enhance expressivity. Likewise, the ability to easily set up the created models for animation and simulation would enable users to produce functional content for computer games and virtual worlds.

Acknowledgments

We are grateful to Aaron Hertzmann, Sergey Levine, Jonathan Laserson, Suchi Saria, and Philipp Krähenbühl for their comments on this paper, and to Daphne Koller for helpful discussions. Chris Platz and Hadidjah Chamberlin assisted in the preparation of figures and the supplementary video. Niels Joubert narrated the video. This work was supported in part by NSF grants SES-0835601, CCF-0641402, and FODAVA-0808515, and by KAUST Global Collaborative Research.

References

CHAUDHURI, S., AND KOLTUN, V. 2010. Data-driven suggestions for creativity support in 3D modeling. In *Proc. SIGGRAPH Asia*, ACM.

EPIC GAMES. 2011. *Unreal Development Kit*. `http://www.udk.com`.

FISHER, M., AND HANRAHAN, P. 2010. Context-based search for 3D models. In *Proc. SIGGRAPH Asia*, ACM.

FREY, B., AND MACKAY, D. 1997. A revolution: belief propagation in graphs with cycles. In *Advances in Neural Information Processing Systems*, 479–485.

FUNG, R. M., AND CHANG, K.-C. 1990. Weighting and integrating evidence for stochastic simulation in Bayesian networks. In *Proc. Conference on Uncertainty in Artificial Intelligence*, North-Holland Publishing Co.

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. In *Proc. SIGGRAPH*, ACM.

GOLOVINSKIY, A., AND FUNKHOUSER, T. 2009. Consistent segmentation of 3D models. *Computers & Graphics 33*, 3, 262–269.

KALOGERAKIS, E., HERTZMANN, A., AND SINGH, K. 2010. Learning 3D mesh segmentation and labeling. In *Proc. SIGGRAPH*, ACM.

KOLLER, D., AND FRIEDMAN, N. 2009. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press.

KRAEVOY, V., JULIUS, D., AND SHEFFER, A. 2007. Model composition from interchangeable components. In *Proc. Pacific Graphics*, IEEE Computer Society.

LEE, J., AND FUNKHOUSER, T. 2008. Sketch-based search and composition of 3D models. In *Proc. Eurographics Workshop on Sketch-Based Interfaces and Modeling*.

MAXIS SOFTWARE. 2008. *Spore*. `http://www.spore.com`.

MERRELL, P., SCHKUFZA, E., AND KOLTUN, V. 2010. Computer-generated residential building layouts. In *Proc. SIGGRAPH Asia*, ACM.

MORI, G., BELONGIE, S., AND MALIK, J. 2001. Shape contexts enable efficient retrieval of similar shapes. In *Proc. IEEE Computer Vision and Pattern Recognition*, vol. 1, 723–730.

NEALEN, A., SORKINE, O., ALEXA, M., AND COHEN-OR, D. 2005. A sketch-based interface for detail-preserving mesh editing. In *Proc. SIGGRAPH*, ACM.

PEARL, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

PODOLAK, J., SHILANE, P., GOLOVINSKIY, A., RUSINKIEWICZ, S., AND FUNKHOUSER, T. 2006. A planar-reflective symmetry transform for 3D shapes. In *Proc. SIGGRAPH*, ACM.

SCHMIDT, R., AND SINGH, K. 2010. Drag, drop, and clone: An interactive interface for surface composition. Tech. Rep. CSRG-611, Department of Computer Science, University of Toronto.

SCHWARZ, G. 1978. Estimating the dimension of a model. *The Annals of Statistics*, 2, 461–464.

SHAPIRA, L., SHALOM, S., SHAMIR, A., ZHANG, R. H., AND COHEN-OR, D. 2010. Contextual part analogies in 3D objects. *International Journal of Computer Vision 89*, 2-3, 309–326.

SHARF, A., BLUMENKRANTS, M., SHAMIR, A., AND COHEN-OR, D. 2006. SnapPaste: an interactive technique for easy mesh composition. *Visual Computer 22*, 9, 835–844.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. In *Proc. SIGGRAPH*, ACM.

SHIN, H., AND IGARASHI, T. 2007. Magic canvas: interactive design of a 3D scene prototype from freehand sketches. In *Proc. Graphics Interface*.

SIMARI, P., KALOGERAKIS, E., AND SINGH, K. 2006. Folding meshes: hierarchical mesh segmentation based on planar symmetry. In *Proc. Eurographics Symposium on Geometry Processing*.

SPROULL, R. F. 1990. Parts of the frontier are hard to move. *Computer Graphics 24*, 2, 9.

TALTON, J. O., GIBSON, D., YANG, L., HANRAHAN, P., AND KOLTUN, V. 2009. Exploratory modeling with collaborative design spaces. In *Proc. SIGGRAPH Asia*, ACM.

TORRALBA, A., MURPHY, K. P., AND FREEMAN, W. T. 2007. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 5, 854–869.

UNITY TECHNOLOGIES. 2011. *Unity*. `http://unity3d.com`.

A Geometric Descriptors for Components

For each shape component C from source mesh M in the repository, we extract a $30 + |\mathcal{L}|$-dimensional feature vector containing: *a)* mean and variance of the mean curvature over the surface of C (the curvature is estimated at multiple scales over neighborhoods of point samples of increasing radii: 1%, 2%, 5% and 10% relative to the median geodesic distance between all pairs of point samples on the surface of M); *b)* mean and variance of the shape diameter over the surface of C, and of its logarithmized versions w.r.t. normalizing parameters 1, 2, 4 and 8; *c)* the following entries, derived from the singular values $\{s_1, s_2, s_3\}$ of the covariance matrix of sample positions on the surface of C: $s_1/\sum_i s_i$, $s_2/\sum_i s_i$, $s_3/\sum_i s_i$, $(s_1+s_2)/\sum_i s_i$, $(s_1+s_3)/\sum_i s_i$, $(s_2+s_3)/\sum_i s_i$, s_1/s_2, s_1/s_3, s_2/s_3, $s_1/s_2 + s_1/s_3$, $s_1/s_2 + s_2/s_3$, $s_1/s_3 + s_2/s_3$; *d)* surface area and volume of C, divided by total surface area and volume of M; *e)* for each label $l \in \mathcal{L}$, the average geodesic distance between points on C and points on M with label l, measuring the proximity of C to siblings with label l. Finally, we denoise the data by retaining only the principal components capturing 70% of the variance in the data.